TOOLBOXES FOR ANALYSIS OF SOUND AND VIBRATION SIGNALS WITHIN MATLAB

A. Brandt[1], J.Tuma[2], T. Lagö[1], and K. Ahlin[3]

Axiom EduTech AB[1], Technical Univ. of Ostrava[2], Blekinge Institute of Technology[3]

Abstract

Commercial tools for measurement and analysis of noise and vibration signals have traditionally been very expensive. In the last decade, however, multi-channel measurement systems have become relatively inexpensive. The analysis functionality in most inexpensive instruments is limited. Therefore, many companies are using alternatives for post processing of measurement results. MATLAB is a platform that is popular for this purpose and which offers many advantages over dedicated, menu driven systems. The open functions in MATLAB assure flexibility and the possibility to modify functions for specific needs. In addition, a command based programming environment provides for traceability and quality assurance, including qualification of used algorithms, important aspects particularly in the aerospace industry. In the past, basic functions for physically scaled signal analysis and vibration analysis have had to be developed before taking full advantage of the MATLAB platform. In this paper we present a new suite of toolboxes, developed by Axiom EduTech, that turn MATLAB into bleeding edge software for noise and vibration analysis. The toolboxes presented here have been well proven in applications in automotive and aerospace industry. Parts of the toolboxes are provided as freeware, and everyone is encouraged to download this from Internet.

1 INTRODUCTION

In the past couple of decades, commercial systems for multichannel noise and vibration data acquisition have developed from large, workstation based, and expensive systems, to small, PC based, and inexpensive ones.

For post processing and analysis of experimental vibration data, commercial systems offer menu-driven interfaces for many common analysis functions. These systems are good at handling large amounts of data during measurements, but they also have some drawbacks. Menus are handy for standard calculations, but they give limited functionality for the user who wants to do something outside the given scope. The flexibility of a menu-based system is necessarily limited, and it is difficult to know exactly what the software performs behind the menu choices. There is also a lack of traceability with any menu selected analysis procedure.

Lately, MATLAB has become a standard for mathematical processing of data and models in universities as well as in industry and has proven to be a powerful mathematics tool. A great advantage using MATLAB for the analysis of experimental noise and vibration signals, is that the user is forced to understand the analysis process more comprehensibly than when using menu-based software. This means that there is a learning threshold to pass, but once this threshold is passed, the user is richly rewarded. The MATLAB path offers automatic traceability, especially important in many aerospace applications. Provided that the scripts used are open, the user can also see what is done in each function. This provides for an increased learning process, but is also imperative in many certification procedures.

A drawback until now with using MATLAB for the processing of noise and vibration signals has been that the user has had to implement the involved analysis procedures, as there are no direct, physically scaled functions for spectrum analysis, etc. in MATLAB. In order to succeed with this, the user has had to be rather experienced in mathematics, signal analysis, and structural dynamics. With the introduction of commercially available toolboxes, such as VibraTools™ from Axiom EduTech, designed for noise and vibration analysis, using MATLAB is greatly simplified.

In the VibraTools™ suite of toolboxes, functions for data filtering, spectral analysis, octave analysis, order tracking, modal analysis, simulation of mechanical systems, durability analysis and much more are available as high-level commands.

2 DATA ACQUISITION AND IMPORT

An increasing number of data acquisition systems support data acquisition from within MATLAB. Furthermore, almost all modern data acquisition systems have a possibility to transfer acquired data to MATLAB. In many cases, though, the transferred data lack information about measurement degrees-of-freedom, engineering units, date and time of acquisition etc. In a more complete data format such as the Universal File Format, this kind of information is contained in a header. In the toolboxes, the header concept has been implemented to full extent, and the header format and MATLAB commands for handling the header are available as freeware, see below. When data are imported to MATLAB, a header is created. In most cases this means that a function is written to import the more complete (binary) specific file format of a particular data acquisition system. As an example, the function datread reads data from SONY PC SCAN files that have been created from a SONY DAT recording. During the import, data may be down-sampled to a lower sampling rate, performed under full anti-aliasing protection. The created header is an ASCII file that may look like shown in Example 1 (The formatting is not correctly reproduced).

Example 1: Data Header

```
Cutting. Test #14
ID number   101
98-02-13    08:44:29
SONY PCscan MK II file
Voltage range    10 V
   201        1        1    480000        0        0        0        0
0  0.0000e+000      0  0.0000e+000      0  0.0000e+000      0  0.0000e+000
Time              sTime             s  0.0000e+000  4.1667e-005      0      0
acc. x             m/s2
```

The format of the header is the same regardless of the data source and the functions in the toolboxes know where to look for information. In the example above, 480000 is the number of samples of data and 4.1667e-005 is the sampling interval, corresponding to a sampling frequency of 24 kHz.

3 DATA ANALYSIS

In order for a relatively inexperienced user to be able to use the toolboxes, the functions in the toolboxes are implemented at a “practical level”. Each function performs a high-level task, but still forces the user to understand the variables involved. The data corresponding to the header shown in Example 1 above are used to illustrate the level of the commands, see Example 2. In this example, commands for plotting the time function, as in Figure 1, are shown.

Example 2. Commands to plot time data with header info.

```
>> load cutting14
>> fs = headfs(Header);
>> t = maketime(Data,fs);
>> plothead(t,Data,Header)
```

The load function loads the data file, which has two parts (MATLAB variables): Header and Data. The sampling frequency fs is extracted from the header, and a time vector with the same length as the data is produced. The plothead function gives a first rough (but properly scaled) plot of the signal, see figure 1.

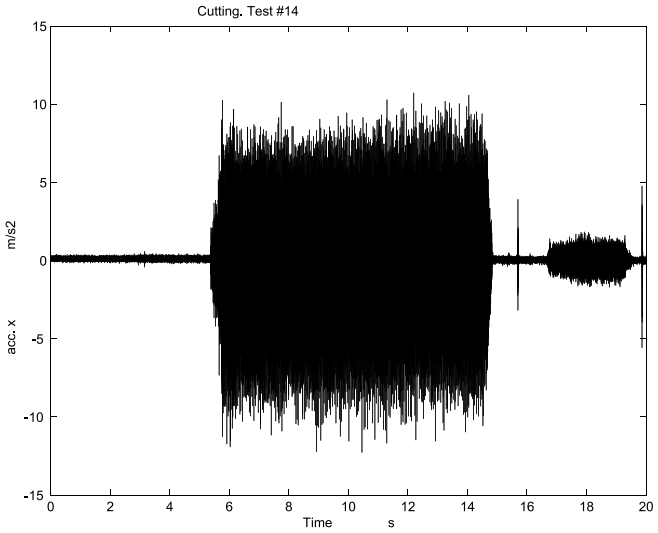

Figure 1: Plot of time history.

The data presented in Figure 1 are vibrations from a cutting process. To extract the main part of the signal (between 6 and 14 seconds) the function sigtrunc (signal truncation) is used. A normalized frequency analysis using an ISO standardized flat top window is then performed and the result is plotted. All the standard MATLAB graphic features may be used to put labels and text to the figure, adjust scales and colors, etc.; the basic commands are shown in Example 3.

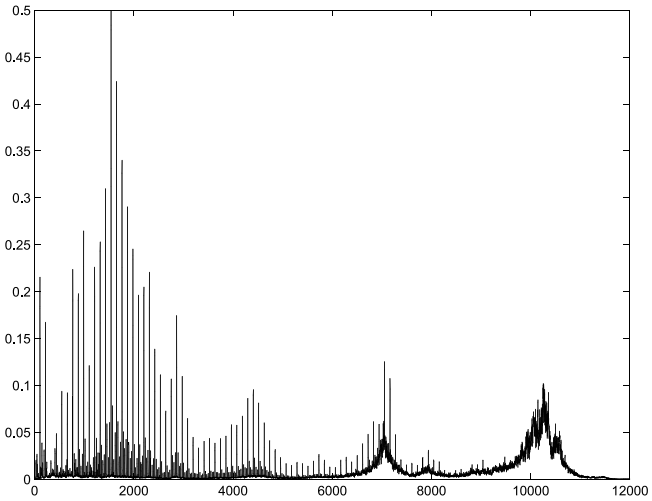

Figure 2: Raw plot of frequency analysis result.

Example 3: Frequency analysis.

```
>> x = sigtrunc(Data,fs,6,14);
>> [z,f] = fanflat(x,fs,32768,10,75);
>> plot(f,z,'k');
```

In the fanflat function (frequency analysis with flat top window) 32768 is the FFT block size, 10 is the number of averages and 75 is the overlap percentage. The raw plot is shown in figure 2.

If a one-third octave band analysis is needed, this can be produced either by digital filters on the raw time data, or by converting a power spectral density spectrum. Commands for the latter procedure are shown in Example 4.

Example 4. Converting a PSD into 1/3-octave spectrum

```
>> [p,f] = psdnorm(x,fs,8192);
>> [zt,ft] = psd2ters(p,f);
>> plotters(ft,zt);
```

The result of Example 4 is shown in Figure 3, where labels etc have been added using standard MATLAB graphics functions.

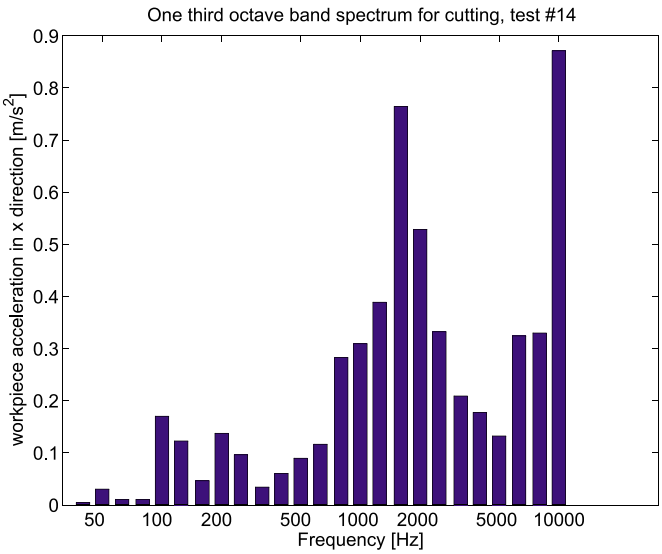

Figure 3: One third octave band analysis of vibrations from a cutting process

4 Whole-Body Vibration

The international standard ISO 2031 provides methods to characterize vibrations experienced by humans, for instance whole-body vibration. For example, the vibration measured at the driver's seat in a truck should be filtered with a frequency-weighting filter, called Wk, which is defined in the standard. Then the "running rms" of the filter output should be calculated with a time constant of one second. The maximum value of the running rms is one of the parameters sought for. All ISO 2631 filters are implemented in the toolboxes and so is the running rms calculation. In Example 5, commands for loading data from a measurement on a truck using a seat pad accelerometer, and commands for calculating the sought maximum value are shown.

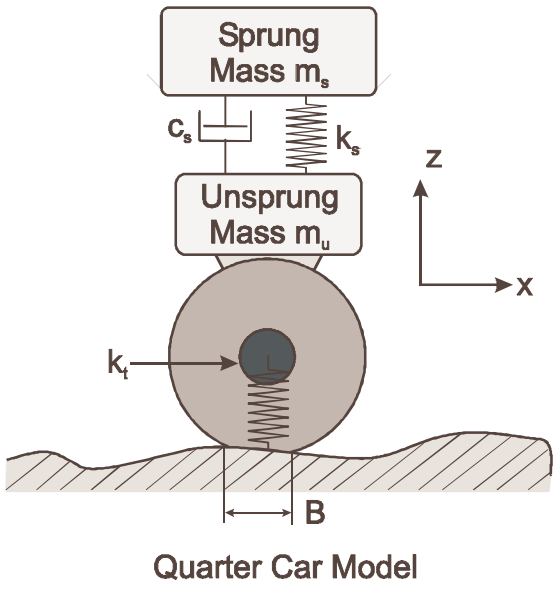

Figure 4: Quarter car model used for IRI calculation

Example 5. Comfort analysis of driver seat acceleration

```
>> load dat8546
>> fs = headfs(Header);
>> y = isofiltw(Data,fs,'Wk');
>> z = runrmsln(y,fs,1.0);
>> a = max(z);
```

5 Forced Response Computation

Measured truck seat vibrations may be compared with simulated vibrations using a measured road profile and a mechanical model of the truck. In the toolboxes there are several functions that calculate forced response of mechanical systems. The systems may be defined in several ways:

- Mass matrix M, damping matrix C and stiffness matrix K,
- Mass matrix, stiffness matrix and modal damping, or
- Poles and residues, for example from FEA modeling or experimental modal analysis.

The IRI is defined as the total damper stroke (in inches) divided by the traveled distance (in miles). In Europe the unit is usually mm/m. The simulated vehicle speed should be 80 kmph. So, to calculate IRI, the system is set up and IRI computed as described in Example 6.

Example 6. Calculation of IRI

```
M = [ms 0;0 mu];
C = [cs -cs;-cs cs];
K = [ks -ks;-ks kt+ks];
fs = 80*1000/3600/step;
[y1,t] = timeresv(z*kt,fs,M,C,K,2,1,[1 2]);
[y2,t] = timeresv(z*kt,fs,M,C,K,2,2,[1 2]);
IRI = 1000*sum(abs(y1-y2))/fs/(length(z)*step);
```

In Example 6, first the mechanical 2-degree-of-freedom system is set up, and the velocity responses of the two masses are calculated with the measured profile z as input. The profile is sampled with the step “step” in distance. The function timeresv (time response, velocity) computes the vibration velocity. z*kt is the applied force, fs is the sampling frequency, the system is defined by M, C and K. The next parameter defines the input and the following defines the output. [1 2] tells the function to use modes 1 and 2 (which is all there is in this case). Then the relative velocity is integrated (summed) into total stroke and IRI (in mm/m) is found by dividing with the traveled distance. The algorithm used for computing the forced response, is a unique modal superposition method which applies digital filters in a way similar to what is commonly used for computing shock response spectra [1, 2].

6 ROTATING MACHINERY ANALYSIS

In many applications of sound and vibration analysis, engines, pumps, generators and other rotating or reciprocating equipment are commonly analyzed using *order tracking*. This analysis method can be implemented using many different methods [3], including synchronous resampling and Vold-Kalman filters. An advantage by using MATLAB as the tool for order tracking, is that all analysis can be done on raw time data, whereby different analysis techniques can be applied to the same data for comparison purposes. In Figure 5 a color intensity diagram of spectrum data from a runup is shown along with an order track of order 2.

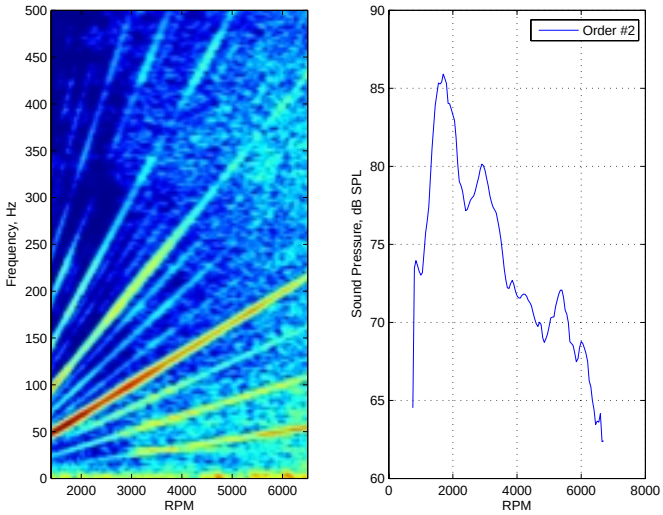

Figure 5: Plot of spectra vs. Rpm (right) and extracted second order, overlayed with second order extracted using Vold-Kalman filters.

7 MODAL ANALYSIS

There are many powerful functions in the toolboxes for simulation of mechanical systems, forced response computations, modal parameter extraction, animation and more. To give an example of the level of the functions, the steps from measured FRFs to poles and residues are given in Example 7. The FRFs are stored in Universal File Format in a file T1.unv.

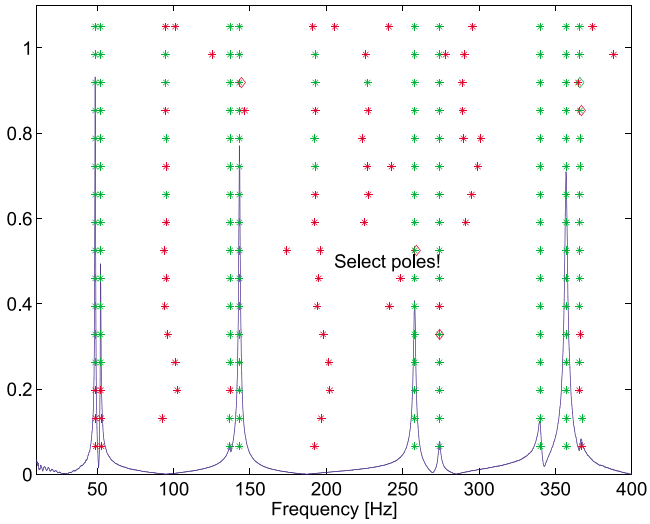

Figure 6: Stability diagram in complex exponential calculation.

Example 7. Modal parameter extraction

```
univread(1,'T1.unv');
[hmat,f] = uf2frf(1,20);
H = cvfrfa2v(hmat,f);
[H,f] = frftrunc(H,f,0,400);
[h,t,fs] = impresp(H,f);
poles = complexp(h,fs,250,44,abs(H(:,1))/.9,f,10,400);
[residues,residuals] = pol2resf(H,f,poles,10,400,1);
```

The function complexp produces a stability diagram, from which the poles are selected by the user, see Figure 6. Thereafter the functions animcalc and animate can be used to produce animated

displays of the modes. There are also functions to produce Windows AVI movie files of an animated mode.

8 DURABILITY ANALYSIS

There is a separate toolbox for analysis of data for prediction of fatigue and for experimental environmental engineering and durability data analysis. There are cycle count methods such as level crossing (lvlcross), range pair count (rangepair) and rain flow cycle count (rainflow). Once the cycles have been counted, a simple function fatigue calculates the damage according to Miner's rule.

For environmental engineering analysis of vibrations, there are functions for shock response spectra, and classification of signals as defined by ISO TC 10811-1 and TC10811-2. There are also functions for generating environmental test specifications from live data.

An example of shock response spectrum of a half sine pulse is shown in Example 8. The resulting diagram produced by Example 8 is shown in Figure 7.

Example 8. Producing a shock response spectrum

```
>> fs=
>> [M,R,mx,rx] = rainflow(Data,40,-300,300,2);
>> rainplot(M,mx);
```

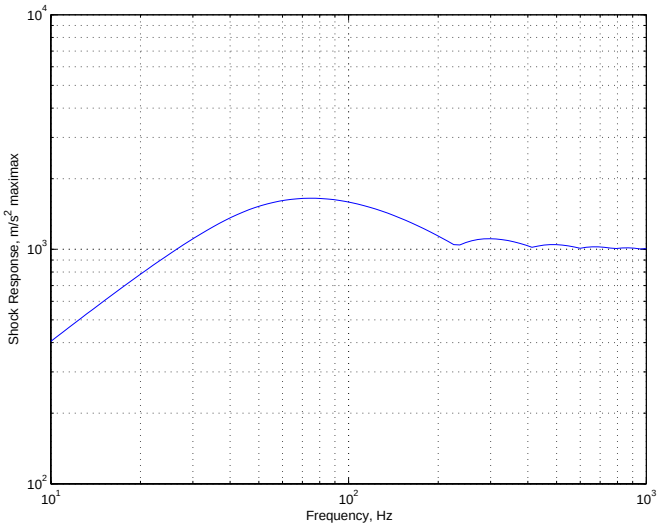

Figure 7: Mean-range diagram from Example 8. Labels have been added using standard MATLAB commands.

9 CONCLUSIONS

In this paper we have shown some examples of how MATLAB can be used for analyzing data from some different vibration applications. The main advantages with using MATLAB are that:

- it forces the user to a certain level of understanding of the analysis procedure,
- certification of routines is simplified by open functions, and
- traceability is ensured, using the same MATLAB program for each analysis.

With the presented toolboxes now available for noise and vibration analysis, also the less experienced user can take advantage of the power and flexibility of MATLAB, as a complement to an existing system, or as the only tool for analysis.

References

[1] Ahlin, K., “On the use of Digital Filters for Mechanical System Simulation,” Proc. Shock and Vibration Symposium, 2003.
[2] Brandt, A., Ahlin, K., “A Digital Filter Method for Forced Response Computation,“, Proc. IMAC XXII, Orlando, FL, 2003.
[3] Brandt, A., Lagö, T., Ahlin, K., and Tuma, J., “Main Principles and Limitations of Current Order Tracking Methods,” Proc. IMAC XXIII, Orlando, FL, 2005.

Anders Brandt
Axiom EduTech AB, Laggarsvägen 36, SE-184 97 Ljusterö, Sweden. Email: anders.brandt@axiom-edutech.com. Vibratools Freeware can be downloaded from www.vibratools.com.

Jiri Tuma
Dept. of Control Systems and Instrumentation, ul. 17.listopady c. 15, 7.patro, 708 33, Ostrava-Poruba, The Czech Republic. Email: jiri.tuma@vsb.cz.

Thomas Lagö
Axiom EduTech Inc., 70 Himalaya Ct, Alpine, Utah 84004, U.S.A. Email: Thomas.lago@axiom-edutech.com.

Kjell Ahlin
Dept. of Mechanical Engineering, Blekinge Institute of Technology, SE-371 79 Karlskrona, Sweden. Email: kjell.ahlin@bth.se.