A Smart Way to Analyze Dynamic Data

Kjell Ahlin and **Anders Brandt,** Saven EduTech AB, Täby, Sweden

In recent years MATLAB® has become a common software tool for general computational mathematics, in universities as well as in industry. In the field of noise and vibration analysis, MATLAB is very common in universities, but perhaps a little less common in the industrial world. In this article some ideas are presented on how MATLAB can be successfully used for analyzing experimental noise and vibration data. Through the introduction of toolboxes in this field, the less experienced user can take advantage of the powerful functionality of MATLAB, either as the main tool or as a complement to the many excellent menu driven systems available on the market. Some of the advantages and disadvantages of using MATLAB versus menu driven systems are also discussed.

In the past ten years, commercial systems for multichannel noise and vibration data acquisition have developed from large, workstation based, expensive systems, to small, PC based, 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 does behind the menu choices. There is also a lack of traceability – who knows what buttons were pressed to obtain a particular plot?

On the other hand, in the last few years MATLAB has become a standard for mathematical treatment of data and models in universities as well as industry and is a powerful mathematics tool. When using MATLAB, the user is forced to know what is being done. This means that there is a threshold to achieve on the learning curve. But once that has been passed, the user is rewarded. The MATLAB path offers automatic traceability and as the scripts are completely open, the user can see what is done in each function.

One drawback until now has been that users had to be experienced in mathematics, signal analysis and structural dynamics knowledge to properly use MATLAB for noise and vibration analysis. With the introduction of toolboxes designed for noise and vibration analysis, even a relatively inexperienced user can benefit from MATLAB in this field. In the VibraTools™ suite of toolboxes, functions for data filtering, spectral analysis, octave analysis, modal analysis, simulation of mechanical systems, durability analysis and much more are available as high-level commands.

Data Acquisition and Import

There is an increasing number of data acquisition systems that support data acquisition from within MATLAB. Furthermore, almost all modern data acquisition systems are capable of transferring acquired data to MATLAB. In many cases though, the transferred data lack information about channel names, 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 its full extent. 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

---

MATLAB® is a registered trademark of The MathWorks Inc.

data from SONY PC SCAN files that have been created from a SONY DAT recording. During the import, the 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:

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 the 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.

Data Analysis

The functions in the toolboxes are meant to be implemented at a practical level. There is no extreme high level function such as **analyze_the_data**. The functions do make life easier for the user, but still force him or her to have full control over what is happening.

The data corresponding to the header above are used for a simple example. The following MATLAB lines produce a first plot of the signal.

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

The **load** function loads the data file, which has two parts: 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. Figure 1 was generated with these functions and shows vibration from a cutting process. To extract the main part of the signal (between 6 and 14 sec) 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.

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

In the **fanflat** function (**f**requency **an**alysis 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 desired, the program starts with a normalized power spectral density spectrum, which is converted to a one third octave band spectrum.

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

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

ISO 2631 gives methods to characterize vibration experienced by a human such as whole body vibration. For example, the vibration measured at the seat of 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 sec. The maxi-
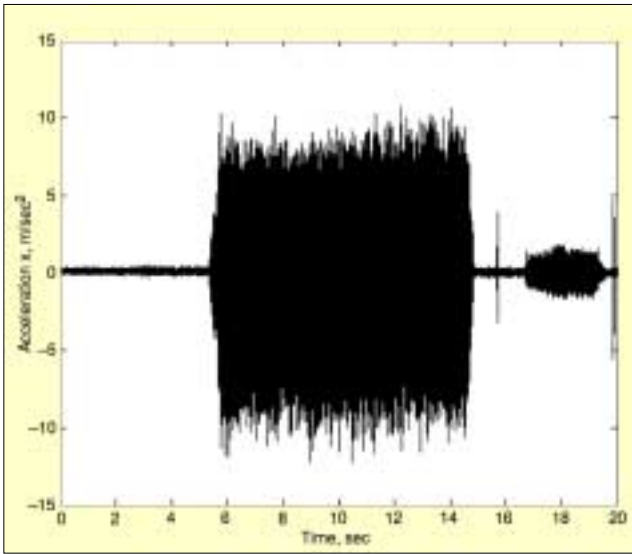
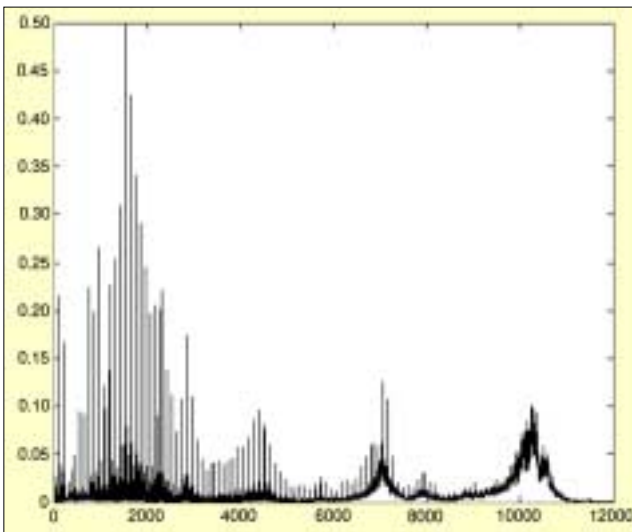Figure 1. Time history of cutting test #14.

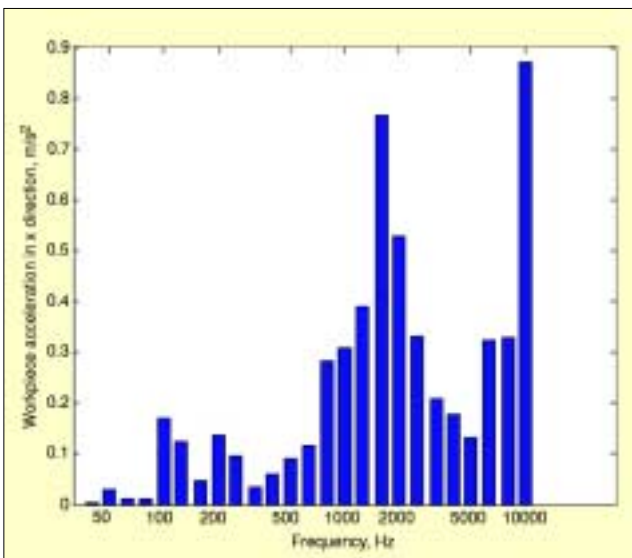Figure 2. Raw plot of a frequency analysis result. Labels and text may be added using standard MATLAB functions.

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

mum value of the running rms is one of the parameters desired. All ISO 2631 filters are implemented in the toolboxes including the running rms calculation. The following lines load a
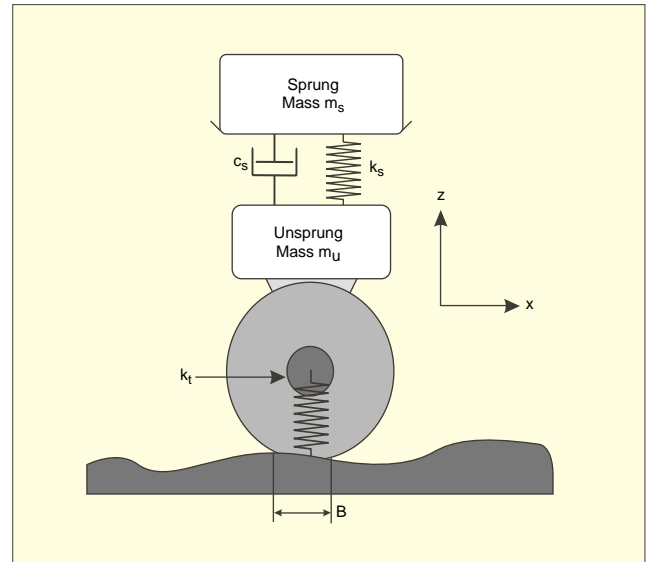

Figure 4. Quarter car model used for IRI calculation.

measured truck seat vibration and calculate the maximum value.

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

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 the 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
- Poles and residues, for example from FEA modeling or experimental modal analysis

A commonly used measure for road roughness is the International Roughness Index (IRI). To obtain IRI, a measured road profile is applied to a simple model, the quarter car (Figure 4). The so-called "golden car" defines the parameters of the model.

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 km/h. To calculate IRI, the system is set up:

```
    M = [ms 0;0 mu];
    C = [cs –cs;–cs cs];
    K = [ks –ks;–ks kt+ks];
```

Then 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.

```
    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]);
```

The function **timeresv** (time response, velocity) calculates the vibration velocity. z*kt is the applied force, fs is the sampling frequency and 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 (all there are in this case). Then the relative velocity is integrated (summed) into total stroke and IRI (in mm/m) is found by dividing by the traveled distance.

```
IRI = 1000*sum(abs(y1-y2))/fs/(length(z)*step);
```

Modal Analysis

In the ModalTools™ toolbox there are many powerful functions for simulation, forced response, modal parameter extraction, animation and more. To give a hint of the level of the functions, the steps from measured FRFs to poles and residues are given. The FRFs are delivered in Universal File Format in
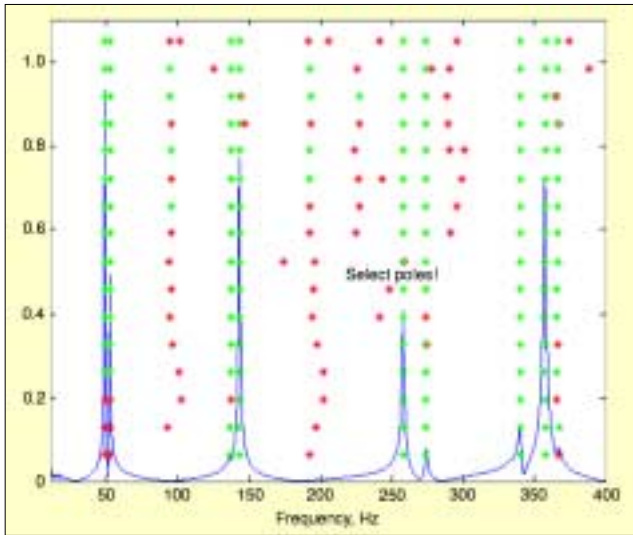
Figure 5. Stability diagram in a complex exponential calculation.

a file T1.unv:

```
univread(1,'T1.unv');           % reads FRFs
[hmat,f] = uf2frf(1,20);        % sets up a FRF matrix and fre-
                                % quency vector
H = cvfrfa2v(hmat,f);           % converts FRF from acceler-
                                % ance to mobility
[H,f] = frftrunc(H,f,0,400);    % truncates FRF matrix to freq.
                                % 0 400 Hz
[h,t,fs] = impresp(H,f);        % calculates impulse response
                                % matrix
poles = complexp(h,fs,250,44,abs(H(:,1))/.9,f,10,400);
                                % computes poles using global
                                % complex exponential routine
[residues,residuals] = pol2resf(H,f,poles,10,400,1);
                                % calculates residues and re-
                                % siduals in frequency domain
```

The function **complexp** delivers a stability diagram, see Figure 5. Thereafter the functions **animcalc** and **animate** may be used to produce animated displays of the modes. A function **avimode** produces a video *.avi of an animated mode for export as shown in Figure 6.

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. An example of a rain flow cycle count analysis is shown in Figure 7. The following lines are used to produce the figure:

```
» 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)
```

Conclusions

In this article we have shown some examples of how MATLAB can be used to analyze data from some vibration applications. The main advantages of using MATLAB are:
1. It forces the user to a certain level of understanding of what the analysis procedure is doing.
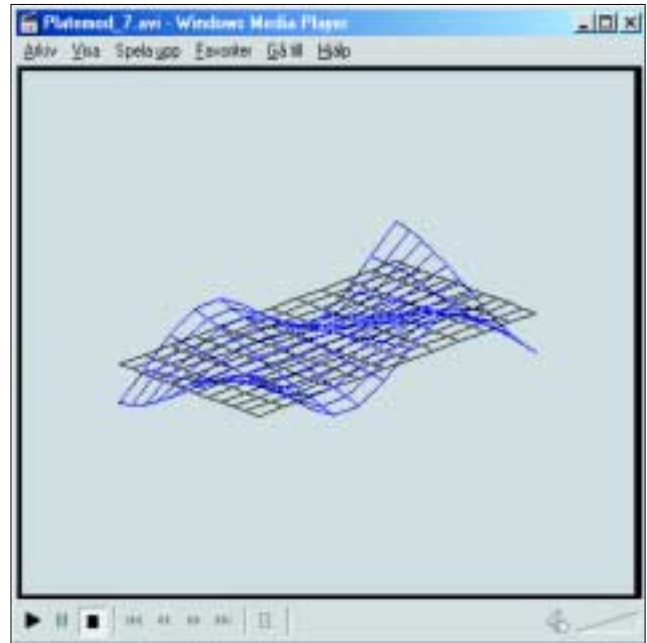2. Traceability is ensured by using the same MATLAB program for each analysis.

*Figure 6. Example of a movie showing an animated mode. The file is generated with the function **avimode**.*
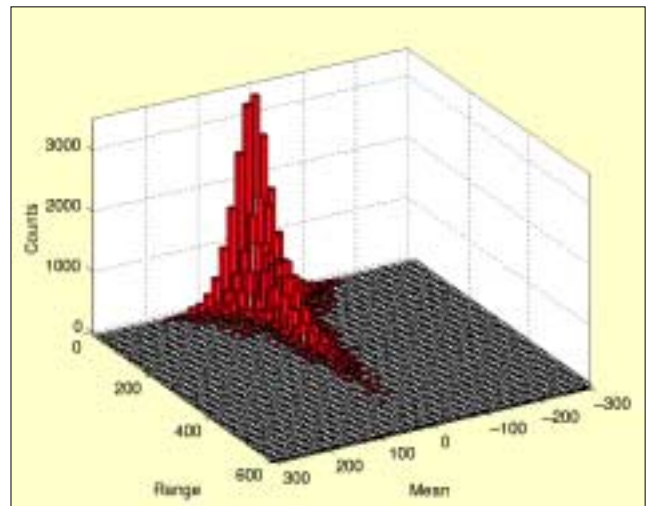

Figure 7. Example of a rain flow cycle count on a test signal.

3. It is very flexible.

The toolboxes now available for noise and vibration analysis allow the less experienced user to take advantage of the power and flexibility of MATLAB, either as a complement to an existing system or as the only tool for analysis. **SV**

The authors may be contacted at: kjell.ahlin@saven.se or anders.brandt@saven.se.

Saven EduTech AB (www.vibratools.com) offers three add-on packages for analyzing vibration data in MATLAB. They are:
- **VibraTools**, for spectrum analysis, statistics and more.
- **ModalTools**, for modal analysis, mechanical simulation and system identification.
- **DuraTools**, for experimental fatigue and durability analysis, and environmental engineering computations.
All three packages require MATLAB 5.1 or higher and the Signal Processing Toolbox from The MathWorks Inc. (www.mathworks.com). Additionally, both ModalTools and DuraTools require VibraTools.