

Sampling and Time-Domain Analysis

Anders Brandt, University of Southern Denmark, Odense, Denmark

Kjell Ahlin, Blekinge Institute of Technology, Karlskrona, Sweden

Most noise and vibration measurement and analysis systems are able to record time history signals for subsequent processing. This article deals with some important aspects of recording and processing these data streams in order to maintain analysis integrity.

Sampling Theorem

The sampling theorem was presented by Nyquist¹ in 1928, although few understood it at the time. It was not until Shannon² in 1949 presented the same ideas in a clearer form that the sampling theorem was more generally understood. (Reprints of both these papers can be found on the web for the reader interested in history.) Actually, Shannon stated that the sampling theorem was “common knowledge in the art of communication,” but he is widely acknowledged for formalizing the mathematics of the sampling theorem in a precise and accessible way. The principle of the sampling theorem is rather simple, but still often misunderstood. As described by Shannon, when a time history is sampled at a frequency f_s , the spectrum of the sampled signal is the spectrum of the analog signal, plus repetitions of that spectrum (and its negative frequency conjugate), spaced f_s apart. For the sampled signal to have the same spectrum as the analog signal within, $-0 \leq f \leq f_s/2$ the bandwidth B of the sampled signal must be:

$$B < f_s / 2 \quad (1)$$

Half the sampling frequency $f_s/2$ is generally called the Nyquist frequency, after being named so by Shannon. It is important to understand that if the sampling theorem is fulfilled before sampling an analog signal, then the sampled signal is an exact representation of the analog signal. In other words, the sampled signal contains all information in the analog signal. Why this is so, is not easily understood intuitively, but a signal with limited bandwidth cannot vary much between two samples.

Mathematically the sampling theorem can be written in terms of the interpolation formula expressed by Equation 2:

$$x(t) = \sum_{n=-\infty}^{\infty} x(n) \frac{\sin[\pi f_s(t - n\Delta t)]}{\pi f_s(t - n\Delta t)} = \sum_{n=-\infty}^{\infty} x(n) \cdot \text{sinc}[f_s(t - n\Delta t)] \quad (2)$$

where Δt is the sampling interval $1/f_s$. This equation (or an approximation of it) is used for interpolation in modern systems with resampling-based order tracking³ and can also be used to change the sampling frequency of a signal. Equation 2 may at first appear rather complicated, but is actually quite easy to understand once the math is revealed. It says that if you know the sampled signal (right-hand side), then you can calculate the original signal $x(t)$ at any instance in (analog) time t . To understand what Equation 2 actually does, the two factors $x(n)$ and the sinc function inside the sum in Equation 2 are plotted in Figure 1 for a limited time span around t . Equation 2 implies that a *sinc* ($\sin(\pi x)/\pi x$) is centered at the time instant t where we want to calculate the signal $x(t)$, and then the values of the sinc are calculated at the sampling instances $n\Delta t$ (where the sampled signal $x(n)$ exists). The new value of $x(t)$ is the sum of the product of the sampled signal $x(n)$ and the sinc. Note also that the width of the sinc is such that its zero crossings appear a distance Δt apart.

Three implications of the sampling theorem are discussed here. First, we must make sure before sampling a signal that it has no frequency content above half the sampling frequency. This is done by an analog antialias filter before the A/D converter (ADC).⁴ The antialias filter must have a cutoff below $f_s/2$, as the slope of any analog filter above the cutoff frequency is finite. The ratio between the sampling frequency and the cutoff frequency of the antialias filter is called the oversampling ratio (sometimes oversampling factor), and the steeper the slope of the antialias filter, the lower the oversampling factor can be. Traditionally the oversampling

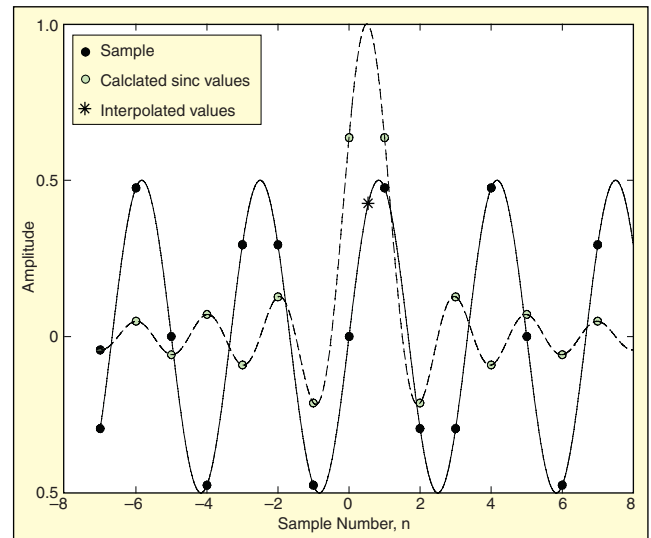


Figure 1. Illustration of principle of interpolation as described by sampling theorem.

factor was always 2.56 in FFT analyzers, but in some modern analyzers with sigma-delta ADCs, the oversampling factor has been reduced.

Second, when resampling a signal in practice, a truncation error will occur because of the finite record length, since Equation 2 includes a sum of infinite length. The sinc function falls off (decreases) by $1/t$ away from the center point. So in practice, the truncation error is most prominent at the ends of the data stream where there is limited (or no) data on one side of the interpolation point t . Some data at the ends should preferably be excluded after resampling; 100 samples are normally sufficient for time-domain analysis. If very accurate data are needed, for example, to perform input/output analysis in the frequency domain, many thousands of points must be discarded at each end.

Third, the sampling theorem says that the signal has to be sampled at more than twice the bandwidth of the signal (and not twice the highest frequency, as is often erroneously stated). A band-limited signal, where the frequency range of the signal does not start at 0 Hz and go up to B , but rather lies in some frequency range $(f_c - B/2) \leq f \leq (f_c + B/2)$, where $f_c \gg B$ can be sampled at a sampling frequency much lower than the frequency f_c . Though of little practical use for noise and vibration analysis, it is frequently used in modern cell phones, where signals in the 1.8 GHz band of the GSM network are sampled at a few MHz (typically after down conversion to a base-band signal).

Data Quality Analysis

In the noise and vibration analysis community, it has been common to reduce time history data to spectra in real time during the measurement. This was originally implemented in the first FFT analyzers in the 1970s due to the high cost of memory, but has since been widely adopted as a standard. Although time history recording is possible in most commercial software today, the software architecture is still usually based on frequency domain analysis. This is an unfortunate habit, since there are very good reasons to save time history data for later processing. One reason is due to the nature of frequency analysis, since all information about the spectral content of a signal cannot be analyzed with one set of parameters (time window and frequency increment). Another reason is that diagnosing problems with transducers, cables, and electrically generated noise can generally only be done in the time domain. So we strongly advocate always recording time data, possibly with the exception of applications like online monitoring or

shaker testing.

Data quality analysis aims at finding problems with transducer (accelerometer) mounting, cable problems, etc., that are commonly experienced in noise and vibration measurements. In principle, data quality analysis is based on calculating a suitable number of statistical parameters of the signal and comparing them with what is empirically known (or learned) to be “normal.” Accelerometer signals are most common, and we will focus our attention on such signals. But our discussion here is largely applicable to any dynamic measurements. Dynamic acceleration signals tend to have zero mean; otherwise, double integration to displacement would be nonzero, meaning the test specimen moved permanently during the test. Dynamic measurements on missile launches, for example, could be exceptions to this if the piezoelectric transducers used have a response down to DC (static). The use of accelerometers with frequency performance down to DC, such as piezoresistive or capacitive transducers, may require extra care.

Most acceleration signals should have approximately symmetric amplitude distribution. This may not always be the case, such as vertical acceleration measured on a vehicle driving over potholes, or where severe, one-directional shocks are included in the data. Measurement equipment often adds a small offset voltage to the signal, so good practice is to always remove the mean of the entire record before proceeding with the analysis described here.

A first check is to assure that data are approximately symmetric around zero. This can be done by observing the minimum and maximum values of the signals. Another particularly useful statistical function is *skewness*. Skewness of a signal x , denoted S_x , is the third statistical moment, normalized by the standard deviation to the third power:

$$S_x = \frac{1}{\sigma_x^3} \frac{1}{N} \sum_{n=1}^N (x_n - \mu_x)^3 \quad (3)$$

where x is the measured signal, N samples long; μ_x is the mean of x , and σ_x its standard deviation (rms value, if mean is zero). Skewness resembles mean value in the sense that the third power keeps negative numbers negative. A skewness value not equal to zero indicates that the probability density function of the signal is not symmetric around the mean. Normally for dynamic signals, the skewness will be zero. A difference between skewness and mean is that larger values are emphasized in the skewness by the third power. For example, skewness will be more sensitive to electrical spikes in the signal in one direction only (positive voltage, or negative).

Normalized kurtosis is the fourth moment divided by the standard deviation to the fourth power:

$$K_x = \frac{1}{\sigma_x^4} \frac{1}{N} \sum_{n=1}^N (x_n - \mu_x)^4 \quad (4)$$

Here we have something entirely different from skewness, since the fourth power makes all values in the sum positive. Indeed, the kurtosis resembles the variance, for which the power is two. However, kurtosis is more sensitive to large values in the signal than the variance. It can be shown that the kurtosis of Equation 4 for a Gaussian-distributed signal is exactly three. This causes some people to define the kurtosis as Equation 4 *minus* 3, which is also often referred to as excess kurtosis; so be sure to check your software.

The kurtosis depends on the statistical distribution of the data so that if the data exhibit large values more often than corresponding Gaussian data, the kurtosis is higher than three. If the kurtosis is less than three, then the data exhibit large values less often than corresponding Gaussian data. Another way of putting it is that the kurtosis is greater than three if the tails of the probability density are above the probability density curve of a Gaussian distribution (based on the same mean and variance) as that of the signal in question, and less than three if the tails are below the same density curve. For data quality purposes, there is little need to present the probability density, since the kurtosis is easier to interpret.

There is no such thing as a “normal” kurtosis value for “real-world” measurements. In a mixed signal composed of random, modulated tonal and impact components, the kurtosis can vary

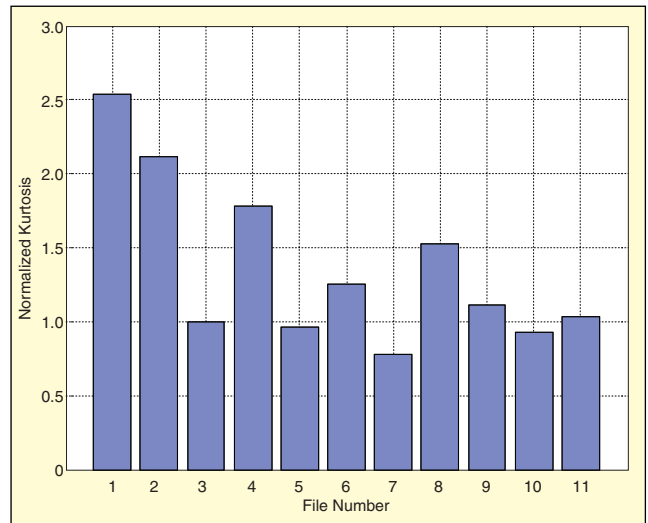


Figure 2. Bar graph with kurtosis values for each of 11 channels normalized to that of channel 3. (Channels 1, 2, 4, 6, 7, and 8 are inconsistent, and should be considered suspicious in terms of quality.)

considerably. For data quality purposes, it is therefore necessary to compare a particular measurement to normal kurtosis values. Usually, comparing the kurtosis between channels is reasonable as long as the vibration characteristics are the same for different measurement locations. (One case where this would not be true is if one transducer is close to some rattling part and other transducers are not.)

The statistical metrics you choose should preferably be computed two ways; first, the overall values, using the entire recording of the signal, should be calculated. Secondly, the same statistical metrics should be calculated for shorter time frames, such as one second intervals, for example. (A suitable record length depends on the bandwidth of the data.) A single spike will tend to disappear in the overall kurtosis, where it is more likely to give a significant contribution in the frame where it appears. To reduce the amount of values to look at from the frame analysis, the largest (without sign) of the values could be extracted. Since the intention is to find intervals with problems, it may be quite sufficient to find if there is any such interval at all in the data.

When recording many channels, or many measurements, a useful trick is to normalize each statistical metric by the metric of one channel that you assume (or perhaps confirm via manual analysis) is without quality errors. A plot of all those normalized metrics will clearly show a channel with a potential problem, as plotted in Figure 2, where data from an 11-channel measurement are presented, and where the kurtosis of at least Channels 1, 2, 4, 6, 7, and 8 are clearly different than the rest. (The example was taken from a case with many problems in the data and is not representative of any normal measurement). Those channels should be more thoroughly checked.

Time-Domain Analysis

In cases where minimum and maximum values in the time signal are to be estimated in an analysis procedure, it is important to consider the oversampling ratio. Such procedures are most commonly used in fatigue analysis, where different forms of reduction processes are used for cycle counting, such as range pair and rainflow reductions. Transient analysis, for example in pyroshock or shock-table applications, is also a common example where time-domain analysis is used. The shock response spectrum is another related example. With a low oversampling ratio of 2.56 or less, as used in FFT analyzers, minimum and maximum values will be seriously in error. It is generally accepted to use 10-times oversampling, which yields less than 10% error in minimum and maximum estimates.⁵ From the above discussion of interpolation, it is clear that 10 times oversampling can be obtained by resampling the data immediately prior to the time domain analysis. If storage space is limited, data with an oversampling of approximately 2.56 (whatever the hardware manufacturer has implemented) can be

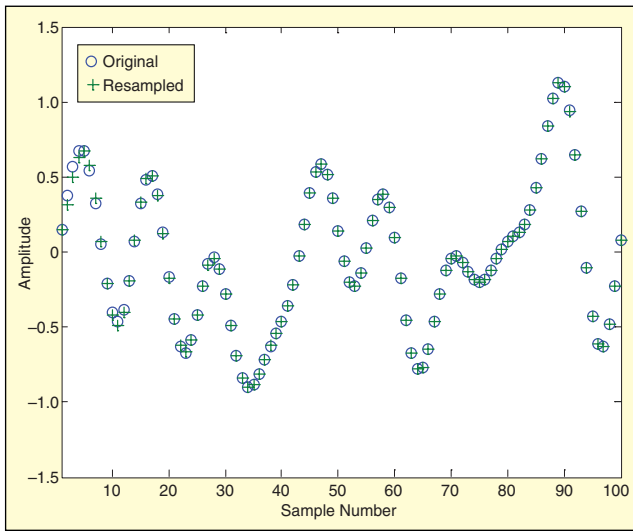


Figure 3. Result plot of Example 1 showing the first 100 original samples and samples recreated by the resample process.

stored without losing any significant data quality. This oversampling ratio is sufficient to allow for accurate resampling when so needed (see Example 1).

Another important aspect of data acquisition for time-domain analysis is the phase characteristics of the antialiasing filters used. In this case, the analog and digital (decimating) antialias filters are important. For a transient to come out of a filter undistorted, the filter must have linear phase characteristics as a function of frequency. Typically, analog antialias filters have significantly nonlinear phase characteristics. With modern data acquisition systems using sigma-delta ADCs,⁴ it is quite easy to accomplish linear phase characteristics over the entire frequency range, from 0 Hz to (nearly) $f_s/2$. Not all manufacturers implement this, however, because it involves some additional costs for the digital filters. Therefore, users must check the documentation and verify that the phase characteristics of the measurement system are indeed linear prior to using it for any type of transient recording or other data where min/max analysis is to be used.

Note that for amplitude probability density estimation, although done in the time domain, it is not necessary to use oversampling greater than 2.56. The same applies to statistical metrics, such as standard deviation and skewness, which are insensitive to the oversampling ratio.

Digital Filters

Digital filters are frequently used when analyzing noise and vibration data. Their design is a whole discipline of its own and here we will touch only on some simple facts that are important to understand from a user perspective.

First of all, a rather general definition of a digital filter between an input x_n and an output y_n is shown:

$$y_n = \sum_{k=0}^{N_b} b_k x_{n-k} - \sum_{l=1}^{N_a} a_l y_{n-l} \quad (5)$$

where the filter coefficients a_l and b_k define the filter characteristics. If all the a_l coefficients are zero; that is, the filter is using only old input values to compute the output, then the filter is called a finite-impulse response (FIR) filter, since it will have a finite-impulse response of length N_b+1 . On the other hand, if there are a_l coefficients in the filter, it is called an infinite-impulse response (IIR) filter. IIR filters are generally more time efficient than FIR filters; that is, more powerful filter characteristics can be provided using fewer filter coefficients. In some special cases, however, FIR filters are preferred, especially if linear phase characteristics are desired.

Many times, we would like to create a digital filter with characteristics equivalent to a specified analog filter, because most filter theory was developed in the analog days. However, there is no such exact transformation. Therefore, the science of digital

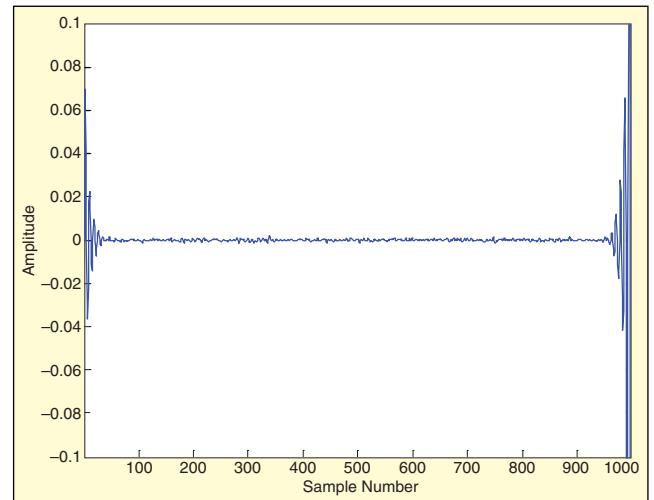


Figure 4. Difference between original signal and resampled signal from Example 1 (see text for details).

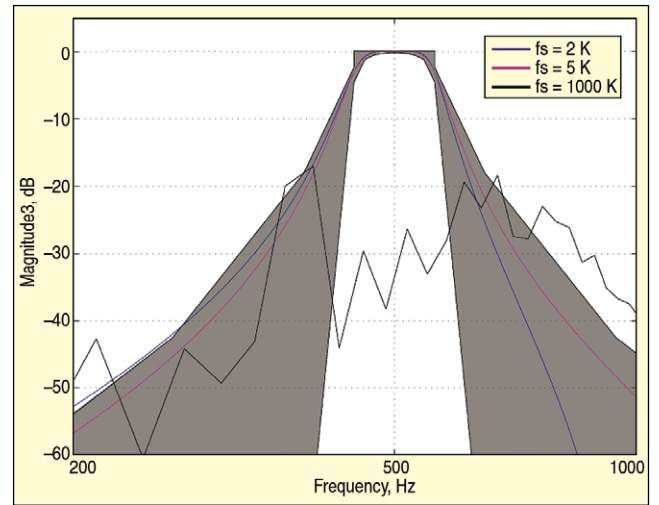


Figure 5. Filter frequency response for a 500-Hz center frequency, 1/1-octave filter, using three different sampling frequencies. Too low or too high sampling frequency results in erroneous filter shape. Gray shape boundaries are specified by ANSI/ISO standards.

filters deals mainly with how to make the digital filter that best approximates analog filter characteristics. Here it is sufficient to mention a few basic facts about digital filter performance that a user must be aware of.

First, the digital filter approximation of an analog filter performs more poorly the closer one gets to the Nyquist frequency. So if the filter characteristics are defined in the analog frequency domain, one should use sufficient oversampling prior to filtering the signal. In the IEC/ANSI standard for octave filters,^{6,7} for example, a minimum oversampling of five times the highest center frequency is recommended. We recommend using a 10-time oversampling when performing digital filtering of data unless you are sure some other factor is better.

Second, the oversampling factor should not be too large, since this produces larger filters (more filter coefficients) and potential numerical truncation problems.

Third, a useful trick to produce linear phase characteristics even for an IIR filter can be achieved by filtering the data first in the normal (time) direction and then running the filter backward in time. The scaling of the filter has to be considered when using this method. In MATLAB®, there is a function *filtfilt* that performs this type of filtering, including scaling the output so that the efficient (amplitude) filter characteristics are the same as for the IIR filter used in the “normal” way. As a curiosity, note that an analog version of *filtfilt* was sometimes used in the old analog days by running tape recorders forward and backward. Digital computers have certainly made things easier.

Example 1. MATLAB script illustrating effect of resampling.

```
N=1000; % Number of samples to start with
fs=1000; % Sampling freq. for simulation
x=randn(N,1); % Gaussian noise, oversampling ratio is 2
% Low pass filter data to 10 times oversampling
fc=100;
wc=fc/(fs/2); % Cutoff as part of fs/2
[B,A] = ellip(8,0.1,70,wc); % 0.1 dB ripple, 70 dB suppression
x=filter(B,A,x); % oversampling ratio of x is 10
x25=x(1:4:end); % Oversampling ratio of x25 is 2.5
xr=resample(x25,4,1); % oversampling ratio of xr is 10
% Plot 'original' data x, and the resampled xr
plot(1:100,x(1:100),'o',1:100,xr(1:100),'+')
legend('Original','Resampled')
xlabel('Sample Number')
```

To show an example of filtering and resampling, the MATLAB script in Example 1 is used to produce random data with Gaussian distribution. An elliptic digital filter is used to filter the data to an oversampling ratio of 10 in the variable x , defining the “true signal.” The MATLAB function *filter* is used to produce a filter with linear phase (not needed but an example of its use). A “simulated” sampled signal, x_{25} , is then produced taking every fourth sample in x , which will essentially produce the result you would get from your measurement system if it uses the traditional oversampling ratio of 2.5. Finally, this sampled signal, x_{25} , is used to recreate the “thrown away” samples, using the MATLAB *resample* function, producing the new signal x_r . The *resample* function essentially uses Equation 2 to change the sampling frequency of any signal to a new sampling frequency, which is a rational fraction of the original sampling frequency.

The resulting plot of Example 1 is shown in Figure 3, where the ‘o’ symbol shows the original samples, and the ‘+’ symbol shows the recreated samples after resampling. As can be seen in the figure, the first few samples are not accurate, which is due to the truncation effects discussed previously. After those first samples, however, the recreated samples are seemingly identical to the original. This example shows that 2.5 times oversampling is high enough to be able to use the *resample* function to temporarily get a higher oversampling ratio whenever needed with sufficient accuracy, at least for time domain analysis.

An interesting result of Example 1 is plotted in Figure 4, where the difference signal $x - x_r$ is plotted over the entire record (1000 samples). From this difference signal, it is clear what is happening at the ends, where the oscillations of the sinc function at both ends is revealed.

As a second example of filtering, we will use octave filters. This example will illustrate the importance of choosing a suitable sampling frequency and will also show how easy it is to construct a filter in MATLAB. Example 2 shows a script that produces filter coefficients for a (1/1) octave filter at center frequency f_c for three different sampling frequencies. The filter conforms to ANSI S1.11 and IEC 1260 if the sampling frequency is chosen properly.^{6,7} According to the standards, octave (all $1/n$ octaves) filters are calculated as third-order Butterworth filters, which is done by the MATLAB *butter* function. In Example 2, we also use the function *freqz*, which computes the analog frequency response of a

Example 2. MATLAB script to calculate filter coefficients for 1/1-octave filter with center frequency of 500 Hz for three different sampling frequencies – 2, 5 and 1000 kHz.


```
fc=500; % 1/1 octave center freq.
flo = fc/sqrt(2); % low cutoff freq. (definition)
fhi = fc*sqrt(2); % high cutoff freq. (definition)
fsl=[2 5 1000]*1000; % List of sampling frequencies
N=16*1024; % Number of frequency lines for H
for n=1:3
    fs=fsl(n);
    [b,a] = butter(3,[flo/(fs/2) fhi/(fs/2)]);
    f=(0:fs/2/N:fs/2-1/N)';
    H=freqz(b,a,f,fs);
End
```

digital filter. Note that the example only shows the essential code to produce the filter coefficients. It should be followed by a filter function if you want to use it for filtering your data.

In Example 2, the filter shape (frequency response) is calculated for a 500-Hz center frequency and for the sampling frequencies of 2 kHz, 5 kHz, and 1 MHz. These frequencies were selected to show what happens if the sampling frequency is too low (2 kHz is less than five times the center frequency as specified by the standards), a suitable sampling frequency (10-time oversampling), and finally a sampling frequency that is too high compared to the center frequency. In Figure 5, the shapes of the three filters are plotted together with the limits as specified by the ANSI and IEC standards. As can be seen, only the 5 kHz sampling frequency gives a filter shape that conforms to the standard.

This article would not be complete without a few words about editing time data. In general, editing time data should be avoided if frequency analysis is to be performed on the data, since virtually any tampering with the data will change its frequency content. A better approach for frequency analysis is to simply skip the time block of data containing whatever you want to remove in the averaging process. When other types of analyses are performed, great caution always has to be taken when editing data.⁵

References

1. H. Nyquist, “Certain Topics in Telegraph Transmission Theory,” *Trans. AIEE*, Vol. 47, pp. 617-644, Apr. 1928. Reprint as classic paper in: *Proc. IEEE*, Vol. 90, No. 2, Feb. 2002.
2. C. E. Shannon, “Communication in the Presence of Noise,” *Proc. Institute of Radio Engineers*, Vol. 37, No.1, pp. 10-21, Jan. 1949. Reprint as classic paper in: *Proc. IEEE*, Vol. 86, No. 2, Feb. 1998.
3. A. Brandt, *et al.*, “Main Principles and Limitations of Current Order Tracking Methods,” *Sound & Vibration*, March 2005.
4. A. Brandt, *et al.*, “Noise and Vibration Measurement Systems Basics,” *Sound & Vibration*, April 2006.
5. H. Himmelblau, *et al.*, *Handbook for Dynamic Data Acquisition and Analysis*, IES Recommended Practice 012.1, 1993.
6. IEC 61260:1995 *Electroacoustics – Octave-Band and Fractional-Octave-Band Filters*, International Electrical Committee.
7. ANSI S1.11-2004, *Specification for Octave-Band and Fractional-Octave Band Analog and Digital Filters*, American National Standards Institute. 

The author can be reached at: abra@ib.sdu.dk. Some useful MATLAB® scripts used to produce the plots in this article and some additional scripts that help understand the topics of this article are available for download at www.SandV.com.