# Chapter 2

## AVERAGING

So far, we've considered the spectral density of a single "record" in time, say $N$ points sampled at $f_s$. If the signal components are steady ("stationary"), then there may be advantages in averaging the spectral density over many of these $N$-point records. In fact, texts on signal analysis often assume that averaging can and will be done.

Why is averaging useful? Perhaps the most common reason for averaging is to reduce the scatter in background-noise spectral density. This can make it easier to distinguish signals (especially sinusoidal or tonal signals) in the presence of noise. In addition, averaging reduces the volume of data (at the expense of losing some detail); this alone may justify averaging when handling large data sets.

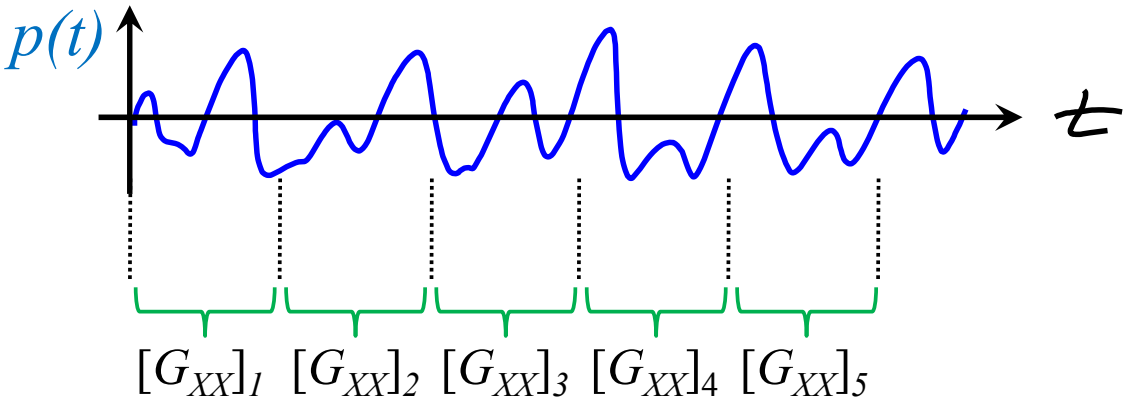
If the desired signal is known—if we are generating the signal or if there is a synchronizing trigger signal—then a specific variety of averaging (synchronous averaging) can be used to extract weak signals from a noisy background. When applicable, synchronous averaging is powerful.

## Ordinary averaging

Because the spectral density of a single record is a meaningful quantity, I'll distinguish the averaged $G_{XX}$ by a bar over the $G$. Eventually, I'll drop this notation—you'll be able to figure out if it's averaged by the context—but this notation will be useful at first. The averaged spectral density is then

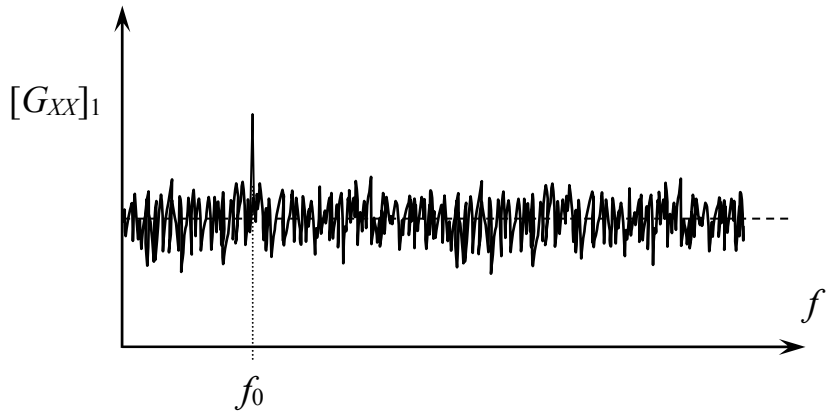
$$\bar{G}_{XX} = \frac{1}{N_r}\sum_{r=1}^{N_r}\left[G_{XX}\right]_r . \qquad \text{(II-1)}$$

This expression indicates that there are $N_r$ records (each having, say, $N$ samples, $\Delta t$ apart in time) and that you have computed a spectral density for each of those records. A specific $G_{XX}$ associated with record, $r$, is denoted $\left[G_{XX}\right]_r$. For example, if $N_r$ is 5, there would be five segments of the time series, each $T$ seconds long and each segment would produce a $G_{XX}$.

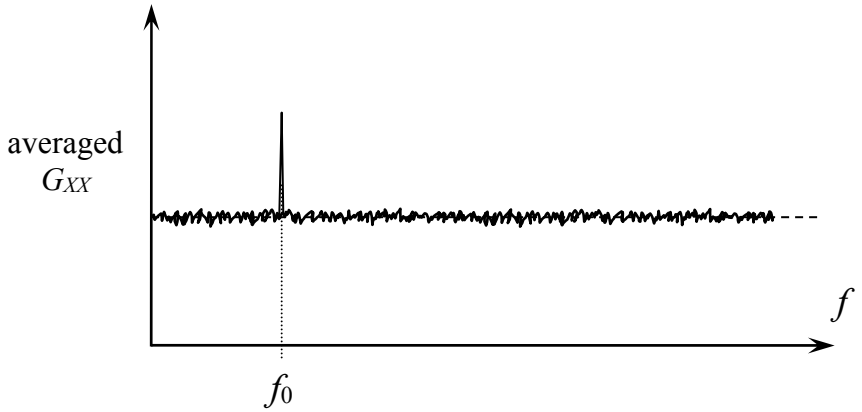



The averaged spectral density would be the average of the five $G_{XX}$'s. If each record has $N$ samples, the averaged spectral density will also have $N$ samples. This process is sometimes called *mean-square* averaging (and sometimes, *root-mean-square* or *rms* averaging) because, by averaging spectral density, we're averaging the magnitude-squared values of the linear spectrum. The phase associated with the linear-spectrum values is lost.

Let's say that $x_n$ is a sine wave at frequency, $f_0$, in random noise. Suppose you have 16 seconds of data sampled at 1024 samples per second. To average, we might select one-second records ($N = 1024$). You would calculate a $G_{XX}$ for each of the 16 records. The $G_{XX}$ for a single record might look something like this,




If you then averaged the 16 $G_{XX}$'s and plotted the result, the averaged spectral density might look something like this,




After averaging, the height of the peak at the sine-wave frequency, $f_0$, is about the same as for any of the individual $G_{XX}$'s (except for minor fluctuations from the added noise). The average noise level for the averaged spectral density will be about the same as the average noise level in any of the individual spectral density's, too, but with much less "scatter" than in the individual spectral density.

If we calculate the noise power by $\sum G_{XX}\,\Delta f$ over all the bins (all frequencies) except the one that contains the sine wave, we'd get just about the same value from any of the individual $G_{XX}$'s as we would from the averaged $\overline{G}_{XX}$. If we calculated the signal power (as $G_{XX}(f_0)\Delta f$) we'd also get nearly the same answer from any individual $G_{XX}$ as we would from the averaged $\overline{G}_{XX}$. (For the

signal power calculation, the added noise may cause significant fluctuation from record to record but the value, on average, will be similar.)

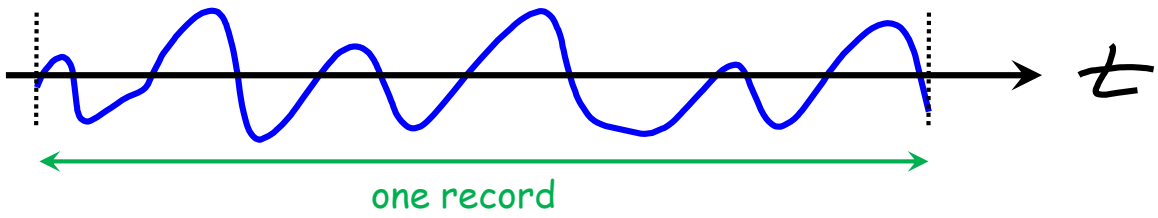
Consequently, ordinary averaging doesn't change the ratio of signal power to noise power. The signal may be easier to *distinguish* in the noisy background in the averaged spectral density but we haven't reduced the noise power or increased the signal power.

Now consider two ways of processing the same 16-second time series—the sine wave in random noise. With 16 seconds of data sampled at 1024 points per second, you could,

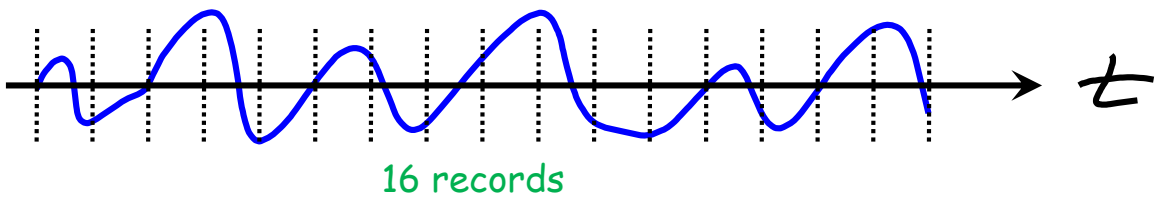
- Find $G_{xx}$ for a single record 16 * 1024 samples long, or,
- Find $\bar{G}_{XX}$ by averaging 16 $G_{XX}$'s where each $G_{XX}$ represents one second (1024 points) of time data (as we did previously).

The two processes use the same data set. Would you get the same answer?

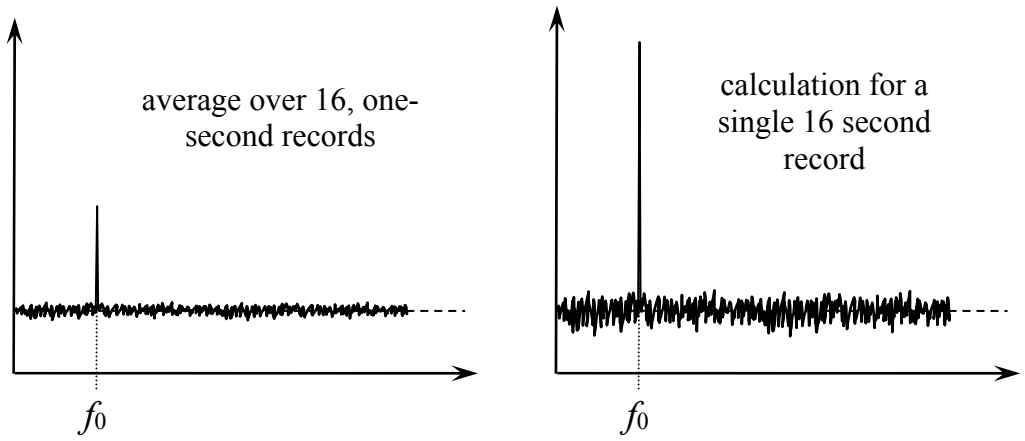
Suppose that we calculate the spectral density for the entire 16 seconds of time-domain data in one transform




and compare that to the result from averaging 16 one-second records,




The spectral density plots would look something like these:




The average level of the spectral density of the noise is roughly the same (although there's more scatter from the single 16-second record) but the signal at $f_0$ for the right-hand plot is much larger.

In this example, the noise is uniformly distributed over frequency. Spectral density is a measure of the "power" density—something like pascals-squared per hertz or volts-squared per hertz. We've *defined* $G_{XX}$ to make this spectral density calculation correctly so, however we analyze the same physical data, we should get similar answers for power that is distributed over frequency. That's why the noise levels on these two spectral density plots should be nearly equal. In contrast, the sine wave component is NOT distributed over frequency—it is completely contained in a single spectrum bin regardless of the width, $\Delta f$, of that bin—so the spectral density isn't an appropriate measure; therefore, we see the spectral density equivalent of the sine wave component changing between the two plots. However, the *product* of the spectral density value for the sine wave and the bin width would be very nearly the same for both analyses.

The signal-to-noise ratio is considerably higher for the $G_{XX}$ based on the single 16-second record than for the $G_{XX}$ based on the 16-record average. Why?

First, the noise is uniformly distributed in frequency so the spectral density has a specific value regardless of the $\Delta f$ of the $G_{XX}$ calculation. Second, the signal is a stable sine wave so all of its power is in a single "bin" in the frequency domain regardless of the width, $\Delta f$, of that bin. In both methods of processing used above,

the *mean-square* value of the sine wave is the same but $G_{XX}$ is proportional to the *mean-square* value squared *divided by the bin width*,

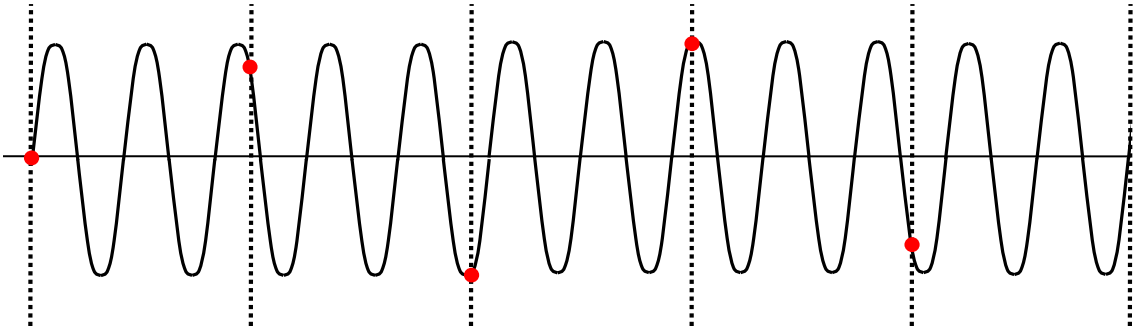
$$G_{XX}(f_0) = \frac{(rms)^2}{\Delta f}. \qquad \text{(II-2)}$$

Since the duration of the single 16-second record is 16 times that of the one-second records, the $\Delta f$ for the 16-second $G_{XX}$ is one sixteenth that of the averaged $\overline{G}_{XX}$. So the "per-hertz" value of the sine wave at $f_0$ is much higher for the 16-second record.

In effect, the $G_{XX}$ of the 16-second record is equivalent to using a much narrower filter around $f_0$ so the signal-to-noise ratio really is better than for the average of 16 one-second records. An alternate view is that the averaged version, $\overline{G}_{XX}$, throws away any relative phase information about the signal from record to record so the coherence of the sine wave cannot be exploited. With the single record, the continuity of the sine wave is maintained for the entire 16 seconds.

## Synchronous averaging

The $G_{XX}$ calculated for a single-record is an array of real numbers proportional to voltage squared (if $x$ is in volts). If this $G_{XX}$ was calculated from the time series from 0 to $T$ seconds, the next record would extend from $T$ to $2T$, and so on. The $G_{XX}$ for the next record is also an array of real numbers – *any phase relationship from record to record is lost*. In contrast, the phase of the linear spectrum depends on the phase of the starting point of the record. In the figure below, each of the five records begins at a different phase of the sine wave; consequently, at the frequency of the embedded sine wave, the phase of the complex linear spectrum would change from record to record. $G_{XX}$ doesn't preserve those phases.



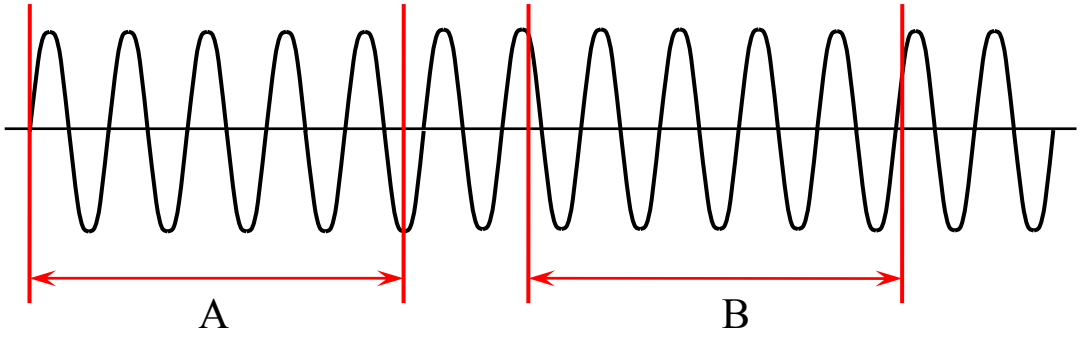
If we add $N_r$ of the corresponding $G_{XX}$'s, the sum would give $N_r$ times the power of any component whether that component was coherent or incoherent from record to record. The average is the sum divided by $N_r$ so neither the power of a coherent signal nor the power of incoherent noise is changed by this type of averaging. (Another way to say this is that the signal-to-noise ratio is unaffected by "mean-square" averaging.)

If we could exploit the coherence of a signal (its "predictability" or its record-to-record continuity), then we could enhance the signal while suppressing the noise.

There are several ways to do this:

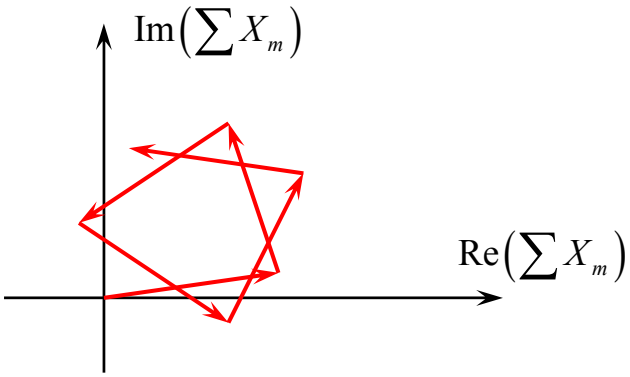
1) Take $G_{XX}$ of the entire time series instead of breaking it up into many records. For the much larger time duration, the effective resolution, $\Delta f$, in the frequency domain is much finer. So the frequency bin that contains the sine wave contains much less noise and the sine wave component appears to be much higher above the noise. This method is only successful in practice when the desired signal is a sine wave or a collection of discrete sine waves.

2) Instead of averaging $G_{XX}$ from record to record, we could average the linear spectrum, $X$, from record to record and then calculate the $G_{XX}$ based on the averaged $X$. For this strategy to work, the time records must be "synchronized" with respect to the signal that we're trying to extract. As mentioned above, the phase of the value $X_m$ (where $m\Delta f$ is the center frequency of the bin that contains our "model" signal) depends on the phase of the sine wave with respect to the time record:

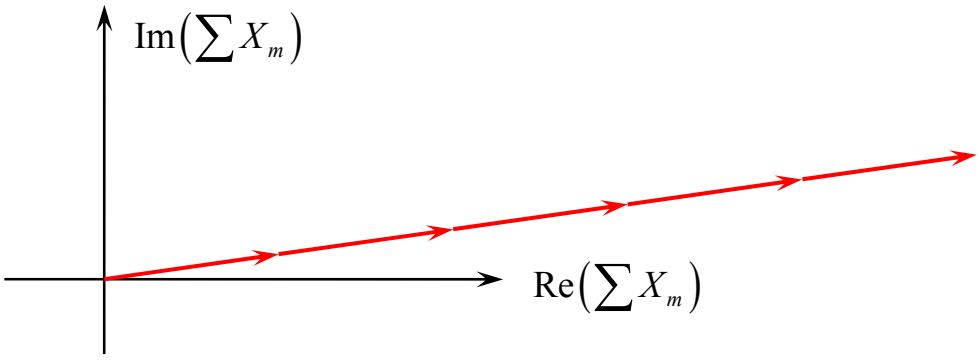



The transform of record A will produce a different phase for the $m^{th}$ component of $X$ than the transform of record B; both records are the same length but they start at different points in the sine wave's cycle.

If the records are not synchronized (that is, if every record is not forced to start at the same place in the signal's time cycle), then the addition of the complex $X_m$ values—they add like vectors—might look something like this in the complex plane:




If the records *are* synchronized (so that the phase of each $X_m$ is the same), then the vector addition would look like this:



In the unsynchronized case, the signal sums the same way noise does – the magnitude of the sum of the $X_m$'s is much less than the sum of the magnitudes. When synchronized, the magnitude of the sum is equal to the sum of the magnitudes.

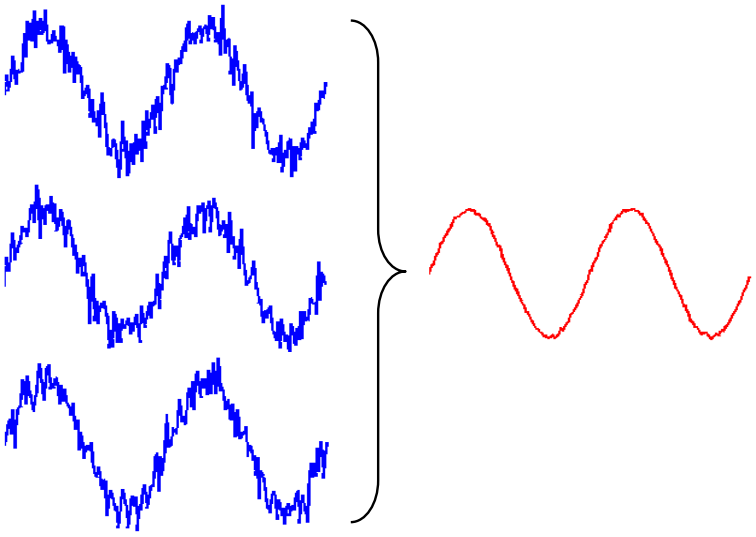
In the synchronized case, the signal "power" (e.g., volts *squared*) grows as the number of records, $N_r$, *squared* whereas the noise (any incoherent component) grows[1] as the number of records, $N_r$. Therefore, the signal-to-noise power ratio grows as $N_r$ (and the amplitude ratio grows as the square root of $N_r$).

Although the example here is a sine wave, the signal can be any repetitive waveform. As long as we have some synchronizing "trigger" to tell us when to start each record, we can average the linear spectra and enhance the coherent (or synchronous) signal over the incoherent noise[2].

3) Often the best approach is to average in the time domain *if we have a synchronizing trigger*. In the example below, each record contains two full cycles of the sine wave and 64 records were averaged to produce the smooth sine curve on the right. (Only three of the 64 original records are shown on the left. Notice the degree to which the noise is reduced.)

[1] I will not prove this. If you don't believe it, write a program to sum random complex numbers (equal magnitude, random phase), run the code many times, and calculate the magnitude of the sum.

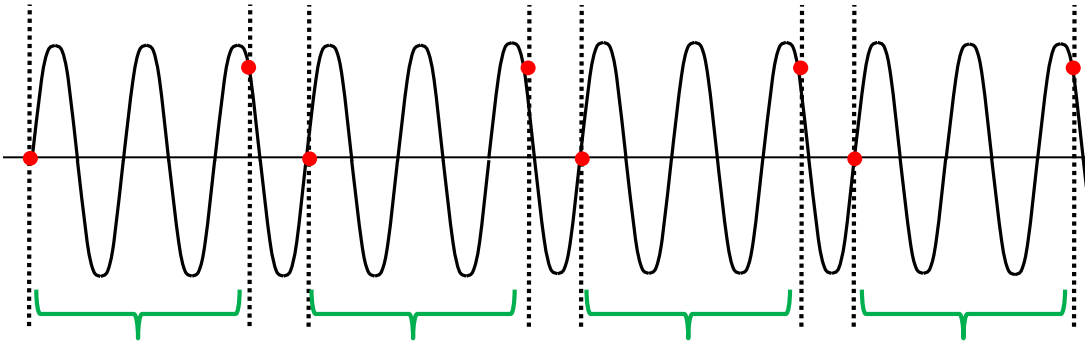
[2] It's tempting to think that you could derive the synchronizing trigger from the signal itself. But remember that you're going to want to use these techniques when the desired signal is buried in noise. The presence of the noise superimposed on the signal will usually frustrate any such attempts. What we need is a clean trigger signal. If we are generating the signal ourselves, we can often arrange to provide a separate, clean trigger, too.



Averaging in the time domain allows us to see what the "clean" waveform looks like. Once the smoothed version of the time waveform has been generated by synchronous averaging, the spectral density can be computed on that smoothed waveform. (And, the spectral density would be computed from the single record—the averaging is already done.) The result should be identical to that produced by Method 2 above.

In either Method 2 or Method 3[3], there are several options for synchronization.

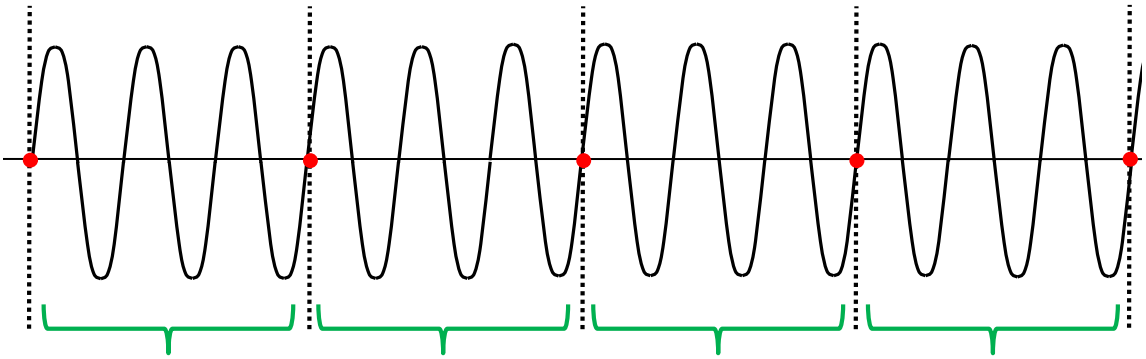
(a) Wait for the same starting phase:



Notice that some of the original data is skipped. If the process is not carried out in real time, overlapping records can be taken to avoid skipping data.

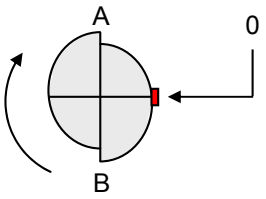
[3] Method 2 is sometimes called "vector averaging" and Method 3 is often called "time averaging".

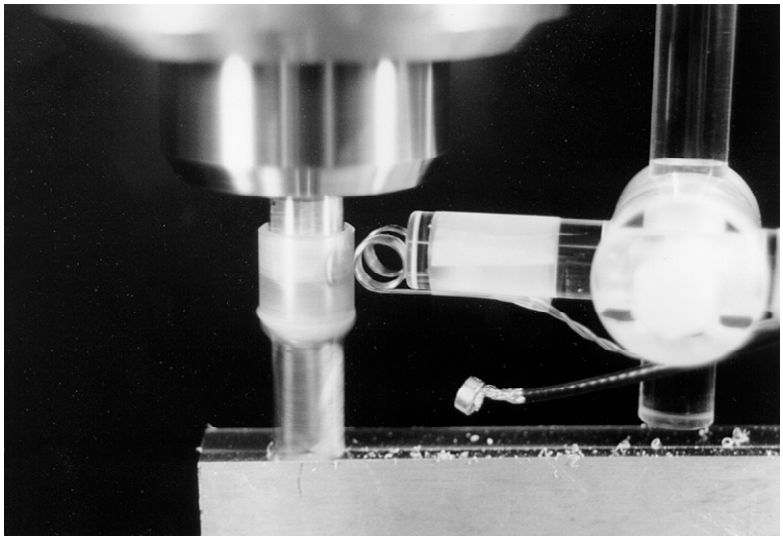
(b) Alternatively, the sample rate could be adjusted to match the periodicity of the signal (to match the trigger points, that is), which also avoids skipping data.



As an example of synchronous averaging, consider analysis of the vibrations produced in a piece of metal as a milling machine makes a cut in the metal. The metal block is at the bottom of the photograph. The signal is the output of an accelerometer fixed to the metal block. The coil assembly (attached to the acrylic fixture on the right and positioned close to the cutter) in conjunction with a small, rare-earth magnet, which is inside the small plastic sleeve on the milling cutter, generates a synchronization signal with a period of one pulse per cutter revolution.

magnet on tool and fixed coil produce the trigger signal

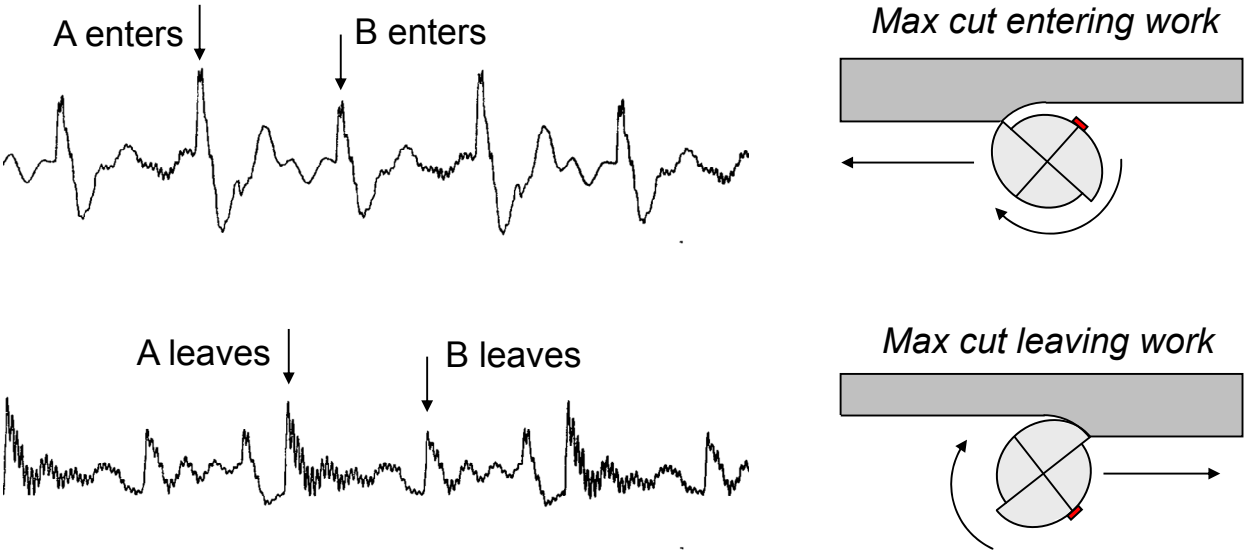


The cutter has two cutting edges ("flutes" A and B) as shown in the cross-section sketch (a top view looking down toward the metal block) to the left of the photograph and the magnet is shown as a small red rectangle at position "0".

Without the trigger signal, we can either examine the raw time series or apply mean-square averaging. We cannot separate the cutter vibrations from the vibrational noise in the machine shop and we cannot distinguish one cutter face from the other. With the trigger signal, we can average synchronously in the time domain (a) to improve the signal-to-noise ratio of the desired signal, and (b) to examine the effects of the two cutter faces separately. Synchronous averaging provides a wealth of detail that would be lost without synchronization.

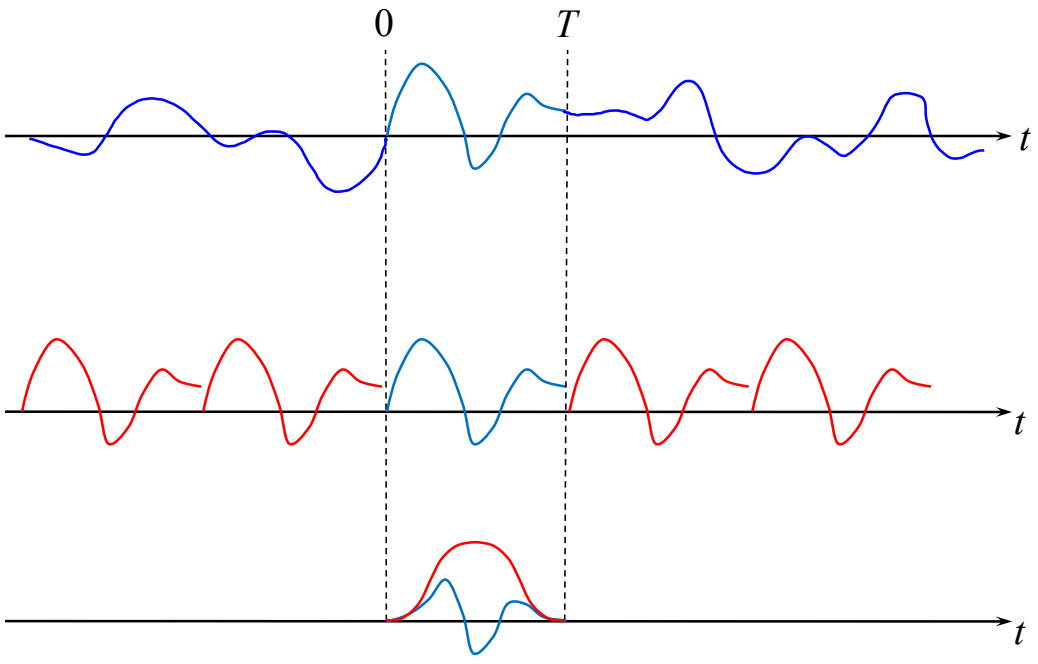
The figure below shows the results for two types of cuts. In the upper portion, the cutter is passing from right to left. With the indicated cutter rotation, the deepest cutter "bite" is taken as the cutter enters the work. In the lower portion, the cutter is passing from left to right. Here, the deepest bite is taken as the cutter face leaves the work. In this second case, the deep cut followed by the cutter leaving the work produces significant "ringing" (residual vibration) in the metal and that appears as the higher-frequency vibrations in the averaged signal. Also, the A face is cutting more deeply than the B face—compare the peak amplitudes.



Even in this relatively simple implementation, the advantages of synchronous averaging are readily apparent. Synchronization is not always practical but, in designing a measurement of periodic signals, it is often worth considering. We will re-examine time-synchronous techniques when we review noise from rotating machinery in a subsequent chapter.

## Time-Domain Window Functions

In the previous chapter, we considered one of the consequences of sampling signals—the implied periodicity of sampled functions. In the figure below, an original signal is shown at the top in blue. By sampling the segment from 0 to T, we select a particular "record" and subsequent processing behaves as though that record were actually repeated over and over (middle trace with repetitions shown in red). This implied periodicity introduces discontinuities at the edges of the record.



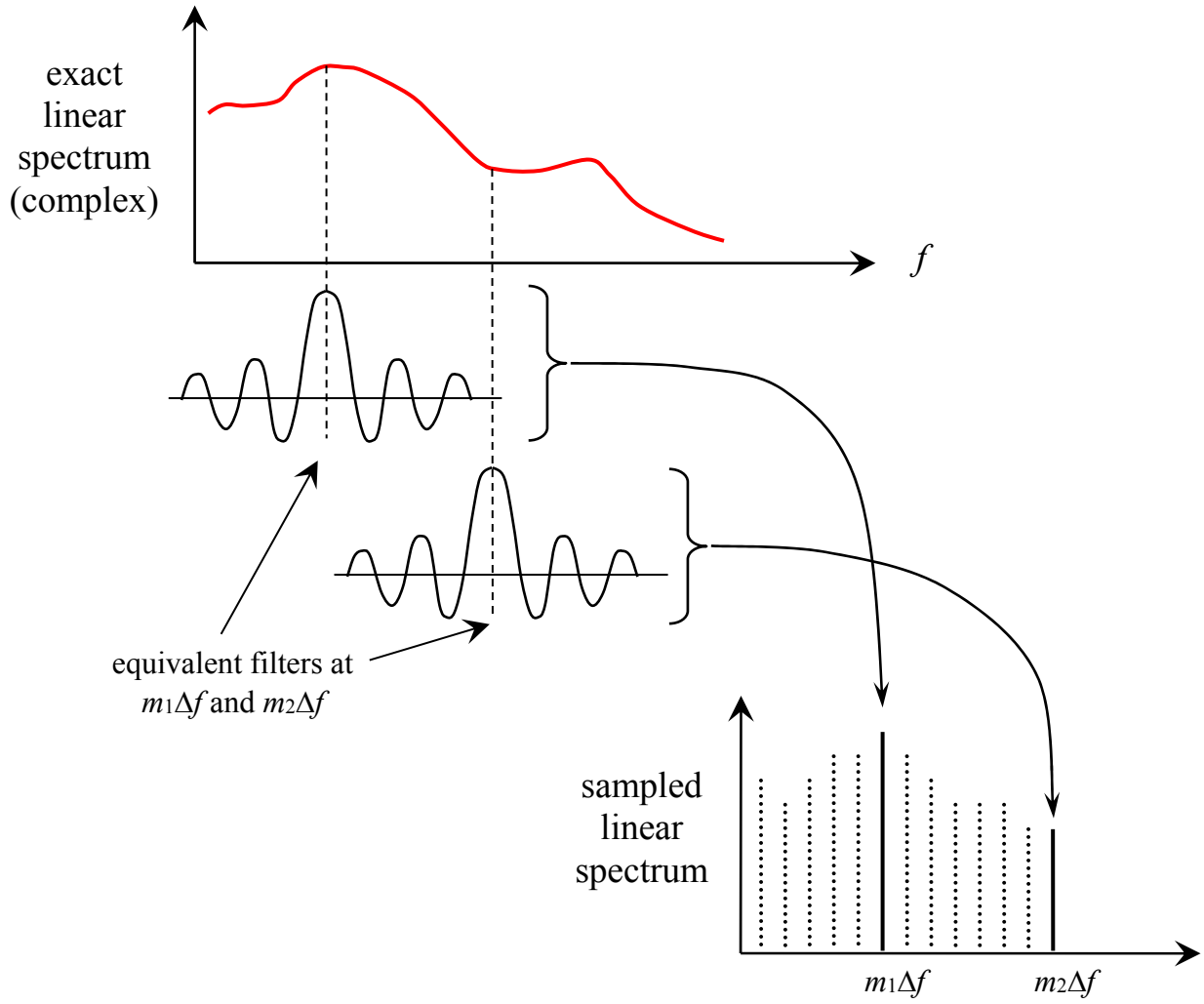
These discontinuities introduce spectral content that isn't a part of the original signal. One way of coping with these discontinuities is to taper the leading and trailing edges of the record (see the bottom trace) with a smoothing function (red curve in the bottom trace).

These smoothing functions applied in the time domain are known as "window" functions. These functions taper the beginning and end of the records to zero (or near zero). This "removes" discontinuities in between signal repetitions in the implied periodicity but, of course, introduces other effects. We will consider these effects in the discussion below.

## Frequency-Domain Shape of Effective Bin "Filters"

Each point in $G_{XX}$ represents the spectral density of the time waveform as it would be calculated from the mean-square of the time-domain output of an equivalent narrow filter. (The actual mean-square output of that filter would be divided by the width of the filter to make it a spectral *density* value.)

If we had the infinite, continuous time waveform, we could generate the exact linear spectrum and the exact spectral density by the Fourier transform integral. In effect, the sampled time series and discrete transforms filter the exact linear spectrum to produce the sampled spectrum:



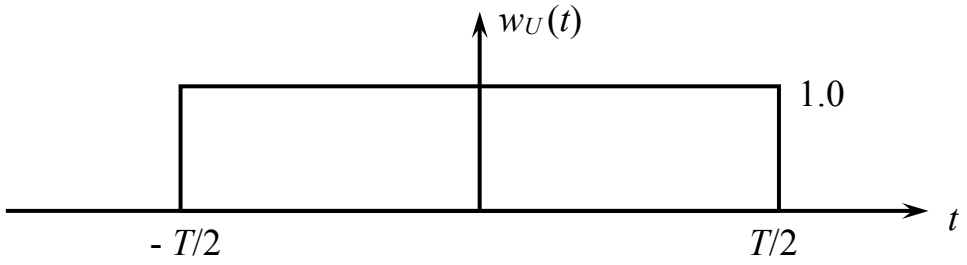

The continuous time function, $x(t)$, is sampled so that the samples are exact values of $x$ at $t = n\Delta t$; however, *the points in the sampled spectrum are not samples of the exact spectrum*, they are *estimates* of the actual spectrum values.

Once the sampled linear spectrum, $X_m$, is found, the sampled spectral density, $G_{XX}$, can be calculated. It would certainly be convenient if the effective bin filter was exactly $\Delta f$ hertz wide with infinitely steep drop-offs on both sides. Then the sampled $G_{XX}$ at $m\Delta f$ would represent only the power from $m\Delta f - \Delta f/2$ to $m\Delta f + \Delta f/2$ in the true (continuous) spectrum.

But that's not the case. We can find the actual equivalent filter shape by taking the Fourier transform of the time "window" function. (And, to make the

integration easier, we can take the transform for a window from $-T/2$ to $T/2$ instead of from 0 to $T$.)

The "uniform" window (also called the "rectangular" or "boxcar" window) simply selects the values in the time series from $-T/2$ to $T/2$ without weighting those values. That window looks like a simple rectangle with unit amplitude:




where $w_U(t) = 1$ for $-T/2 < t < T/2$ and zero otherwise. The Fourier transform of the uniform window is a sine-$x$-over-$x$ ("sinc") function,

$$W_U(f) = \int_{-T/2}^{T/2} e^{-j2\pi f t}\, dt = T\left(\frac{\sin \pi f T}{\pi f T}\right). \qquad \text{(II-3)}$$

Notice that $W_U = T$ at $f = 0$ and $W_U = 0$ at any integer value of $f/\Delta f$. (Since $\Delta f = 1/T$, the quantity, $f/\Delta f$, is also equal to $f \cdot T$.)

Multiplication of any time series by a window function in the time domain is equivalent to convolution[4] in the frequency domain. The convolution would be between the transform of the time series and the transform of the window function. At any specific frequency, $f_0$ (= $m\Delta f$), the result of that convolution is

$$X_m = \int X(f) \cdot W_U(f_0 - f)\, df\,. \qquad \text{(II-4)}$$

The left-hand side above is the *sampled* linear spectrum value at $m$, the index that corresponds to the frequency, $f_0$ (so that $m\Delta f = f_0$).

[4] Don't be put off by the word "convolution". We'll discuss convolution later; for now, accept that it's the appropriate mathematical operation.

Fortunately, we don't need to do this integral. We just want to see what the shape of the filter $W_U$ is. Dividing $W_U$ by $T$ normalizes the equivalent filter to a "passband" amplitude of one:

$$\frac{W_U(f_0 - f)}{T} = \frac{\sin \pi (f_0 - f)T}{\pi (f_0 - f)T} = \frac{\sin \pi (m\Delta f - f)/\Delta f}{\pi (m\Delta f - f)/\Delta f}. \qquad \text{(II-5)}$$

This function is one when $f$ is in the center of the $m^{\text{th}}$ bin ($f = m\Delta f$) and it is zero when $f$ is in the center of any other bin ($f = (m+m_x)\Delta f$ where $m_x$ is a non-zero integer). The equivalent filter has the shape of sin*(x)/x* with the peak at the center of the bin in question and zero crossings in the centers of all the other bins. This is why there is no "leakage" for a sine wave that is *centered* in one of the spectral bins; all the other bin filter functions have zeros at that frequency. If the sine wave is off-center, though, then *every* other bin contain some contribution from that sine wave. (This is spectral "leakage".) Likewise, for a signal with energy spread over some range of frequency, a bin value at a specific frequency, $f = m\Delta f$, has contributions from all other parts of the spectrum. The filter function does decay in overall amplitude with distance from the center but, for the uniform window, the decay is relatively slow.

*Window functions*

Another way of thinking about leakage is that it is produced by the discontinuity at the ends of the time record. Our use of sampled time series and the discrete transform are equivalent to repeating the single record in time over and over. If we've selected the record so that the series of repeating records introduces a discontinuity into the overall signal, then this discontinuity influences the spectrum.

For example, if the time series is a sampled sine wave and the frequency is an integer multiple of $\Delta f$, there will be no discontinuity if we were to repeat the record over and over and, therefore, no leakage. But, if the sine wave doesn't

begin and end at the same phase in its cycle in the record, then there will be a discontinuity in value every $T$ seconds.

We can reduce the impact of such discontinuities if we taper the ends of the time series to zero; even if there's a discontinuity in the original, repeated series of records, the tapered version won't have a discontinuity. Of course, this tapering operation has its own consequences.

Two commonly used windows[5] are the Hann (or Hanning or von Hann) window and the flat-top window. The equation[6] in the time domain for the Hann window is,

$$w_H(t) = 1 - \cos(2\pi t/T), \tag{II-6}$$

and the equation for the flat-top window[7] is,

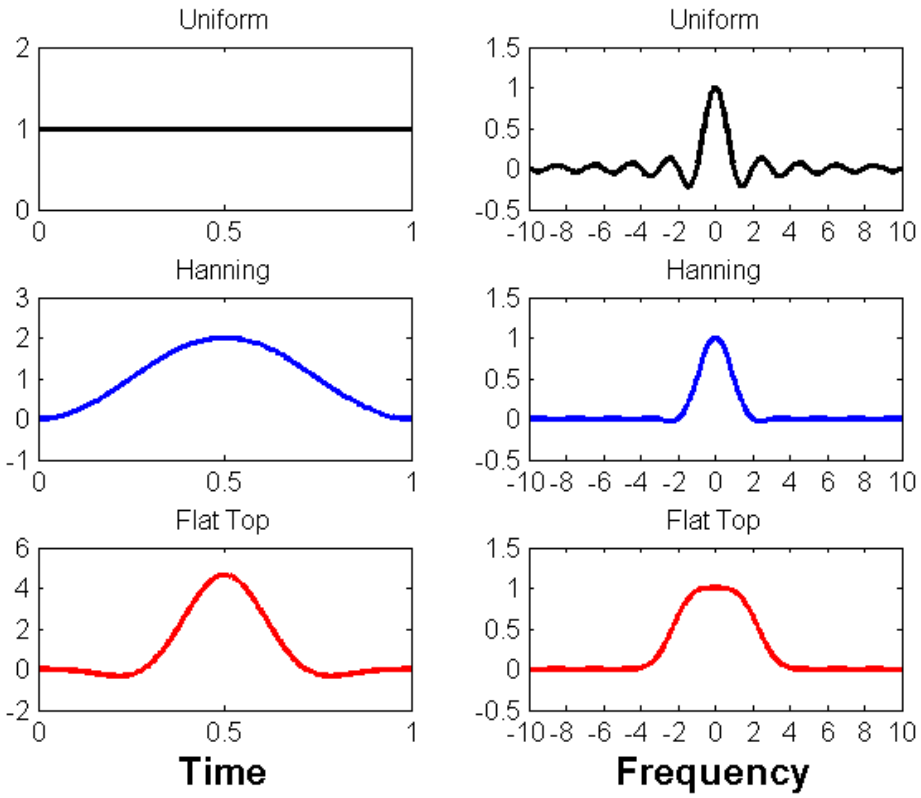
$$\begin{aligned} w_{FT}(t) = {} & 1 - 1.93\cos(2\pi t/T) + 1.29\cos(4\pi t/T) \\ & - 0.388\cos(6\pi t/T) + 0.0322\cos(8\pi t/T) \end{aligned} . \tag{II-7}$$

The time-domain shape and the shape of the equivalent bin filter for the uniform, Hann, and flat-top windows are shown here:

[5] There are many other windows but these two illustrate the most useful features of windows. See https://en.wikipedia.org/wiki/Window_function for an extensive list of windows and their properties. With the wide variety of windows available, you can easily spend too much time agonizing over the choice. The Hann window may not be optimal (depending on your application) but it is a safe choice—a reasonable starting point. I've included the flat-top window not as a recommendation but as an example of an extreme window.

[6] In some texts, you will see this equation with a factor of 0.5. Shortly, we will introduce the proper compensation for any scale factor in a window function.

[7] There is no standard definition for the flat-top window. The function definition that I've used here is the one used in Bruel and Kjaer Signal Analyzers.




The plots in the left column are the weighting (windowing) functions as they would be applied to the time series. The Hann and the flat-top both taper the time values to zero at $t = 0$ and $t = T$. Keep in mind that the sampled time series has samples at $t = m*\Delta t$, where $m$ runs from 0 to $N$-1. Consequently, the last point in the sampled time series is at $t = (N\text{-}1)* \Delta t = T - \Delta t$. Properly implemented[8], the window function forces the first point to zero *but not the last point*. The last point is one $\Delta t$ short of $T$; the point that represents $T$ is the *first point in the next periodic repetition* of the time series.

Notice how much of the time series is reduced in amplitude by the flat-top window compared to the other windows.

[8] Here is another instance where care is required in using packaged routines. The MatLab Signal Processing Toolbox has windowing functions but the default is for zero values at both ends. In the implied periodic repetition of the sampled series, that would be equivalent to setting two adjacent points equal to zero – the last point of a record and the adjacent first point of the next periodic repetition. While this makes little difference when $N$ is large, for small $N$ the errors can be substantial. The MatLab window functions have an option called "periodic" that returns a window function properly matched to a sampled time series.

The plots in the right column are the effective filter shapes in the frequency domain. The tick marks on the horizontal scale are index numbers for the centers of the bins (of $G_{XX}$, for example, although the plots are of the filter amplitude response, not amplitude squared). The flat-top window equivalent filter is virtually flat over the entire central bin (± 0.5 on the horizontal axis) and includes very little power from bins beyond about ± 5, but the main "lobe" of the equivalent filter is *much* wider than either of the other two equivalent filters.

[Note: In writing the equations for the window functions, I picked the amplitude scaling so that the peak of the equivalent filter response would be one after normalizing by $T$. You may see these windows written with other scaling – a factor of 0.5 multiplying the Hann window equation, for example. We'll take care of these scaling issues later.]

*Equivalent filter shapes for DFT bins*

As we did for the uniform window, the equivalent filter shape for a frequency bin of the discrete Fourier transform can be found by taking the Fourier transform of the time-domain window function that was applied prior to the DFT. The equations for these equivalent filter shapes for the three windows – uniform, Hann, and flat-top – are given here.

Define a non-dimensional frequency offset from bin center,

$$\alpha \quad = \quad \frac{f \; - \; f_i}{\Delta f}, \tag{II-8}$$

where $f_i$ is the bin center frequency of the bin and $\Delta f$ is the nominal DFT bin width ($1/T$). The normalized frequency, $\alpha$, is 0.0 if the frequency is at the bin center, 0.5 if the frequency is at the upper bin edge, and 1.0 if the frequency is actually in the center of the next higher bin. Also define

$$S_k \quad = \quad \frac{\sin\left[\pi\left(k+\alpha\right)\right]}{\pi\left(k+\alpha\right)} \quad . \tag{II-9}$$

In terms of the normalized frequency, $\alpha$, the filter function generated by the uniform ("rectangular" or "boxcar") window is

$$H_{unif}(\alpha) \;=\; S_0(\alpha). \tag{II-10}$$

For the Hann ("Hanning") window, the equivalent filter shape is

$$H_{hann}(\alpha) \;=\; S_0 \;+\; \frac{1}{2}\left[S_1 \;+\; S_{-1}\right], \tag{II-11}$$

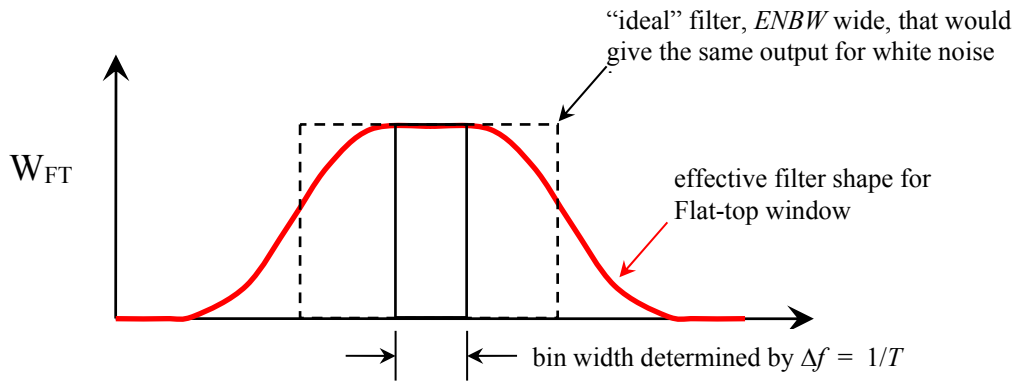
and for the flat-top window,

$$\begin{aligned} H_{flat}(\alpha) \;=\; S_0 \;+\; \frac{1}{2}\Big[1.93\left(S_1 \;+\; S_{-1}\right) \;+\; 1.29\left(S_2 \;+\; S_{-2}\right) \;+ \\ 0.388\left(S_3 \;+\; S_{-3}\right) \;+\; 0.0322\left(S_4 \;+\; S_{-4}\right)\Big] \end{aligned} \;. \tag{II-12}$$

*Equivalent noise bandwidth*

Because the central region of the flat-top equivalent filter is so much wider than that of the uniform equivalent filter, each bin in $G_{XX}$ for a flat-top windowed time series would contain more power if the time series was, for example, white noise.

One way to quantify this effect is to calculate the *equivalent noise bandwidth* (*ENBW*) of the equivalent filter. If we pretend that the filter is operating on white noise, we can figure out how wide a "perfect" rectangular filter bin would have to be to take in the same noise power:




If the response of the frequency-bin filter[9] is $W(f)$ and you pass white noise with a spectral amplitude of one through that filter, the time-domain mean-square value would equal the integral of the spectral density associated with the filter,

$$ms\left[w(t)\right] \;=\; \frac{1}{T}\int_0^T w^2(t)\,dt \;=\; \int_{-f_s/2}^{f_s/2} \frac{W^* W}{T}\,df \quad , \tag{II-13}$$

The "ideal" filter (the black dashed rectangle in the figure above) has a constant equivalent spectral density of $W(0)^*W(0)/T$ over the width, which we'll call the Equivalent Noise Bandwidth (*ENBW* in Hz), and is zero outside that range. Consequently, the corresponding mean-square value is equal to the area under that equivalent spectral density,

$$\frac{1}{T}\int_0^T w^2(t)\,dt \;=\; \frac{W(0)^2}{T}\,ENBW \quad . \tag{II-14}$$

The peak value, $W(0)$, is simply related to the average value of the time-domain function by the Fourier transform at zero frequency,

$$W(0) \;=\; \int_0^T w(t)\,dt \;=\; T\;avg\left[w(t)\right] \quad , \tag{II-15}$$

[9] The frequency response of a filter (or any system for that matter) is the linear spectrum of the time-domain impulse response. We'll consider impulse response later; for now, treat the frequency response, $W$, as a complex linear spectrum from which we can determine a spectral density ($W^*W/T$).

Combining these last two results, solving for *ENBW* and replacing one *T* factor with $\Delta f$ gives,

$$\frac{ENBW}{\Delta f} = \frac{ms[w]}{\left(avg[w]\right)^2} = \frac{\frac{1}{N}\sum w_n^2}{\left(\frac{1}{N}\sum w_n\right)^2} \quad . \qquad \text{(II-16)}$$

Therefore, the ratio of the equivalent noise bandwidth to the actual bin width is equal to the ratio of the mean-square of $w_n$ to the square of the mean of $w_n$. This ratio is fixed for any given window. For our three "model" windows, these ratios are:

| | | |
|---|---|---|
| uniform | → | 1.00 |
| Hann | → | 1.50 |
| Flat-top | → | 3.77 |

Historically, the flat-top window has been recommended when the objective is to determine accurately the amplitude of a sine wave. This is only true if the sine wave is not at one of the bin-center frequencies of the transform and if the signal-to-noise ratio is large. The flat-top window has a very large noise bandwidth so, if the signal is noisy, the amplitude estimate may be corrupted by noise. If the sine wave frequency is equal to one of the frequency-bin frequencies, then there is no amplitude error (the amplitude error is the result of the shape of the equivalent filter) so using a Hann or uniform window would generate a better estimate with a noisy signal[10].

[10] You should always be aware of assumptions. If you understand the underlying assumptions of a process, then you may be able to improve your strategy based on factors in your measurement that you can control. If you have no control over the sine wave frequency, the flat-top window may indeed give a more accurate amplitude estimate but, if you do have control over the frequency, you can do better with another window (and with synchronous averaging).

## Spectral Density for Windowed Time Series

With time-domain window functions, we can reduce the impact of the implied periodicity in sampled and transformed functions. We do need to be able to predict the effects of these tapering or weighting functions in the frequency domain, though.

In particular, we need to know how to calculate spectral density so that the "rule" for $G_{XX}$ still holds: an integration of $G_{XX}$ over frequency must still give the mean-square value (the "power") of the original (*before* weighting) time-domain waveform. We can find a single correction factor for the spectral density but it will only be strictly valid if the signal's characteristics (its "statistics") don't change with time. If the signal, for example, is white, random noise with a more or less constant mean-square value, then the time-domain window will always "extract" a segment with similar properties no matter where in the time series it is applied. If, on the other hand, the signal is a short burst of random noise, then the alignment of the time-domain window with the burst can have a substantial effect on the result.

For the moment, we'll assume that the signal's characteristics don't change over the analysis period. With this assumption of "stationarity" the simplest way to find the required correction factor for the spectral density is to operate on a "DC" or constant value in the time domain:

Let $x_n = 1$ for $0 \le n \le N-1$ (that is, for all $n$). The mean-square value of this $x_n$ is 1.

Let $w_n$ be the weighting function so that $y_n = w_n \cdot x_n$ is the new, weighted ("windowed") series. We'd like to find the correction factor, $A_W$, that we can apply to the spectral density, $G_{YY}$, of $y_n$ so that the resulting spectral density integrates properly to the mean-square value. Call the correction factor, $A_w$, and the corrected spectral density, $G_{W\text{-}XX}$, so that

$$G_{W-XX} = A_W \cdot G_{YY} . \qquad \text{(II-17)}$$

Since $x_n = 1$, $G_{YY} \equiv G_{WW}$, where $G_{WW}$ is the spectral density of the window function, $w_n$. We want the integral (actually, the sum times $\Delta f$) of $G_{W\text{-}XX}$ to be equal to the mean-square value of $x_n$ (that is, equal to one) so,

$$1 = \sum_m A_W \cdot G_{YY} \cdot \Delta f = A_W \sum_m G_{WW} \cdot \Delta f . \qquad \text{(II-18)}$$

But, from the discrete form of Parseval's Theorem,

$$\frac{1}{N}\sum_n w_n^2 = \sum_m G_{WW} \cdot \Delta f . \qquad \text{(II-19)}$$

Therefore,

$$1 = \frac{A_W}{N}\sum_n w_n^2 \quad \text{or} \quad A_W = \frac{1}{\frac{1}{N}\sum_n w_n^2}. \qquad \text{(II-20)}$$

The correction factor, $A_W$, is the reciprocal of the mean-square value of the window function. In order to apply a weighting function and still generate the proper spectral density:

1) Generate the weighting function, $w_n$. (The amplitude scaling of $w_n$ doesn't matter; the correction factor, $A_W$, takes care of that.)

2) Calculate the spectral density correction factor, $A_W$.

3) Apply the weight function to the time series: $y_n = w_n \cdot x_n$.

4) Calculate the spectral density: $G_{W-XX} = A_W \cdot G_{YY}$.

In this section, I've used a special notation to indicate the spectral density produced from a windowed time series. From this point on, I will drop the *W-XX* subscript on *G*: when you see $G_{XX}$, it may or may not be generated from a windowed time series but you can assume that it has been corrected, if necessary,

for any time-domain windowing. *Make sure that, if you use a time-domain window, you correct the spectral density properly*.

Example MatLab code: A better window function

Although I do not recommend using packaged functions blindly, you can adapt packaged functions once you understand them. For example, if you have the MatLab Signal Processing Toolbox, you can write a function around the packaged hann function to use the proper option and to include the missing features. The following function generates the window and also returns the spectral-density correction factor and the noise-bandwidth factor.

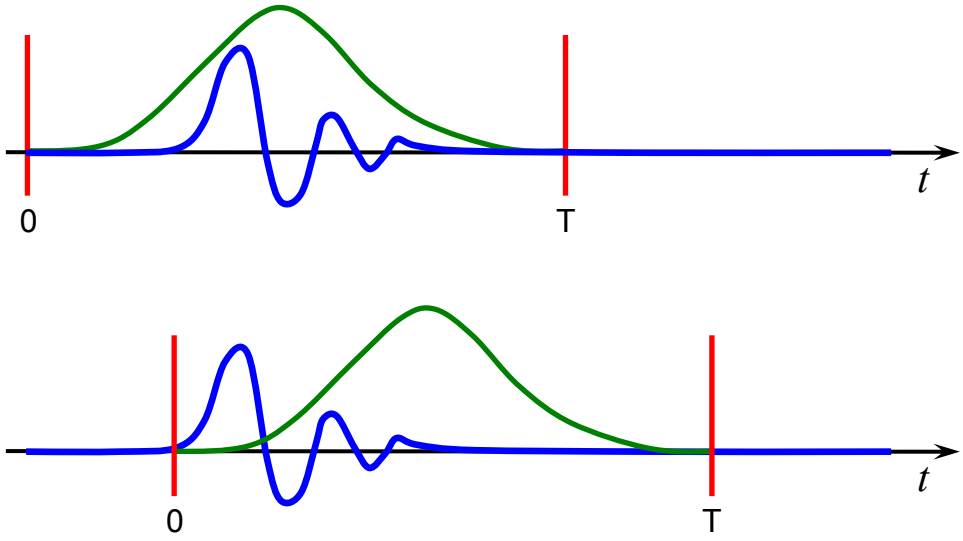
```
function [W, Aw, nbw_factor] = Hann_window(N)
%      Generates a Hann window function with N points.  Also returns the
%   spectral-density correction factor, Aw, and the effective noise
%   bandwidth factor, nbw_factor.  The effective noise bandwidth is
%   df*nbw_factor.
%
%   Usage:  [W, Aw, nbw_factor] = Hann_window(N);  %  to get all options
%   Usage:  [W, Aw] = Hann_window(N);
%   Usage:   W = Hann_window(N);   %  for window only
%

W   = hann(N, 'periodic');   %  use MatLab's hann function properly!
ms  = sum(W.*W)/N;           %  mean-square of W
avg = sum(W)/N;              %  average of W
Aw  = 1/ms;                  %  spectral-density correction factor
nbw_factor = ms/(avg^2);     %  noise-bandwidth factor
```

## The approximation associated with $A_w$

The spectral density correction factor, $A_w$, was derived above by assuming a constant-value time series. If we had worked instead with a random-noise time series, we would have discovered that *this factor makes the integral of the spectral density only **approximately** equal to the mean-square value of the random-noise time series*. If the random-noise time series has statistics that are constant over the interval of the spectral density calculation (i.e., "stationary"), we would find that the approximation is quite good. If, on the other hand, the statistics of the time

series change significantly over the interval of spectral density calculation, there can be a substantial difference between the integral of the corrected spectral density and the mean-square of the unwindowed time series.

This should not be a big surprise if you think about the action of the time-domain window. Consider a short burst of noise in the time domain. The burst has a certain mean-square value. If we take a block of time that's larger than the duration of the burst and then apply a time-domain window, the resulting spectrum depends on the location of the burst with respect to the window. If the burst is centered in the record, then a window like the Hann window would have little effect. If the burst is at one edge of the record, then the Hann window would reduce the amplitude of the burst dramatically. The spectra from these two cases would be substantially different but, in each case, the pre-windowed mean-square value is the same. These two cases are shown in the diagram below:




The signal is a short burst (blue). In the upper trace, the Hann window (green) is applied so that the burst is centered in the window and the amplitude of the burst will not be affected much. In contrast, the lower trace has the window applied in such a way that the burst would be reduced considerably in amplitude. Clearly one correction would not be appropriate for both cases. Furthermore, the correction derived earlier is probably inappropriate in both cases!

The spectral density window correction factor contains the underlying assumption that the signal's characteristics do not change over the entire extent of the record. For the noise burst this is not true. If we compared the integral of the "corrected" spectral density to the mean-square value of the time series prior to windowing, we might see large differences depending on the location of the burst with respect to the window function. On the other hand, if we computed the mean-square of the time series *after* windowing and compared that to the integral of the spectral density *before* application of the window correction factor, the results would be exact to within MatLab's ability to perform the arithmetic – that's Parseval's theorem. But the $A_w$ factor is based on an assumption that is never exactly true unless the time-domain signal is a constant so we can't count on the same degree of agreement using the pre-windowed mean-square value and the integral of the corrected spectral density. For stationary random noise, the correction is fairly accurate and the assumption of stationary, random noise underlies the usual textbook derivation of the spectral density correction[11].

*An exact (but impractical) approach*

If we work out the derivation more carefully, we would start with an arbitrary time series, $x_n$, and a window function, $w_n$, so that the windowed time series, $y_n$, is

$$y_n \quad = \quad w_n \cdot x_n \,. \tag{II-21}$$

The mean-square of the time series before windowing is

$$ms\left[x_n\right] \quad = \quad \frac{1}{N}\sum_n x_n^2 \,, \tag{II-22}$$

and the mean-square after windowing is

$$ms\left[y_n\right] \quad = \quad \frac{1}{N}\sum_n w_n^2 \, x_n^2 \,. \tag{II-23}$$

[11] In fact, many texts do not even mention this issue with the spectral-density correction factor.

By Parseval's theorem, the mean-square after windowing is equal to the integral (the sum times $\Delta f$) of the spectral density of $y_n$:

$$ms\left[y_n\right] \;=\; \sum_m G_{YY}\cdot\Delta f \;. \tag{II-24}$$

In order to make this expression equal to the mean-square of the original time series, multiply by the ratio of the mean-square of $x_n$ to the mean-square of $y_n$:

$$ms\left[x_n\right] = \;\frac{\frac{1}{N}\sum_n x_n^2}{\frac{1}{N}\sum_n w_n^2\, x_n^2}\cdot\sum_m G_{YY}\cdot\Delta f \quad . \tag{II-25}$$

Consequently, the exact correction factor would be

$$A_W^{exact} = \;\frac{\sum_n x_n^2}{\sum_n w_n^2\, x_n^2}\,. \tag{II-26}$$

If $x_n$ is a constant, this expression reduces to the usual form:

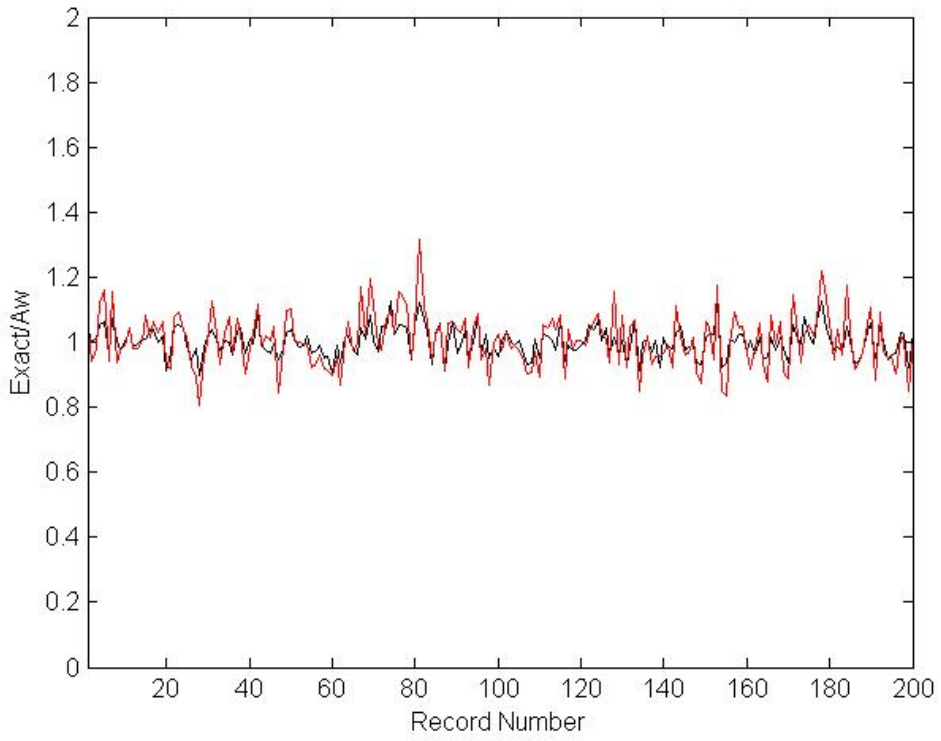
$$A_W = \;\frac{1}{\frac{1}{N}\sum_n w_n^2}\,. \tag{II-27}$$

The exact correction factor, though, is a function of the details of the time-domain signal and this makes the more accurate expression impractical. We also want our correction to be compatible with the spectral density correction used in commercial signal analyzers, which is the approximate form. While you could, in principle, calculate a spectral density correction for each record, this is never done in practice (to my knowledge). If the correction factor were different for each individual record, measurements would be difficult to interpret.

In this course, we will follow the acoustics and signal processing communities and use the common definition (II-27) for $A_w$. However, recognize the approximation involved and realize that the mean-square value of an unwindowed time series will no longer equal exactly the integral of the corresponding corrected spectral density of the windowed time series.

*An example*

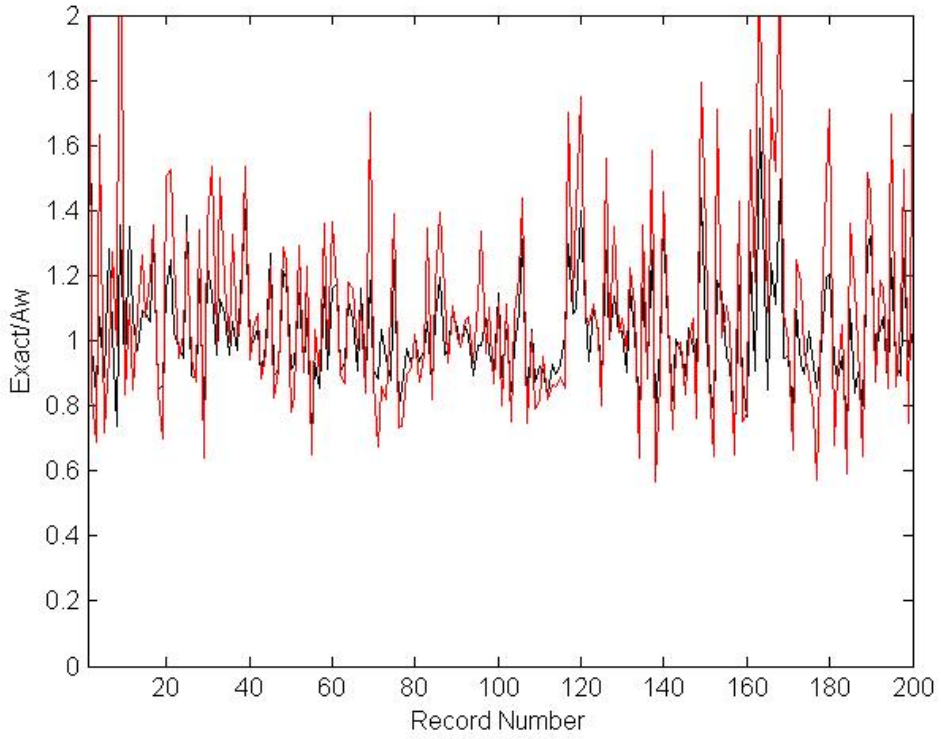
As an illustration of the approximation associated with $A_w$, I generated a random-noise time series and computed the spectral density record-by-record for 200 records. For each record, I computed the exact correction factor and then I plotted the ratio of this exact result to the standard definition for $A_w$. I did this for both the Hann window and the flat-top window. The results are in the plot below:




The black curve is for the Hann window; the red curve is for the flat-top window. For the most part, the error is less than ten percent for the Hann window and somewhat larger for the flat-top window. If the results were averaged over all 200 records, the $A_w$ factor would produce results virtually identical to those with the

exact correction. So, for averaging of random noise (having statistics that don't change with time), the $A_w$ factor is fine. (Also, bear in mind that if you haven't windowed in the time domain, then there is no correction to the spectral density and no error to worry about.)

As a second illustration, I used measurements of the sound of a race car passing a stationary observer. I used both a Hann window (black curve) and a flat-top window (red curve) for the plot below:



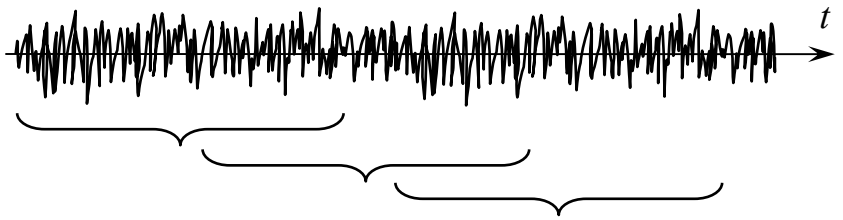

Notice that the "error" is much larger. Although this signal is noise-like, its statistics change significantly over the time associated with 200 records. It's also interesting to note that there appears to be a bias – the exact value seems to be more likely higher than lower compared to $A_w$.

## Windows: Their Consequences

Keep in mind what a window (or weight) function does in both the time and frequency domains:

*Time Domain*: The time sequence is tapered toward the ends so,

- Information near the ends of each record is lost. You may want to **overlap** your choice of records in the time domain to make up for this loss:




  An overlap of 50% is typical for the Hann window.

- A transient event or a short pulse that fits entirely within a single record may not need windowing. If a window is used, make sure that important characteristics of the pulse are not suppressed by the window function. Plot the unwindowed and the windowed pulse to compare. Also, the spectral-density correction factor may not be applicable.

*Frequency Domain*: The more severe the tapering in the time domain, the wider the effective bin filter is in the frequency domain. But, at the same time, "leakage" from components far from the bin in question is reduced.

- A larger Equivalent Noise Bandwidth (*ENBW*) means that the measurement of, for example, a sine wave is noisier: of the windows discussed above, the flat-top window gives the worst signal-to-noise for sine wave measurements.

- A larger *ENBW* can "smear" features like very high Q resonances in a system response measurement[12].
- A large sine wave component (even if it's 60 Hz interference!) can obscure a large part of the spectrum from "leakage" if no window ("uniform") is used.
- Small tonal components only a few bins apart may not be resolved if the *ENBW* is large.

Exercise 2.1: Recording Signals, Windowing, and Overlapping

This exercise combines averaging, windowing, and overlapping.

Write a MatLab script to record 10 seconds of data from the internal or external microphone of your computer (see the appendix on recording at the end of this chapter). Plot the recorded time series. The usual maximum range for computer-recorded files (and wav files) is ±1. If the record volume is set too high, the signal will be clipped; if the volume is set too low and the signal is far below the ±1 range, the system noise could corrupt your measurement. You can use the time-domain plot as a guide to adjusting the record volume. Call the units digitizer units (DU) or working units (WU) – your choice but you must have units.

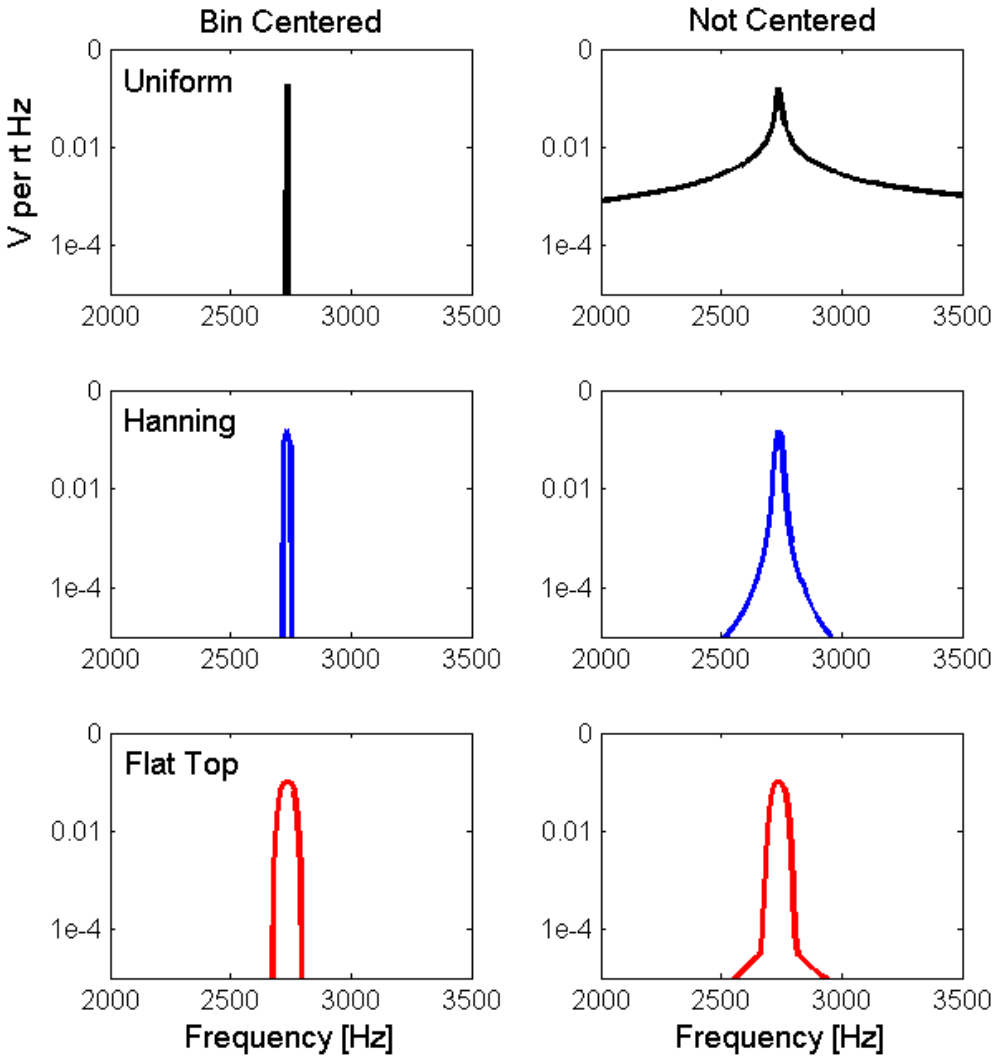
Select a window length ("record length") and then compute and plot the averaged spectral density for that 10-second recording. What would the units be? You should try this for a few different record lengths – short records, many averages; long records, few averages. First, try the averaging with no time-domain window (that is, use the "uniform" window) and no overlap. When you are sure that is working properly (how would you check?), modify the code to use the Hann window and 50% overlap. Don't forget the correction factor for the spectral density when using a time-domain window.

## Amplitude error in sine-wave measurements

Another aspect of windowing and its impact on spectral estimates is the potential for amplitude error when measuring sine-wave components that are not exactly at bin frequencies (i.e., do not have an integral number of cycles in each record). Shown below are the root-spectral density results, $\sqrt{G_{XX}}$, with units of

[12] Measurements on systems that have resonances that are very lightly damped (high-Q resonances) must be made carefully. Roughly, Q is the number of oscillatory cycles required for the system to respond to an input

volts per root hertz, for a "bin-centered" sine wave (on the left) and for a sine wave that is not bin centered (on the right). The bin-centered frequency is $200.0 \cdot \Delta f$ and the non-centered frequency is $200.3 \cdot \Delta f$. ($\Delta f$ is 13.672 Hz for this example.)




(The vertical scales on the root-spectral density plots are logarithmic. The same scales are used for all plots.)

Using the single largest value of the spectral density and the appropriate effective noise bin width, *ENBW*, to find the *rms* value (which, for the unit-amplitude sine waves, should be 0.7071) gives:

| | Centered | Not Centered |
|---|---|---|
| UNIFORM | 0.7071 | 0.6065 |
| HANN | 0.7071 | 0.6670 |
| FLAT-TOP | 0.7071 | 0.7071 |

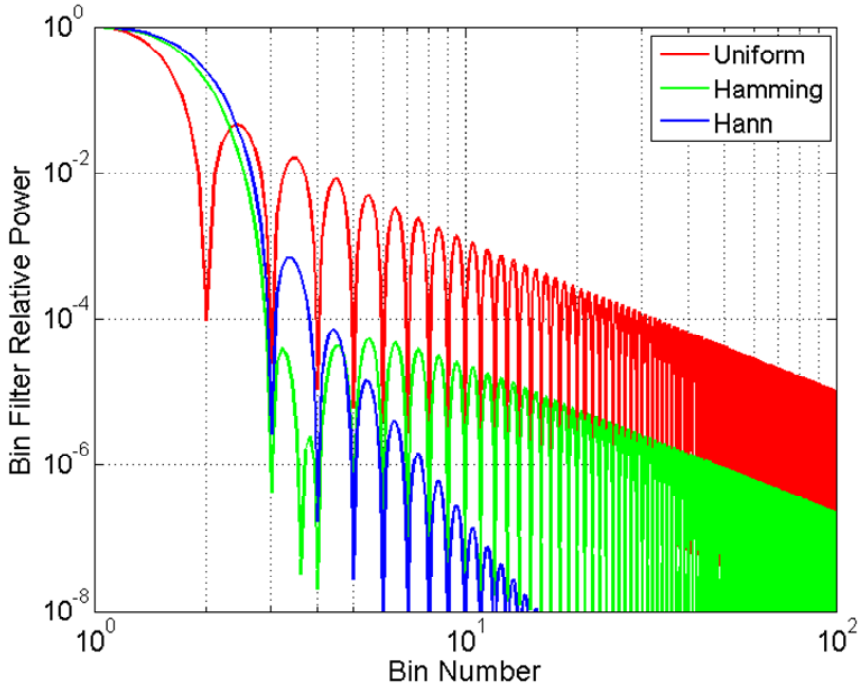
From the figure, notice the extent of spreading ("leakage") to other bins for the non-centered sine wave for the different windows. Also notice that, if the sine wave is bin-centered, the measurement of *rms* level based on the peak bin is the same (and correct) for all three windows[13]. For the non-centered sine wave, there is considerable error in the *rms* estimate for the uniform window and less, but still significant, error for the Hann window. (Shortly, we'll consider better ways to estimate the amplitude of a sine wave.)

Discussions of spectral leakage are often confined to the effects that result from a single, strong tone in the spectrum. If a uniform window (that is, no window) is used, the leakage from a single, strong line can raise the apparent noise floor of the spectrum and prevent seeing structure in the weaker parts of the spectrum. While it is important to understand the leakage associated with strong tones, there is another circumstance that is often overlooked in which leakage can contaminate a measurement.

In many measurements, the background ("ambient") noise increases in magnitude with decreasing frequency. For outdoor sound at very low frequencies, the spectral density is roughly proportional to $f^{-2.5}$. There may be no strong tonal components in the background noise; however, with a poor choice of window, the spectral leakage from the high levels at low frequency may decrease more slowly with frequency than the environmental noise. Details in the high-frequency spectrum may be obscured and the analysis noise floor may be higher than expected.

[13] It is often said that the flat-top window gives the most accurate measure of a sine wave. That's only true if the sine wave is not centered in the bin. For a bin-centered sine wave, the uniform of Hann windows give more accurate measures if noise is present because of their smaller noise equivalent bandwidths.

We can use the fact that the effective filter shape of a time-domain window function is the Fourier transform of that window to examine the spectral leakage well away from a particular bin in the spectrum. The following plot shows the effective filter function for three windows: uniform, Hann, and Hamming[14].




In order to emphasize the behavior far from the bin center, the horizontal axis is logarithmic in bin number (where the bin center is at one—the far left). The leakage in spectral density for the uniform window (red) drops as $f^{-2}$ while the leakage for the Hann window (blue) drops as $f^{-6}$. The spectral leakage for the Hann window drops much faster than typical atmospheric low-frequency ambient noise so the Hann window would be a safe selection for those measurements. The Hamming window (green) is a minor modification of the Hann window—a small constant value is added to the Hann window—to reduce the level of the first sidelobe: compare the green and blue curves between bin numbers 3 and 4. This small modification, though, results in a frequency dependence of the leakage equal to that of the uniform window. Sometimes, using a more "sophisticated" window makes the analysis worse!

[14] A special-purpose window function that does not taper all the way to zero value at record boundaries.

## Single-Tone Calibration[15]

Because of the amplitude accuracy of the flat-top window for sine wave components, the flat-top window can be useful for accurate determination of calibration signals. Consider, as an example, "pistonphone" calibration of a measurement microphone – a common procedure in acoustics. The pistonphone produces a sinusoidal acoustic pressure with well-controlled amplitude. The frequency is not so well controlled – it may only be within a few hertz of the design frequency so you can't count on the sine wave being centered in a particular frequency bin.

For calibration, a segment of data is recorded with the calibrator attached. In almost all applications, the sine wave would be far above the system noise floor (so the large effective noise bandwidth of the flat-top window may not be an issue).

From the known acoustic pressure produced by the calibrator and the recorded voltage, a calibration factor (in pascals per volt for this example) is found. This factor is then applied to the actual acoustic data measured to convert the recorded voltage into acoustic pressure. The success of the calibration depends on an accurate determination of the *rms* voltage of the recorded calibration signal. In the next section, we'll consider several methods for making this estimation. For now, assume that we know that *rms* voltage.

Example: A particular pistonphone produces 30.0 pascals-rms at 250 Hz. By some method, you determine the recorded calibration signal to be 0.135 volts-rms. The calibration factor, $M_{250}$, is then

$$M_{250} = \frac{30}{0.135} = 222 \text{ pascals/volt}.$$

[15] The term "calibration" is being used loosely here. This is really just a check of the response of a sensor at a single frequency.

If the actual data (your recording of an aircraft flyover, for example) has significant power only in the range of frequencies for which the microphone frequency response is flat, then you can convert the time series data, $x_n$, from volts to pascals directly,

$$p_n = M_{250} \cdot x_n \quad . \tag{II-28}$$

If the measured data extends beyond the "flat" portion of the microphone's response, then the calibration is best applied in the frequency domain. The microphone's manufacturer may supply the variation in microphone response[16] with frequency, which we'll call the function, $M(f)$. As provided, this function is usually normalized so that its value at some reference frequency is one (which might be expressed as 0 dB). Find the spectral density of the recorded voltage and apply the calibration factor and the frequency response function:

$$G_{PP}(m\Delta f) = M_{250}^2 \cdot M^2(m\Delta f) \cdot G_{XX}(m\Delta f). \tag{II-29}$$

The resulting spectral density, $G_{PP}$, has the units of pascals-squared per hertz.

## Amplitude of a sine wave

Sometimes the objective of a measurement is to find the amplitude (or the mean-square or the root-mean-square) of a pure sine wave component. One important application is in the measurement of a calibration tone provided to a microphone or an accelerometer. There are a number of ways of doing this either in the time domain or in the frequency domain. In no particular order, these methods are described here.

For this discussion, we will be concerned primarily with high signal-to-noise (but not noise-free) single-frequency components as would be encountered in a

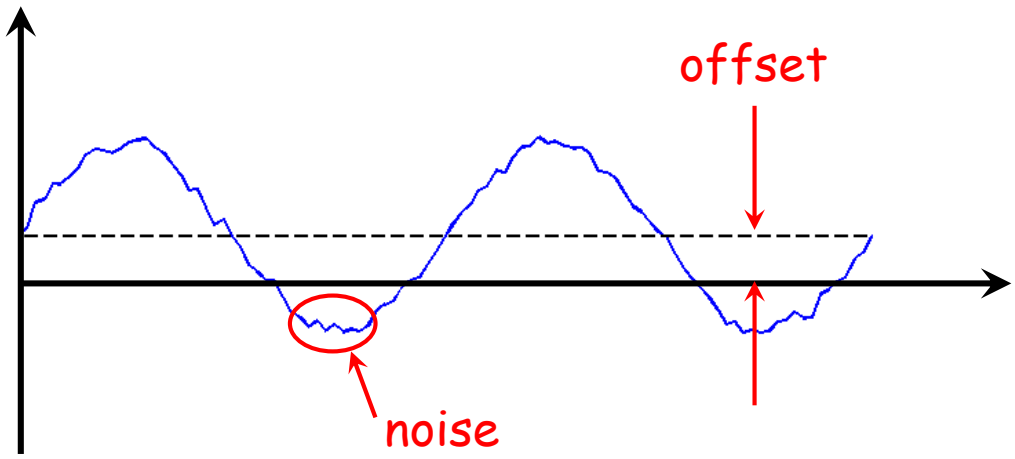
[16] The manufacturer usually supplies the inverse of $M$: the response in volts per pascal. Manufacturers typically provide only the magnitude of the response, not the phase.

single-frequency calibration. At the end of this section, I will discuss coping with noise in these measurements. Also, the goal of these analyses is to provide an estimate of the amplitude of the single-frequency sine wave. We can find the amplitude (the peak value) or the mean-square (*ms*) value or the root-mean-square (*rms*) value. The latter two can be converted to amplitude as long as the waveform is truly sinusoidal. For a sine wave, the *rms* value is equal to the amplitude divided by the square-root of two. (This relationship is only true for a sine/cosine function; don't try to convert between *rms* and amplitude for any other type of waveform.)

Methods A, B, C, and G are applied directly to the time series.

*Method A*

If the power in the single-frequency tone is considerably higher than the noise power, then the time series will look like a sine (or cosine) wave. In a real measurement, there will be some noise (some "fuzz" on the sine wave) and there will often be a static offset—that is, the average value may not be zero. You need to recognize these aspects to reduce the error in your estimate of the sine-wave amplitude.

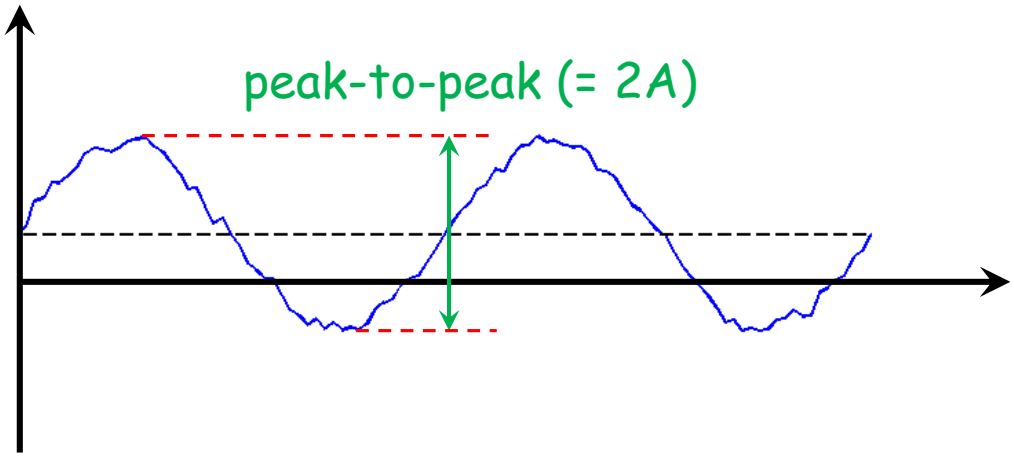



If the sine component is much larger than the noise and there is negligible static (or "DC") offset, then we can make a direct mean-square calculation on the time series. Significant noise or significant offset will cause this estimate to be

high. You can minimize the error associated with an offset by subtracting the mean value prior to calculating the mean square. Also, if the calculation extends only over a few cycles, make sure to perform the calculation over an integer number of cycles otherwise the fractional cycle will bias the mean-square calculation.

*Method B*

The amplitude can be extracted directly from the time series (again, as long as the noise component is small). Simply picking the peak value is not good practice, though. It's better to pick the maximum and the minimum value. If there is a DC offset, using the maximum and minimum values to find the peak-to-peak value eliminates the influence of the DC offset. The amplitude is then half the minimum-to-maximum (or peak-to-peak) value.




This method is an excellent first check on a sine-wave measurement. Since you're looking directly at the time-domain waveform, you can see how influential noise may be and you can see if the waveform is distorted or clipped. Even in the presence of noise, this technique can produce results with reasonable accuracy and can serve as a check on more sophisticated methods. Note: many of the other methods will not uncover distortion or clipping in the waveform. Distortion or clipping can introduce large errors if not accounted.

*Method C*

While we have not discussed filters yet, if you can imagine having a filter that passes energy only in a narrow range of frequencies around the frequency of the sinusoidal component, pass the time series through that filter first and then compute the mean-square of the filtered time series. The narrowband filter will remove any DC offset and reduce the noise power by rejecting noise that is not in the immediate vicinity of the sine signal. This method is capable of substantial accuracy as long as the signal to be measured is not too noisy and is fully within the passband of the filter.

*Method D*

This method has great flexibility and is capable of good accuracy for high signal-to-noise ratio signals. First, calculate the spectral density ($G_{XX}$, for example). If there is a sine wave with an amplitude well above the background noise, then there will be a peak in $G_{XX}$. That peak might be a single value (if there are an integer number of cycles of the sine wave in the record length, $T$) or the peak may consist of several neighboring values that are well above the noise. This peak in $G_{XX}$ represents the sine wave and we can determine the sine-wave amplitude from the value or values in that peak. The sine wave's mean-square value is equal to the sum of the values that make up the peak in $G_{XX}$ times $\Delta f$. (This is the equivalent of an integration over the region with the peak in contrast to an integration over all frequency.)

The root-mean-square (*rms*) value of that sine (or cosine) wave is then

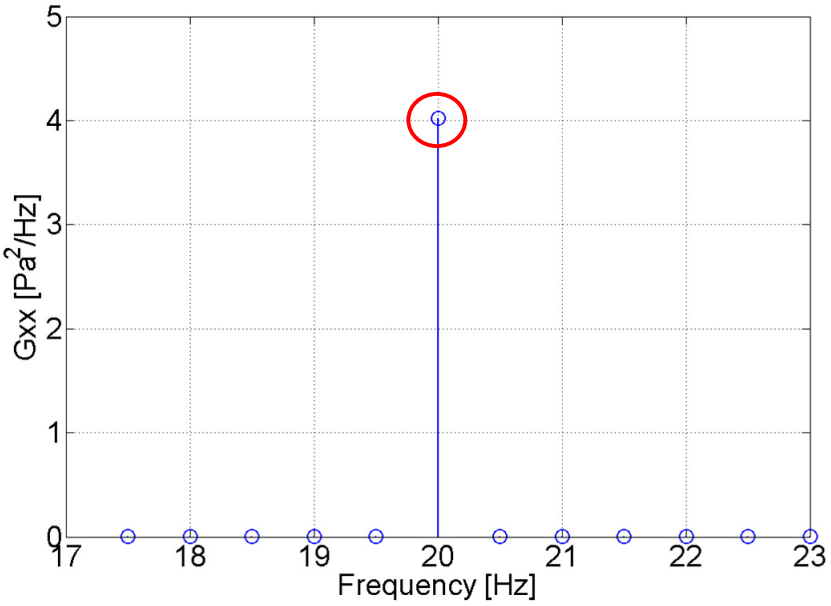
$$rms = \sqrt{\sum_{m=m_1}^{m_2} \left(G_{XX}\right)_m \cdot \Delta f} \quad , \tag{II-30}$$

where the summation is only over the values of $G_{XX}$ that make up the peak in the spectral density[17]. If the sine wave has exactly an integer number of cycles in the

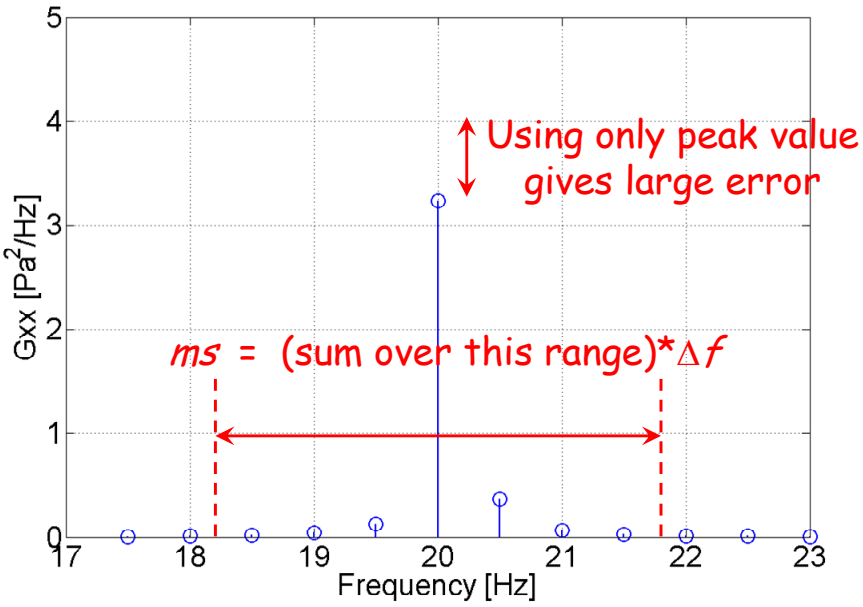
[17] If the peak is not very far above the background noise, then your result for the sine-wave amplitude will not be accurate. This technique is only accurate for sine waves that are well above the noise.

time record, then there will only be one large value of $G_{XX}$—the value at the frequency of the sine wave.

The next two figures give examples of the two cases: (a) a bin-centered sine wave (an integer number of cycles in the record), and (b) a sine wave with a frequency that is not a bin frequency. In these examples, $\Delta f$ is 0.5. In (a), the sine-wave frequency is 20.000 Hz; in (b), the sine-wave frequency is 20.125 Hz. For the sine wave at 20.000 Hz, virtually all of the "power" is in the peak of $G_{XX}$ at 20.000 Hz:




The mean-square of the sine wave is equal to the product of the maximum value of $G_{XX}$ times $\Delta f$, which gives 2.0 in this case. There is some error because there is some noise—I'm using the same slightly noisy sine wave as in the time-domain examples; however, only the noise in the single frequency bin affects the estimate. When f = 20.125 Hz, the power is distributed over a number of bins in $G_{XX}$. We can still find a reasonable estimate, though, by summing over a range of bins:




This process is more sensitive to noise in that several bins are summed. System calibration signals are often relatively clean, high signal-to-noise ratio signals so this method is still capable of excellent estimation.

If noise is present and we can make a reasonable estimate of the noise power, the estimate for the sine wave power can be corrected (see below for Subtracting Noise).

*Method E*

If you compute the power spectrum (see Power Spectrum above), you can use the peak value as the mean-square of the sinusoidal component. Commercial signal analyzers often provide this option. If the frequency of the sine component is not equal to one of the analysis bin frequencies, there will be an amplitude error. We will consider this issue in more detail in the next chapter.

Exercise 2.2: Amplitude of a Sine Wave in Noise

Pick some record length, $T$, and some sampling frequency, $f_s$. Generate a sine wave $T$ seconds long with an *rms* value of 1.00 working units (WU). Generate a random-noise waveform $T$ seconds long with an *rms* value of 0.20 WU. Add the two waveforms.

Use methods (A), (B), (D) and (E) above to estimate the *rms* value of the sine wave from the noisy waveform. Since you know what the actual rms value of the noise-free sine wave is, you can assess the different methods. Later in the course, we will consider the design of filters so that method (C) could be implemented.

*Method F*

This method is rarely used but is useful to know how to find the linear spectrum of a pure sine wave. This method not recommended unless the sine wave has an integer number of cycles in the time record. (Another way of saying this is that the sine wave frequency should equal one of the spectrum-bin frequencies.)

Let $x(t) = A\cos(2\pi f_0 t)$ so that the sampled time series is

$$x_n = A\cos(2\pi f_0 n\Delta t) \quad . \tag{II-31}$$

We'll also assume that $f_0 = m\Delta f$ for some integer, $m$. (Integer $m$ ensures that the sine wave is centered in a frequency bin.) The linear spectrum at $m$ is

$$X_m = \sum_{n=0}^{N-1} A\cos(2\pi m\Delta f\, n\Delta t)\, e^{-j2\pi\frac{mn}{N}}\,\Delta t \quad . \tag{II-32}$$

As before, the product $\Delta f\Delta t$ is $1/N$. With that substitution and expanding the complex exponential,

$$X_m = \sum_{n=0}^{N-1} A\cos\left(2\pi\frac{mn}{N}\right)\left[\cos\left(2\pi\frac{mn}{N}\right) - j\sin\left(2\pi\frac{mn}{N}\right)\right]\Delta t$$

$$= \sum_{n=0}^{N-1}\left\{A\frac{1}{2}\left[1 + \cos\left(4\pi\frac{mn}{N}\right)\right] - jA\cos\left(2\pi\frac{mn}{N}\right)\sin\left(2\pi\frac{mn}{N}\right)\right\}\Delta t \quad . \tag{II-33}$$

The terms with cosine or sine functions sum to zero, therefore,

$$X_m = A\frac{N\Delta t}{2} = \frac{AT}{2} \quad \text{or} \quad A = \frac{2X_m}{T}, \tag{II-34}$$

so the *rms* value would be

$$rms = \frac{A}{\sqrt{2}} = \frac{\sqrt{2}X_m}{T}. \tag{II-35}$$

Don't forget that the square-root-of-two relationship between the amplitude and *rms* value is ONLY true for a sine wave. (With what you know about the relationship of $G_{XX}$ to the *fft*, you should be able to show that the *rms* calculation by Methods D and F give the same answer as the *rms* calculation by Method A.)

When a sine wave has an integer number of cycles in the time record, the spectrum is markedly different from the more general case. If the frequency is an integer multiple of $\Delta f$, then the sine wave will satisfy this special case. Sampling of a finite time record is equivalent to repeating that same time record over and over and, if there are an integer number of cycles in the record, then there will be no discontinuity at the points of repetition. As discussed earlier, these are called "bin-centered" sine waves.

*Method G*

A powerful method sometimes used in precision measurements of sine waves is a model-fitting method. In this method, a cosine function is assumed with unknown amplitude, frequency, phase, and offset. Then these four parameters are adjusted to minimize the mean-square error between the model sine wave and the measurement data. The linear spectrum provides initial guesses for these parameters and an iterative method converges on the best estimates.

For estimation of a sine wave, the model function, $y_0(t)$, is[18]

[18] It's convenient to use a cosine as the model function rather than a sine because the phase of a linear spectrum value is zero for the cosine form. This is a minor point but makes estimation of the initial value for phase simpler.

$$y_0(t) \;=\; A_x \cos\left(2\pi f_x t \;+\; \phi_x\right) \;+\; B_x \;, \tag{II-36}$$

where $A_x$, $B_x$, $f_x$, and $\phi_x$ are the as-yet-unknown "fit" parameters. If $y(t)$ represents the measured data and the model function, $y_0$, is close to being correct, then the difference, $z$, between the two,

$$z \;=\; y \;-\; y_0 \;, \tag{II-37}$$

should be small. Since it is small, we can write this difference as a Taylor series expansion from the desired value (which is zero):

$$z \;=\; \frac{\partial z}{\partial A_x}\Delta A_x \;+\; \frac{\partial z}{\partial \omega_x}\Delta\omega_x \;+\; \frac{\partial z}{\partial \phi_x}\Delta\phi_x \;+\; \frac{\partial z}{\partial B_x}\Delta B_x \;. \tag{II-38}$$

The measured data, the model function, and the difference, $z$, are all time series with $N$ points. These functions are all $N$x1 matrices. The series expansion above can be re-written as a matrix equation with the following definitions:

A 4x1 parameter matrix,

$$\mathbf{P} \;=\; \begin{bmatrix} A_x \\ \omega_x \\ \phi_x \\ B_x \end{bmatrix} \;, \tag{II-39}$$

a 4x1 parameter-differential matrix,

$$\Delta\mathbf{P} \;=\; \begin{bmatrix} \Delta A_x \\ \Delta\omega_x \\ \Delta\phi_x \\ \Delta B_x \end{bmatrix} \;, \tag{II-40}$$

and an $N$x4 matrix of partial differentials of $z$,

$$\mathbf{dz\_dP} \;=\; \left[\frac{\partial z}{\partial A_x} \quad \frac{\partial z}{\partial \omega_x} \quad \frac{\partial z}{\partial \phi_x} \quad \frac{\partial z}{\partial B_x}\right] . \qquad \text{(II-41)}$$

(Note: z is an $N$x1 column vector so each partial derivative is also a column vector. The combination of four column vectors in the above expression gives the $N$x4 matrix.)

The series-expansion for $z$ can then be written as a matrix multiplication,

$$\mathbf{z} \;=\; \mathbf{dz\_dP} \cdot \Delta\mathbf{P} \;. \qquad \text{(II-42)}$$

This is equivalent to a set of $N$ equations in 4 unknowns. As long as $N$ is greater than 4, the equation set is over-determined but we can find a solution by minimizing the mean-square error. In linear algebra, this is done by finding the pseudo-inverse of **dz_dP**; however, MatLab (and other analysis packages) provides a convenient shortcut. By "dividing into" the equation[19] on both sides and then swapping sides,

$$\Delta\mathbf{P} \;=\; \mathbf{dz\_dP} \setminus \mathbf{z} \;, \qquad \text{(II-43)}$$

MatLab recognizes that, dimensionally, this is an improper matrix operation, so MatLab assumes that a least-squares solution is desired.

Once the increments, $\Delta\mathbf{P}$, for the parameters are found, the matrix of parameters, **P**, can be updated,

$$\mathbf{P} \;=\; \mathbf{P} \;-\; \Delta\mathbf{P} \;, \qquad \text{(II-44)}$$

the model function, $y_0$, can be re-calculated based on the new parameters, and the process can be repeated. Continue this iterative process until the mean-square error (between the measured data and the model function) reaches a steady value. The mean-square error is

[19] Note the "slash" rather than the "back slash" to indicate this "divide into" operation. This is MatLab language, not standard mathematical notation.

$$\varepsilon^2 \;=\; \frac{1}{N}\sum\left[y(n\Delta t) \;-\; y_0(n\Delta t)\right]^2 \quad , \tag{II-45}$$

where the summation is over all of the $N$ points. The model function is well behaved and, as long as the measurement is not overly noisy, this method converges rapidly as long as the initial guesses for the parameters are reasonably close.

The initial parameter values can be determined from the linear spectrum of the time series. Find the linear spectrum, $Y$, of the measured data, $y$. Find the value, $Y_{max}$, of the linear spectrum that has the largest magnitude and find the frequency, $f_0$, corresponding to that maximum value. The initial parameter values (the "guesses") are then,

$$\begin{aligned} A_x &= \frac{2}{T}\left|Y_{\max}\right| \\ \omega_x &= 2\pi f_x \\ \phi_x &= angle\left(Y_{\max}\right) \\ B_x &= \frac{1}{T}Y(1) \end{aligned} \quad , \tag{II-46}$$

where *angle* is the MatLab *angle* function (which returns the phase of a complex number in radians) and $Y(1)$ is the first value (the zero-frequency value) of the linear spectrum.

Also required are the four columns in the partial-derivative matrix, **dz_dP**. These come from the partial derivatives of $z$, which are also the partial derivatives of $-y_0$ since the measurement, $y$, has no dependence on the parameters. For the model function given above, the partial derivatives are:

$$\begin{aligned}
\frac{\partial z}{\partial A_x} &= -\cos\left(2\pi f_x t + \phi_x\right) \\
\frac{\partial z}{\partial \omega_x} &= A_x t \sin\left(2\pi f_x t + \phi_x\right) \\
\frac{\partial z}{\partial \phi_x} &= A_x \sin\left(2\pi f_x t + \phi_x\right) \\
\frac{\partial z}{\partial B_x} &= -1
\end{aligned} \quad . \tag{II-47}$$

Each of these is an $N$x1 column vector including the last one, which is a column vector of -1 values.

Iteration toward a minimum mean-square solution is a powerful technique and can be applied to other functions and other measurements. Estimation of an exponentially decaying sinusoid or estimation of a resonance should also be possible with suitable modification of the process described above.

## Subtracting noise

Measurement of the amplitude of a sine wave is straightforward (and discussed above) when the amplitude of any background noise is much lower. When the background noise is not negligible, the frequency bin that contains the sine wave also contains noise. The amplitude of the sine wave can be "corrected" by subtracting the background noise level.

Working with mean-square values has a practical advantage when treating noise. If the noise is not correlated with the signal[20] (say, the sine wave in the above discussion), then the signal power (mean-square value) and the noise power simply add. Altermately, the signal spectral density and the noise spectral density

[20] Correlation is covered in a later chapter. Frequently, the specific circumstances suggest whether noise is correlated with signal or not. Wind noise on a microphone is not correlated with exhaust emission noise from an aircraft flying overhead; traffic noise is not correlated with the sound from an air-conditioner fan on a building. Later, we'll consider a quantitative test for correlation but don't be afraid to think about your measurement!

add. This means that an estimate of signal "power" can be corrected for the presence of noise by subtraction of either mean-square values or of spectral density values.

The first problem is how to estimate the noise. This is often done by averaging the spectral density values in nearby bins that do not contain the signal. You must be careful to move far enough from the signal bin that spectral leakage from the signal is negligible. This estimate is easily corrupted by the presence of other signals. An alternate approach is to use the median value of the "noise bins" instead of the average value. Infrequent large values influence the average value but not the median value.

If we have control over the signal, the signal can be turned off. If the signal is intermittent, we can pick a period for noise estimation when the signal is absent.

Once the noise is estimated, the subtraction still requires care. As a result of normal fluctuations in the noise, the noise value may occasionally be larger than the signal-plus-noise value in the signal bin. In that case, the subtraction would produce a negative number—meaningless as a spectral density value. The usual "solution" is to simply reject negative numbers. *It should be kept in mind that this introduces a bias in the measurement* – every case in which the noise fluctuation produces a higher level in the noise estimate than in the signal bin is rejected and every case in which the noise fluctuation produces a lower level than the signal bin is accepted.

Be reasonable with your expectations, though. Even if the result of the subtraction is always positive, unless the noise power is significantly lower than the signal power, the uncertainty in the result (after subtraction) can be high and the answer unreliable.

## Zero padding

As discussed earlier, the constraint, $\Delta f \Delta t = 1/N$, which can be re-written in terms of the sampling frequency, $f_s$,

$$f_s/\Delta f = N, \qquad \text{(II-48)}$$

is a hard constraint. If a certain frequency resolution, $\Delta f$, is desired from a time series sampled at $f_s$, then there is no choice regarding the number of points, $N$, per time record.

If we take an $N$-point time series and add $N$ zeros to the end of that series, the new frequency resolution is $\Delta f/2$. While this implies that we can see more detail in the frequency domain, we haven't added any new information. The operation is called "zero padding" and the result is simply an interpolation in the frequency domain. In some cases, an interpolation may be useful.

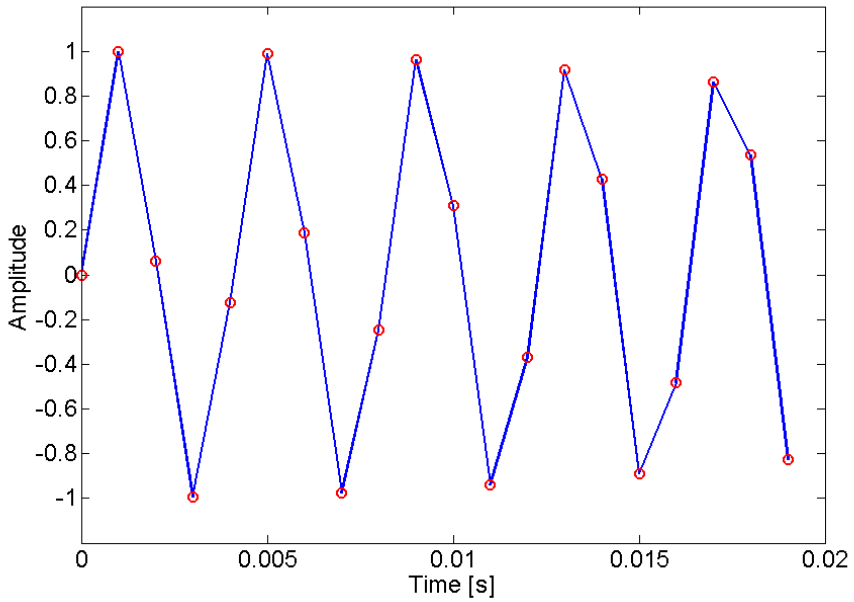
Another application for zero-padding a time series relates to the efficiency of the algorithm that computes the *fft* or *ifft*. The algorithm is most efficient when the number of points is an integer power of two. If we have many transforms to take (to construct a spectrogram, for example, or for spectral averaging over large numbers of records), it may be worth forcing the record length to be a power of two. If the original records are 800 points long, the process may run faster if we zero pad those records to make them 1024 ($2^{10}$) points long[21]. Keep in mind that the change in $N$ will change $\Delta f$ for a constant sample rate.

If we are extracting $N$-point records from a long, continuous time series, then we can pick $N$ to be 1024 rather than 800. In that case, zero padding would be pointless.

[21] Recognize, though, that modern *fft*/*ifft* routines are fast whenever N is "highly composite" (that is, if N can be factored into powers of small integers). Notice that $800 = (2^5)*(5^2)$, a highly composite number. Zero padding to 1024 points might give only marginal improvement in performance.

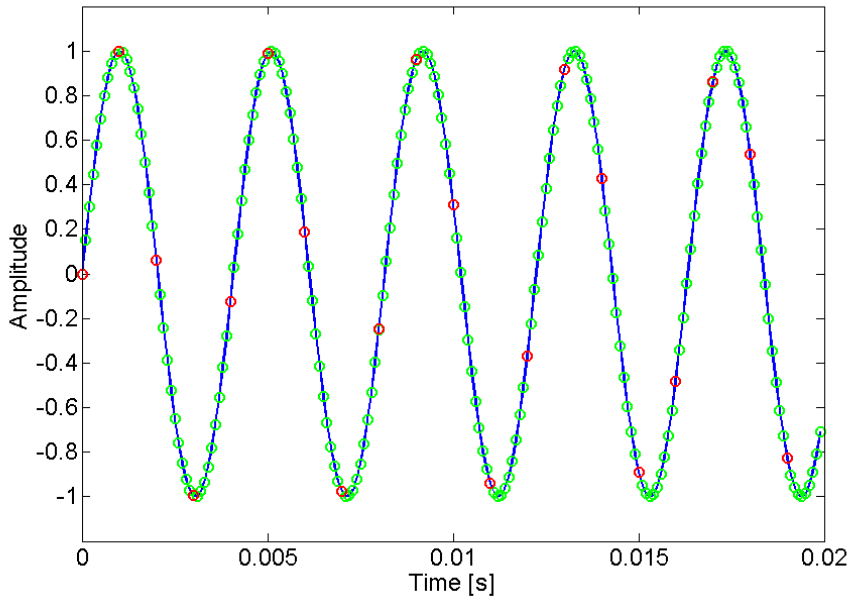
Another consequence of sampling is that we can only represent faithfully frequency components that are lower than half the sampling frequency. A sine wave with a frequency of 0.49* $f_s$ will be properly represented in the frequency domain and can be reconstructed exactly from the linear spectrum; a sine wave with a frequency of 0.51* $f_s$ will appear in the frequency domain as if the frequency axis had been folded at fs—the peak will appear at 0.49* $f_s$ and reconstruction in the time domain would result in a sine wave with the wrong frequency. This is aliasing.

However, *the sampling principle does not mean that the sampled version of the* 0.49* $f_s$ *hertz sine wave looks much like a sine wave*. The figure below shows the sampled time series for a 245 Hz sine wave sampled at 1000 samples per second. The actual sample points are shown as red circles.




Although the wave shape is far from sinusoidal, there is, in fact, enough information in the samples to define the sine wave.

If we generate the linear spectrum for this time series and zero-pad that linear spectrum[22] to increase its length by a factor of ten (i.e., increasing the sampling rate by a factor of ten), the time series that results from an inverse transform of this padded linear spectrum looks much more like a sine wave:




The original sample points are shown as red circles. These are duplicated in the inverse transform of the padded linear spectrum along with nine additional points in between each original sample. The curve looks much more like a sine wave now even though simply adding zeros does not add additional information about that sine wave.

[22] Zero padding a linear spectrum is more complicated than zero padding a time series. For the linear spectrum, the zeros must be inserted at the $f_s/2$ point near the middle of the linear spectrum. This process is described in the previous chapter.

## Final notes

While it is important to understand the processes described in this and other chapters, recognize that the goal is to make good measurements. An important aspect is that the correct numerical answer is important. If your conceptualization is correct but your implementation is not, you have not succeeded. Learn and practice techniques for checking your answers:

- use a second (and a third) approach to see if you get the same answer;
- use special values for which you know the answer;
- use very small data sets initially so you can "watch" every value;
- build your code in very small steps and check each step;
- write code for small test cases to evaluate operations before embedding those operations in a lengthy script;
- introduce "bad" values intentionally; and
- test, test, test.