Vibration Analysis of Car Engine

Description

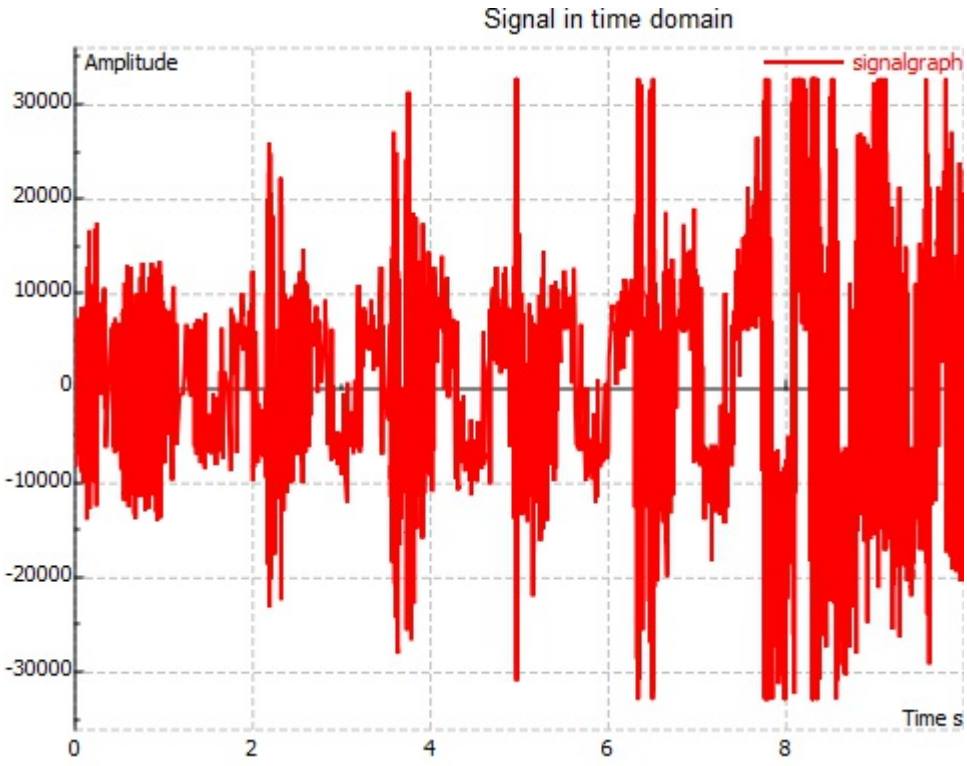
FT, power spectral density -PSD, and spectrogram are common tools for vibration analysis. In the following segment we show how these tools are used in MatDeck to perform vibration analysis on a real world example. The real world example deals with vibration analysis of a car engine which is recorder by a sensor called accelerometer. The data is recorded in a .wav file and the MatDeck functions which are used for reading this type of files are also taken under consideration. After that, we go through the important differences between an FFT, PSD, and spectrogram and we illustrate when it is appropriate to use each type of vibration analysis tool.

Data acquisition

As explained above, we deal with data captured with an actual accelerometer and recorded within the carengine.wav file. In order to reproduce examples the .wav file should be in the same folder as this document. The file carengine.wav contains 10 seconds of a car engine while it was idle with the sampling rate equal to 20kHz. The signal is first analyzed in the time domain by loading it from the file and showing it graphically. The signal is also played and we can hear the sound of the vibrations before we perform deeper analysis.

```
fileID := "carengine.wav"                    File name containing the signal
props := wave properties(fileID)             Properties of the .wav file
signal1 := wave read(fileID)                 Read data from .wav file
Fs := props[0]                               Sampling frequency
audio play(Fs , props[1] , signal1)          Play the content
time := ynodes(t , 0 , 10 , 20000)           Time axis for the signal

signalgraph := join mat cols(time , subset(signal1 , 0 , 0 , 19999 , 0))   Signal graph
```



FFT analysis

Any signal is actually the sum of sinusoids of different frequencies, amplitudes, and phases. We use Fourier analysis or spectrum analysis to deconstruct a signal into its individual sine wave components. The result of FFT analysis is the acceleration/vibration amplitude as a function of frequency, which lets us perform analysis in the *frequency domain* (or spectrum) to gain a deeper understanding of our vibration profile. Most vibration analysis are typically done in the frequency domain. The number of discrete frequencies that are calculated using fft() as part of a Fourier transform is directly proportional to the number of samples in the original signal, However the FFT analysis can be calculated for a smaller number of frequency points as well. One has to make the trade-of between spectrum precision and number of operations. If N is the length of the calculated spectrum, the frequency samples are given in intervals Fs/N from 0Hz up to (N-1)Fs/N. Real world signals have conjugate symmetric spectrum, which means that the amplitude spectrum is symmetric and phase spectrum is anti-symmetric around the origin. Therefore, it is very common to show the amplitude spectrum from 0Hz up to the half of the sampling rate.

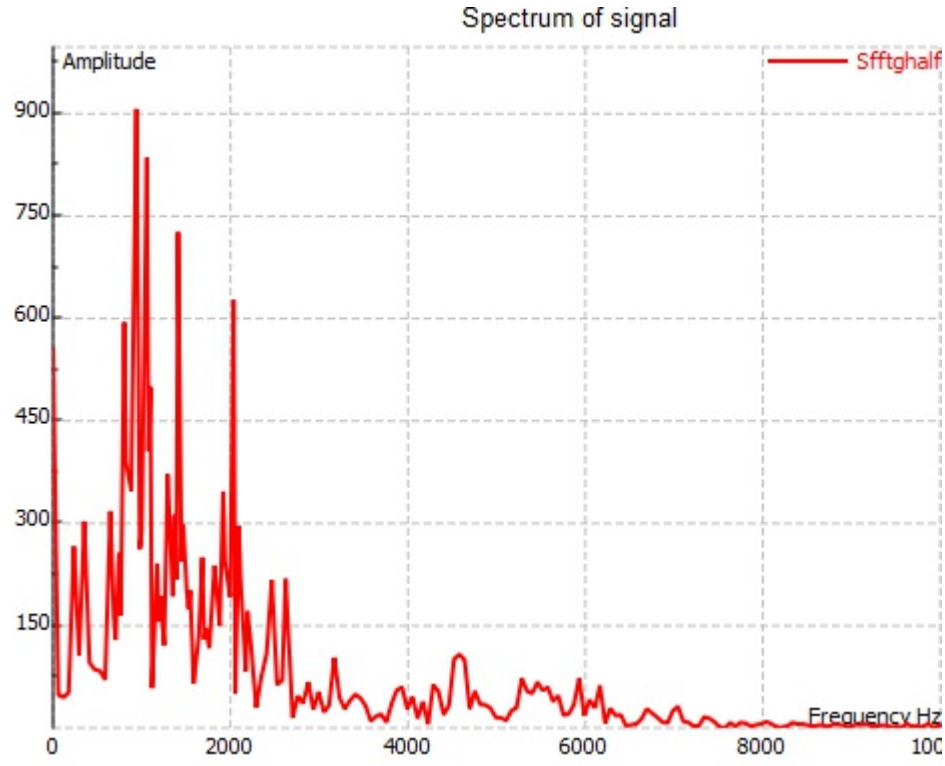
$N1 := 1024$ Number of frequency points for FFT

$fre := ynodes(f, 0, N1 \cdot Fs/(N1-1), N1)$ Frequency axis

$Sfft := fft1n(signal1, N1)$ Calculation of fft

$Sfftgraph := join\ mat\ cols(fre, |Sfft|/N1)$ Grpah of the frequency spectrum

$Sfftghalf := subset(Sfftgraph, 0, 0, N1/2, 1)$



From the frequency spectrum of the signal we can see the dominant frequency components of the signal. However we can not see the time domain position when certain frequency components occur.

Power spectral density-PSD

We can determine the dominant frequency components by using PSD analysis. However, one again time domain information is lost. Welch's method for estimating the power spectrum is carried out by

dividing the time signal into successive blocks, forming the periodogram for each block and then averaging them. MatDeck contains a function powspectwelch() for this purpose. MatDeck function powspectwelch() has the following arguments: input signal as a vector for which the power spectral density is estimated, window function determined by string name of the window used and number of samples to generate the result which is also number of samples to perform fft() withing each periodogram function. The last two arguments are: the block length in samples, and number of overlapping samples between consecutive blocks. These two arguments define the overall number of periodograms which are averaged.

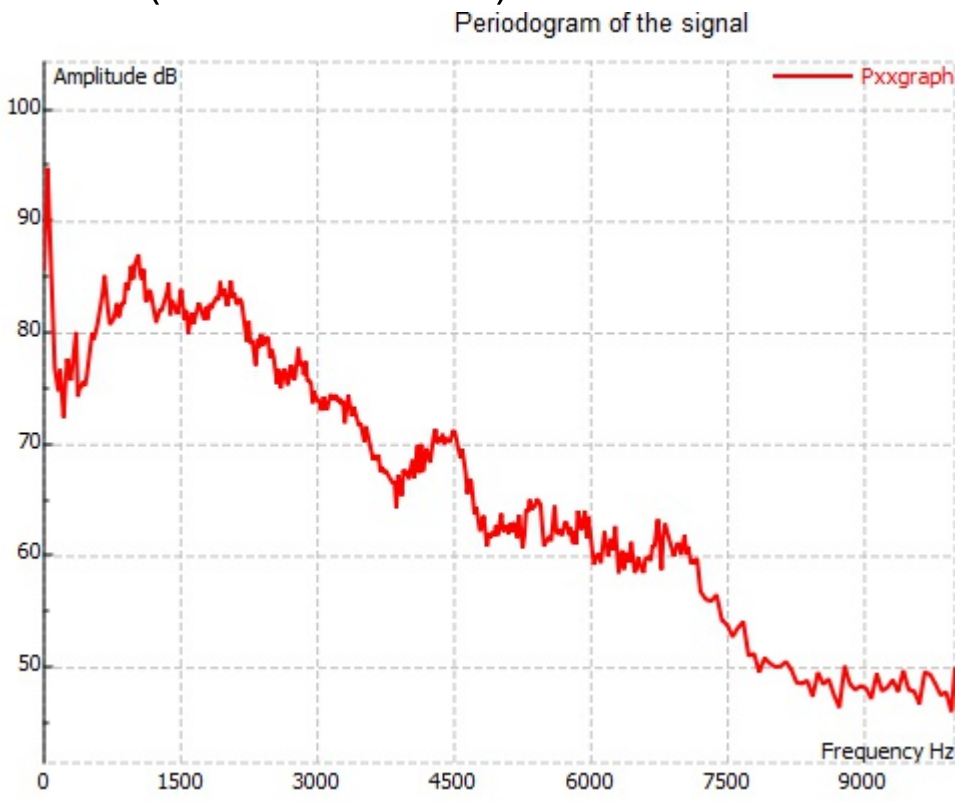
block := 10000 Block length

nover := 5000 Overallping samples between segments

Pxx := powspectwelch(signal1 , "rectangular" , N1 , block , nover) Welch's method, rectangular window

ff := ynodes(f1 , 0 , 1 - 1 / N1 , N1 / 2 + 1) Frequency axis

Pxxgraph := join mat cols(ff · Fs / 2 , 10 · log(Pxx))



We can use spectrum analysis of the vibration profile displayed above to determine what the engine's crank shaft rotation speed was. The experiment has been performed on a 4-cylinder 4-cycle engine. The engine operates with two pairs of pistons moving out of phase with each other and there are two piston combustions per crank shaft rotation. This means the dominant frequency of the engine's vibration is twice the crank shaft rotation speed. Since the dominant frequency is 30Hz we conclude that rotation speed is 15Hz which makes 900 rotations per minute however, we do not know the exact timing of the rotation.

Spectrogram

In this example, and whenever vibration frequency changes with time, we need a spectrogram. A spectrogram works by breaking the time domain data into a series of chunks and taking the periodogram of these time periods. These series of periodograms are then overlapped to visualize

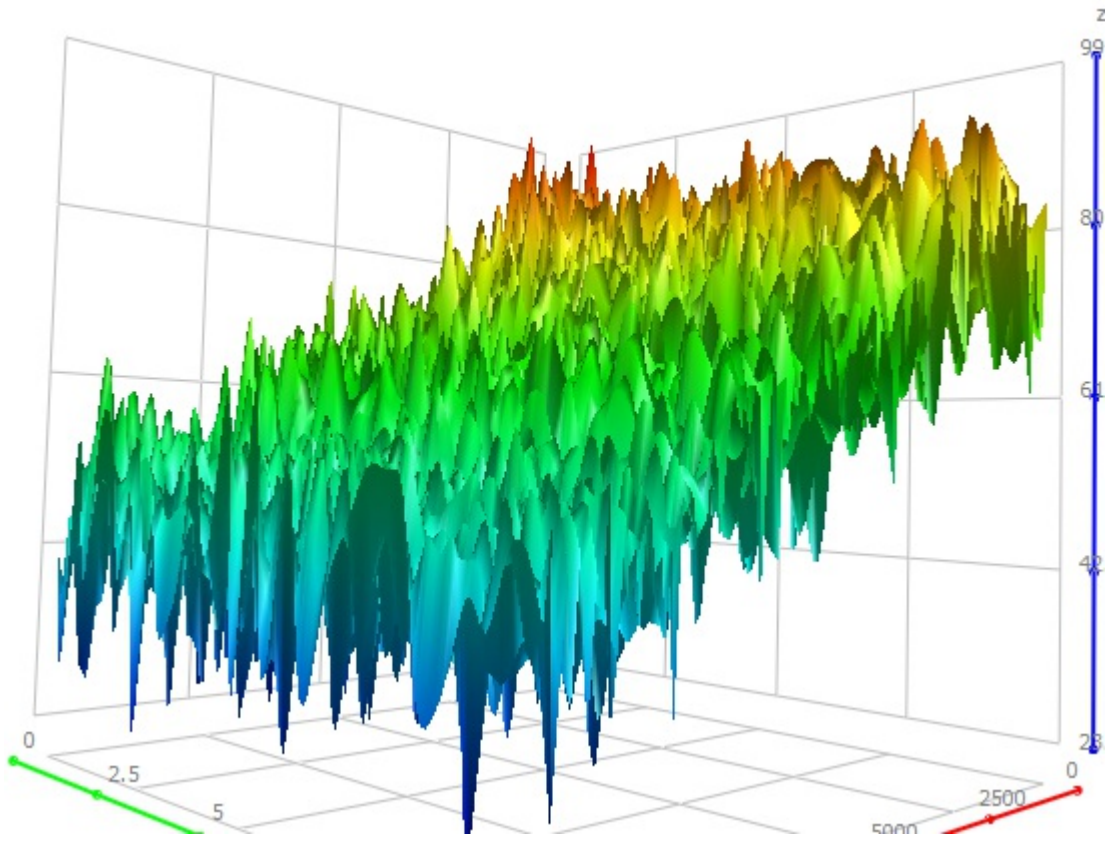
how both the amplitude and frequency of the vibration signal changes with time. We need three dimensional graph to show the spectrogram versus time and frequency. MatDeck contains a function spectrogram() which shows the results in a 3D graph. MatDeck function spectrogram() has the following arguments: input signal as a vector for which the power spectral density is estimated, window function determined by string name of the window used and number of samples to generate the result which is also number of samples to perform fft() withing each block in time. The last two arguments are: the block length in samples, and number of overlapping samples between consecutive blocks. These two arguments define the overall number of points in the time taken to calculate the spectrum.

```
Spect := spectrogram(signal1 , "rectangular" , N1 , block , nover)   Spectrogram, rectangular window is used
Sp := data3d(10 · log(Spect) , xx , 0 , Fs/2 , y , 0 , 10)           Prepare data for 3D graph
```

The 3D graph is shown next.

```
gr1 := graph3d(0 , Sp)

set size(gr1 , 550 , 450)
```



In this example the engine was revved for a short while during the experiment. The spectrogram shown above illustrates how the dominate frequencies change with time in relation to when the car engine was idled and revved. Using a spectrogram, we can get a deeper analysis of the vibration profile and how it changes with time.