Vibration Diagnostics Signal Analysis Using GNU Octave to Calculate

Howard A. Gaberson, Consultant, Oxnard, California

Octave is a licensed free signal processing package that can do basic machinery diagnostics FFT-based operations. There is also a companion front end called GUI Octave that makes the computer screen look familiar. This article will give those interested a big start in doing digital signal processing on their own. I've always felt less than helpful giving presentations on how to do a signal analysis operation, knowing that only if the user has a $2000 program can he use the material. This changes the game entirely. You download Octave and GUI Octave, and with those programs you have a powerful package. This article shows you how to use it. Once you get started doing these things, you gain considerable understanding of data analysis.

I have been fascinated with digitized shock and vibration signals since 1969. It's just extremely interesting, and makes a lot of the seemingly useless math come alive and solves problems. I've been using the expensive digital signal arithmetic program, MATLAB®, since the early '90s. I had learned a little Fortran in graduate school and started using it to calculate shock spectra in the early '70s. Now I understand certain aspects of the FFT very well. Fourier series went in one ear and out the other all through graduate school, and maybe I began to figure it out in a plates and shells course where, thank heaven, they taught me to read Timoshenko's "Plates and Shells." Trying to keep up with the Joneses, I had to learn FFT in the early '70s, but there wasn't anything in it for shock work at that time. Later, in the late '80s, I was tasked to dig into diagnostic machinery vibrations, and the FFT became indispensable. What I can do now in recent years is try and show what interested me and give some ideas for people to try. I'm sure the calculation possibilities will continue to offer useful opportunities.

I think the obstacle of the high cost of MATLAB for routine signal arithmetic may prevent many potential innovators from learning the ideas, and they are not very difficult. The free program Octave may be enough to do it, so here I want to show some basic machinery diagnostic calculations with MATLAB[1] and Octave[2] so you can try them out. Maybe I can attract a subgroup to develop this further, and then we can make it easier for all to learn. Knowing how to do the calculation yourself develops an intimate relationship with the signal arithmetic. You can call it math, but I want you to see that we're talking about just a few concepts: add, subtract, multiply, divide, and raise to powers; on long lists of numbers, while keeping track of where we are. So let's start doing some simple vibration diagnostic digital signal analysis.

I think what you have to do is to learn how to calculate something, go to the program that I used to calculate it and then convince yourself that I am doing it right. A potentially difficult job is to download Octave and GUI Octave and get them to work together.

So here I'm going to pretend we have a means to get a copy of the digitized data, and I'm going to try and explain how you can do the signal analyzer functions on your laptop with Octave, a GNU-licensed signal analysis program. I'm hoping this is understandable enough so that you can try out everything I discuss.

Vibration monitoring is the most widely used machinery diagnostics technology. Machinery problems usually occur within the rotating members. Dynamic forces from the rotating members communicate to the machine foundation through the bearings. Thus bearing housing vibrations are logically a good source of diagnostic vibrations.

Most vibration-based machinery condition monitoring means a frequency or spectrum analysis of bearing housing vibration to permit evaluation of changes. Machinery vibration spectra are an FFT (fast Fourier transform) that show that the vibrations are composed of frequencies that are typically multiples of the shaft rotational speed (RPM.) A simplified explanation of the analysis might be that when a trend of abnormal growth of constituent frequencies is observed, a machinery problem exists. The pattern of frequency growth usually indicates the type of problem; the level of the signals more or less indicates the time remaining before failure. Problems with unbalance, alignment, looseness, drive belts, bent shafts, and rubbing have known documented spectral patterns. Ball bearings have four known characteristic frequencies. If there are gears, there is a tooth mesh frequency, and with vanes there is a vane-passing frequency. Sidebands appear at multiples of the shaft rotational speed on either side of gear mesh and bearing fault frequencies. Harmonic peaks appear at multiples of shaft speed and bearing frequencies.

Many excellent references to the practical technology are available and listed in the Vibration Institute Publications Catalog at www.vibinst.org.

Based on a paper presented at the Vibration Institute 36th annual Meeting, Orlando, FL, 2012.

Digitizing and Aliasing

To process vibration signals in a computer, they must be digitized. Just prior to digitizing, the analog transducer signal must be analog filtered to avoid aliasing. Shannon's sampling theorem[3] states that a signal is adequately sampled if it is sampled at a rate at least twice the highest frequency present in the signal. By frequency content he means that the Fourier transform is zero beyond the maximum frequency present, and this condition is essential in his proof. A signal that is adequately sampled is completely defined for all points even those between the samples; one and only one function can pass through those points and be band limited. A signal digitized without analog antialiasing has been essentially destroyed. Any little spike or spurious high-frequency squiggle might be selected by the digitizer as a value and totally corrupt the digitized signal. Insist on antialiasing.

Loading Data

MATLAB expects and accepts numerical data in ASCII format to be a matrix of numbers. Equal numbers of rows and columns. Thus a column of 1000 numbers is fine. If the file name has any extension other than "*.mat," MATLAB expects it to be a table or matrix consisting of an equal number of rows and columns.

The "%" causes everything on the remainder of that line to be ignored. So if your data file has a lot of important info in the header section, you can leave all that in by preceding or beginning each line by a "%." Now I'll show a portion of a diary of the computer screen where I have loaded a file called pnscrng.ch2 into MATLAB and inspected the data.

```
load pnscrng.ch2
who
 Name    Size     Bytes  Class
 pnscrng 10241x2 163856 double array
Grand total is 20482 elements using 163856 bytes
xxdd=pnscrng(:,2);
tt=pnscrng(:,1);
%pnscrng.ch2 is a file I recorded on 940502
%File had 2 large columns, the first is probably time
fs=10240/(tt(10241)-tt(1))
fs = 5120
%This is typical and is probably correct.
%Column 2 is acceleration.
```

This loaded the file. The 'who' command showed what file was loaded.

```
14 plot(tt,xxdd),grid
15 xlabel('Time, seconds')
16 ylabel('Acceleration, g')
17 title('Pnscrng Acceleration 5 May 1994')
18 axis([.58 .61 -.08 .08])
19 axis([.585 .59 -.08 .08])
```

On line 4, the "who" command shows I loaded pnscrng, which has 10,241 rows and 2 columns of data values On line 6, I named the acceleration xxdd and guessed that it's every row and the second column. In this case, the colon means every row. Similarly in line 7, I name the time tt and set it equal to every row and the first column of pnscrng. On line 10, I calculate the sampling rate. There are 10240 spaces between the first sample and the last sample. I take the number of spaces and divide it by the time between the first sample and the last sample. On line 14, I plot the data with a grid as shown in Figure 1. Lines 15, 16, and 17 add axis labels and a title. Line 18 expands the time axis for better detail as shown in Figure 2. Line 19 expands the time axis even further to show the coarseness of sampled data.

Figure 3 is a blow-up of a portion of the pnscrng time history. It is amazing to see the coarseness of the digitizing sampled data that has been anti-aliased to 2.56 times the highest frequency present. However this is enough for the fft to be able to evaluate the magnitudes and phase of the spectrum.

Peak, RMS, Overall, Crest Factor, Kurtosis

The vibration signal itself can be evaluated by statistical measures such as its RMS level (root mean square, or square root of the mean of the squared values.) This overall measure is the basis of many vibration shutdown switches or alarm lights. The value of RMS velocity over the frequency range of 10 to 1,000 Hz is used as a measure of vibration severity in ISO Standards 2372 and 3945. The peak value of a pure sine wave is 1.414 times its RMS level, and this factor is used too often to convert between RMS and peak levels. The crest value, which is the ratio of the absolute peak to RMS level, is frequently reported as indicative of impacting in the signal. Kurtosis[4] is given by Equation 1:

$$k = \left[\frac{1}{N} \sum_{i=1}^{N} \left(\frac{(x_i - \bar{x})}{\sigma} \right)^4 \right] \tag{1}$$

Here the standard deviation or RMS value is given by Equation 2:

$$\sigma^2 = \frac{1}{(N-1)} \sum_{i=1}^{N} \left(x_i - \bar{x} \right)^2 \tag{2}$$

Sometimes a "–3" is added to Equation 1 to make the value zero for a normal distribution. Both crest factor and kurtosis measure the "peakiness" of a signal.[5] A very nice study on the use of these measures is given by Pachaud, *et al.*[6] Now lets take a look at a short program to calculate the RMS value, the crest factor, and the kurtosis of a data file called nsa1300.

```
%kurtosis.m Do kurtosis anal on file nsa1300
load nsa1300;
x=g21300;
xx=x(1:2048);
N=length(xx);
xx=xx(:)';%assure x is a row
xx=xx-mean(xx);
sigma=sqrt(1/(N-1)*(xx)*(xx)');
kurt=(1/N*sum((xx/sigma).^4))
rms=sqrt((xx*xx')/N)
crest=max(abs(xx))/rms
```

Now let's examine what the program is doing line by line

```
load nsa1300;
x=g21300;
```

Loads file nsa1300.mat (with no extension, *.mat is assumed). File contains list of numbers called g21300, set x equal to that list

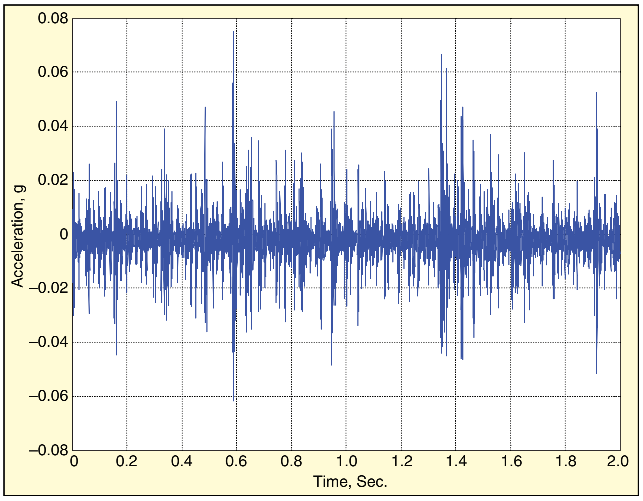

Figure 1. First plot of data file pnscrng.ch2, as received.

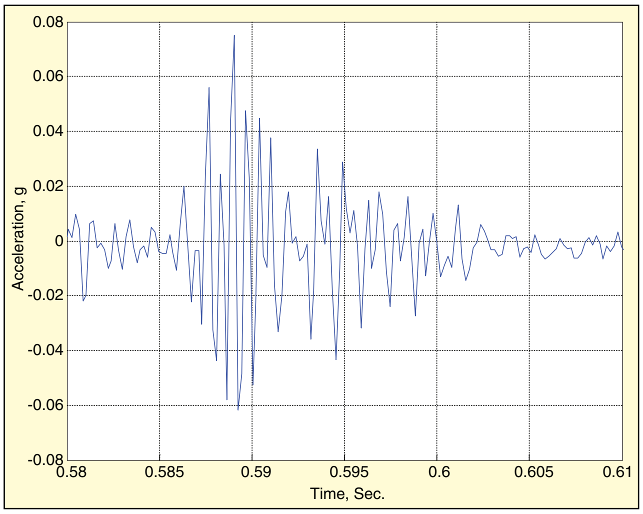

Figure 2. Plot of file pnscrng.ch2 with time expanded as shown.

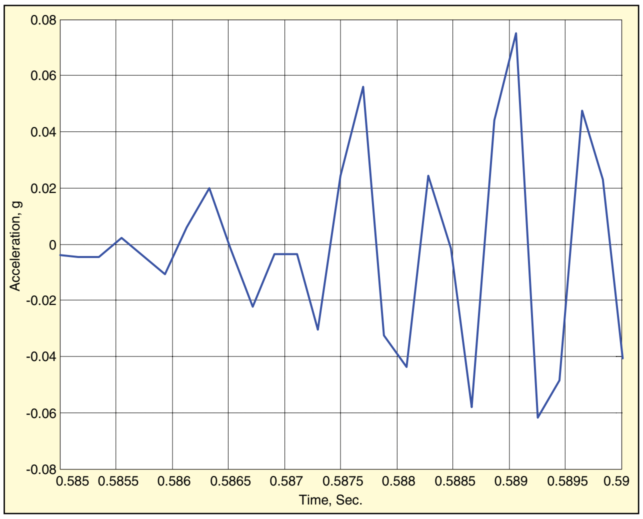

Figure 3. Plot of file pnscrng.ch2 with time further expanded to show coarseness of sampling at 2.56 times the highest frequency present.

```
xx=x(1:2048);
```

Make xx a row of the first 2048 numbers in x

```
N=length(xx);
```

Make N equal the number of values in xx

```
xx=xx(:)';%assure x is a row
```

xx=xx(:) makes xx a column; the ' transposes it to a row

```
sigma=sqrt(1/(N-1)*(xx)*(xx)');
```

xx*xx' the inner product of the row times a column is the sum of the squares. Divide that sum ny (N-1) and take the square root. That's the number sigma.

```
kurt=(1/N*sum((xx/sigma).^4))
```

Kurtosis is the sum of all the xx's divided by sigma and raised to the fourth power; then the sum is divided by N.

```
rms=sqrt((xx*xx')/N)
```

RMS value is square root of mean of sum of squares of xx.

```
crest=max(abs(xx))/rms
```

Crest factor is max absolute value of xx divided by RMS.

Spectrums

A spectrum of a vibration signal is a graph that indicates content as a function of frequency. Generally spectra are formed from averaged windowed discrete Fourier transform magnitudes of vibration time histories. They are normally computed in a signal analyzer using a fast Fourier transform (FFT) computer algorithm on a digitized (equally spaced samples, sampled at a sampling rate, f_s segment of the vibration time signal. As a result of the FFT, the transform is a complex function of frequency; complex because each frequency component has a magnitude and phase.

It's convenient to use complex numbers for a quantity that has amplitude and phase. Normally the magnitude is plotted as a function of frequency as the spectrum; phase is also computed and can be plotted when needed. The FFT is actually a Fourier series of a segment of a vibration signal. This is good because the FFT will indicate frequency content as the actual sine wave components that make up the signal. Add them all up and you have the original signal. But the FFT considers everything in the segment periodic. This causes errors in evaluating components that do not have an integral number of periods in the segment.

Also assuming the segment periodic implies a discontinuity, where the assumed periodic segments join, end to beginning. The errors are called leakage errors. To diminish these errors, the time segments are "windowed" with bell-shaped functions (Hanning, Kaiser-Bessel, flattop or other.)[7,8] This means that the segment values are term by term multiplied by the window function, which goes to zero at its ends. This in turn reduces the signal amplitude and must be compensated for by multiplying the spectra values by a window factor. This also effectively wastes part of the data, because the ends are attenuated (see Figure 4).

Figure 5 shows how we put this into practice. On the top line is a segment of a signal to be analyzed. The five lines below it show five segments of signal, 50% overlapped. To the right, approximately over time = 0.05, are the same five segments windowed with or multiplied by a Hanning window. These five windowed segments are ready for FFT and averaging. The rightmost column is spectrums of the individual windowed data segments. The number of values in the digitized time segment to be transformed is equal to a power of two, which gives a huge computational efficiency. Typically, 1,024 samples are used to yield 400 frequency values or lines. The bandwidth is the frequency range over which the data are analyzed.

From the sampling theorem you would expect values all the way to $f_s/2$, and indeed these are computed. But since the analog antialiasing filter can't abruptly stop frequency content at half the sampling rate, the higher frequency values will be inaccurate. The number of frequency lines (values, bins) resulting from the FFT is related to the number of time history values by 2.56. The 400-line spectrum requires 1024 points, 800 lines require 2048, etc. The sampling rate must also be 2.56 times the highest frequency to be accurately analyzed. A spectrum with a 1,000-Hz upper frequency was obtained from a time history sampled at 2,560 samples per second. In general, many spectra are taken and averaged together to yield the spectrum to be stored or analyzed.

The y-axis or ordinate of the spectrum is the amplitude, and frequency is plotted on the x-axis. The ordinate or y axis can be plotted linearly, logarithmically, or in decibel notation. Decibels are a ratio of the log of the amplitude to the log of a typically very small reference amplitude. Logarithmic plotting makes the low amplitude values visible. Sometimes the frequency is plotted logarithmically, and advocates of this refer to it as a constant percentage bandwidth presentation.

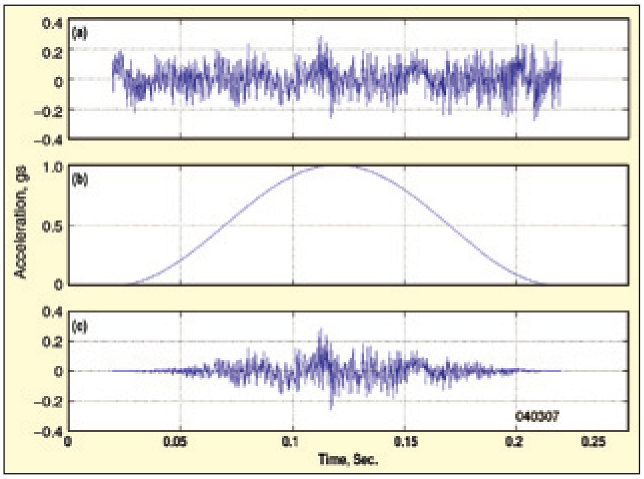

Figure 4. Application of Hanning window to a segment of data; (a) segment of data; (b) plot of the Hanning window; both window and signal have same number of values. For each time instant; (c) signal value is multiplied by the window value.

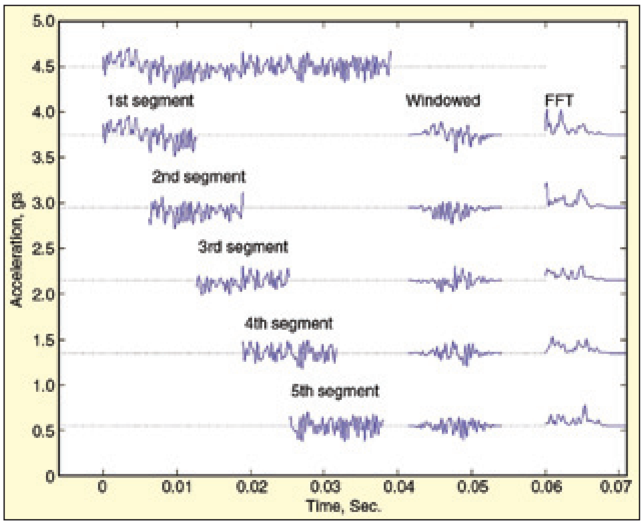

Figure 5. Spectrum development is shown by taking segments, windowing them and then applying FFT. We generally average the third column of spectra, but we can sometimes cascade them and make a waterfall plot. And sometime we calculate hundreds of them and plot them almost touching each other for a mesh or 3D plot. One other format I like is the contour plot, where instead of showing the 3D mesh, we show a top-down picture with contours indicating the various levels or elevations.

Inner Product and Transpose

The inner product is a helpful concept for understanding vibration signal processing, and it's an easy calculation for a computer using high-level signal analysis software. Your data collectors and signal analyzers are doing it all the time. We can only analyze a digitized version of our vibration signal (which will be a long list of numerical values) because we have digital computers. A very frequent calculation we perform on our data is to form an inner product of it with an analyzing function. This is really a term-by-term multiplication and then an adding of those products. The matrix multiplication rule for a row times a column is as follows. Let's define x and y as in Equation 3:

$$x = \begin{bmatrix}1 & 2 & 3\end{bmatrix}, \text{ and } y = \begin{bmatrix}4 \\ 5 \\ 6\end{bmatrix} \tag{3}$$

The inner product of x and y, or x times y, is (and the usual inner product symbol is these corner brackets) given by Equation 3a:

$$\langle x * y \rangle = 1\times 4 + 2\times 5 + 3\times 6 = 4 + 10 + 18 = 32 \tag{3a}$$

It's a term-by-term multiply and add of the two equal element lists of numbers. We call each list of numbers a vector. Our vibration signal could be a 40,000-point list of digitized acceleration values. We will analyze it with another 40,000-point analyzing list of numbers (an analyzing function) by forming an inner product of our signal with the analyzing function. MATLAB and Octave will do an inner product if you say *x*y,* and *x* is a row and *y* is a column and they both have the same number of elements.

Transpose is a manipulation we need. The transpose of a row is a column, and the transpose of a column is a row. Transpose is denoted in MATLAB and Octave with a prime or apostrophe as shown on Equation 4:

$$\text{If } x = \begin{bmatrix}1 & 2 & 3\end{bmatrix}, \text{ then } x' = \begin{bmatrix}1\\2\\3\end{bmatrix} \tag{4}$$

Although we won't use matrices, which are rectangular arrays of numbers, transposes are used here as well. If we had a 3 x 3 matrix called r:

$$r = \begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}, \text{ its transpose is } r' = \begin{bmatrix}1 & 4 & 7\\2 & 5 & 8\\3 & 6 & 9\end{bmatrix} \tag{4a}$$

Each row becomes a column. The matrix is flipped about its upper left lower right diagonal. However its best use for us is to make a row a column and vice versa. We denote the transpose of *x* as *x'*. If *x* is a row, *x'* is a column (and vice versa). If *x* is a row of numbers, and you give MATLAB and Octave the instruction *x*x'*, which is the inner product of *x* and *x'*, it will give you back the sum of the squares of *x*. There is a time associated with each of those *x* values if they are digitized vibration signal values; lets call *t* the list of times corresponding to each of those *x* values. Again *t* could be a row or a column; it's up to us to say which. If *t* was a column, *t'* would be a row. For example, the FFT can be expressed as Equation 5:

$$X_k = \sum_{i=1}^{N} x_n e^{\frac{-i2\pi(k-1)(n-1)}{N}}, \quad 1 \le k \le N \tag{5}$$

If *n* is a column of numbers from 1-1024, then $e^{-i2\pi kn/N}$ is also a column, because it has a value for each value of *n*. Each value of the FFT can be viewed as an inner product of, for example, 1024 values of our signal as a row, with 1024 values of $e^{-i2\pi kn/\mathrm{N}}$ as a column. In the FFT, we accomplish one of these inner products for each bin or frequency. It's actually even worse than this, because we do it several times to get many averages.

Relation of DFT Xs to a Spectrum, Harmonic Content

Symbols:

k	frequency index
n	time index
f_s	sampling rate
N	number of sample in the shock
x_n	data value
X_k	DFT value
a_k, b_k	Fourier series coefficients

These DFT values are the amplitudes and phases of the sine waves that when added together build the exact shock.

You can configure (arrange) FFT results to be a Fourier series analysis of the data segment considered periodic. There is a duality between time and frequency of signals; we can precisely specify a signal segment in terms of either time or frequency. As an example, a 1024 value list of a digitized vibration signal can exactly be expressed as a 1024 list of frequency values. The two lists say exactly the same thing with the same precision. The DFT or FFT is the calculating procedure we use to transform one to the other. This will be a spectrum of the sine waves composition of the segment. The DFT so configured actually tells you the sine wave composition of the data segment analyzed. To get started, I've copied the pertinent portions of the 'help fft' response from my MATLAB R12 below.

```
 ">> help fft

  FFT Discrete Fourier transform.
  FFT(X) is the discrete Fourier transform (DFT)
of vector X....
   ...For length N input vector x, the DFT is a
length N vector X,
   with elements
         N
   X(k) = sum x(n)*exp(-j*2*pi*(k-1)*(n-1)/N), 1 <=
k <= N.  (6)    n=1

  The inverse DFT (computed by IFFT) is given by

                   N
   x(n) = (1/N) sum X(k)*exp( j*2*pi*(k-1)*(n-1)/N),
1 <= n <= N.     (7) k=1


   The relationship between the DFT and the Fourier
coefficients a and b in


                     N/2
   x(n)  =  a0  +  sum  a(k)*cos(2*pi*k*t(n)/
(N*dt))+b(k)*sin(2*pi*k*t(n)/(N*dt))   (8)
                        k=1
   is


   a0 = X(1)/N, a(k) = 2*real(X(k+1))/N, b(k) =
-2*imag(X(k+1))/N,(9a,9b,9c)


  where x is a length N discrete signal sampled at
times t with spacing dt. ..."
```

Notice, since the sum in Eq. 8 is from one to *N*/2, the above is exact if the file length is even; throw away one point if necessary. I have never run into anyone using *N* odd, and I try to not FFT an odd length segment.

In their first equation they are saying the FFT exactly calculates a set of *N* X_k's from the *N* valued data set x_n according to Equation 10a:

$$X_k = \sum_{n=1}^{N} x_n e^{\frac{-i2\pi(k-1)(n-1)}{N}}, \quad 1 \le k \le N \tag{10a}$$

Their Equation 7 states that the data can be exactly recovered from the X_k according to Equation 10b, the discrete Fourier synthesis:

$$X_n = \frac{1}{N}\sum_{k=1}^{N} X_k e^{\frac{i2\pi(n-1)(k-1)}{N}}, \quad 1 \le n \le N \tag{10b}$$

And they further say that if you want the set of (*N*/2 + 1) Fourier coefficients a_k, and b_k, for that data segment considered periodic, they are given by Equations 11a, 11b and 11c:

$$a_0 = \frac{X_1}{N}, \quad a_k = \frac{2}{N}\mathrm{Re}(X_{k+1}), \quad b_k = -\frac{2}{N}\mathrm{Im}(X_{k+1}) \tag{11a, b, c}$$

Where the a_k, and b_k, are defined in Equation 11d:

$$x(n) = \frac{a_0}{2} + \sum_{k=1}^{N/2}\left(a_k \cos\frac{2\pi k t_n}{Ndt} + b_k \sin\frac{2\pi k t_n}{Ndt}\right) \tag{11d}$$

This *t*(*n*)/*dt* is clumsy and is equal to the time index, *n*, so write this as Equation 11e:

$$x(n) = \frac{a_0}{2} + \sum_{k=1}^{N/2}\left(a_k \cos\frac{2\pi kn}{N} + b_k \sin\frac{2\pi kn}{N}\right) \tag{11e}$$

The frequency of each X_k, a_k, and b_k is kf_s/N. To show this, remember a single frequency constant vibration is a sine (or cosine) wave that we write as Equation 12a:

$$x = A\sin(2\pi f t) \tag{12a}$$

The quantity in parenthesis in Equation 12a is called the argument of the sine. The quantity in the argument between the 2π and the *t* is the frequency in cycles per unit time. We want the frequency associated with the arguments in terms such as the complex exponential relation to sines and cosines in Equation 12b. (See Ref. 4, p474, Circular Functions in Terms of Exponentials.)

$$e^{-\frac{i2\pi kn}{N}} = \cos\frac{2\pi kn}{N} - i\sin\frac{2\pi kn}{N} \tag{12b}$$

Multiply the quantity $2\pi kn/N$, of Eq. 12b, by hf_s (where h is the time per sample), which equals 1, and group the terms as in Equation 12c:

$$\frac{2\pi kn}{N} hf_s = 2\pi \frac{kf_s}{N} nh \tag{12c}$$

Compare this with $\cos 2\pi\phi$ ft. In digitized terms, nh corresponds to the time, and kf_s/N, corresponds to the frequency. So a sine wave with an argument $2\pi kn/N$, has a frequency of kf_s/N.

Now I must relate X_k of the DFT to content or the spectrum. The content is the magnitude of the Fourier series coefficients. We say this periodic segment, $x(n)$, can be exactly built from its Fourier series coefficients a_k and b_k , as stated in Eq. 11d, which says the segment is composed of a DC term and $N/2$ sine waves. That's all it takes to go back to the data you started with. Each value of k specifies one sine wave. We can write the kth harmonic sine wave as Equation 13:

$$x_k(n) = a_k \cos\frac{2\pi kn}{N} + b_k \sin\frac{2\pi kn}{N} \tag{13}$$

Its frequency is kf_s/N. We can also write the kth harmonic in terms of amplitude and phase as in Equation 13a:

$$x_k(n) = A_k \cos\left(\frac{2\pi kn}{N} - \phi_k\right) \tag{13a}$$

Here A_k is the amplitude, our content, and ϕ_k is the phase. The phase is the angle in radians to the first positive peak. We can get the relation of the amplitude and phase to the a_k and b_k by using the formula for the cosine of the difference between two angles, which gives us Equation 14a:

$$\begin{gathered} x_k(n) = A_k \cos\left(\frac{2\pi kn}{N} - \phi_k\right) = \\ A_k\left[\cos\phi_k \cos\frac{2\pi kn}{N} + \sin\phi_k \sin\frac{2\pi kn}{N}\right] \end{gathered} \tag{14a}$$

Comparing the coefficients of cosine and the sine in the right side of Eq. 13 and the second right side of Eq. 14a, we see that Equations 14b and 14c must be true.

$$a_k = A_k \cos\phi_k, \text{ and } \mathrm{b}_k = A_k \sin\phi_k \tag{14b, c}$$

By squaring Equations 14b and 14c and adding we see that the amplitude is given by Equation 14d:

$$\begin{gathered} a_k^2 + b_k^2 = A_k^2\cos^2\phi_k + A_k^2\sin^2\phi = A_k^2\left(\cos^2\phi_k + \sin^2\phi_k\right) = A_k^2 \\ A_k = \sqrt{a_k^2 + b_k^2} \end{gathered} \tag{14d}$$

Dividing Equation 14c by 14b, we obtain the phase as Equation 14e:

$$\phi_k = \tan^{-1}\left(\frac{b_k}{a_k}\right) \tag{14e}$$

Using the values of a and b from Eq: 11b and 11c, we get Equation 14f:

$$\phi_k = \tan^{-1}\left(\frac{b_k}{a_k}\right) = \tan^{-1}\left(\frac{-\frac{2}{N}\mathrm{Re}\left(X_{k+1}\right)}{\frac{2}{N}\mathrm{Im}\left(X_{k+1}\right)}\right) \tag{14f}$$

After simplifying, Equation 14f becomes Equation 14g:

$$\phi_k = \tan^{-1}\left(-\frac{\mathrm{Re}\left(X_{k+1}\right)}{\mathrm{Im}\left(X_{k+1}\right)}\right) \tag{14g}$$

We evaluate A_k of Eq. 14d by substituting Eqs. 11b and 11c into the radical in Eq. 14d in Equation 15a:

$$\begin{gathered} A_k = \sqrt{\left(\frac{2}{N}\mathrm{Re}\,X_{k+1}\right)^2 + \left(\frac{2}{N}\mathrm{Im}\,X_{k+1}\right)^2} = \\ \frac{2}{N}\sqrt{\left(\mathrm{Re}\,X_{k+1}\right)^2 + \left(\mathrm{Im}\,X_{k+1}\right)^2} \end{gathered} \tag{15a}$$

After simplifying, this becomes Equation 15b:

$$A_k = \frac{2}{N}\left|X_{k+1}\right| \tag{15b}$$

Calculate the Spectrum

Maybe this section should be called MOPFD for "Matlab/Octave Programming for Dummies." The trouble with the dummies books is that they're often not written by dummies. I won't apologize for my programming weakness, because that's what you need. Only a dummy can teach a dummy. Over 50 years ago when I was a teaching assistant and graduate student working for Professor Crandall at MIT, he told me that "You have to be a little bit dumb to teach." That stunned me, and I thought it was at best an unimportant thought, maybe to confuse me. I remembered it and gradually got the idea, but I don't think I fully understood it for 40 years. I do now. You have to be able to guess what the dummy can't understand, and then help. If you can't understand what the dummy can't understand, you can't help him. So you are lucky to be trying to learn Matlab/Octave programming from a dummy. I'll proceed.

```
  1     %fft Hanning window with k averages
  2
  3     fs=2560;%*****set
  4     k=6; %***** the number of spectra we will
compute
  5     m=4096; %***** the size of the fft we will
be computing
  6     index=(1:m)';x=x(:);%assure x and index a
column
  7     p=m/2+1; % number of frequency values com-
puted
  8     KMU=4/(m*k); % this has the N/2, and 2x for
the Hanning window; tested 21504
  9     w=hanningc(m);%I corrected Matlab's Hanning
window.
  10    skip=2048; %***** m-noverlap; or the number
of points we skip for ea spectrum
  11    % N = m + (k-1)*skip
  12    XX=zeros(p,1); %this is the vector we fill
with fft values
  13    for l=1:k
  14            xw=w.*(x(index));
  15            index=index+skip; %advance index by
skip
  16            Xx=abs(fft(xw));
  17            X=KMU*Xx(1:p);%must half X(1) and
X(p)
  18            X(1)=X(1)/2;X(p)=X(p)/2;
  19            XX=XX+X;
  20    end
  21    d=(1:m/2+1);
  22    f=(d-1)*fs/m;
  23
  24    g=max(XX);
  25    plot(f,XX),grid
  26    title(['Maximum vel level is ',num2str(g)])
  27    xlabel('Frequency, Hz');
```

Now for some line-by-line comments:

1. %fft Hanning window with k averages [the beginning "%" means the whole line is a comment. The computer ignores everything on the line following a "%." It explains what the program does.]
3. fs=2560;%*****set [Here is where I insert the sample rate in samples per second. The ";" at the end prevents the value from being printed on the screen.]
6. index=(1:m)';x=x(:);%assure x and index a column [Creates index as a row of numbers from 1 to m. "(1:m)" makes a row from 1 to m. Then transposes it to a column with "(1:m)'" the apostrophe transposes it to a column. The ";" ends the line and indicates the start of a new line. So we actually have two lines on this line 6. "x=x(:);" The "(:)" makes x a column. It's one of the uses of the colon. Importantly the ";" indicates the line end, but also prevents printing the list on the screen.]
7. p=m/2+1; [Creates a number "p", the number of frequency values computed.]
8. KMU=4/(m*k); % this has the $N/2$, and $2x$ for the Hanning

window; tested 21504 [Creates the constant KMU taking into account the fft length *m*, and the number af averages *k*. Again the ";" at the end prevents it from being printed on the screen.]

9. w=hanningc(m);[This creates a column of the Hanning window values *m* long.]
10. skip=2048; %***** m-noverlap; or the number of points we skip for ea spectrum; [On line 5 I made the fft length 4096, so if we skip 2048 values for each average we will be doing 50% overlap.]
11. % N = m + (k-1)*skip [The line is a comment. I'm reminding me that when you set *m*, and *k*, and skip, you need at least *N* values of data. The program expects you have a list of data values in the workspace named x. We've had to load the data and name it x.]
12. XX=zeros(p,1); %this is the vector we fill with fft values [makes a column of *p* rows and one column of zeros that the program will fill with spectrum values.]
13. for l=1:k [this starts a 'for' loop that will execute lines 14 to 18 *k* times.]
14. xw=w.*(x(index)); [This takes each value of *w* and multiplies it by corresponding value of *x* for each value of index. It is a term by term multiply which is indicated by the ".*" Without the dot it would be a vector multiply like a row times a column or a row times a row. A row times a row is an outer product, a square matrix.]
15. index=index+skip; %advance index by skip [Index started as a column of numbers from 1 to *m*. Now we add skip to each of those numbers so the second time through index runs from 1+skip to *m*+skip. This is how we advance through the data.]
16. Xx=abs(fft(xw)); [Here we use calculate absolute value of the fft of the current xw. Xx is now a column of *m* real transform values.]
17. X=KMU*Xx(1:p);%must half X(1) and X(p) [We multiply the first 1 through *p* values of Xx by the constant KMU.]
18. X(1)=X(1)/2;X(p)=X(p)/2;[This divides the first and last value of X by 2.]
19. XX=XX+X; [This adds the newly calculated values of X to XX to build up the averages. KMU is divided by the number of averages so each average is correctly apportioned.]
20. end [This ends the for loop and XX now contains our spectrum values.]
21. d=(1:m/2+1);[This makes a row of numbers from 1 to *p*, the number of spectrum values calculated.]
22. f=(d-1)*fs/m; [The calculates the frequencies in Hz for each spectrum value calculated.]
24. g=max(XX);[This makes *g* equal to the maximum value of the spectrum.]
25. plot(f,XX),grid [This plots the spectrum linearly with a grid. If we wanted a semilog plot we would use semilog y(f,XX),grid.]
26. title(['Maximum vel level is ',num2str(g)]) [This is not much of a title but it illustrates how we can put the maximum spectrum value in the title. "num2str(g)" makes the maximum spectrum value, *g*, a string that can be printed along with text.]
27. xlabel('Frequency, Hz');[This prints a label for the *x* axis. A label for the *y* axis would be printed with ylabel('Amplitude'). The single quote marks are needed whenever text is being printed.]

Conclusions

That's it. I've presented a first lesson in doing vibration diagnostics on a general-purpose signal analysis program with MATLAB and Octave. It's disappointing how little I can get done in one article. I hope it's a tempting start and I can entice some of you to try it. Cost is no longer an excuse.

I wasn't able to give a nice recipe for downloading Octave and GUI Octave, because I'm still not certain which version to work with. The latest version of GUI Octave is 1.5.4 and can be downloaded. Version 4.6.1 is the latest Octave version, and I am struggling to download that. I was advised to test Version 3.4.2. Even though being a dummy has advantages for teaching, it's troublesome for learning the material you teach. Many manuals for Octave can be found by searching the internet for GNU Octave. Colleges are being forced to use Octave due to the high cost of MATLAB, so there is a great deal a material available. I even found a video of a math teacher showing his class how to download Octave.

But here I have tried to talk you into taking some time and downloading Octave and trying out some simple good machinery diagnostic data. I hope I showed you how to input the data, plot it up and see what you have. I've gone into detail on how we arrange the FFT results into the spectrum. I went over all the steps on how to do an averaged spectrum of the data. I wanted to get in the STFT (short-time Fourier transform) and the waterfall and couldn't fit it in. They are relatively simple extensions to the spectrum concept, so a first step is to learn how to do your own spectrum.

Maybe all I got done was to take apart and explain the spectrum calculation. If so, do that. I've tried to give you a start. It goes into the simplest yet most important of our operations, the spectrum. We calculate it with the FFT, which is easy for the computer to do. When you get comfortable with signal analysis, go over my past articles and papers and try those methods.

References

1. MATLAB® for Windows, Version 6, High-Performance Numeric Computation and Visualization Software, The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA, 2000
2. GNU Octave and GUI Octave are widely discussed on the internet. Search with Google on GUI Octave and GNU Octave.
3. Shannon, C. E., "Communication in the Presence of Noise," *Proceedings of the IRE*, January 1949.
4. Zwillinger, D., *CRC Standard Mathematical Tables and Formulae*, 30th Ed., CRC Press, New York, 1996.
5. Braun, S., *Mechanical Signature Analysis, Theory and Applications*, Academic Press, Orlando, FL, 1986.
6. Pachaud, C., Salvetat, R., and Fray, C., "Crest Factor and Kurtosis Contributions to Identify Defects Introducing Periodical Impulsive Forces," *Mechanical Systems and Signal Processing*, Vol. 11, No. 6, pp 903-916, 1997.
7. McConnell, K. G., *Vibration Testing; Theory and Practice,* John Wiley & Sons, Inc. 1995.
8. Gaberson, H. A., "How Windows Work for Vibration Analysis, Vibration Institute Annual Meeting, Bloomingdale, IL, June 21, 2004.

The author may be reached at: hagaberson@att.net.