Windowing Functions Improve FFT Results, Part II

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

Also see Part I of the article.

The first article in this two-part series (Footnote 1) explained the cause and spectral smearing effect of "leakage" in the fast Fourier transform (FFT). In the first article, I explained how you can use window functions to minimize that frequency-domain distortion when you're performing spectral testing. The article explained how a simple window function applied to time-domain data reduces inherent FFT spectral leakage distortion. But that reduction comes at the expense of degraded frequency resolution.

The window functions discussed in Part I (Hamming, Hanning, Blackman-Harris, and rectangular) have a limitation. Their amounts of leakage (side-lobe level reduction) and resultant loss in frequency resolution (main-lobe widening) are fixed. Now you will find out how two popular window functions let you strike a compromise between widening a window's main lobe and reducing its side-lobe levels.

Adjust Windows Parameters

The two window functions that provide such flexibility go by the names Chebyshev (or Dolph-Chebyshev) window function and Kaiser (or Kaiser-Bessel) window function. Two formidable-looking expressions define their time-domain samples (**Eq. 1**).

a) Chebyshev Window Function

$w(n)_{Cheb}$ = the N-point inverse DFT of

$$\frac{\cos\left[N * \cos^{-1}\left[\alpha * \cos\left(\pi \frac{m}{n}\right)\right]\right]}{\cosh\left[N * \cosh^{-1}(\alpha)\right]}$$

where

$$\alpha = \cosh\left(\frac{1}{N}\cosh^{-1}(10^{\gamma})\right)$$

$m = 0, 1, 2, \ldots, N-1$

b) Kaiser Window Function

$$w(n)_{Kaiser} = \frac{I_0\left[\beta\sqrt{1-\left(\frac{n-\rho}{\rho}\right)^2}\right]}{I_0(\beta)}$$

for

$n = 0, 1, 2, \ldots, N-1$

$\rho = (N-1)/2$

I_0 is the 0^{th} order modified Bessel function of the first kind

Equation 1

In both expressions, N represents the number of time-domain points in the window functions. Don't let the complexity of these equations intimidate you—at this point, you need not be concerned with their mathematical details. In fact, most commercial digital-signal-processing (DSP) software packages have built-in functions that evaluate $w(n)_{Cheb}$ and $w(n)_{Kaiser}$. You only need to realize that the "control" parameters g and b let you adjust the windows' main-lobe widths versus their side-lobe levels.

The graphs in **Figure 1** show how different g values can affect the window and the frequency response for a Chebyshev window function. **Figure 1a** shows the windows that result for three values of g, and **Figure 1b**shows the corresponding frequency responses for each window.

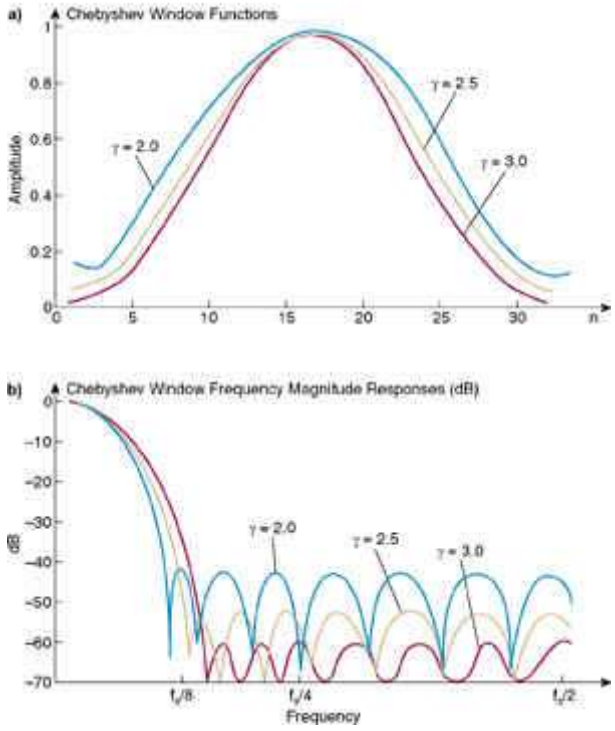

Figure 1

The Chebyshev window function's stop-band attenuation, in decibels, is equal to

$$\text{Atten}_{\text{Cheb}} = -20g$$

If you needed side-lobe levels to be no greater than –60 dB below the main lobe, you'd use the above equation to establish a g value of 3.0 and let your FFT software generate the Chebyshev window coefficients. Some commercial DSP software packages merely require that you specify $\text{Atten}_{\text{Cheb}}$ in decibels rather than specify g. If you use one of these packages, you don't need the above equation.

Software Generates the Coefficients

The same general process applies to the Kaiser window functions (**Figure 2a**). Commercial software packages let you specify b, and they then

generate the associated window coefficients. The frequency-response curves (**Figure 2b**) obtained for the three Kaiser window functions show that you can adjust the desired side-lobe levels and see what effect each level has on the main-lobe width. The control over the value of b (and g for the Chebyshev windows) gives you a capability you don't have when you use the fixed-response window functions discussed in Part I.

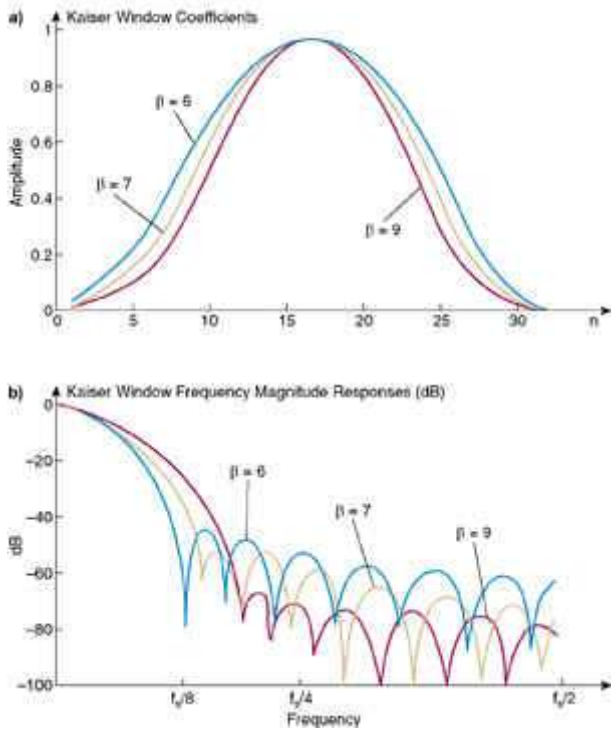

Figure 2

Analyzing Window Functions

There are two ways to determine the frequency response of a window function: *analytically and numerically* . In the analytical approach, you use pencil and paper to derive the continuous Fourier transform of your continuous window function equation. That is, you take the finite integral of a complex variable. Albert Nutall has demonstrated this approach in a paper (Footnote 2).

Most people find the analytical approach difficult for all but the simplest window functions. Using this method is like tearing down the “top end” of a motorcycle engine; it’s time consuming and loaded with opportunities to make mistakes, but very satisfying when you’re successful.

Whereas the analytical approach gives the correct frequency response of a window, the numerical (FFT) method provides merely an approximation. For all practical purposes, this approximation works just as well. To use the numerical method, you define the time-domain window function in software using K points and then append sufficient zeros (called zero padding) to those K non-zero points to make the entire vector N points long. Then you take an N -point FFT of the data. (Remember, N must be an integer power of two for standard radix-2 FFT routines.) The FFT result provides the approximate frequency response of the window function.

Check the Approximation

How do you make sure the approximation is close enough? I recommend using hundreds of sample values to define your window function and then pad those samples with enough zeros so you’ll be taking a several-thousand-point FFT. For example, you could define a Chebyshev window function having 256 points, append 3840 zeros to that time function, and then perform a 4096-point FFT on the data.

These numbers let you see the very low-level side lobes and subtle side-lobe structures in a window’s frequency-magnitude response. Using too few points to define a window function yields incorrect side-lobe levels,

particularly for low side-lobe level windows, once you perform an FFT. (Beginners often make the mistake of using too few points when trying to duplicate the window responses they see in articles and textbooks.)

To best represent information in the frequency domain, I suggest you use a normalized logarithmic vertical-axis scale when you plot your FFT results. That makes it easier to gauge your side-lobe levels, in decibels, relative to the window's main lobe peak. The equation to use for this normalization and dB conversion is

$$\text{normalized } \mathbf{x}_{dB}(m) = 20 \cdot \log_{10}\left(|\mathbf{x}(m)| / |\mathbf{x}(m)|_{max}\right)$$

where $X(m)$ is the complex result for the FFT's m th frequency bin, $|X(m)|$ is the magnitude of that bin's value, and $|X(m)|_{max}$ is the largest individual FFT output magnitude value. In practice, plotting normalized $X_{dB}(m)$enhances the low-magnitude resolution afforded by the logarithmic decibel scale.

Go Window Shopping

By now you may be wondering, "Which is the better window to use?" The window you choose depends on your testing application, but you can begin to answer this question by comparing the Chebyshev and Kaiser windows. Refer back to Figure 1b and Figure 2b, and you will see that unlike the constant side-lobe peak levels of the Chebyshev window, the Kaiser window's side lobes decrease as the frequency increases. The Kaiser side lobes, however, are higher than the Chebyshev window's side lobes near the main lobe.

Your primary goal is to reduce the side-lobe levels without broadening the main lobe too much, which would degrade your frequency resolution. While the most popular windows have main-lobe widths that vary by a factor of 1.5 to 2, their side-lobe attenuation varies by as much as 5 to 6 orders of magnitude. So in general, it's the side-lobe levels that determine which window meets your application's requirements.

Here's a way to select the appropriate window function for detecting low-level spectral components that are located near high-level components: First, determine the frequency resolution requirement of your spectral testing application. This frequency will determine the maximum window main-lobe width you can tolerate for a given FFT size. Next, estimate the highest and lowest level spectral components you need to measure and how far they are in terms of FFT bins from the main lobe. This estimation will let you determine if a given window's side lobes are low enough for your testing application.

If a high-power interfering signal—whose leakage you're trying to minimize—resides close to the low-power signal you're trying to detect, then you need a very low first-side-lobe level. Conversely, if the high-power signal is sufficiently separated in frequency from the low-power signal of interest, then a window with rapid side-lobe roll-off is more important.

Using Windows with Real-World Signals

If you need to improve your windowed FFT frequency resolution while using a window function, you can double the number of collected time

samples (while maintaining the original sampling rate). If you can do this, you'll improve your frequency resolution by a factor of two.

Even when you use a windowing function, very high amplitude spectral components can obscure nearby low-amplitude spectral components of the signal you want to test. This obscuring is especially pronounced when the original time data has a non-zero average—that is, when it's "riding" on a DC bias. When you perform an FFT on the data, the high-amplitude DC spectral component at 0 Hz will overwhelm its low-frequency spectral neighbors. This effect is particularly true for large FFTs.

To eliminate this problem, calculate the average of the original signal and then subtract the average from *each* sample in the original signal. This step removes the DC bias by making the new signal's average (mean) value equal to zero. Thus, you eliminate any high-level 0 Hz component in the FFT results.

To determine how many time- domain samples you must collect, you first must decide what spectral resolution you need in your test application. Using the following equation, you can determine the number of required data samples, N :

$$N = f_s/(\text{desired frequency resolution})$$

Remember, for the standard radix-2 FFT, N must be an integer power of 2. The value of f_s represents the sampling rate of your data-acquisition system.

In some cases, you can't control the length of the FFT input data sequence, and the length of the data might not be an integer power of two. Don't simply discard data samples to shorten the length of the data sequence so it is a power of two. Instead, append zero-valued samples (zero stuffing) to match the number of points of the size of the next largest radix-2 FFT. For example, if you have 1000 time-domain samples to transform, don't discard 488 data samples and use a 512-point FFT. (It's usually not a great idea to throw away data!) Rather, append 24 trailing zero-valued samples to the original sequence and use a 1024-point FFT.

If you perform both zero stuffing and windowing, be sure to append the zeros *after* you window the original non-zero time-domain samples. Applying a window function to a sequence containing appended zeros will distort your FFT results. In **Figure 3** , you can see how windowing the zero-stuffed data distorts the window.

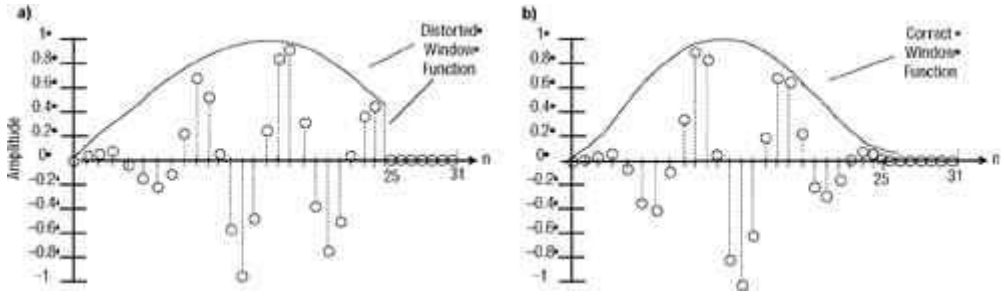

Figure 3

Hardware or software implementation may constrain your choice of window functions. If you use floating-point values in your spectral-analysis software, you can use any window function. If your hardware or

software restricts you to representing data in fixed-point (integer) format, you can't use some high-performance window functions because of window coefficient quantization. For example, to achieve the g=3 Chebyshev window performance (where the first side lobe occurs at –60 dB) you must use 12-bit words to represent the high-precision window coefficient values. Some hardware designs for instruments and computers do not provide sufficient memory to work with such values.

Getting More Information About Window Functions

Three good papers can help you delve into window functions in more detail. The "mother" of all window papers was written by frederic Harris (Footnote 3). Don't be put off by Harris' mathematical rigor, because studying his classic paper pays great dividends. Do be aware, though, that Harris' paper contains two typographical errors in the 4-term (–74 dB) window coefficients column on p. 65. Albert Nuttall's paper[2] specifies those coefficients as 0.40217, 0.49703, 0.09892, and 0.00188.

In addition to correcting several of Harris' typographical errors, Nuttall's paper provides a useful discussion of the Cosa window functions. If you experiment with the useful window functions in Nuttall's paper, pay attention to his equation (5). The a_k coefficients with odd subscripts are negative.

Finally, Charles Gumas presents an informative, and very readable, window discussion in his paper (Footnote 4) and provides a useful tabulation of the performance of many popular window functions. *T&MW*

FOOTNOTES

1. Lyons, Richard, "Window Functions Improve FFT Results," *Test & Measurement World* , June 1998, p. 37.

2. Nuttall, Albert H., "Some Windows with Very Good Sidelobe Behavior," *IEEE Transactions on Acoustics, Speech, and Signal Processing* , Vol. ASSP-29, No. 1, February 1981.

3. Harris, Fredric J., "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proceedings of the IEEE* , Vol. 66, No. 1, January 1978.

4. Gumas, Charles C., "Window Choices Become Crucial in High-Dynamic Range FFT Processing," *Personal Engineering and Instrumentation News* , May 1997.

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 I of the article.