

ME220 Lab #4 Create VIs for Accelerometer Analysis

Due: 5/7/08, before 5:00 pm.

Goals of this Lab:

This is the first part of a two-part lab. We're going to build up some LabVIEW VIs to analyze the frequencies present in two signals. In the next lab we will use these VIs to characterize an accelerometer and the actuator used to excite it. This lab will mostly consist of work in LabVIEW. You will also be measuring the response of a low-pass filter and coupling noise as a reference.

Acknowledgement:

Zachary Nelson of National Instruments orchestrated a donation of the LabVIEW Software and the National Instruments Data Acquisition hardware that we'll be using in the lab this quarter. Zach will visit sometimes during the quarter, so please be nice to him!

Part 1 - A little bit more about your friend, the Fast Fourier Transform (FFT)

In previous labs, we have used the FFT (which is a type of discrete Fourier Transform) to transform a time-domain signal into its frequency-domain components. If we know what frequency our signal should be at, we can read off the amplitude of the signal at that particular frequency, ignoring the FFT values at other frequencies. In using the FFT, we've been brushing a few things under the rug, which we hope to address briefly below.

First, the FFT value computed at each frequency is really a complex quantity, having both real and imaginary components. LabVIEW reports the magnitude of that complex value, which is what we've been using (correctly) as the amplitude of that particular frequency component. You've most likely used the RMS value of that amplitude (picked in the Spectral Measurements Express VI). If you want the amplitude of the particular sinusoidal component instead of its RMS value, simply pick the "Magnitude (Peak)" option.

If you are curious, this complex value comes from the use of Euler's equation in the formula used to compute the discrete FFT. For $X_k = \text{fft}(x(N))$:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi}{N}nk} \quad k=0, 1, \dots, N-1$$

where x_n is the data sequence and N is the number of samples.

There is also a phase associated with each FFT value. The Spectral Measurements Express VI gives you an output named phase, which is a series giving the phase at with each frequency. If the input is a cosine wave with frequency F (*i.e.*, at time 0 the amplitude is at its maximum), then the corresponding phase at frequency F is 0. If it is a sine wave instead, there is a phase shift of $-\pi/2$ radians (LabVIEW uses radians by default, but you can change it to degrees), because

$$\cos\left(2\pi Ft - \frac{\pi}{2}\right)$$

In other words, if the peak magnitude output of the FFT block at frequency F is A and the phase output is θ , then the component of the time-domain signal at frequency F (with reference to $t=0$ at the start of the time-domain signal) is:

$$A \cos(2\pi Ft - \theta)$$

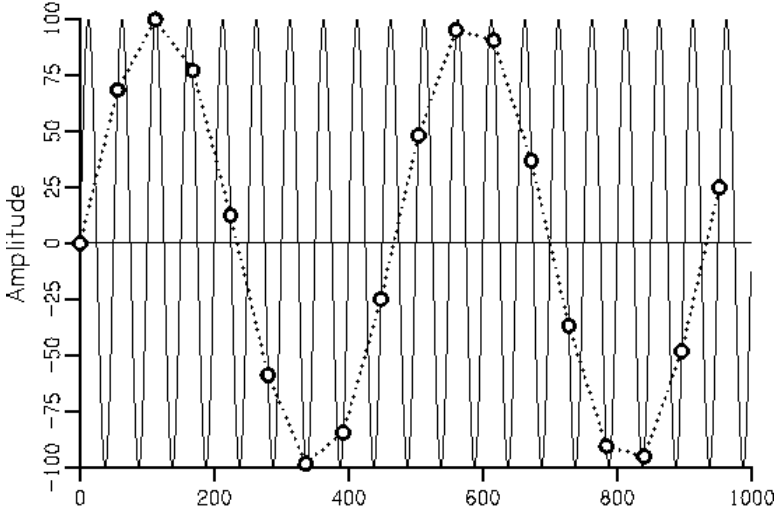
Because 0 radians is the same as 2π radians, there is no unique way to express phase. The LabVIEW Spectral Analysis block reports the phase shift between positive and negative π by default.

In practice, when we acquire an experimental signal, we often don't have precise control over whether we start recording data at the start of the cosine wave or somewhere in the middle. Thus, the phase at a particular frequency is probably not predictable from one measurement to another. However, the relative phases of two data points in the same FFT or the relative phases of two FFT values where the data was acquired at the exact same time can contain useful information. The relative phase difference between different frequency components can also be important in designing filters, measuring audio components, measuring circuit impedances (which have a phase), in image processing, and in structural dynamics.

Signal characteristics that ensure a meaningful FFT result are:

1. Sampled data contains an integer number of cycles of a periodic waveform.
2. Signal is band-limited (has frequency components only over some range, not everywhere)
3. Signal has stable frequency content.

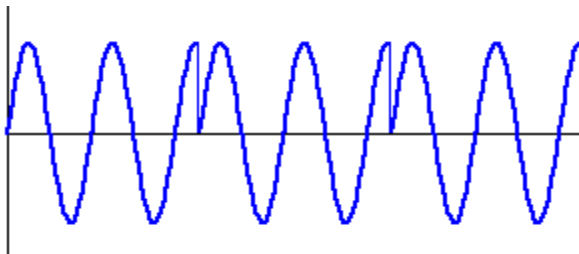
A note on point #2 above: When you sample at frequency F, the resulting signal has frequency components from 0 to F/2 (i.e. you have to sample at least twice for every period of a sine wave in order to reconstruct it, or you will be trying to represent the wave by a single point - this is the famous Nyquist relation). Frequencies in the signal larger than F/2 get "aliased" down and appear at frequencies between 0 and F/2. Here's a picture to drive that point home, where the sampling rate is much slower than the signal frequency, so the resulting discrete samples look like a signal of a lower frequency (lower than F/2).



In order to increase the frequency resolution of the FFT increase the number of points you feed to the FFT (you saw this in Lab #1). In order to improve the accuracy of the FFT magnitude and phase, take averages over longer times. As a general rule of statistics, the average error associated with measuring a quantity

goes down as $\frac{1}{\sqrt{N}}$ where N is the number of measurements you average.

The way the FFT is defined it considers its input to be a periodic signal; (i.e., it pastes another copy of the signal onto the end of the signal). So if you had 2.25 cycles of a sine wave repeated periodically it would look like the picture below. Note the sharp edges created by the repeats.



In order to get around this problem, often the input signal is multiplied by a "window function" before performing the FFT computation, with the intend of smoothly tapering the signal to 0 on both ends so that when the FFT operation implicitly repeats it there are no sharp edges created. Now you know what the "Window" pull-down menu in LabVIEW is specifying! Common smoothing windows used include:

1. Rectangle (no window) - Noise measurement
2. Hanning and Hamming - Frequency measurement
3. Flat Top - Amplitude measurement

LabVIEW FFT functions allow us to do the following:

1. FFT spectrum with peak amplitude (i.e. magnitude)
2. FFT spectrum with RMS amplitude (peak amplitude divided by $\sqrt{2}$)
3. Power spectrum (RMS magnitude squared - gives the energy)
4. Power spectrum density (measure the energy content within the frequency band)

For the write-up: Write, in your own words, what the Nyquist criterion means.

Part 2 - Build a VI to find the amplitude and phase at a particular frequency

In general, and in the rest of this lab in particular, we will be concerned with the amplitude and phase components of a signal at a particular frequency (like the frequency at which we expect our signal to be)

In this part, you will build a VI that will take two inputs: a waveform and a particular frequency F. It should output the magnitude ("magnitude (peak)") and the phase of the FFT at this frequency F.

Start by putting down the input control to read in the waveform. Find it under by right-clicking in the Front Panel and then selecting "All Controls" => "Classic Controls" => "classic I/O" => "Waveform."

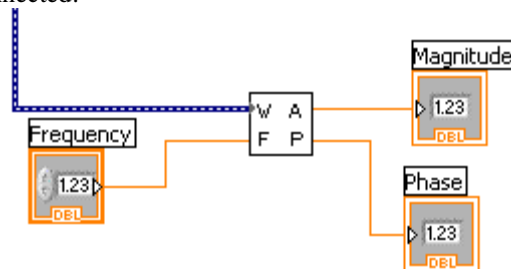
In order to use the VI you are creating, you later will hook up a signal like one coming from a "Simulate Signal" block, but now we need a control that will accept such a signal. Now the rest is up to you!

Make sure you output the phase in radians and report the peak FFT amplitude on a linear (not dB) scale. Also make sure to label your connectors in a way that makes it intuitive to use.

Hint #1: You may find the VI "Get Y Value" helpful. (It can be found under "All Functions" => "Waveform"). Be careful that you feed it the correct input at the mode pin. We don't expect you to know all about it just by looking; try the help information (right click on the box, then select "Help").

Hint #2: If you don't recall how to create a VI and define connectors to it, review exercises 1 and 2 that you did back in Lab #1. You'll need to create connectors in order to be able to test it.

Make sure your VI has two inputs (the waveform and the frequency) and two outputs (the peak amplitude (i.e., complex magnitude) and phase at the supplied frequency). Your input should be able to hook up to a dotted blue bus coming from a "DAQ Assistant" VI or "Simulate Signal" VI and should have two outputs that are simply numbers (so when you wire it up you should see thin orange wires). It should look something like this when connected:



The final step is to test out your VI. Do this by creating a new VI in which you generate a sine wave from a "Simulate Signal" box and feed it to the VI you just created. In the box, set the frequency to 100, amplitude to 1, and add noise of amplitude 1. Set the frequency input to 100 initially, but experiment with other values as well. It may be helpful to make a plot of the whole FFT (both amplitude and phase) that you can look at to see if your results match. Convince yourself that your VI works as it should.

This will look very much like the above picture (plus some other stuff not shown above). There are many ways to insert one VI as a sub-part of another. One way is to open both VIs in such a way that you can

click and drag the icon from the upper right corner of the sub-VI's front panel onto the block diagram of your new VI. (If the upper-right corner is still showing the connector instead of the icon, you may need to select "Show Icon").

For the write-up:

- Include a picture of the block diagram of your VI.
- Include a picture of the block diagram of your VI in your test circuit.
- Include values for the amplitude and phase at 100 Hz as determined by your test circuit, as described above. Provide brief justifications for their values based on the input signal to your VI.

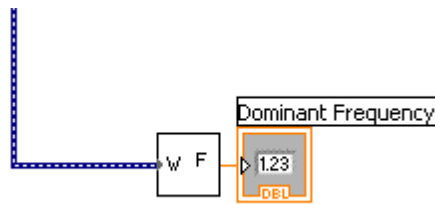
Part 3 - Build a VI to measure the dominant frequency:

In many applications, our input data is relatively clean, meaning that if you look at the magnitude output of the FFT you'll see one dominant peak (plus other tiny peaks that are noise). It may be of interest to determine which frequency that is. For example, if you hook up a lab function generator and acquire data from it using LabVIEW, you may have a rough idea of what that frequency is from the knob on the function generator but may want a more precise estimate of its frequency.

In this part, you will build a VI that will take a waveform as its input signal and identify the dominant frequency component. This is defined as the frequency at which the magnitude of the FFT is the greatest. It should output the value of that frequency.

Hint: You may find the "Statistics" VI helpful (under "Signal Analysis").

Make sure your VI has one input (the waveform) and one output (the frequency of the dominant component). Your input should be able to hook up to a dotted blue bus coming from a "DAQ Assistant" VI or "Simulate Signal" VI and should have one output that is simply numbers (so when you wire it up you should see a thin orange wire). It should look something like this when connected:



The final step is to test out your VI. Do this by creating a new VI in which you generate a sine wave from a "Simulate Signal" box and feed it to the VI you just created. In the box, set the frequency to 100, amplitude to 1, and add noise of amplitude 3. It may be helpful to make a plot of the FFT of the input waveform. Convince yourself that your VI works as it should (by changing the "Simulate Signal" block and seeing how the output responds, for example). The block diagram of your test VI will look very much like the above picture (plus some other stuff not shown above).

For the write-up:

- Include a picture of the block diagram of your VI.
- Include a picture of the block diagram of your VI in your test circuit.

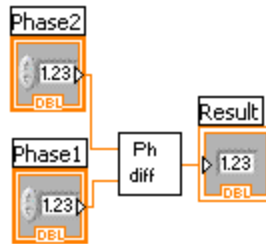
Part 4 - Build a VI to determine the relative phase of two signals:

Often it is important to know the relative phase of two signals. For example, if you want to verify that a capacitor introduces a phase shift of $-\pi/2$ as predicted you could apply a sinusoidal input and measure the phase of the signal before and after the capacitor.

In this part, you will build a VI that will take two phases and compute the relative phase. If the inputs are Phase1 and Phase2, we define the relative phase as $(\text{Phase2} - \text{Phase1})$. However, we also want to report the phase in convenient range, so if the calculated value falls outside of the range $[-\pi, \pi]$ then add whatever multiple of 2π necessary to create an output within the range. The modulo function may be helpful here. If you express your initial result X as $(2\pi * n) + \theta$ for some integer n and θ in $[0, 2\pi)$, then $\text{mod}(X, 2\pi) = \theta$.

Hint: You may find the "Formula" Express VI helpful (under "Arithmetic and Comparison").

Make sure your VI has two inputs (the two phases) and one output. It should look something like this when connected:



The final step is to test out your VI. Do this by creating a new VI in which hook up your VI to two inputs and one output. Your block diagram should look basically the same as the picture shown above. Perform the calculations listed in the write-up section by manually inputting the inputs in the Front Panel and recording the corresponding outputs. Convince yourself that your results are correct.

For the write-up, include the results of the following computations:

1. Phase2 = 1, Phase1 = -1, Result = ?
2. Phase2 = 3, Phase1 = -1, Result = ?
3. Phase2 = -2, Phase1 = 2.5, Result = ?
4. Phase2 = -1, Phase1 = 1, Result = ?
5. Phase2 = 6, Phase1 = 0, Result = ?

Part 5 - Build a VI to acquire two analog channels and separate them:

One final thing we need to get set up for the next lab is to be able to acquire two signals simultaneously. It turns out that LabVIEW only lets you have one "DAQ Assistant" VI that takes analog inputs (yes, it's the same box you've used before in LabVIEW). So we have to do a little trickery to read in the two signals with the single DAQ Assistant block and then separate them to do analysis on them separately.

Start by placing a DAQ Assistant block on the block diagram. Select analog input, then voltage, then channel 1 ("ai1") just as you have in the last two labs. Set the range from -10 to 10, differential configuration. Set the sampling frequency to 1000 Hz and to acquire 100 samples for now, although you may wish to change this later on.

Now, to add another channel of analog input: Still within the DAQ Assistant dialog box, click on the green plus sign above the channel list. Choose channel "ai3" this time. This adds another channel to the channel list. Set the range to -10 to 10 volts, again in differential configuration.

A few notes to help later on:

- You can change which physical pins a particular channel is connected to by right clicking on the channel name.

- Setting the sampling rate and number of samples affects all channels. For example, there is no way to sample one channel at 1000 Hz and another at 2000 Hz, at least not with our current set-up.
- The DAQ Assistant box only contains one output pin labeled "data" no matter how many channels it is acquiring. It just adds an extra row to the output array, if you want to think in those terms. More on this later on.
- The physical connection to what LabVIEW calls "ai3" is between pins 30 and 63 on the green connector block. "ai1", which we've used on previous labs, is between pins 33 and 66. Pins 63 and 66 are considered as reference to the two inputs at pins 30 and 33 (i.e. they would usually be hooked up to ground).

Hook up the function generator to channel ai1. Feed it a sine wave at about 100 Hz with amplitude of about 8 volts with no offset. Turn on your power supply as well and feed the +5 volt input to channel ai3.

Place a Waveform Graph at the output of the DAQ Assistant block. You should see two waveforms superposed on the graph, both the sine wave and the constant 5 volts.

So the DAQ Assistant block is outputting both waveforms on that single output (which looks like a thick blue dotted line). Also, there is a sequence of times associated with that data, which is the same for both waveforms. LabVIEW is smart enough to plot the signals versus time when you tell it to draw a waveform.

Now we want to separate the two waveforms so we can manipulate them separately. Create a "Select Signals" block (under "Signal Manipulation" on the functions palette). Click OK on the dialog box that comes up without doing anything.

Now wire up the output of the DAQ Assistant block to the Select Signals block. Run the VI. Now the Select Signals block has an idea of what channels the DAQ Assist block is outputting. Double-click on the Select Signals block (or right-click and go to Properties) and then move the first channel ("Voltage" unless you've changed the default name) from the "Unselected Signals" side to the "Selected Signals" side. Close the dialog box and connect a waveform graph to the output of the Select Signals block. Run the VI again. On the output of the Select Signals block you should only see the signal that is coming in on ai1 (it should be the sine wave at 100Hz).

Now repeat the process to create another Select Signals block that isolates and displays the signal coming in on ai3 (it should be about +5 volts). The only thing to do differently is which of the signals you select within the dialog box.

Save your results to use as a starting template for Part 6 and for the next lab.

For the write-up:

- Include a screen shot of your VI.
- Answer the following question: if you wanted to use ai4 instead of ai3 as your physical input channel, what pins on the connector block would you use?

Part 6 - Use your VIs to measure the response of a low-pass filter

In this part of the lab we will measure the attenuation and phase shift associated with a low-pass filter.

Construct a low-pass filter on your breadboard using a 0.01 μF capacitor and a 1 $\text{M}\Omega$ resistor. Use a sine wave from the function generator as your input signal. Connect the physical wires for ai1 to the input of the filter and the physical wires for ai3 to the output of the filter. See the notes for Lecture #2 if you can't recall what a low-pass filter looks like.

Start with the LabVIEW file you created in part 5, but be sure to save a copy of the original.

Add the VIs you created in parts 2-4 of this lab to the block diagram (and possibly more things) in order to see the relative amplitude (output amplitude / input amplitude of the component at the driving frequency) and the relative phase (output phase - input phase). You may find it helpful to corroborate your findings with waveform graphs, at least when debugging.

Sweep frequencies from roughly 5 Hz to 500 Hz (take about 6 data points) and record the relative amplitude and relative phase at each data point. Be sure to also record the actual frequency used as determined by your VI (not just eye-balled from the function generator knob). Compare those with the theoretical values (see Lecture #2 Notes for formulas). Comment on any deviations. Plot your data and corresponding theoretical values as a function of frequency.

Try setting the input frequency of the function generator at about 1000 Hz. Is your sampling rate fast enough to represent this signal? (Hint: remember Nyquist? Even then, is this *really* fast enough?) If not, change your DAQ Assist block so that it is fast enough. At this frequency, the low-pass filter should have attenuated the input signal to about 1.6% of its original value. What relative amplitude do you observe? Try removing the resistor so as to disconnect the input from the output of the filter, so theoretically the output is at ground (connected through the capacitor). Thus we would ideally find that 0% of the input signal is found at the output. Is this the case? If not, offer some brief explanation. How does this result tie into the interpretation of the frequency response curve you just generated?

For the write-up:

- Include a picture of your LabVIEW block diagram
- Include your frequency response curve showing both actual and theoretical curves.
- Include your comments about deviations of actual data from theoretical.
- Include your figures for the amount of cross-coupling you observed at 1000 Hz without any connection and comment on how that may affect your interpretation of the frequency response curve and the deviations from theory.