ENGR 325 | **Lab 4** | **Walla Walla University**
Winter 2021 | **Sampling theory and the FFT** | **Inst. Mark Haun**

In this lab you will learn how to use the Fast Fourier Transform (FFT) to examine the frequency spectrum of sampled data. You will use the same A/D converter, connection board, and MATLAB support function as last week, but now the input signal will be provided by a Digilent *Analog Discovery* module. You will observe the effects of changing the sampling rate and the length of the captured data, and learn how to produce plots of the frequency content with properly scaled X and Y axes (frequency and amplitude).

Objectives:

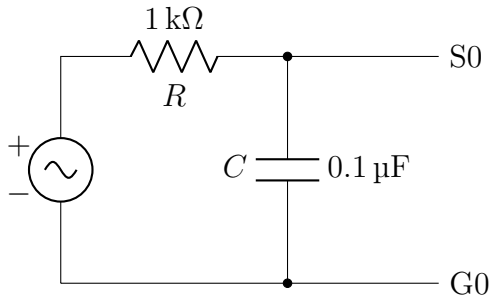
- Continue gaining experience with A/D converters.
- Learn proper technique for sampling an analog signal so the Fourier transform can be used.
- Observe the effect on the spectrum of changing the sampling rate and the data record length.
- Use the Fourier transform to identify the frequencies in a signal.

Equipment provided:

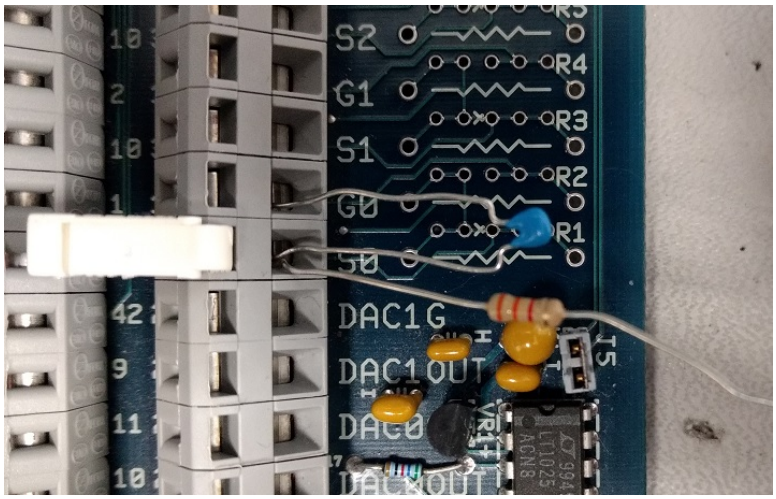
- Digilent Analog Discovery 2
- Linux PC with Digilent Waveforms software
- Windows PC with ADC card and MATLAB
- ADC connection board with resistor and capacitor for anti-alias filtering
- Tektronix TBS2000 series oscilloscope

Procedure:

1. Use the supplied 1 kΩ resistor and 0.1 μF capacitor to form a simple anti-alias filter at the input to the same S0/G0 terminals we used in last week's lab:

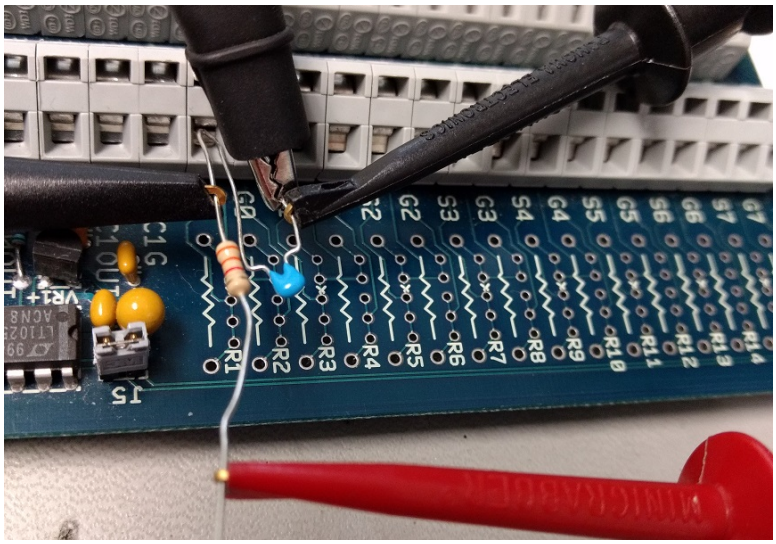


   The capacitor is connected across S0 and G0, and the resistor is in series with the signal input. This forms a low-pass filter with a corner frequency of $1/2\pi RC \approx 1600$ Hz. A correct circuit should look like this:



2. Make sure the BNC connector board is firmly seated into the side of the Analog Discovery, and the USB cable from the pod is plugged into one of the ports on the front of the Linux PC. Using the same cable as last week, run a connection from "W1" on the BNC connector board to the input of the anti-alias filter: Connect the positive clip to the free end of the resistor and connect the negative clip to the capacitor leg in G0.

3. Connect an oscilloscope probe across S0 and G0 (retracting clip to S0, alligator clip to G0). The oscilloscope is not used for measurements in this lab; it is only there to verify that a signal is present. Your connections should look like this when you are done:



4. Log into both the Linux and Windows PCs at your lab bench—we will be using both of them. The same keyboard, mouse, and display are connected to both machines; press the Scroll Lock key twice to switch between them.

5. On Linux: Find the `test_signal.csv` and `mystery_signal.csv` files on the course web site (in content under lab), and download them to your home directory. Launch the Waveforms application by clicking on the grid icon and typing letters until you see the W icon, then selecting it. Inside the Waveforms application, select Wavegen from the options on the left-hand side.

6. Inside the Wavegen window, find the dropdown selection which says Simple and change it to Custom. Click Import and choose `test_signal.csv` from your home directory. Leave all of the parameters as-is except for Sample Rate—change that to 100 kHz, then click OK. In the dropdown menus next to the waveform display, change Amplitude to 5V. Then, click the Run button (green arrow). Verify that you have a $\sim 10\,V_{pp}$ waveform on the oscilloscope.

   The `test_signal` waveform going into the A/D converter is a sum of five sinusoids of varying amplitudes as follows:

70.2 Hz	1.0 V
268.6 Hz	0.4 V
396.7 Hz	0.7 V
399.8 Hz	0.7 V
476.1 Hz	1.4 V

   The overall waveform has a period of 0.32768 seconds. (Each of the sinusoids completes many cycles during that time, but the entire pattern takes 0.32768 s to repeat.)

7. On Windows: Download the `lab4.m` MATLAB script template from the course web site and save it to your STEM drive or other convenient location. Start MATLAB. Test the `msgets0` function at the command line to verify that the ADC is getting good data:

```
[d,t] = msgets0(3125, 1024);
plot(t,d); grid on;
```

   You should see a complicated signal which looks smooth when you zoom in on it.

   For reference, here is the documentation for the `msgets0` function:

```
[data,time] = msgets0(rate, Npoints, gain);  % Reads A/D chan 0 "single-ended"
```

   Inputs:

`rate`	number of samples per second
`Npoints`	number of data points
`gain`	optional selectable gain (1, 10, or 100 with a default of 1)

   Outputs:

`data`	vector of samples from the ADC, as signed integers (see below)
`time`	time vector corresponding to the data samples

READ THIS: It is highly recommended that you run any scripts containing calls to `msgets0` by typing their name on the command line. If you run them from the editor (using the GUI button), you may get bad data and/or leave the ADC in an unknown state requiring a reset.

---

8. `lab4.m` is a template for a MATLAB script which you can use throughout this lab:

```
close all;
clear all;

sampleRate = 3125;   % in Hz
```

```
numSamples = 1024;

[d,t] = msgets0(sampleRate, numSamples);
d = k * d;  % Convert to volts; use your k from last week

f = % Create a vector of frequencies here

% This calculates the FFT and then takes the magnitude of the complex output.
% The (2/numSamples) scaling is necessary to get amplitude in volts.
D = (2/numSamples) * abs(fft(d));

plot(f, D, '.-'); grid on;

set(gca,'FontSize',14);
xlabel('some text');  % Fill these with proper axis labels and title
ylabel('some text');
title('some text');
```

The template has some missing parts which you will need to fill in:

(a) Use the conversion factor you derived last week to convert the `d` vector from raw ADC values to voltage. Recall that the ADC has an input range of ± 5V ($10\,V_{pp}$) which is expressed at the output of the `msgets0` function as 16-bit signed numbers from -32768 to +32767.

(b) Create a vector of frequencies `f` for the x axis of your plot. Recall from lecture that the FFT takes N time samples as input and produces N frequency samples as output, evenly spaced from 0 to f_s Hz.

Hint: In MATLAB the expression `[0:N-1]` will produce a vector of integers from 0 to $N-1$ inclusive; the length of this vector is N. The syntax `[a:b:c]` will generate a vector with arbitrary spacing, starting at a, stepping by b, and ending at or before c .

(c) Edit the axis labels and the plot title to match School of Engineering standards. Each axis label should state the variable and its unit. The title should be as specific as possible.

9. Using the default parameters (1024 samples @ $f_s = 3125$ Hz), run your script and zoom around the plot. Do the peaks in your FFT output match the frequencies and amplitudes of the sinusoids comprising the input signal? (Within the plot window, if you turn off zoom you can click on the peaks to show a box with X and Y values.)

   You will notice that the full plot is always symmetric around the middle, which should be the Nyquist frequency $f_N = f_s/2$. Because the frequency spectrum of sampled data is periodic, the second half of the plot, from $f_s/2$ to f_s, is equivalent to the negative frequency range of $-f_s/2$ to 0. In fact, you will sometimes hear that the FFT output goes "from 0 to $f_s/2$, then from $-f_s/2$ to 0." For real input signals, the magnitude spectrum (which is what we are looking at, because of `abs(...)`) is always the same for positive and negative frequencies.

10. Experiment with different sampling rates and numbers of samples. Lots of learning can occur in this step. Don't rush. Document what you observe, and ask your instructor for help understanding what you are seeing.

(Hint: You can use the `save` command (try `help save`) to save all of the variables in your workspace, and restore them with `load`. This could be useful if you want to continue exploring later at your own pace. In this case, it might be wise to save to a different filename after each trial, so they won't overwrite each other. You can also save figures to `.fig` files from the plot window, which will let you customize things like zoom and axis labels offline.)

11. Two of the input frequencies are quite close together (396.7 Hz and 399.8 Hz).

    (a) If you keep the sampling rate (f_s) constant and change the number of samples (N), what effect does that have on your ability to distinguish two separate peaks for these frequency components?

    (b) Write down the expression for the FFT frequency spacing (frequency resolution) in terms of f_s and N. Show how this depends on the total recording time in seconds.

12. Change your script to collect 1024 samples @ $f_s = 600$ Hz. At this sampling rate some of the higher-frequency sinusoids in the input will alias to lower frequencies. Calculate the expected [apparent] frequencies for all five sinusoids (showing your reasoning), then compare your predictions with the plot.

13. Go to your Linux screen and press the Stop button to stop the waveform generation. Then use Import to load the other waveform file, `mystery_signal.csv`. Follow the procedure in step 6 to activate this waveform, and make sure you can see it on the oscilloscope. It contains a different set of sinusoids, but the repetition period is still the same at 0.32768 s. The highest frequency in this waveform is less than 1 kHz.

    Choose your preferred sampling rate and number of samples and make a plot of the FFT spectrum. Record all of the frequency components that you find, with frequencies to the nearest 0.1 Hz and amplitudes to the nearest 0.1 V.

14. Run the function `msstop` to shut down the connection with the A/D board, then exit MATLAB before signing off the Windows PC. On the Linux side, stop the waveform generator by pressing the red stop button and exit the Waveforms application before signing off.

Report guidelines:

As always, there is no need to copy/paste from this document, but you should briefly summarize what you did in your own words. Show your results in context, and describe your observations. Include frequency plots (made by your script) as necessary to support your results. One to two pages, plus graphs and a copy of your MATLAB script, should suffice.