MATLAB TOOLBOXES FOR ADVANCED NOISE AND VIBRATION ANALYSIS

Anders Brandt,	Axiom EduTech AB	Sweden
Thomas Lagö,	Axiom EduTech Inc.	USA
Kjell Ahlin	Blekinge Institute of Technology	Sweden

anders.brandt@axiom-edutech.com

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 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 that turn MATLAB into state-of-the-art software for noise and vibration analysis. A great advantage with using the toolboxes is that the included open functions allow the user to get a deeper insight into analysis procedures. This ensures a learning experience during the use of the toolboxes. The toolboxes include comprehensive functionality for statistics and quality assurance of large data sets and are designed to be used with large amounts of data and large channel counts. Spectrum analysis with, for example, general narrowband and octave band filters are included. There are also functions for analysis of engine vibrations with fixed frequency as well as synchronous resampling, and order tracking. In addition, advanced applications include experimental modal analysis and simulation of mechanical systems, durability analysis, and environmental engineering with response spectrum analysis etc. High-level commands make it possible to perform advanced analysis of vibration signals without being a specialist in numerical methods and signal processing. In the paper we present a new method to compute forced response. The method, developed by Axiom EduTech, is based on a modal superposition, where the single-degree-of-freedom systems are approximated by state-of-the-art digital filters. An overview of the involved errors are presented, and it is shown that the algorithm is approximately 2,000 times faster than a standard Runge-Kutta solution, for a typical system.

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 applications, 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.

Matlab has become a standard for mathematical processing of data and models in universities as well as in industry. A great advantage using Matlab for the analysis of experimental noise and vibration signals is that the user is encouraged to understand the analysis process more comprehensibly than when using menu-based software. Although there is a slight learning threshold to pass, once this threshold is passed, the user is rewarded. Using Matlab offers automatic traceability, especially important in research but also in other areas, for example in aerospace. Provided that the scripts used are open, the user can also see what is done in each function. Thus Matlab provides for an easy learning process, 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 analysis procedures involved, as there are no direct, physically scaled functions for spectrum analysis, etc. in Matlab. With the introduction of commercially available toolboxes designed for noise and vibration analysis, even a relatively inexperienced user can use Matlab for analysis of noise and vibration signals. 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

Data acquisition with the Data Acquisition Toolbox available from MathWorks, the makers of Matlab, supports many data acquisition products suitable for noise and vibration measurements. 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 for 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 VibraTools toolboxes, the header concept has been implemented and the header format and Matlab commands for handling the header are available as freeware, [1]. When data are imported to Matlab with the built-in import functions, a header is created. 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 in **Example 1** (the formatting is not correctly reproduced).

2.1 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
     1         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 in the data and 4.1667e-005 is the sampling interval, corresponding to a sampling frequency of 24 kHz.

3 DATA ANALYSIS

In order for a reasonably 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, a concept that encourages the user to understand the variables involved. The data corresponding to the header above are used to illustrate the level of the commands, see **Example 2**. In this example, commands for plotting the time function, as in Fig. 1, are shown.

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

```
>> load cutting14
>> t = headtime(Header);
>> 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 Fig. 1.

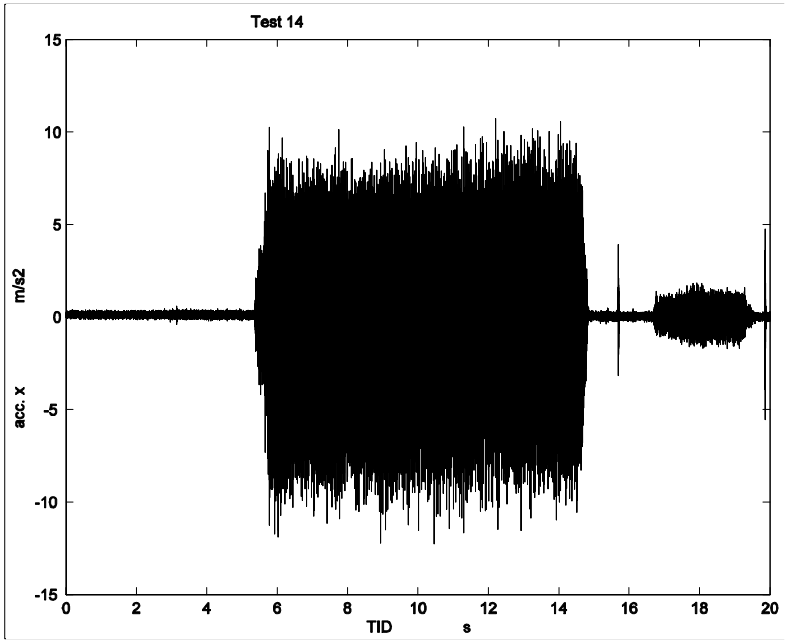

Figure 1. Plot of time history.

The data presented in Fig. 1 are the vibrations of 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**.

3.2 Example 3: Frequency analysis.

```
>> x = sigtrunc(Data,fs,6,14);
>> [z,f] = fanflat(x,fs,16384,10,75);
>> 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 Fig. 2.

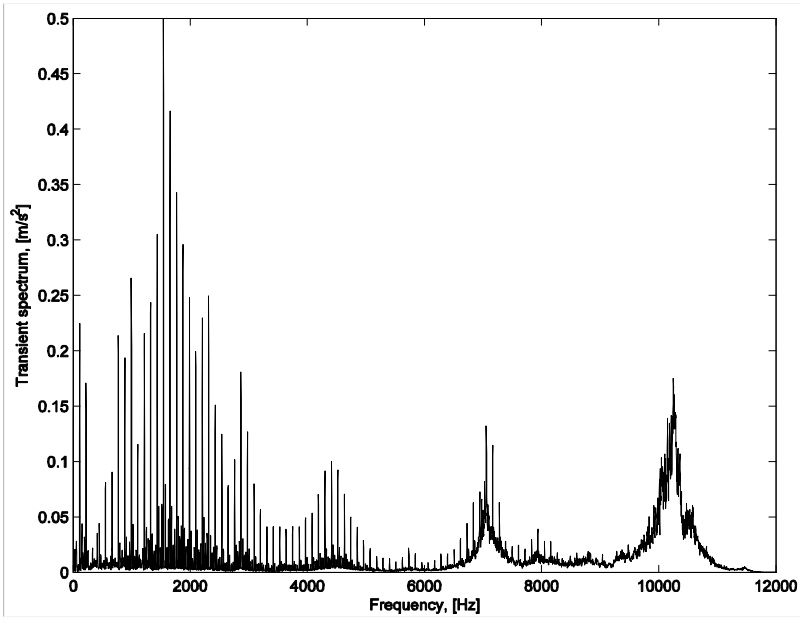

Figure 2. Raw plot of frequency analysis result.

If a one-third octave band analysis is required, this can be produced either by digital filters operating on the raw time data, or by converting a power spectral density spectrum. Commands for the former procedure, for all third-octaves between 20 Hz and 4 kHz, are shown in **Example 4**.

3.3 Example 4. Time domain 1/3-octave spectrum

```
>> [Lp,f] = fanters(y,fs,20,4000);
>> plotoct(f,Lp,std(y))
```

The result of **Example 4** is shown in Fig. 3, where labels etc. are automatically applied by the command plotoct. The "total level" that is, the rms value of the time signal, is included in the rightmost bar in the figure.

The international standard ISO 2031 provides methods to characterize vibrations experienced by humans, for instance whole-body vibrations. 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 a measured truck seat vibration and calculating the sought maximum value are shown.

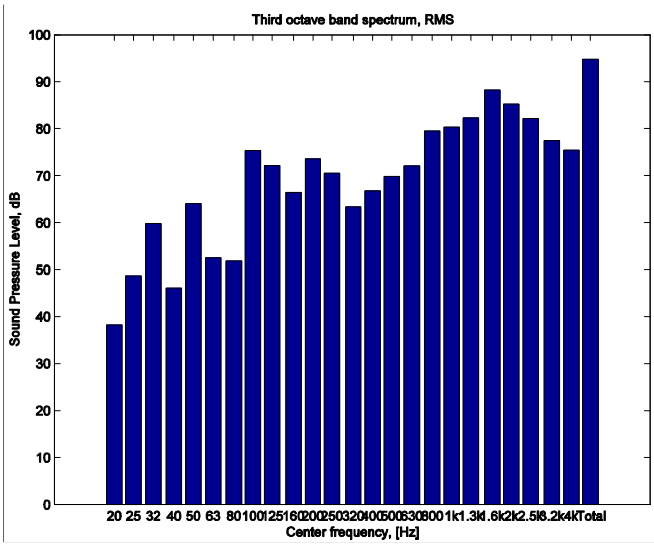

Figure 3. One-third octave band analysis of vibrations from a cutting process

3.4 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);
```

4 Order tracking

In analysis of rotating equipment, order tracking is a common application. It usually involves producing plots of spectra at various rpm, and from such plots usually the *order*, that is the instantaneous rms value of a particular multiple of the fundamental frequency, is extracted as a function of rpm. Functionality for order tracking is included in the toolboxes [2], and Fig. 4 shows sound pressure from a microphone signal measuring exhaust noise from an automobile.

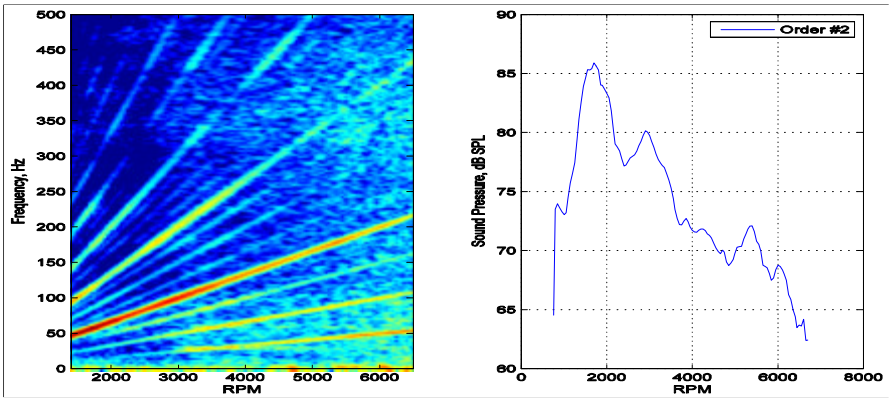

Figure 4. Plot of spectrum content as a function of rpm in a color intensity plot (left), and order 2 extracted from this plot (right).

5 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 delivered in Universal File Format in a file T1.unv.

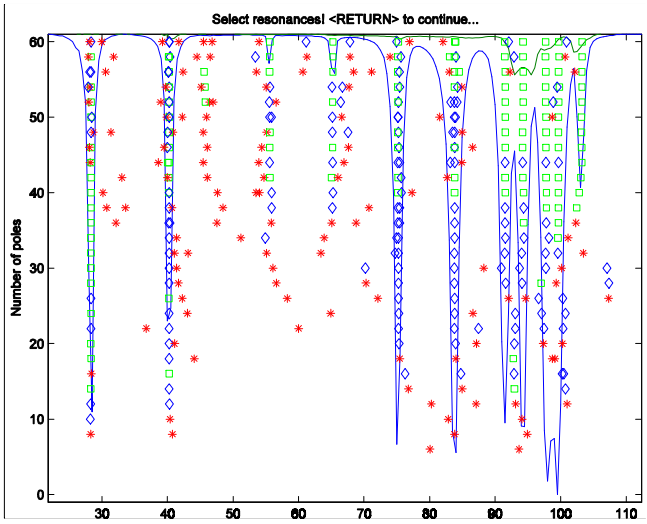

Figure 5. Stability diagram as part of Polyreference Time Domain calculation on an aeroplane model.

5.1 Example 7. Modal parameter extraction

```
>>univread(1,'planemodel.unv');
>>[H,f,ForceNo,ForceDir,RespNo,RespDir]=uf2frf(lo,hi,'no');
>>Hv=cvfrfa2v(H,f);
>>[h,t,fs] = impresp(H,f);
>>MIF = muvamif(Hvt); [poles,MPF] = polyref(h,fs,10,30,MIF,ft,flo,fhi,ft(1));
>>ModeShapes = lsfd(Hvt,ft,poles,MPF,Type,DispTime);
```

The function polyref produces a stability diagram, from which the poles are selected by the user, see Fig. 5. Thereafter functions are available to produce animated displays of the modes. There are also functions to produce Windows AVI movie files of an animated mode.

6 FORCED RESPONSE COMPUTATION

Computing the responses in one or more locations, using a theoretical model is referred to as forced response computation. This can be accomplished in a variety of ways. Computing forced response from time data, is usually done in one of principally two different ways; either using a differential equation solver based on a variant of a Runge-Kutta solver, and directly solving the Newton equations, or by using modal superposition, where each mode is simulated separately, and the solution is produced as a superposition of the contribution of all modes. In the VibraTools Suite, the latter approach has been implemented, but with a revolutionary technique [3], using digital filters instead of the usual convolution by the impulse response of each mode. This makes a computational saving of hundreds of times. As an example, computing all three responses for 2 Mega samples of data from two independent forces, on a 3-degree-of-freedom model, takes

approximately 0.5 seconds on a laptop computer with a 1.5 GHz Pentium processor. In addition, the modal superposition formulation allows several different options for the model used. You can thus use either the

1. mass matrix, damping matrix, and stiffness matrix, or
2. mass and stiffness matrices and modal damping, or
3. poles and residues, for example from FEA modeling or experimental modal analysis.

7 DURABILITY ANALYSIS

There are several functions in the toolboxes to analyze data for prediction of fatigue. 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. An example of a rain flow cycle count analysis is shown in **Example 8**.

7.1 Example 8. Producing range-mean diagram

```
>> load sf11mp6
>> [M,R,mx,rx] = rainflow(Data,40,-300,300,2);
>> rainplot(M,mx);
>> set(gca,'FontSize',14
>> axis([-300 300 0 600 0 3500])
>> title('Rain Flow Cycle Count on test signal','FontSize',14)
>> ylabel('Range','FontSize',14)
>> xlabel('Mean','FontSize',14)
>> zlabel('Counts','FontSize',14)
```

The resulting diagram produced in **Example 8** is shown in Fig. 6.

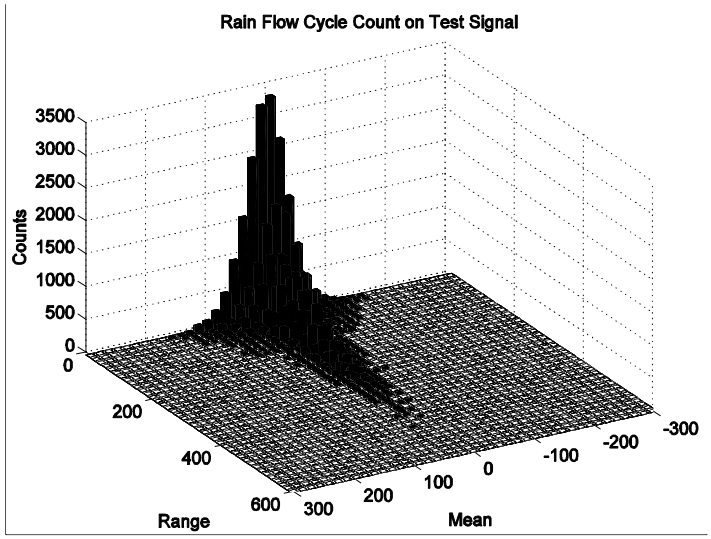

Figure 6. Mean-range diagram from Example 8.

8 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:

1. it encourages the user to a certain level of understanding of the analysis procedure,
2. many analysis procedures are greatly reduced in time
3. certification of routines is simplified by open functions, and
4. 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 measurement system, or as the only tool for analysis.

9 REFERENCES

1. Downloadable freeware from www.vibratools.com.
2. Brandt, A., et Al., "Main Principles and Limitations of Current Order Tracking Methods,", Proc. IMAC XXIII, 2005.
3. Ahlin, K., "On the use of Digital Filters for Mechanical System Simulation," Proc. Shock and Vibration Symposium, 2003.