# Windowing Functions Improve FFT Results, Part I

JUNE 1, 1998
BY RICHARD LYONS, TRW, SUNNYVALE, CA

COMMENT 1

*Also see Part II of the article .*

When your test requirements involve digitizing analog signals and then using an FFT to analyze their spectral content, you must become familiar with windowing functions. These simple functions improve the sensitivity of FFT spectral-analysis techniques. In the first part of this two-part series, you'll learn what windowing functions do and how you can apply them with fast Fourier transforms (FFTs). The second part of the series (scheduled for September) will cover additional window functions and their adjustable parameters.

Although scientists and engineers have used Fourier analysis techniques for many years to enhance frequency analyses, you may not fully appreciate how well Fourier analysis can extract frequency information from a signal. Consider a nonelectronic example described by Walter Munk, who was studying ocean turbulence:

*. . . we were able to discover in the general wave record a very weak low-frequency peak which would surely have escaped our attention without spectral analysis. This peak, it turns out, is almost certainly due to a swell from the Indian Ocean, 10,000 miles away. Physical dimensions are: 1 mm high, 1 kilometer long.*[1]

Only by performing careful frequency analysis on his data could Munk determine the presence of, and the characteristics of, such a small disturbance in a sea full of other waves.

### Windows Reduce FFT Leakage

By using windowing functions, you can further enhance the ability of an FFT to extract spectral data from signals. Windowing functions act on raw data to reduce the effects of the leakage that occurs during an FFT of the data. Leakage amounts to spectral information from an FFT showing up at the wrong frequencies. The people who first studied the effect thought of the spectral information as “leaking” into adjacent frequency values.

You can’t avoid leakage, but by applying windowing functions to your data prior to performing an FFT, you can reduce its ill effects. As you learn about windowing, you’ll also learn about how leakage arises and how it affects the results of an FFT.

If you could perform a Fourier analysis on a signal that goes on forever, the results would represent perfectly the frequencies and the amplitudes of the frequencies present in the signal. Computers, however, have limited space for data storage, so you usually acquire only a few hundred or a few thousand values.

When sampling a sine wave with an analog-to-digital converter (ADC), a computer acquires discrete samples, as shown in **Figure 1a** . Saving only 32 samples (**Fig. 1b** ) in effect multiplies an infinitely long sequence of sine-wave values by a sequence that contains 32 values, all equal to 1. As a result, you truncate—or window—the original sine-wave values by sampling during a finite period.

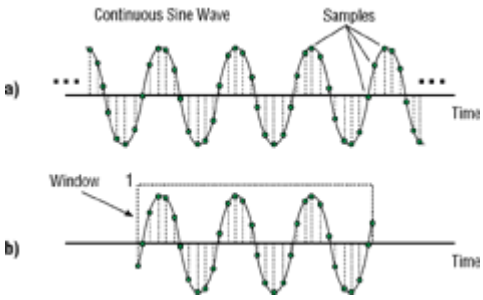


**Figure 1.** a) The dots represent the sine-wave samples acquired by an ADC. By storing a finite number of samples in memory, you effectively window the original signal. When you execute an FFT on the samples, you actually analyze the windowed samples shown in (b).

Next, you execute an FFT on the 32 sine-wave samples to see how well the spectral data from them compare with the theoretical spectral data from the original infinite sine wave. You may recall that multiplication in the time domain—the windowing operation—equals convolution in the frequency domain. But what does that mean?

It means that the frequency spectrum of the 32 sine-wave values is the convolution of the Fourier transform of the infinite-duration sine wave and the Fourier transform of 32 1's. The Fourier transform of the infinite sine wave yields a single value at the sine wave's frequency. The Fourier transform of the 32 1's yields the function, sin(x)/x. The convolution of

those two transforms is just the sin(x)/x function (**Fig. 2** ). Some authors call the continuous sin(x)/x curve the discrete-time Fourier transform (DTFT) of an all-1's sequence. All of the FFT results I will show come from a discrete sampled version of this DTFT curve.

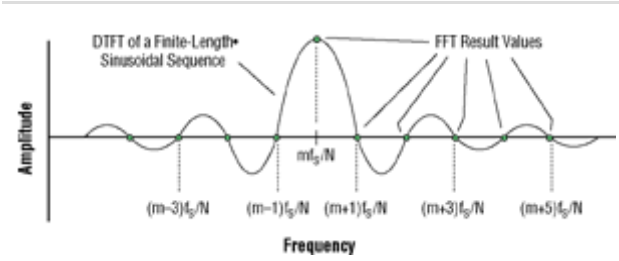


**Figure 2.** The amplitude curve represents a discrete-time Fourier transform (DTFT) of sine-wave samples that have *m* cycles inthe sample interval. The value of *m* is an integer and $f_s$equals the sample frequency. The discrete points represent the results of performing an FFT on the sine-wave samples.

#### An FFT Distorts Spectral Data

So, even though you acquired 32 samples from a pure sine wave, an FFT of these data produces spectral values "riding" on a curve, not a single point at a single frequency as you might expect. Performing an FFT on values acquired during a finite sampling interval spreads out and distorts

the results. Just by acquiring a finite number of values of a signal, you distort the signal's spectrum.

You'll better understand how sampling affects the results from an FFT by examining some data. Assume that you have a 16-Hz sine-wave signal to measure. You set the sampling rate ($f_s$ ) of your computer's ADC to 1024 Hz and you acquire data for 0.5 s (**Fig. 3a** ).

During that period, the signal produces eight complete cycles ($m$ ), and the computer acquires 512 time samples. After you perform an FFT, you obtain the results shown in **Figure 3b.**

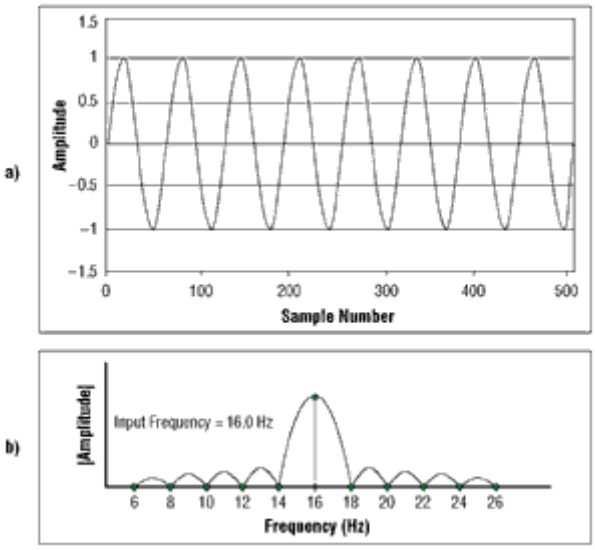

**Figure 3.** a) The FFT results for 512 samples of a 16-Hz sine-wave sampled during a 0.5-s period yields (b) these discrete points. The superimposed curve represents the DTFT of the sampled sine wave.

#### You See Only the Points

Although a DTFT actually yields a sin(x)/x curve for the 16-Hz signal, you see only the individual points from the FFT at the discrete frequency-axis bins. In effect, the center of the DTFT's sin(x)/x curve landed exactly on the center of the 16-Hz bin. The results of the FFT look correct. After all, you started with a 16-Hz signal, and the FFT indicates that the sampled data represent a 16-Hz signal. Although you processed a set of sampled data values, you saw no leakage into adjacent bins. You got lucky—everything lined up perfectly.

Consider what happens, though, when you use the same sample rate and sampling interval to measure a 17.5-Hz sine wave. You no longer acquire an integer number of complete cycles (**Fig. 4a** ). If you plotted the results of the FFT for the 17.5-Hz signal, you would see the center of the sin(x)/x curve at 17.5 Hz (**Fig. 4b** ).

The FFT points in Figure 4b seem to show that energy from the pure 17.5-Hz sine wave "leaked" into the FFT bins at 16 Hz and 18 Hz, and

to a lesser extent into other bins. That leakage arises because the FFT processes a finite number of sample values. In some cases, FFT leakage can cause a strong signal to swamp adjacent weak signals, which reduces spectral sensitivity.

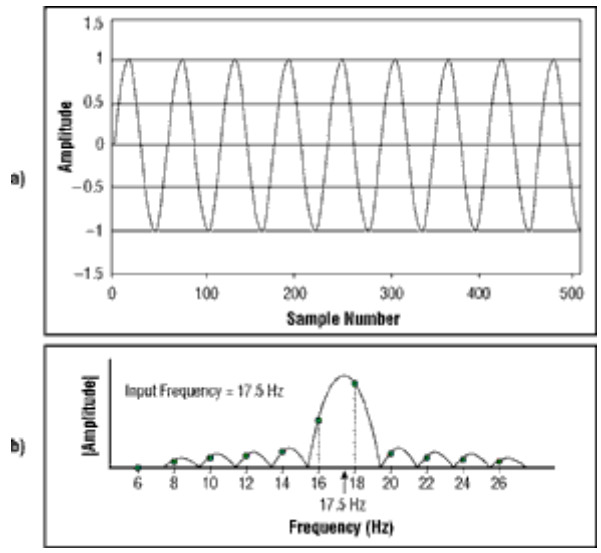


**Figure 4.** a) The FFT results for 512 samples of a 17.5-Hz sine-wave sampled during a 0.5-s period yields (b) these discrete points. The frequency data for the 17.5-Hz peak seams to leak into the adjacent FFT bins.

#### Reduce the Side Lobes

Reducing the high side lobes on the sin(x)/x curve would minimize leakage into the adjacent FFT bins. The side lobes arise from the effect on the FFT of the abrupt rising and falling edges of the rectangular

window. Remember that you “apply” the rectangular window to your data simply by starting and stopping the sampling process.

The first people to use digital signal processing discovered that smoothing the sharp rising and falling edges of the rectangular window reduced the side lobes in the sin(x)/x curve and thus greatly reduced spectral leakage. But you can’t smooth the data just any way you want. You need to use special math functions, or windowing functions, to get your samples ready for an FFT.

The sequence of plots in **Figure 5** shows how a windowing function works. You multiply the 32 sine-wave values by the corresponding values for a Hanning window. The windowing operation “compresses” the starting and ending values to zero (Fig. 5c). Thus, the set of 32 values no longer has abrupt start or end transitions. Keep in mind that you perform a windowing operation on the digital data after the computer acquires them.

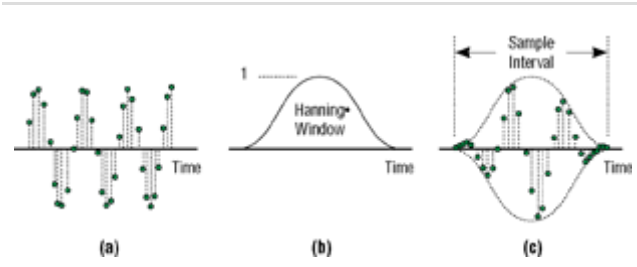


**Figure 5.** a) Multiplying each value in a sampled signal by a corresponding window value (b) results in a signal that does not have abrupt transitions (c).

#### Windowing Boosts Performance

When you perform an FFT on the windowed sine-wave data, the resulting sin(x)/x curve shows reduced side lobes, and thus the FFT data exhibit less leakage. In effect, by windowing the data before you run them through an FFT routine, you improve the sensitivity of your spectral measurements.

Luckily, you can find the equations for many popular window functions in reference books.[2] The following equation defines the Hanning window shown in Figure 5b:

$$w(n)\text{Hanning} = 0.5 - 0.5\cos(2\pi n / N)$$

In this equation, N represents the number of sine-wave samples you acquire, and n equals the sample index: 0, 1, 2, and so on up to N–1. You could write a simple Basic or C program to compute the Hanning coefficient for each point and then multiply it by the corresponding data value. You could even do the same thing in an Excel spreadsheet, too. Most commercial FFT software packages include provisions for applying a window to your raw data.

#### Try Another Window

You also may hear about the Hamming window, which is similar to the Hanning window. In the Hamming window, the end points don't reach zero. A plot of the Hamming window looks like a Hanning window raised on a pedestal. The following equation defines the Hamming window:

$$w(n)\text{Hamming} = 0.54 - 0.46\cos(2\pi n / N)$$

You can best see how various windows reduce the FFT's side lobes by plotting their frequency responses along with the frequency response of a rectangular window. A rectangular window provides a reference for measuring relative window performances. The plots in **Figure 6** show the FFT magnitude responses for the rectangular, Hanning, and Hamming windows. I normalized the plots so the main-lobe peak for each curve starts at 0 dB, and the graph shows only positive frequencies. The graph for negative frequencies is simply a mirror image of this graph.

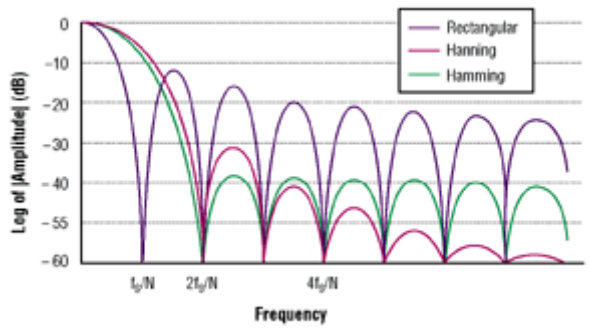


**Figure 6.** The frequency-response plots for the rectangular, Hanning, and Hamming window functions let you compare how well they suppress side lobes that cause leakage.

The rectangular window yields the narrowest main lobe, providing the best frequency resolution for spectral analysis. Unfortunately, the first side lobe for the rectangular window drops only 13 dB below the peak of

the main lobe, which is not good because it increases leakage. The Hanning and Hamming windows reduce the levels of the side lobes, but at a price. Their main lobes are almost twice as wide as the main lobe of the rectangular window (depending on how you define main-lobe width).

#### Windows Can Reduce Resolution

These wider main lobes degrade frequency resolution, yet their small side lobes greatly reduce leakage. You can overcome the loss of frequency resolution by sampling your signals faster during your sample period. Doubling the sample rate, for example, doubles the number of samples you must acquire.

The Hamming window has the lowest first side lobe level of all three types of windows, but after its first side lobe, its remaining side lobes decay slowly relative to those of the Hanning window. The slow decay means that leakage two or three bins away from a signal's center frequency is lower for the Hamming window than for the Hanning window. But leakage a half dozen or so bins away from a signal's center frequency is much lower for the Hanning window than for the Hamming window. You'll need to choose the window function that gives the best results for the type of signals you want to analyze.

You can see the outcome of using a windowing function in **Figure 7** . The graph shows the results of performing an FFT on the 17.5-Hz sine wave mentioned earlier, after the raw data had a Hanning window applied to them. You can compare the results with those shown in Figure 4b.

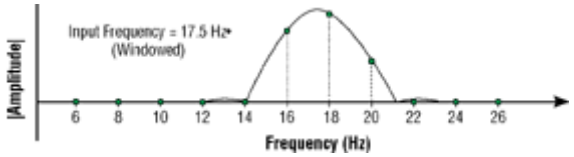


**Figure 7.** Applying a Hanning window to the 17.5-Hz signal in Fig. 4a and then performing an FFT on the data results in less leakage into adjacent frequency bins.

Some high-performance windows show significant differences in their side-lobe reduction. Albert Nuttall studied window functions, called Blackman-Harris windows, which have the general equation:[3]

$$w(n) = a_0 - a_1 \cos(2\pi n/N) + a_2 \cos(4\pi n/N) - a_3 \cos(6\pi n/N).$$

When Nuttall set the $a_k$ coefficients to the "minimum 4-term" values of $a_0 = 0.355768$, $a_1 = 0.487396$, $a_2 = 0.144232$, and $a_3 = 0.012604$, he obtained the window and the frequency response shown in **Figure 8** . The minimum-4-term window loses some frequency resolution because it has a wide main lobe, but the first side lobe drops to –93 db.

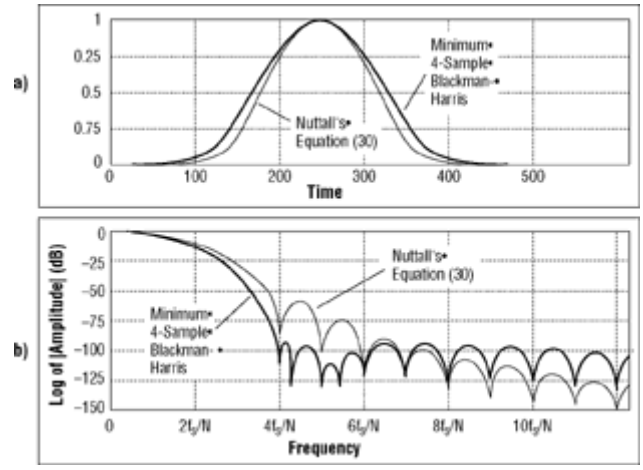


**Figure 8.** a) Two high-performance window functions greatly reduce (b) the side-lobes levels.

If you require fast side-lobe rolloff in an application, consider Nuttall's Equation (30). It applies the following coefficients to the general equation above: $a_0 = 10/32$, $a_1 = 15/32$, $a_2 = 6/32$, and $a_3 = 1/32$. Figure 8 also includes the window and frequency response for Nuttall's Equation (30). This window has a first side-lobe level of –61 dB and a significant side-lobe rolloff of –42 dB/octave.

The first investigators of windowing functions determined that window functions should have zero values at their boundaries and so should their successive derivatives. If a window's $k^{th}$ derivative is zero at the boundaries, the peaks of the window's side lobes will decay at a rate of $6(k+2)$ dB/octave. *T&MW.*

**FOOTNOTES**

1. Blackman, R.B., and J.W. Tukey, The Measurement of Power Spectra, Dover Publications, New York, NY, 1958.

2. Ramirez, Robert W., The FFT; Fundamentals and Concepts, Prentice-Hall, Englewood Cliffs, NJ, 1985.
3. Nuttall, Albert H., “Some Windows with Very Good Sidelobe Behavior,” IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-29, No. 1, IEEE, Piscataway, NJ, February 1981.

**Richard G. Lyons** *works as a systems engineer at TRW. He has been involved in designing and testing electronic signal-processing systems for more than 15 years. He is the author of Understanding Digital Signal Processing.*

***Also see Part II of the article .***