# Windows for Non-coherent waveforms

By Dan P. Bullard, 12 May, 2011

adapted from a Maxim Knowledge Base article originally published 24 January 2007
Published with permission of Maxim.

**Introduction**

Whenever you capture a waveform from a non-coherent source, you should window it before doing any spectral analysis. Whether the source is an oscillator, an RF generator, a self contained digitizer (like a temp sensor), capturing a waveform that came from a non-synchronous or non-coherent source means that you cannot get all the amplitude into a single spectral bin. In that case, you really need to window.

**Non-coherency**

When the waveform source is non-coherent (i.e. not related to the clock you are using to capture the waveform) you will get the characteristic spectral smearing that tells you something is wrong. For example, here is the Time Domain representation of a sine wave that is 3.5 cycles long.

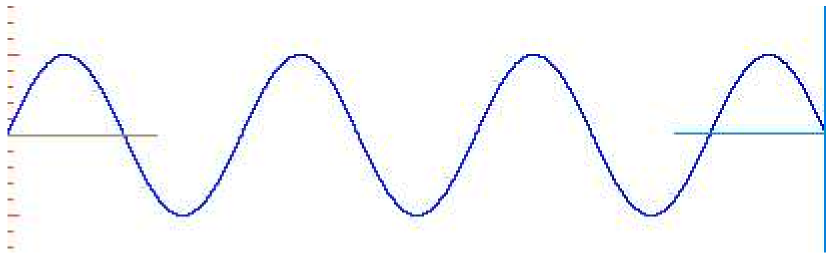


Figure 1: 3.5 cycle sine wave

A sine wave with 3.5 cycles is non-coherent because it does not end after an integer number of cycles. In the first approximation it can be stated that the FFT assumes that the signal that it sees repeats, so imagine what the FFT has in mind for this waveform. Here is what it assumes.

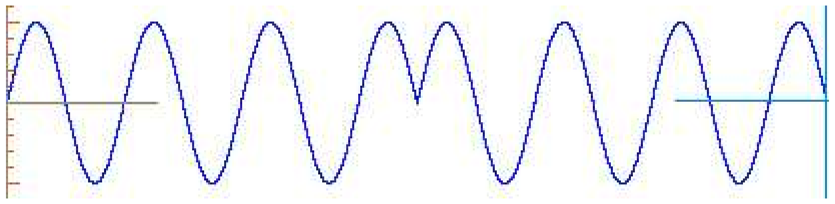


Figure 2: 3.5 cycle waveform repeated

The FFT assumes that the waveform repeats, so it sees this big **discontinuity** when the two endpoints (the first point and the last point) of the waveform don't line up. That discontinuity is assumed to be part of the signal, and therefore causes spectral components to appear that are not really in the signal. The spectrum of the original waveform looks like this:

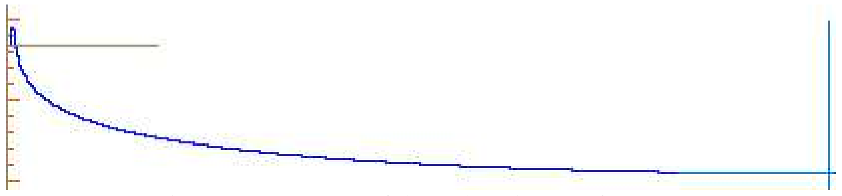


Figure 3: Spectrum of the 3.5 cycle waveform

Notice the **smearing** of the spectrum. This is a big problem, because now our signal to noise ratio (SNR) is really bad about -2.047dB. In other words, all those extra spectral components are assumed to be noise, even though they were caused by the signal. But anything that is not the signal, or not a harmonic is considered noise, so even though this signal has zero noise, the smearing is assumed to be noise by all the DSP functions that look at SNR, SFDR, ENOB, etc.

Whenever you see this smearing, you know you have done something wrong. It's important to note at this point that this can happen to you in a coherent system, if you program your clock dividers incorrectly. If the signal source was not coherent with your tester, then you didn't do anything wrong, you were just unlucky. If the signal source and the capture clock are unrelated, how many cycles you get is pretty much a "roll of the dice".

The only way you can guarantee that you have exactly an integer number of cycles is to have the waveform source driven by clocks that are related to the clocks that **capture** the waveform. If you can't control it, what do you do? Well, the best solution to date is to apply a window before you do the FFT.

**Window basics**

Even when you don't use a window consciously, you are using a window, it's called a rectangular window. A rectangular window is defined as having a value of 1.0 from sample 0 to sample N-1 (the length of your waveform). Windows are applied by multiplying each sample of the window by its counterpart in the waveform. Well, let's face it, if you multiply each sample of the waveform by one, you are going to get the original waveform out again. So saying that everything is windowed by a rectangular window, is like saying that every number is multiplied by one. Even if it wasn't true, you couldn't prove it!

Other than the rectangular window (which doesn't do much for you) there are lots of windows, and unlike a rectangular window, these windows do **good things** for you, when you are non-coherent that is. The trick is to know the good effects and the bad effects of each window so you can choose the best one for the application you have in mind. If you can't find a window that does what you want you are welcome to design your own window, and if other people find it useful, you may end up being famous like Blackman, Hamming, Hann, Nuttall and so on.

I'm not going to show you all the windows there are, many of them are listed and described in Wikipedia under window functions (http://en.wikipedia.org/wiki/Window_Function). What I will do is describe a few of the best windows, tell you how to use them and how to compensate for their failings.

**First Steps**

The first thing you **must do** if you want to use a non-rectangular window is **remove the DC offset**. If you miss this step you will regret it, the window will smear the DC and make the spectrum even uglier that it was without the window. Many primers on windowing miss this step, even a world famous DSP lecturer from Glasgow didn't know you needed to remove the DC offset! In many cases you can get away with it, such as when the signal is AC coupled, but this is no guarantee. So, step one, use a DSP function to measure the average value of the waveform, then subtract that value out of the waveform.

To apply the window, you multiply the window by the waveform sample by sample. That is, sample 0 of the waveform is multiplied by sample 0 of the window, sample 1 of the waveform is multiplied by sample 1 of the waveform and so on, all the way to the end. This of course requires that your window must have exactly the same number of samples as your waveform.

**Getting Windows**

Luckily waveform windows are cheaper than cheap, they are free. Every window is described by a formula, all you have to do is to code that into C or whatever language your tester uses and you are in business. For example, below is the code for a Hann window (there's no such thing as a Hanning window it's just a misnomer for Hann).

```
void create_hannwindow(int NumSamples, double *RealOut)
{
int n;
for(n=0; n<NumSamples; n++)
     {
          RealOut[n] = 0.5*(1-cos((2*PI*n)/(NumSamples-1)));
     }
}
```

Now if you go to the Wikipedia link above you can see the formula I used to create this code. Now, here is what it looks like:

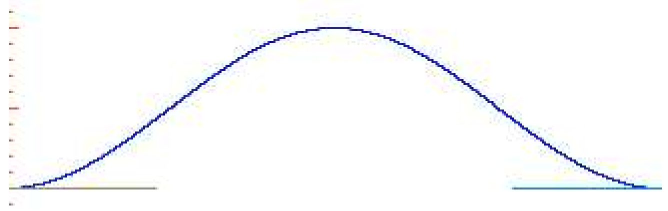


Figure 4: A Hann Window

Notice that the window resembles a bell curve, most windows look like this. The idea is that if we can gently taper the ends of the waveform that don't line up, they can't cause that nasty discontinuity, and the FFT won't know about it. Basically we are sweeping the dirt under the rug, so the FFT can't find out about our bad luck in capturing a non-coherent waveform. As we mentioned earlier, we have to multiply the non-coherent waveform by the window waveform, so here is the result:

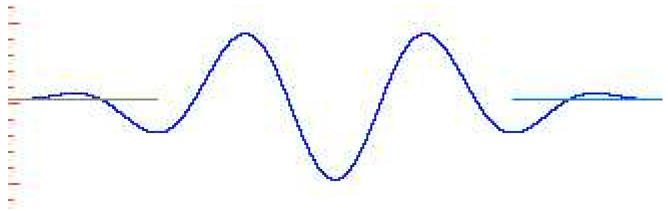


Figure 5: Hann windowed 3.5 cycle sine wave

So here we have the original 3.5 cycle waveform after multiplying it by the Hann window. Notice how the ends are tapered, this way the FFT will never know what we are trying to foist on it. It's almost like when we sweep dirt under the rug, we don't leave it in a big pile, we smooth it out so nobody will notice the lump down there. Now, what does the spectrum look like? Here it is:

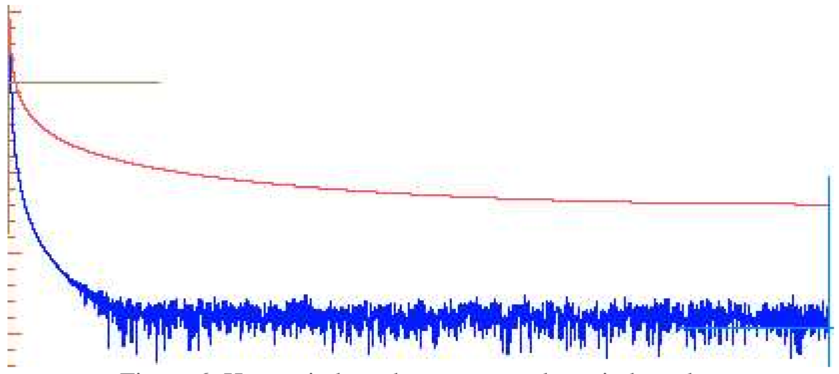


Figure 6: Hann windowed spectrum and unwindowed spectrum

Here in blue is the windowed spectrum of the 3.5 cycle sine wave, and in red is the badly smeared original 3.5 cycle sine wave spectrum. The Hann window made a big difference didn't it?
If we zoom in we can get a better idea of how this helped us.

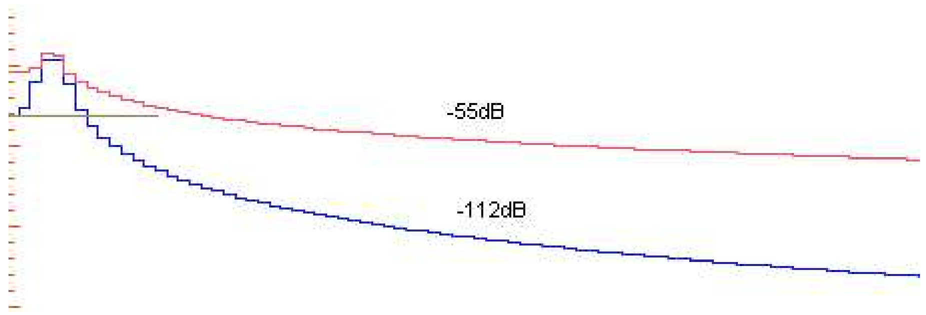


Figure 7: Comparison between windowed and unwindowed spectra

Again in blue is the Hann windowed spectrum, and in red is the unwindowed (rectangular windowed if you insist) spectrum. At the same point in the spectrum the red is about –55dB down from the high point (the signal) and the blue spectrum is about –112dB below the signal amplitude. This is what windows do for you, they "sharpen the skirts" so to speak. They reduce the smearing so that other spectral details like noise, harmonics, spurs and the like can actually be seen. Without windowing we couldn't see much of anything but the signal and that sort of defeats the point of doing the spectral analysis. Take a look at the next plot:

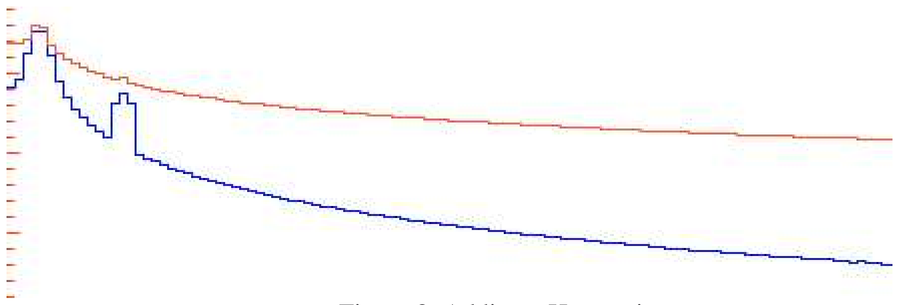


Figure 8: Adding a Harmonic

In this example I added the 4th harmonic to the signal, about 40dB down from the fundamental amplitude.
The unwindowed spectrum barely notices the harmonic, but in the Hann windowed spectrum it stands right out. The reason this harmonic didn't appear in the unwindowed spectrum is obvious, the amplitude was at the same amplitude as the smearing. This is why we window, to clear out the spectrum so we can see the things we need to see to decide whether the part is good or bad. It's kind of like mowing the grass to find the car keys you dropped in the lawn. Of course when you do that, bad things can happen can't they? Speaking of bad things, now you can see one of the bad things that windowing does. It spreads the energy of the signal out into adjacent bins. The interesting thing about the 4th harmonic of this signal is that it is coherent, since the fundamental fell in bin 3.5 (since we captured 3.5 cycles) the 4th harmonic will fall in 4*3.5 or bin 14. Since 14 is a whole number, this signal is (coincidentally) coherent. Without windowing all the energy of this harmonic would have fallen in a single bin, but we did that, and we couldn't see it because of the smearing. So we used the window, but the window spread the energy out into 1 bin below bin 14 and one bin above bin 14. But that's OK, because if we calculate the RMS value of those 3 bins, we get the true amplitude of the 4th harmonic. We can do the same for the fundamental in bin 4 (rounding 3.5 up).
So, this spreading of signals into adjacent bins isn't really too much of a problem, as long as we know we have to find the highest bin, and calculate the RMS value of those three bins. Well, three in the case of the Hann window, other windows spread differently. That's one of the things you have to determine when you use a window, how much does it spread the signals.

**Other windows**

Let's look at some other windows and see how they compare the Hann window.
Next up is the Flattop window. Here is the window in Time Domain:

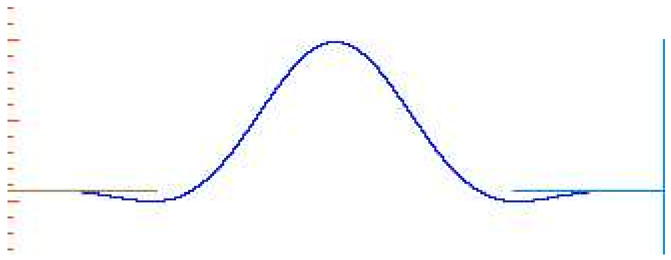


Figure 9: Flattop Window

The code for the Flattop looks like this:

```
// coeffcients for the flattop window
double a0 = 1.0;
double a1 = 1.93;
double a2 = 1.29;
double a3 = 0.388;
double a4 = 0.032;
int n;
for(n=0; n<NumSamples; n++)
{
     RealOut[n] = a0 - a1*cos((2*PI*n)/(NumSamples-1)) +
     a2*cos((4*PI*n)/(NumSamples-1)) - a3*cos((6*PI*n)/(NumSamples-1));
}
```

Notice there are four terms in this equation, that makes this a four term window. Most of the best windows are at least four term windows, and they all seem to be based on a cosine wave. Now, given the same signal (3.5 cycles with –40dB of the 4th harmonic) how does the Flattop compare to the rectangular windowed wave and the Hann windowed wave? Let's see:

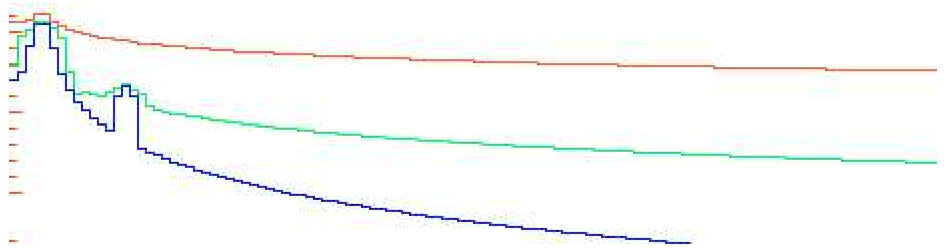


Figure 10: Rectangular vs Hann vs Flattop

The spectrum of the Flattop windowed sine wave isn't very good, it's somewhere between the Hann and the rectangular windowed spectrum. While some claim the Flattop is good for some things, I haven't figured out what they are yet. OK, the next one is better, I promise.

**The Blackman-Harris window**

The Blackman-Harris window is famous because it's a very good window. First though, let's take a look at the code it takes to make one:

```
// coeffcients for the Blackman-Harris window
double a0 = 0.35875;
double a1 = 0.48829;
double a2 = 0.14128;
double a3 = 0.01168;
int n;
for(n=0; n<NumSamples; n++)
{
     RealOut[n] = a0 - a1*cos((2*PI*n)/(NumSamples-1))
     + a2*cos((4*PI*n)/(NumSamples-1)) - a3*cos((6*PI*n)/(NumSamples-1));
}
```
Notice it's another four term window. Now for a look at the window in Time Domain:

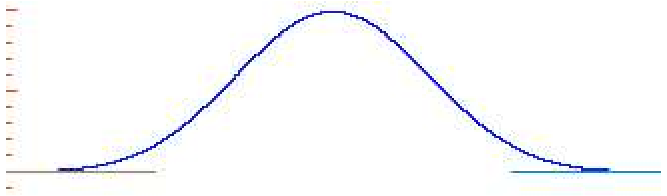


Figure 11: The Blackman-Harris window

It looks pretty much like all the other windows we have seen, but don't let appearances fool you.
And now let's compare the Blackman Harris against the rectangular window and the Hann window:

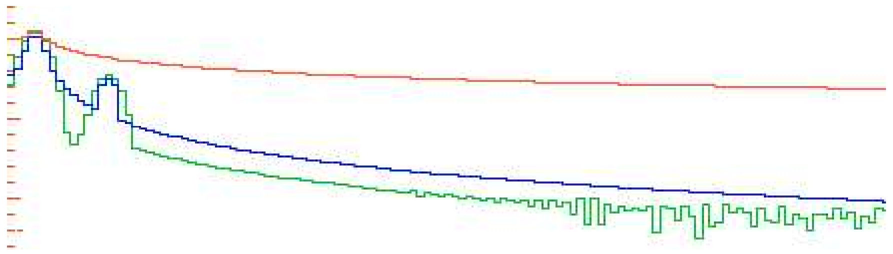


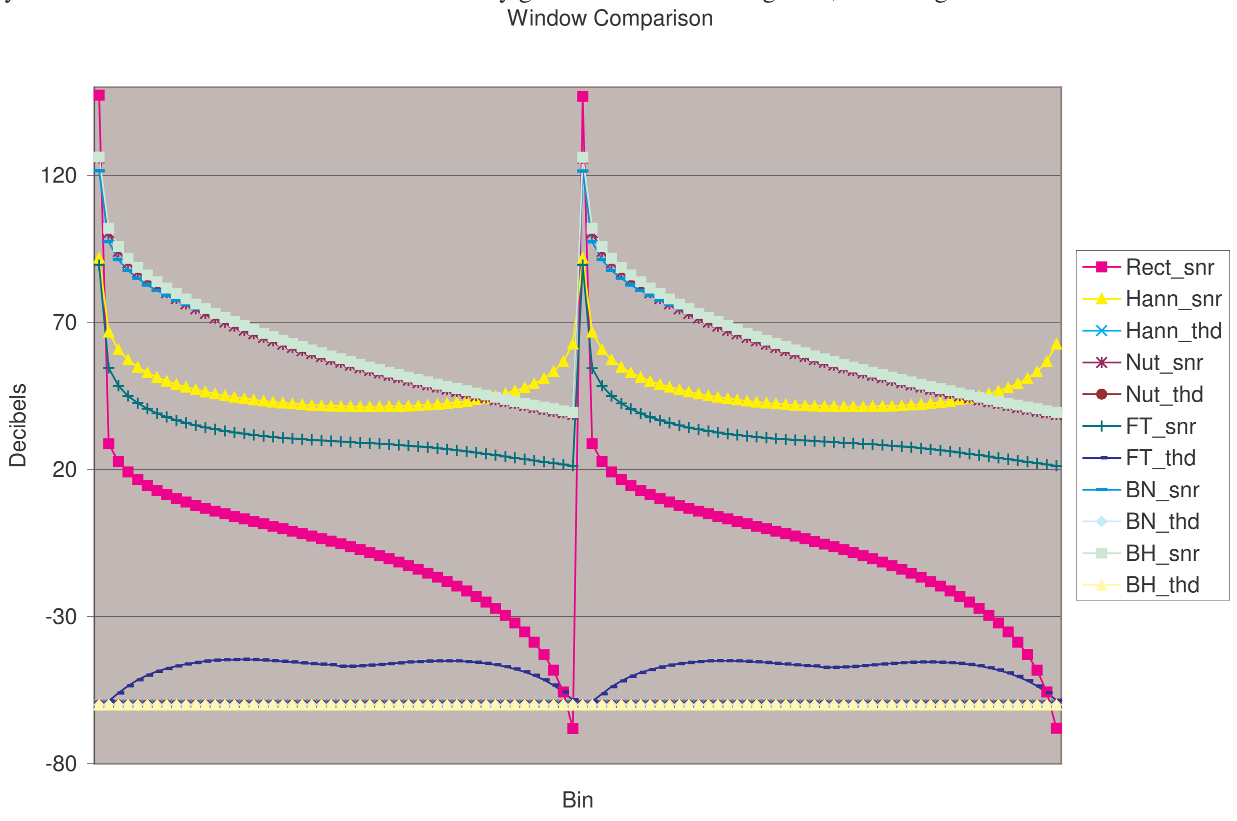
Figure 12: Blackman-Harris spectrum comparison

Again, the rectangular windowed spectrum is in red, the Hann windowed spectrum is in blue, and the newcomer, the Blackman-Harris windowed spectrum is in green. Notice how much lower the skirts are, you can even begin to see the mathematical noise from the FFT off to the right, that's how much the Blackman-Harris window attenuates the smearing. That mathematical noise is down around –155dB, which is clearly better that the Hann or the Flattop window. That means that using the Blackman-Harris window, you will be more likely to be able to see very small signals that other windows may hide just because their skirts are too high above the mathematical noise floor of the FFT.
You may also notice that the Blackman-Harris window caused the signals to spread more, the spreading is about 3 bins on either side of the signal, putting it at a total of 7 bins. So to use the Blackman-Harris your signals need to be far enough apart that the spreading doesn't cause overlapping of one signal's energy onto another. You also have to be willing to RMS all seven bins into a single value for the true answer to the question "how much of that signal do I have". Luckily for you, I have coded that routine.

**Window Dressing**

There are lots of consequences to using windows. For example, when measuring THD, you can't just waltz through the spectrum adding up the amplitudes of each harmonic. Since windows cause the signals to spread, you have to get the energy of each spread harmonic bin, calculate the RMS energy of each harmonic, then RMS those. When calculating Signal to Noise Ratio, you have to go through the spectrum being careful to skip not only the signal bin and the harmonic bins, you also have to avoid the bins that the signals and the signal bins spread into. Managing all this can get kind of hairy, so I did it for you. As mentioned earlier, I've already written the code that does this, it's the same code I used to develop and test these ideas and build functions that take the nastiest work out of your hands. Just email (dan@danbullard.com) me and I'll let you have it.
One of the biggest issues was to deal with the potential that the harmonics and even the fundamental may alias. Much of the code deals with that possibility. You can try to figure that part out if you like, but it is pretty crazy.
Another issue is how non-coherent you are. We were using a signal that was pretty bad, 3.5, a half a cycle was missing. What is really interesting is that using the Hann window, it never gets worse than having a half cycle missing, and if you get 3.8 cycles in your capture, the smearing actually improves. This is not the case with the Blackman-Harris and the other so called High Dynamic Range windows. As the non- coherency gets worse, so does the smearing. That may be one reason to choose the Hann over the others. Below is a chart that shows this, in this

case I ran the “M” (number of cycles) from 21 through 23 (with 22 in the middle). You’ll notice that the Hann SNR gets bad, then gets better as we get closer to 22.0 cycles, then gets worse again, then gets better as we near 23 cycles. The Blackman-Harris and others actually get worse after reaching 21.5, becoming even worse than the Hann.



## Conclusion

Windowing can help you, and it’s not as hard as most people think. Dealing with it comes down to having the functions to sum the bins into signals and avoid summing the signal bins into the noise. If you want to be famous, see if you can come up with your own window, you never know, you might have it in you.