University of Applied Sciences Northwestern Switzerland
School of Engineering

Institute of Microelectronics

IME Report 01/2009: How to use the FFT for signal and noise simulations and measurements

Hanspeter Schmid[*]

Abstract — This short text describes how information on signal and noise levels can be extracted from an FFT when windowing is used. The document does **not** *cover confidence intervals in spectral estimation. The target group of readers are engineers who want to simulate (or measure) signal-to-noise ratios using FFTs or periodograms (pwelch in Matlab) or on a captured signal, e.g., a sigma-delta bitstream.*

1 Introduction

Simulating (or measuring) signals and noise with an FFT is not trivial, because signals and noise do not behave in the same way when analyzed with an FFT.

The main reason for this is easy to explain. Let us do the following experiment: we use a sine signal of frequency 100 Hz and amplitude $\sqrt{2}$ and a white noise source with power spectral density 10^{-8} over the whole frequrency range covered by the FFT.[1] What happens if the simulation[2] time is increased by a factor of 100, but the sampling time is left the same?

The answer directly follows from Parceval's Theorem that states: the total signal power in the time domain and in the frequency domain is the same. If we just increase the simulation time, then the signal power does not change, so the amplitude of the signal stays the same. The noise power *also* does not change, but it is white noise, and occurs in all frequency bins of the FFT. We now have 100 times as many frequency bins as before, so we have to expect that the signal power within one frequency bin is diminished by a factor of 100, or 20 dB. Figure 1 shows these two simulations next to each other. The 20 dB difference is well visible.

To see this in simulation is not trivial, for two reasons: first, the FFT itself also introduces a factor N, the length of the FFT, and second, as can be seen in the left plot of Fig. 1, the signal may obfuscate the noise because it is smeared out. The latter can be fought with windowing.

2 FFT and windowing

Windowing means that the time series to be transfored is multiplied by a window function before the FFT. So instead of $x[i]$, we transform $x[i]w[i]$ for some window function which promises to produce a clearer spectral representation of the signal.

[*]`hanspeter.schmid@fhnw.ch`, Institute of Microelectronics, University of Applied Sciences NW Switzerland.

[1]In Matlab Simulink, this would be a "Sine Wave" block with amplitude $\sqrt{2}$ and frequency $2\pi 100$ rad/sec; and a "Band Limited White Noise" block with noise power $10^{-8}/2$. The reason for the $\cdot/2$ is that we want to have a one-sided power spectral density (PSD) of 10^{-8}, but the Simulink block "Band Limited White Noise" assumes a two-sided PSD.

[2]All of this is also valid for measurements

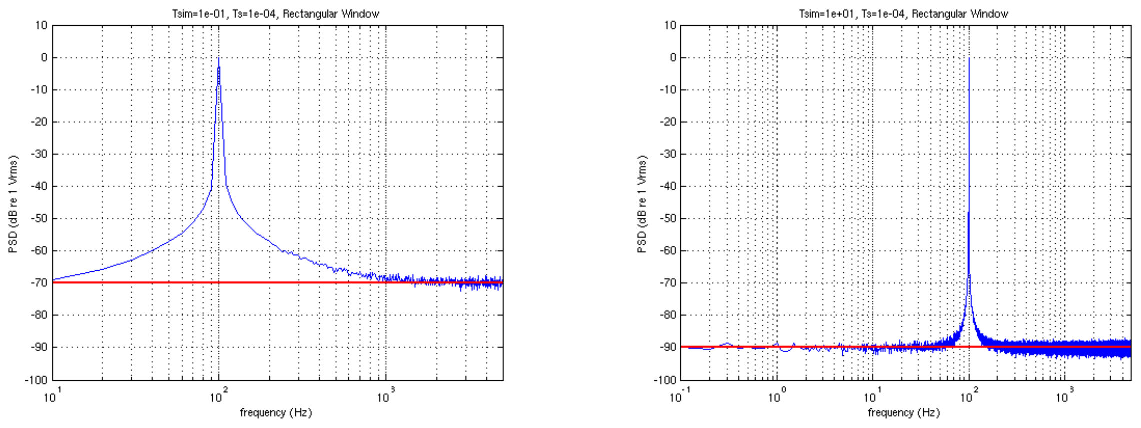

Figure 1: The same system with two different simulation times.

Window	CG	NG
Rectangular	1.0000	1.0000
Hamming	0.5400	0.3974
Hanning	0.5000	0.3750
Bartlett	0.5000	0.3333
Blackman-Harris	0.3587	0.2580
Flat Top	0.2156	0.1752

Table 1: Correction factors for different windows.

The function of the window can be thought of as follows: every single frequency bin of the transformed signal is a linear combination of N time samples. If the signal to be transformed is a sine function, then, ideally, all these N samples add up in one bin and cancel out in all other bins, such that the sine will result as a single peak in the spectrum. For $x[i]w[i]$, the average value in the bin where the time signals add up will be multiplied by

$$\text{CG} = \frac{1}{N}\sum_{i=0}^{N-1} w[i] \tag{1}$$

compared to what happens with a rectangular window ($w[i] = 1$ for all i). CG is the so-called coherent gain of the window. If the signal is white noise, however, then the N time samples are uncorrelated. This means that in every bin, the noise *power* of N input values will add up, and the average value in the bin will be

$$\text{NG} = \frac{1}{N}\sum_{i=0}^{N-1} w[i]^2 \tag{2}$$

compared to what happens when a rectangular window is used. We call NG the noise gain. For a rectangular window, CG $=$ NG $= 1$; the correction factors for a few common windows are listed in Table 1.

2.1 Normalization for reading signal RMS values

If we want to be able to read the RMS value of deterministic signals from an FFT plot, we have to divide the FFT by N times the coherent gain and then calculate the power spectral density. So for an input signal x, we get for the one-sided power spectral density:

$$Y[i] = \frac{\mathrm{FFT}\left\{x[i]w[i]\right\}}{N \cdot \mathrm{CG}} , \tag{3}$$

$$P_{yy}[0] = Y[0] \cdot Y[0]^* \quad \text{and} \quad P_{yy}[i] = 2 \cdot Y[i] \cdot Y[i]^* \quad \text{for } i > 0. \tag{4}$$

Using this scaling, we can read the RMS-value of a deterministic signal directly out of the plot. The 0 dB visible in Fig. 1 correspond to 1 $\mathrm{V_{rms}}$, which was the value used in the simulation.

How can we now read the power spectral density of the noise signal from the plot? When white noise with the power spectral density x_n^2 is fed into the FFT, then the output will be the noise integrated over a frequency range

$$f_{\mathrm{bin}} = \frac{1}{T_{\mathrm{sim}}} = \frac{1}{N T_{\mathrm{samp}}} , \tag{5}$$

(where T_{sim} is the simulation time and T_{samp} is the sampling period) and then multiplied by the noise gain. Since we also divided the result of the FFT by CG, what we plot actually is

$$P_{yy}[i] = x_n^2 \cdot \frac{\mathrm{NG} \cdot f_{\mathrm{bin}}}{\mathrm{CG}^2} . \tag{6}$$

Therefore the power spectral density calculated from the value in the FFT is:

$$x_n^2 = \frac{P_{yy}[i]\mathrm{CG}^2}{\mathrm{NG} f_{\mathrm{bin}}} , \tag{7}$$

or, in other words, if we want to plot a known power spectral density into the same plot, we need to scale it by the factor

$$s_n = \frac{\mathrm{NG} f_{\mathrm{bin}}}{\mathrm{CG}^2} . \tag{8}$$

The red lines in Figure 1 have been obtained like this. For a rectangular window, the calculation is trivial: since $\mathrm{CG} = \mathrm{NG} = 1$ we get $s_n = f_{\mathrm{bin}}$.

2.2 Normalization for reading noise values

Sometimes reading the noise level directly off a plot is more important than being able to read a signal. In this case, the proper way to normalize the FFT is

$$Y[i] = \frac{1}{N} \mathrm{FFT}\left\{x[i]w[i]\right\} , \tag{9}$$

$$P_{yy}[0] = \frac{Y[0] \cdot Y[0]^*}{\mathrm{NG} f_{\mathrm{bin}}} \quad \text{and} \quad P_{yy}[i] = \frac{2 \cdot Y[i] \cdot Y[i]^*}{\mathrm{NG} f_{\mathrm{bin}}} \quad \text{for } i > 0. \tag{10}$$

If we then know that the power at index i comes from a deterministic signal, the power of that signal is

$$P_{\mathrm{sig}} = P_{yy}[i] \cdot \frac{\mathrm{NG} f_{\mathrm{bin}}}{\mathrm{CG}^2} . \tag{11}$$

3 Window functions

Much has been written about window functions, and using the best window function for a certain application requires a lot of specialized knowledge. Without going into the detail, I recommend to use the Hanning window for most applications, except when very low noise has to be observed in the presence of a signal, then the Blackman-Harris window gives better results.

Figures 2 and 3 show this. It can be seen in Fig. 3 that the Hanning window concentrates the signal in a narrower peak, but below some value, the signal is smeared out into so-called side lobes. The Blackman-Harris window creates a wider peak to start with, but has much lower side lobes.

Fig. 2 shows well that using windows is not for free, compared to the Rectangular window, they raise the visible noise floor. In Fig. 2, the rectangular window plots the white noise at -70 dB, the Hanning window at -68.24 dB, and the Blackman-Harris window at -66.98 dB. This means that the latter is least suitable when one wants to see small distortion peaks in the noise floor: distortion peaks are deterministic signals and will not change their magnitude in P_{yy} if we apply a different window.

4 Reading signal RMS values out of a Matlab pwelch periodogram

Matlab's pwelch function (e.g., `pwelch(y,2^13,2^12,[],Fs,'onesided')`) draws a power spectral density by using a Hamming window and calculating a periodogram. The Matlab documentation describes how to set its parameters, but, unfortunately, not how to read values off such a periodogram.

pwelch's output is normalized for reading noise values. Therefore, if one wishes to read the magnitude of a deterministic signal, (11) applies.

So, for example, if one reads the dB-values $x_i = -26.21$, -61.62, -46.19 off a periodogram, then the RMS values can be computed as follows:

```
[Pxx,f]=pwelch(y,2^13,2^12,[],Fs,'onesided');
fbin = f(2);
W=hamming(1024);
CG=sum(W)/1024;
NG=sum(W.^2)/1024;

ff = NG * fbin / CG^2

Xrms = sqrt( 10.^([-26.21 -61.62 -46.19]/10) * ff )
```

5 Behaviour for flicker noise

In the explanation above, we argued that white noise and deterministic signals are treated differently when evaluated with an FFT, because the time samples of a deterministic signal are correlated, which the noise samples are not. For flicker noise, the time samples are not independent, but have some correlation. We have not been able to treat it mathematically yet, but simulations as in Fig. 4 indicate that flicker noise behaves like white noise in an FFT analysis. This hypothesis is not well tested, though.

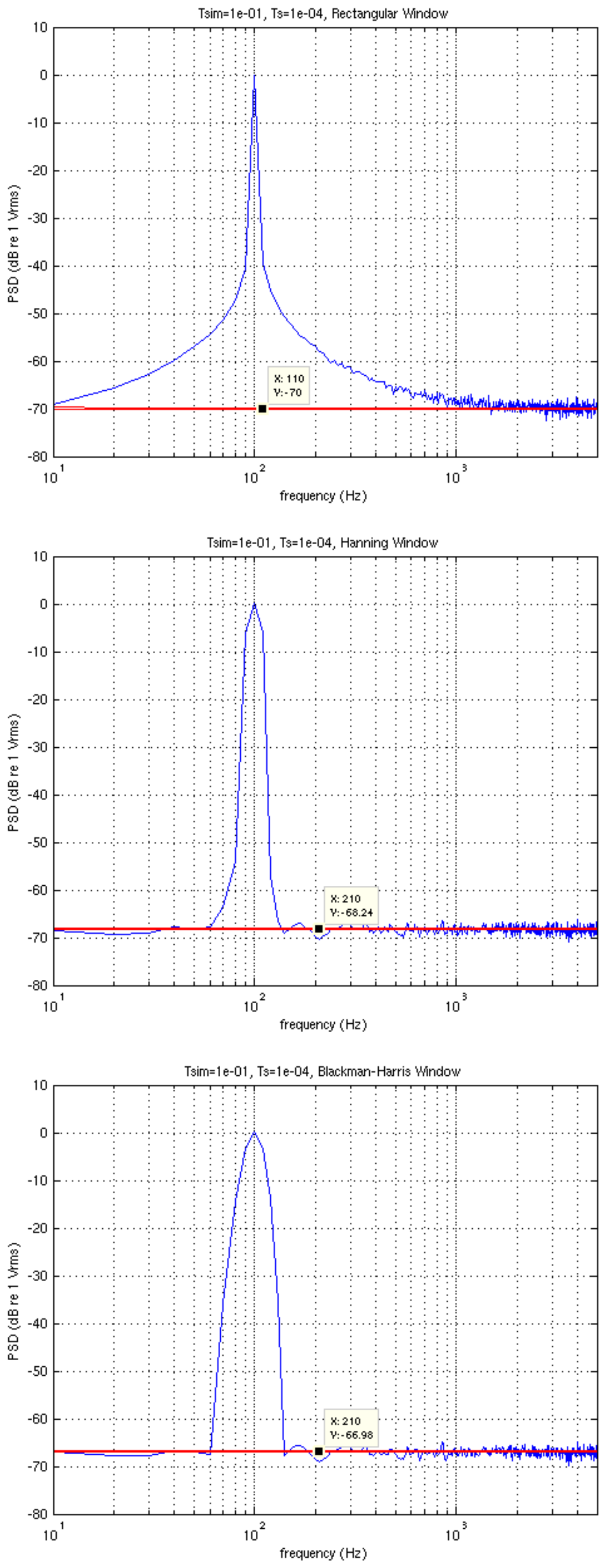

Figure 2: Rectangular, Hanning and Blackman-Harris Window applied to a signal of magnitude $\sqrt{2}$ and white noise with a one-sided power spectral density of 10^{-8}.

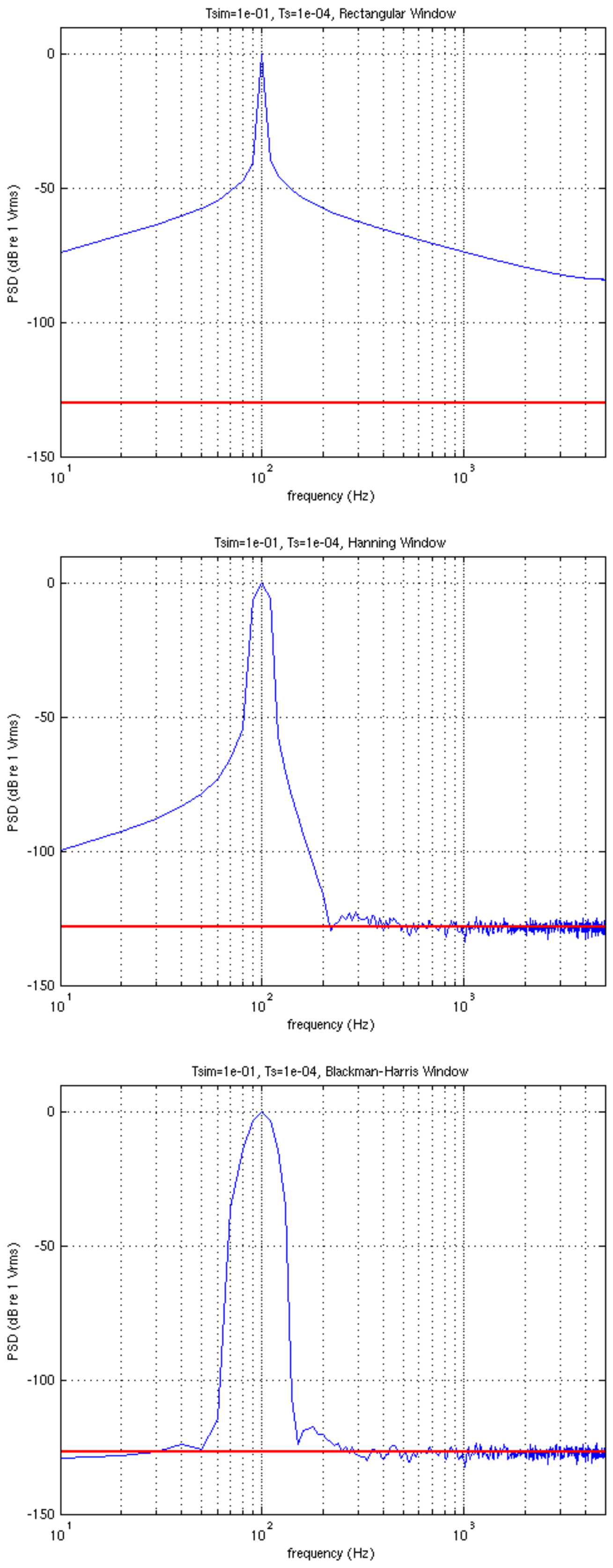

Figure 3: Rectangular, Hanning and Blackman-Harris Window applied to a signal of magnitude $\sqrt{2}$ and white noise with a one-sided power spectral density of 10^{-14}.

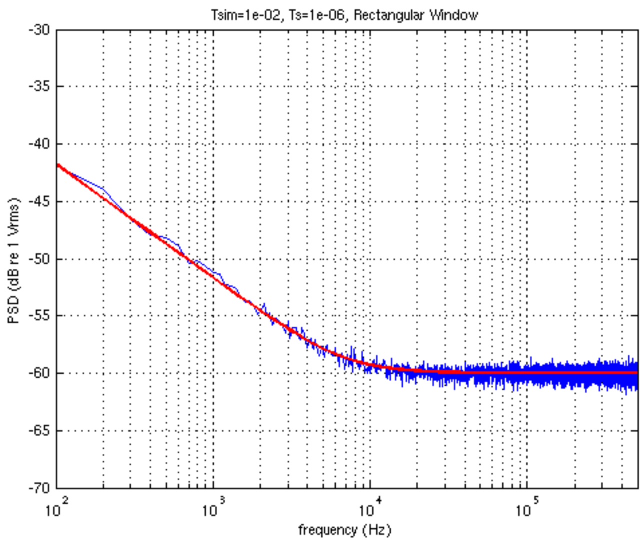

Figure 4: Average of 100 simulations with flicker noise proportional to $1/f$.

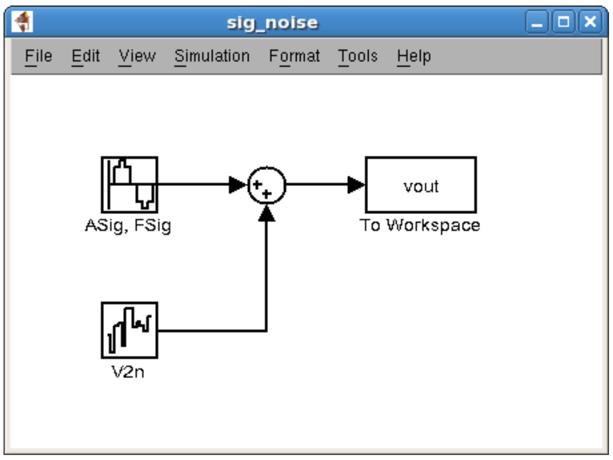

Figure 5: Simulink model used for this document.

6 Sample Matlab code

The plots above were made using the Simulink model shown in Fig. 5 with the following code:

```
clear
%% Set values of noises and signals
V2n    = (1e-4)^2;
ASig   = sqrt(2)*1.0;
FSig   = 100;
TSamp  = 1e-4;
TStop  = 0.1;
NSim   = 32;
rSeed=1454;
sim('sig_noise')

%% Evaluate Fourier Transform and plot
N=max(size(vout));
W = hanning(N);           WN = 'Hanning Window';
W = blackmanharris(N);    WN = 'Blackman-Harris Window';
W = ones(N,1);            WN = 'Rectangular Window';
CG=sum(W)/N;              % Coherent Gain of the window
NG=sum(W.^2)/N;           % Noise Gain of the window
fbin=1/TStop;             % Frequency bin width
NF=fbin*NG/CG^2;          % Factor to be used when plotting an ideal PSD
                          % into the same plot

%Calculate one-sided power spectral density
Y=vout.*W;                % Windowing
YF = abs(fft(Y))/CG/N;    % Transformed signal divided by coherent gain
f = (0:(N-1)/2)/TStop;    % Frequency Axis of half of the FFT
Pyy = 2*YF.^2;            % One-Sided Power Spectral Density
Pyy(1)=Pyy(1)/2;          % The DC component should not be doubled

for k=1:NSim-1,
  k
  rSeed = 169+13*k;
  sim('sig_noise')
  Y=vout.*W;                % Windowing
  YF = abs(fft(Y))/CG/N;    % Transformed signal divided by coherent gain
  Py = 2*YF.^2;             % One-Sided Power Spectral Density
  Py(1)=Pyy(1)/2;           % The DC component should not be doubled
  Pyy = Pyy+Py;
end

Pyy=Pyy/NSim;
PyydB=10*log10(Pyy(1:(N-1)/2+1)); % in dB

semilogx(f,PyydB)
titlestring=sprintf('Tsim=%1.0e, Ts=%1.0e, %s',TStop,TSamp,WN);
title(titlestring)
xlabel('frequency (Hz)')
ylabel('PSD (dB re 1 Vrms)')
axis([1/TStop 1/2/TSamp -80 10])
grid
hold on
plot(f,10*log10(NF*ones(size(f))*V2n),'r','LineWidth',2)
hold off
```