Matlab Exercise: FFT, Spectral Leakage, Zero Padding

1. Set up a simulink spectrum analyser as shown at the end of the Matlab Tutorial notes. For those who did not complete their own during the Tutorial, a simulink file can be downloaded from the web site: http://www.kstio.com/dsp/. Download both of the files ExFFTmatlab.m and ExFFTsimulink.mdl by right clicking on the link in the web page. When running these programs you should run ExFFTmatlab.m first (most easily done by copying the lines to the command page), and then "open" ExFFTsimulink.mdl from the Matlab command window.

2. Generate two separate 64 length buffers of the two sinusoids:

   $x_1(t) = 100\sin(2\pi 101.56t)$ $\qquad$ $x_2(t) = 2\sin(2\pi 156.25t)$

   Use a sampling frequency of 1 kHz.

3. Use the FFT analyser to evaluate the FFT of each of these signals using a rectangular window, and determine which frequency bins have the highest peaks. Compare this result with that expected from calculations, using the method shown in the notes.

4. Repeat the FFT evaluations done in part 2, but using a 1024 point FFT. (N.B. Applying a 1024 point FFT to a 64 length data buffer has the effect of appending (1024-64) zeros to the end of the data, which increases the number of points at which the spectrum is evaluated.)

5. The Hamming window is defined as:

   $$w_{\text{Hamming}} = 0.54 - 0.46\cos\left(\frac{2\pi n}{L-1}\right), \quad n = 0,1,\ldots,L-1$$

   Generate and plot (stem) the Hamming window using the window function generator in simulink.

6. Now repeat parts 2 and 3, but applying first a Hamming window to the 64 length data buffers before evaluating the FFTs. Observe the outputs of the FFTs.

7. Now add these two sine waves together, and apply both 64-length and 1024-length FFTs, with both rectangular and Hamming windows. Observe how "spectral leakage" can mask a small signal in a large one.

8. Calculate what the sampling frequency would have to be to avoid all spectral leakage from the large sine wave into the small one. Use Simulink to prove your result by repeating part 6, but using the new value of sampling frequency.

9. Explore the use of other window functions. Are any others more appropriate to this problem?

10. Now add Gaussian noise to the combined signals, with a variance of 10. Compare the results of evaluating the 1024 point FFT on a 64-length data buffer (i.e. with zero padding), with a 1024 point FFT applied to a 1024-length data buffer (i.e. generate more data).

11. Try using the same FFT analyser but using a *.wav* file as the source (e.g. *start.wav* on the web site). You can also listen to the file as it is analysed by plugging in your own headphones.