PHYS 5061 Lab 5
Signals and the Frequency Domain

Introduction

This lab makes use of A/D tools developed previously to sample signals and the Fast Fourier Transform (FFT) to examine them in the frequency domain. You will explore the role of sampling rates and the phenomenon of aliasing, the frequency content of standard waveforms, and work in the frequency domain to process some recorded data to remove interference and other random noise in order to recover the signal of interest.

Time-dependent signals can be regarded as the superposition of sinusoidal signals. A general form for expressing this for a periodic signal with period T, that obeys $v(t+T) = v(t)$, is

$$v(t) = \sum_{n=-\infty}^{\infty} c_n e^{in\omega_o t} = c_0 + \sum_{n=1}^{\infty} \left(c_n e^{in\omega_o t} + c_{-n} e^{-in\omega_o t} \right),$$

where the fundamental angular frequency $\omega_0 = 2\pi f_0$ and $f_0 = 1/T$. The coefficients c_n are called the Fourier coefficients and may be complex numbers. You should recall that $e^{\pm i\theta} = \cos\theta \pm i \sin\theta$, so that this expression is made up of sinusoidal signals with frequencies that are integer multiples of the fundamental and the multiples are often referred to as harmonics. For a non-periodic waveform, the sum over discrete frequencies turns into a sum over all frequencies and is expressed as an integral, with the weighting of each frequency (its Fourier coefficient, now given as $F(\omega)$:

$$v(t) = \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega.$$

These relationships can be inverted to express the coefficients c_n or $F(\omega)$ in terms of the signal $v(t)$:

$$c_n \propto \int_0^T v(t) e^{-in\omega_0 t} dt$$

or

$$F(\omega) \propto \int_{-\infty}^{\infty} v(t) e^{-i\omega t} dt.$$

These last two equations are written as proportionalities since you will encounter various ways of writing down the basic relationships, and these will include “normalization” factors that vary by author. Such pairs of relationships (between $v(t)$ and c_n or $v(t)$ and $F(\omega)$ are known as Fourier transform pairs.

Discretely sampled signals as recorded by an A/D converter or a digital oscilloscope limit the range of frequencies included in the sum or integral to calculate the Fourier coefficients. If the signal $v(t)$ is measured with N samples taken at equally spaced times Δt apart, the corresponding sampling frequency $f_s = \frac{1}{\Delta t}$. As you will see, this imposes a limit on the frequencies reliably found through the Fourier transform to a maximum of $f_s/2$.

This upper frequency limit is often called the Nyquist frequency. The rule of thumb is that to digitize a signal of frequency f, you must sample it at a rate at least twice this rate – amounting to at least 2 points per cycle. Naturally, recording many points per cycle of a periodic wave gives a better reproduction of the signal, so this Nyquist frequency tells us about the minimimum sampling rate needed to reliably detect a frequency component in a signal.

The calculation of the Fourier coefficients of a digitized waveform is carried out efficiently with the fast Fourier transform algorithm (FFT). Given N samples of v sampled at a rate of f_s, this algorithm calculates the Fourier coefficients c_n for a series of equally spaced frequencies. These frequencies are spaced by $\Delta f = f_s/N$ and range over positive and negative frequencies:
$f_n = -(\frac{N}{2}-1)\Delta f,\ -(\frac{N}{2}-2)\Delta f, \ldots -2\Delta f, -\Delta f,\ 0,\ \Delta f,\ 2\Delta f,\ \ldots (\frac{N}{2}-2)\Delta f,\ (\frac{(N}{2}-1)\Delta f,\ \frac{N}{2}\Delta f.$
The highest frequency corresponds to the Nyquist frequency, $f_s/2$.

This calculation is available in LabVIEW functions under Signal processing - Transforms as a VI, $\mathcal{F}(x)$. Given an input array of measured values of the signal $v(t)$, this VI sends out an array of the Fourier coefficients c_n for the discrete set of frequencies allowed by the number of samples and the sampling rate. To do the calculation efficiently, the FFT algorithm requires that the number of samples, N, be an integer power of 2: $N = 2^m$. While the algorithm can accept input arrays of other lengths, it must then adapt the calculation by truncating the data, padding it with additional zeros, or adapt a slower, less efficient algorithm, all of which introduce their own quirks. Therefore throughout much of this lab we will work with a number of samples $N = 1024 = 2^{10}$.

The inverse process, of taking the Fourier coefficients c_n and producing $v(t)$ is carried out by the companion VI, $\mathcal{F}'(x)$, the inverse FFT. Taking the Fourier transform of the Fourier transform of v should produce the orignal v. However, because of the normalization factors included in the Fourier transform pairs, the two VI's are not interchangeable. So be sure to use $\mathcal{F}(x)$ to take a signal in the "time domain" $v(t)$ to find its frequency spectrum and use $\mathcal{F}'(x)$ to take the frequency domain representation c_n's (its frequency spectrum) back to the time domain.

Chapter 12 of Essick explores the FFT. It lays out some of the details of calculating the Fourier coefficients for a discretely sampled signal and pursues some more sophisticated techniques (particularly using window functions) in using FFT's to extract information about the frequency and amplitude data about a waveform. You may find this a useful reference in understanding the general application of the FFT, but we won't dwell on windowing techniques or follow the exercises of this chapter.

Experimenting with the Fast Fourier Transform

Building the spectrum analyzer

Revive your simple digital oscilloscope from chapter 6 in Essick or make use of the simple one-shot digitizer, SingleDSO.vi, available in a folder of utility VI's. In any case be sure you have convenient controls for the number of samples collected and the sampling rate. Throughout most of this work, the number of samples should be set to 1024, so you might see if you can program the control to have this as its default value. You should display the digitized signal in an X-Y Graph, plotting $v(t)$. If adapting your own DSO VI, you may need to (a) pry out just the digitized voltage data from the output of DAQ Assistant VI you used, and (b) construct an array of time values for the sampled data and bundle these together to feed to the X-Y graph VI. Please save your modifications to your old VI's (or to the utility VI's provided) as new and separate VI's in your own folder.

Add to this digitizing VI the capability to compute (with the FFT.vi) and display (with another X-Y graph) the FFT spectrum of coefficients as a function of frequency. Put this 2nd graph directly below the time domain graph of $v(t)$. Since the Fourier coefficients are complex, you can settle for displaying the magnitude of the c_n. The math palette has tools for manipulating complex numbers - look for the one that converts cartesian $(x + iy)$ to polar $(re^{i\theta})$ form, with handy outputs of magnitude (r) and angle.

If you read the help for the FFT.vi, you'll discover the coefficients are ordered in a non-intuitive way (at least to us non-experts). Recall that our FFT includes coefficients for negative frequencies. The default order for the coefficients starts at $f = 0$, Δf, $2\Delta f$, running up to $f = +f_s/2$ (the maximum) and then the frequencies run through negative values $-(\frac{N}{2}-1)\Delta f$ to $-\Delta f$. There is another utility VI that generates an array of frequency values based on the number of samples and the sampling frequency in the same order as the FFT's output array of coefficients, FFTfreqs.vi – also in the utility folder. You can use this, but take the time to see how it works. (The ordering of array elements in the FFT is another idiosyncratic feature of implementing the algorithm that may vary if you use an FFT routine in another software package; always look carefully at the documentation.) Bundling the frequencies array with the coefficients array from FFT.vi for the second X-Y graph will take care of the weird ordering, since the X-Y graph will be based on the correct pairs of (f_n, c_n) when plotted. *Before* bundling with frequencies for the 2nd X-Y Graph, it may be worthwhile to divide the magnitude of the c_n's by N, the number of samples. This turns out to produce something scaled correctly to correspond to the normal Fourier coefficients c_n in the original expressions above.

Note on the X-Y Graph: some plots here may be very cluttered with lots of points. By right-clicking on the graph, you can choose to make the Graphics Palette visible beneath the graph. This provides tools to select and zoom in on smaller regions of the graph to get a better look the details. You should enable this feature for both the graphs you have.

With the number of samples collected = 1024 and a sampling frequency of 1024 (samples/second), try recording a 20 Hz sine wave, 2 Vpp, from a function generator (a real one on the bench, not one inside LabVIEW).
Double check what you're getting from the function generator with your bench scope.
Make sure the digitized $v(t)$ in your VI matches your expectations.
Look at the frequency spectrum in the FFT graph. It should show two nearly symmetric spikes near $\pm 20\,\text{Hz}$. If everything is in order, your VI is probably wired up right.

Next, record your observations in your lab note book as you try the following:
Doubling the amplitude of the function generator and observe the change in the frequency spectrum;
Add a DC offset at the function generator and observe the FFT output.
Change the frequency of the function generator (changing nothing in your VI) and see that the spikes move in the correct fashion.

Observation of aliasing

Systematically observe the FFT spectrum for 2.0 Vpp sine waves from the function generator (no DC offset) with frequencies of 50, 100, 200, 300, 400, 475, 500, 512, 525, 550, 600, 700, 800, 900, 1000, and 1020 Hz. Sketch neatly in your lab notebook the frequency spectra observed in a way that makes comparison easy and shows the trends (stack them with a common scales for the axes. What happens to the spectrum when the signal frequency increases beyond the Nyquist frequency? Based on your measurements what is the connection between the true frequency from the generator, the apparent frequency as found by the output of the FFT routine, and the sampling frequency for frequencies greater than $f_s/2$?

Harmonic content of non-sinusoidal waveforms - square, triangle waves

Carefully examine the FFT spectrum for a 5 Hz square wave, 2.0 Vpp. Record which frequencies are present, what multiples of the 5 Hz fundamental they are, and the magnitudes of their Fourier coefficients. (dig to get three"significant" figures for the values of the coefficients; don't settle for a truncated fixed point value like .01, coax LabVIEW to display some more digits.) Do the same for a triangle wave. Compare and contrast the harmonic content of these two waveforms. Interpreting the c_n's further can be tricky, since your graph is only the magnitude, and the real and imaginary parts determine sign/phase information about how each harmonic contributes (or the signs of the corresponding a_n and b_n in the series expansion in terms of cosines and sines).

Harmonic content of non-sinusoidal waveforms - amplitude modulation

Examine the FFT spectrum of an amplitude modulated (AM) waveform using the bench-top function generator or your previously built AM waveform generator (there is an AM generator Vi in the utility folder, too). To use the bench-top generator set it to sine wave, $f = 500$ Hz then select MOD = AM, Modulation frequency = 50 Hz, depth = 100%, shape = Sine) to generate data for the FFT. Use a carrier or signal frequency of 500 Hz and a modulation frequency of 50 Hz. Use a sampling frequency of 20000 per second and a large number of samples 2^n, e.g. n=14 or 15. (Be sure to use a power of 2.) What frequencies do you expect to see, and what frequencies do you find? The AM waveform comes from

$$v(t) = A_o \left[1 + m \sin(2\pi f_{mod} t)\right] \cdot \sin(2\pi f_{sig} t)$$

where $m = 1$ corresponds to a modulation depth of 100% on the function generator. A trigonometric identity ($\sin A \sin B = ...$) can help in sorting out the frequency spectrum that this formula implies.

Signal recovery and filtering in the frequency domain.

You are provided with a custom data file containing a signal buried in a lot of noise, both 60 Hz interference and random white noise. Your task is to use the FFT capabilities to identify the probable frequency and amplitude of the true signal. You'll need to read the file into LabVIEW (there's a read-from-spreadsheet-file VI that can read simple text files), then break the data into separate time and signal voltage arrays, and determine the number of points and calculate the sampling frequency (do this in your code, so it can handle different file lengths and sampling rates, and remember $f_s = 1/\Delta t$, the time interval between measurements). Then build a digital filter in LabVIEW, i.e. adjust the Fourier components of the signal to reduce irrelevant noise and interference (notch filters, band pass filters, high pass or low pass are all possible by simply either leaving alone or setting to zero, as appropriate, the various Fourier coefficients of the signal. Once filtered, use the *inverse* FFT VI $\mathcal{F}'(x)$ to convert the filtered Fourier coefficients back into the time domain (= the filtered $v(t)$), and display the filtered signal on a 3rd X-Y graph. From this graph estimate the frequency and amplitude (and maybe phase) of the buried signal assuming it can be written as $C \cos(2\pi f t - \phi)$.

Save and print out the final polished spectrum analyzer / filter VI, showing the original and filtered $v(t)$'s and the original's FFT spectrum. Include both front panel and wiring diagrams in your lab notebook.