On Simulink and the FFT

1 Introduction

This appendix provides a brief explanation on how to display and interpret the frequency domain representation of a signal using Simulink blocks. It is not meant as a theoretical explanation on the Fast Fourier Transform.

2 Building your own FFT display

FFT stands for Fast Fourier Transform. As the name says, it is an algorithm which computes the Fourier Transform (i.e., a transformation from time domain to frequency domain) very fast. It is by nature a block computation, meaning that it is performed on groups, or frames, of data. Therefore, it makes no sense to try to compute the FFT of a single sample. The input to an FFT is a group of samples and the output from the FFT is also a block of data representing the frequency domain in complex format.

You must gather a number of samples to input to an FFT Simulink block. This is done by “buffering” the samples, or “storing” them somewhere in memory. You must first determine which frequency resolution you will require and from that you will decide how many samples to store or buffer in \(^1\). If for some reason you buffer in a smaller number of samples than the size of the FFT you are about to perform, the data will be padded with zeros prior to the FFT operation, i.e., Matlab will pad the buffered data with zeros to extend the buffer up to the number of points you have chosen for the FFT. When the FFT is taken with this padding, it results in an output that was computed with a “partially complete” input. This is to say that your best output will be when the signal without padding is fed to the FFT, and no portion of the input signal is followed by zeroes. The number of FFT points (or FFT length) which you will use for all lab exercises is 1024, unless otherwise specified.

The output from the FFT will be a block, or frame, of 1024 complex values. These will yield magnitude and phase. Remember that now you are in the frequency domain; the first output point represents DC (or 0 Hz) and the last point represents the sampling frequency, which in the lab exercises will likely be 48KHz. This is to say that the frequency interval, or the output resolution obtained is 48,000 divided by 1024. For discrete-time (sampled) signals, the output of an FFT is made of two halves, mirrored at half of the sampling frequency. Frequencies which do not fall on integer multiples of this interval will not be represented precisely, resulting in “spectral leakage”.

How about the amplitude of the output? For the 1024 point FFT of a single sinusoid, whose frequency you will make fall exactly on a multiple of the frequency interval, the amplitude of the

\(^1\) you can think of frequency resolution as how fine you want your output to be. The more points you use to calculate an FFT, the finer your resolution will be.
two components is the amplitude peak of the sinusoid, multiplied by the number of FFT points divided by two.

To summarize now, you know you should buffer data prior to performing an FFT with at least as many samples as the number of points of the FFT, you know that 1024 points will be your FFT length and that you should try to display the output of the FFT from 0Hz (DC) to 48KHz (Fs), realizing that this display is mirrored around Fs/2. For all lab exercises, you can use these numbers and they will yield a good picture.

Things are better learned when they are done. Now you will design your own spectrum analyzer in Simulink. You can achieve this by using three blocks: a Buffer, a Magnitude FFT and a Vector Scope block. These blocks are all found within the Signal Processing Blockset. The Buffer block is under Signal Management - Buffers. The Magnitude FFT is under Transforms, and the Vector Scope is under Signal Processing Sinks. These three will provide you with a good simulated spectrum analyzer. Drag these three into your model, so that you will have a sine wave generator connected to the time domain display as well as your version of a frequency domain display. These displays will be used extensively through the exercises.

You should now have a system resembling the one on Figure 1(a). Double-click on the blocks to adjust the parameters as follows. For your Buffer block use 1024 as buffer size, zero overlap and zero initial conditions. Select “Magnitude” as the output to your FFT block and set 1024 as the FFT length. Finally, on the Vector Scope, select “Frequency” as your input domain, click on the tab “Axis Properties” and set the frequency range to [0...Fs] and the Amplitude Scaling to “Magnitude”. Leave all other parameters as they are.

Run the simulation again and you should see two “extra” windows: one representing your signal in the time domain and another representing it in the frequency domain. Try to explain what you see in the frequency domain, based on what you learned in the above paragraphs. You should see the picture as represented in Figure 1(b), for a 1500Hz, 1 Vp Sinusoid.

![Figure 1: Visualizing Time and Frequency Domain](image_url)

---

2 remember, the output is made of two mirror images. For a single frequency you will see an impulse between DC and Fs/2 and another between Fs/2 and Fs.
There are some things which annoy everyone, including the TAs. One is the surreptitious data point. When the frequency domain display pops up, go under “channels – markers” and select the little circle. After you do that, two circles will appear on your display, representing data that you could swear it was never there before.

So here is a question: why are the sine waves at 1KHz and 1.5KHz displayed differently in the frequency domain if both have 1 as peak value, you have sampled both at 48KHz and used the same FFT length of 1024? Try it out and see if you can explain it. This is an example of the spectral leakage briefly explained above, when the frequency of the input sinusoid does not fall on an integer multiple of the desired resolution (in this case 48,000/1024).