ABRAVIBE – A Toolbox for Teaching and Learning Vibration Analysis

Anders Brandt, University of Southern Denmark, Odense, Denmark

A MATLAB® toolbox has been developed as a tool for teaching and learning vibration engineering and vibration analysis. This free, open software will also run under GNU Octave, if an entirely free software platform is wanted, with a few functional limitations. The toolbox functionality includes 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 article, an overview of the functionality is given and recommended use in teaching is discussed.

Vibration engineering and, even more, vibration analysis are topics often grasped only with great difficulty by many students. So powerful tools that can aid a student's learning process are important. This article presents 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:

- 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,
- Help the students' understanding of mechanical vibrations and analysis of it by being able to go through each step in calculations etc., in an open fashion.
- 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 functionality similar to 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 contains high-level commands for standard tasks needed in this field, and each and every function is open, so that the student can open and investigate it.

All functionality can start with recorded or simulated time data, the latter of which I find to be easier for students to comprehend if they themselves, for example, can go through all the typical steps of analysis such as converting data to spectra, then perhaps to frequency response functions, and then extracting the operating deflection 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, in time-limited lab assignments. Undergraduate students can thus get more "canned" demonstrations, while graduate students can be asked to develop more of the tasks themselves.

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

- Store data with header information in a standardized format, which allows for easy implementation of operating deflection shape analysis and experimental modal analysis.
- Import and export data in universal file format allows 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 and 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 References 1and 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.
- Animation of operating deflection shapes and modal analysis results.

All theory that follows is, of course, described in Reference 1. To simplify the notation below, all toolbox command variables will be written in **bold** type in the general text, while MATLAB code examples will be in the Courier font.

Data Storage Format

The basis for much of the functionality of the toolbox is the way data are stored. First, data from measurements can be stored in a standardized data format that contains information about measurement DOFs etc., to facilitate easy implementation of experimental modal analysis, for example. 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:

- Each function is stored in a separate file to allow processing of as much data as possible without memory limitations. For example. a function can 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; this 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 the section titled "Experimental Modal Analysis."

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

Based on the paper, " The ABRAVIBE Toolbox for Teaching Vibration Analysis and Structural Dynamics," presented at the 31st International Modal Analysis Conference (IMAC), Garden Grove, CA, 2013.

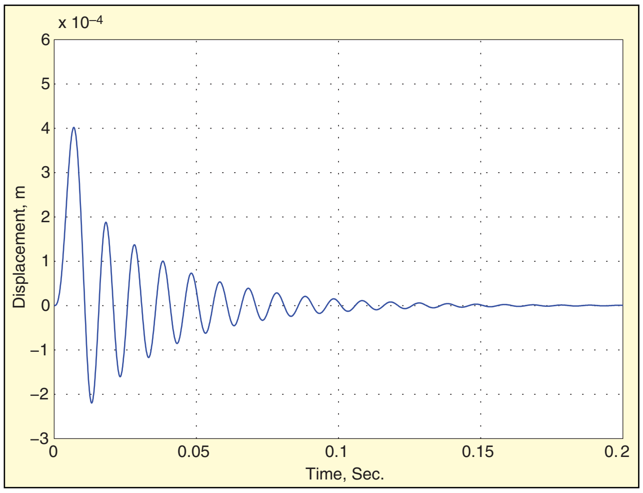

Figure 1. Plot for Example 1; SDOF response to a half-sine input force.

- 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 what 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 reasonable limits) can be computed with high-level commands.

So 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 acceleration. 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 to the command **H=mkz2frf(f,M,K,z,indof,outdof,type)**, where **z** now is a vector with the modal damping of each mode.

Teaching mechanical vibrations often includes a description of (analytical) modal analysis, where modal parameters (natural frequencies, damping ratios, and mode shapes) are computed from the system matrices. Furthermore, analyses 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 that can be used in various ways to reflect the form of damping. 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 then 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 a state-space system formulation if the damping is non-proportional. This is done for pedagogical reasons, of course, since the state-space formulation could indeed be used in both of the latter cases.

Finally, for completion, the toolbox also contains commands to convert from modal parameters to FRFs. There are also 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 using 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[4,5] and has some very important advantages:

- It is much faster and much more accurate than standard methods such as Runge-Kutta variants.[4]
- 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; this makes it very flexible.

The syntax of the command, which is called **timefresp**, depends on which of the input parameters are known. 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 $[N]$:

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

where the pulse time T = 11 ms. The code to generate the output is found in Example 1, and the result is plotted in Figure 1. A more advanced illustration of the use is found in the section titled “Frequency Response Estimation.”

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

```
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 re-
sponse in displacement [m]
```

Time-Series Analysis

Signal analysis is an important part of vibration analysis because most students taking a class in vibration analysis are mechanical or civil engineers who lack a background in signal analysis. The ABRAVIBE toolbox 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 due to 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 that 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. Despite 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 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, since it includes most of the other functions.

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 such as one-second intervals, for example, may instead reveal time-restricted errors due to intermittent errors resulting from things like cable issues, electrical spikes, etc.

To illustrate an example of a data quality assessment analysis, we use eight channels of data from a measurement on a truck supplied with the toolbox. The main code for data quality analysis of these data are shown in Example 2. 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, or 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 Figure 2. Next, the high-level command **statchkf** is used to produce a list of the 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 Figure 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.

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

```
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 com-
    mand produces the plot, see Fig. 2
    s(n)=teststat(F(:,n),0.02,'reverse'); % s con-
    tains 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 chan-
nels, 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 are shown in Fig. 3
```

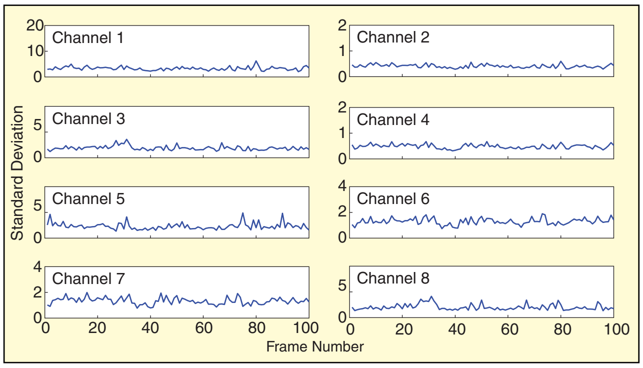

Figure 2. Results of frame statistics in Example 2; standard deviation of each channel is plotted versus frame number.

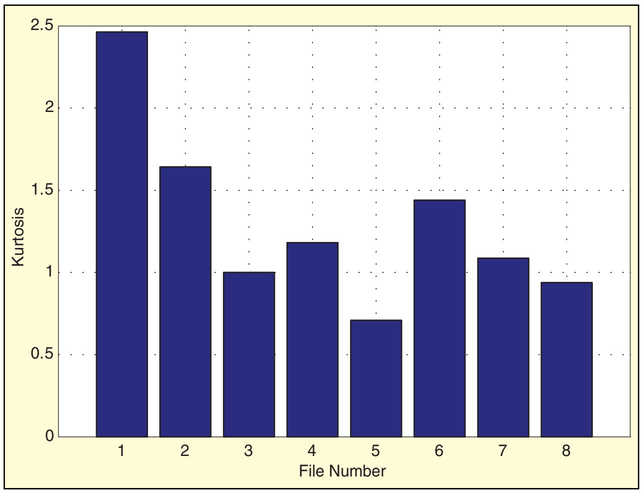

Figure 3. Kurtosis of each channel in Example 2 normalized to the kurtosis of channel 3; assuming the kurtosis of channel 3 is good, plot reveals suspicious kurtosis in at least channels 1, 2, 5 and 6.

Spectral Analysis

Spectra are perhaps the most commonly used functions for vibration analysis. Therefore, the ABRAVIBE toolbox includes a number of high-level commands to produce spectra of multichannel 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 method and also 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.

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

Table 1. Statistical parameters from output of data quality test, Example 2.

File #	Max.	Min.	Mean	Crest	Std. Dev.	Var.	Skew	Kurt.
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

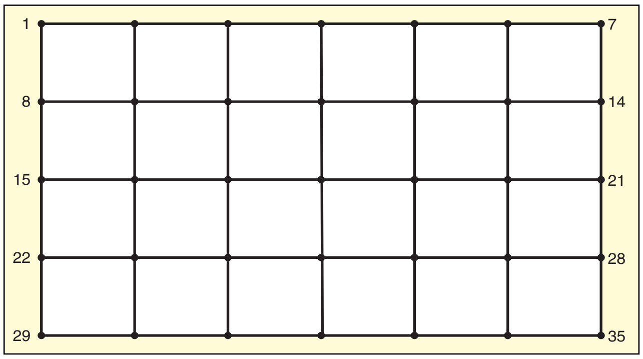

Figure 4. Grid used for reduced mode shapes and for EMA of Plexiglas plate.

method that can make it interesting:

- 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.
- 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 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.[7] This also illustrates the advanced synthesis models available in the toolbox. The code for this pedagogic example was developed in the ABRAVIBE toolbox, and also the free open CALFEM[8] toolbox, which is available in the ABRAVIBE toolbox. The example is used throughout the ABRAVIBE toolbox and is very similar to the IES plate. It is a small, rectangular plate made of Plexiglas (PMMA) and with two first modes at very close natural frequencies. The advantages of using such a simple structure are many:

- The structure geometry is trivial, so the student can focus on the topic.
- The structure is easily measured with high-quality impact testing or shaker testing in relatively short time, which allows for high-quality experimental modal analysis (EMA) to be incorporated.
- The plate can be modeled relatively simply using inexpensive shell elements.[7]
- More than the first 10 modes of the plate can be readily described with a simple 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 (see Figure 4). How to obtain such modes is described in Reference 7 and is also 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.

Example 3a. Create synthesized time data from a 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.

```
% 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 sam-
pling frequency
N=100*1024; % Use 100K samples
Forces=randn(N,length(indofs)); % Create 2 indepen-
dent Gaussian forces
Header=makehead(1,Forces(:,1),1/fs); % Create Nomi-
nal 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 re-
sults, 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 sec-
ond force
Header.Dof=indofs(2);
Header.Dir='Z+';
Header.Unit='N';
Header.Title='Simulation using Plexi FE model re-
sults, 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
```

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

- $\mathbf{G_{xx}}$ – N/2+1-by-R-by-R, input autospectral matrix
- $\mathbf{G_{yx}}$ – N/2+1-by-D-by-R, input-output cross-spectral matrix
- $\mathbf{G_{yy}}$ – N/2+1-by-D, output autospectral matrix (no need for cross-spectral densities between outputs)

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.

```
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
```

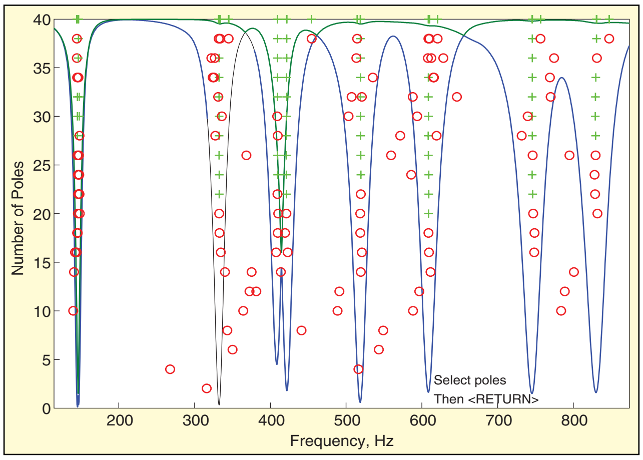

Figure 5. Stabilization diagram of polyreference time domain estimation for Example 4.

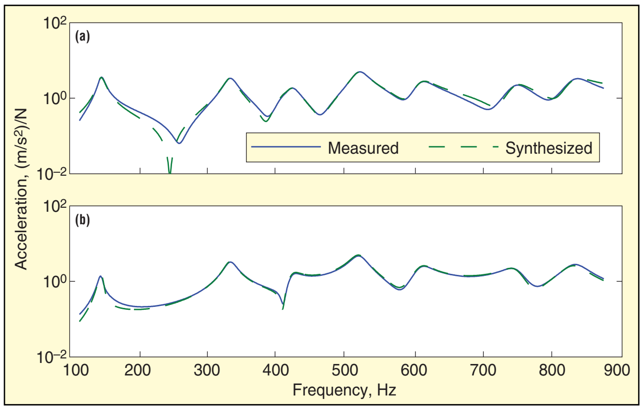

Figure 6. Measured and synthesized FRFs from estimation of residues (mode shapes) for DOF 35.

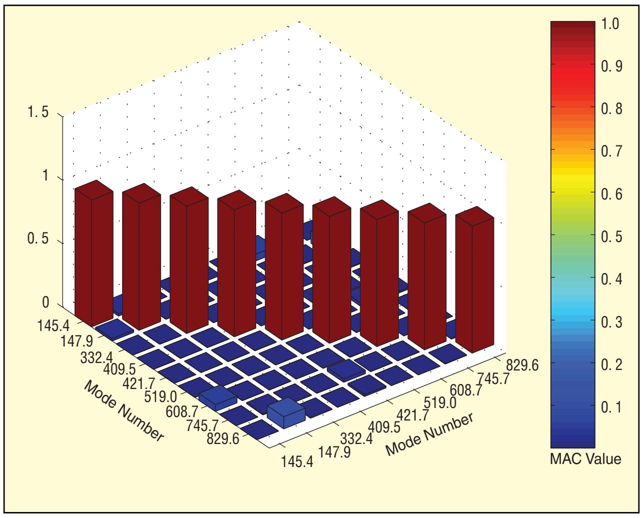

Figure 7. MAC matrix of mode shapes from least-squares frequency domain estimation using poles and modal participation factors from polyreference time-domain estimation for Example 4.

```
,N); % 50% overlap, Hanning window
```

Note that there are a lot of details in the chapter examples about how to optimize the spectral analysis FFT settings, such as the block size for minimizing bias error, etc. Note also 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 could also incorporate known error signals (extraneous noise) either on the input or output signals to analyze bias effects when estimating FRFs. Numerous such examples are available among the chapter examples that deal with FRF estimation (Chapters 13 and 14 in Reference 1).

Frequency Response Estimation

The next natural step after producing auto and cross-spectral density matrices 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.[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 estimate poles and sometimes modal participation factors (MPFs) using either Prony's method (for one FRF at the time[9]), the least squares time domain method (LSCE)[10], or by the polyreference time domain method.[11] Mode shapes can then be computed using the command **frfp2modes**, which includes a multireference least squares 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 using the command **data2hmtrx**. It also rearranges data coming from an impact test with roving impacts so that it looks as if it came 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 – 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 display (see Figure 4).

To illustrate an EMA example, we will use data from an impact test on the Plexiglas plate mentioned previously. Of course, we could use the FRF matrix computed earlier, but that would not allow us to see how actual measurement data are used for EMA. In Example 4, there is 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 (see Figure 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 Figure 6) together with the corresponding measured function if so is requested. After the mode shapes are computed, a MAC matrix is computed and plotted, as shown in Figure 7. All that remains is to save the modal parameters to a file for subsequent animation, which will not be shown here.

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

```
Prefix='PlateH';
[H,f,Rdof,Rdir,Fdof,Fdir,FillMtrx]=data2hmtrx(Pre
fix,1,105);
[p,L,fLimits] = frf2ptime(H,f,400,20,'mvmif','p
td'); % 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);
```

Order Tracking

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

Summary

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 authors, Dan Lazor and Dave Brown, University of Cincinnati, for allowing incorporating and distributing the **animate** GUI-based animation software with the ABRAVIBE toolbox.

Software Availability

The ABRAVIBE toolbox is available for free download from www.abravibe.com. Matlab® is available in a free trial version and under various purchase agreements from www.mathworks.com. The GNU Octave software is available for free download from www.gnu.org/software/octave/. H. A. Gaberson wrote a paper "Free Pseudovelocity Shock Data Analysis Software Using GNU Octave" that includes detailed instructions for downloading and installing GNU Octave and a tutorial on pseudovelocity shock analysis. Gaberson relates, "This paper will give you a big start in doing shock data analysis on your own. It shows how I downloaded and installed GNU Octave 3.2.4 into my Microsoft Windows XP Pro laptop computer, and then shows how to do some of the basic shock data analysis calculations and plotting. Once you can do analysis for yourself, you gain understanding of data analysis. It will give you a start and enough confidence to see that you can learn it." This paper is available on request from sv@mindspring.com. A ZIP file of his "m" scripts is also available from S&V.

References

1. Brandt, A., *Noise and Vibration Analysis – Signal Analysis and Experimental Procedures*, John Wiley and Sons, 2011.
2. Brandt, A., *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., and Brincker, R., "Impact Excitation Processing for Improved Frequency Response Quality," *Proc. 28th International Modal Analysis Conference*, Jacksonville, FL, 2010.
4. Brandt, A., and Ahlin, K., "A Digital Filter Method for Forced Response Computation," *Proc. 21st International Modal Analysis Conference*, Kissimmee, FL, 2003.
5. Ahlin, K., Magnevall, M., and Josefsson, A., "Simulation of Forced Response in Linear and Nonlinear Mechanical Systems Using Digital Filters," *Proc. International Conference on Noise and Vibration Engineering (ISMA)*, Catholic University, Leuven, Belgium, 2006.
6. Brandt, A., and Linderholt, A., "A Periodogram-Based Method for Removing Harmonics in Operational Modal Analysis," *Proc. of the International Conference on Noise and Vibration Engineering (ISMA)*, 2012.
7. Sturesson, P. O., Brandt, A., and 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., and Wernberg P. A., *Calfem – A Finite Element Toolbox*, Version 3.4, Studentlitteratur AB, 2004.
9. Proakis, J. G. and Manolakis, D. G., *Digital Signal Processing: Principles, Algorithms, and Applications*, Prentice Hall, 2006.
10. Brown, D., Allemang, R., Zimmerman, R., and Mergeay, M., "Parameter Estimation Techniques for Modal Analysis," SAE Tech. Papers, 1979.
11. Vold, H, Kundrat, J., Rocklin, T. G., and Russell, R., "A Multiple-Input Modal Estimation Algorithm for Mini-Computers," SAE Tech. Papers, 1982.

The author may be reached at: abra@iti.sdu.dk.