

Dyson School of Design Engineering

DE2 Electronics 2: Signals, Systems and Control

Lab Experiment 1: Introduction to Signal Processing with Matlab(webpage: http://www.ee.ic.ac.uk/pcheung/teaching/DE2_EE/)**Objectives**

By the end of this experiment, you should have achieved the following:

- Have Matlab running on your personal laptop;
- Able to plot signals with Matlab;
- Understand basic Matlab syntax and language;
- Use functions within Matlab;
- Explore how phase affects signal shape;
- Explain the idea of orthogonality and projection of signals.

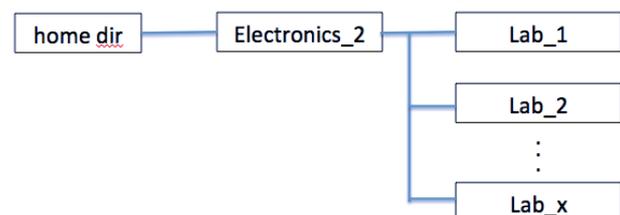
Reminder on Logbook

As in the first year's Electronics 1, you are required to keep an electronic logbook for all the laboratory sessions for this module. You will find a useful instruction on how to keep an electronic logbook on the course webpage. While your logbook will not be formally marked, you are expected to answer some questions during your oral examinations by referring to the electronic logbook.

Getting started with Matlab

For the rest of this experiment, make sure that your laptop computer has Matlab installed and running properly. You would need to include the Signal Processing Toolbox. In this Lab session, you will investigate the ideas covered in Lectures 1 to 3.

Before you start, I strongly recommend that you to create a directory structure for the lab sessions in Electronics 2. A possible structure may look something like this. Trust me – a little effort now will save you lots of time later.

**Exercise 1: Sinusoidal signal generation**

Enter the following Matlab function to generate a sinusoidal signal using the filename: **sine_gen.m**. Test the function to produce the plot as shown.

```

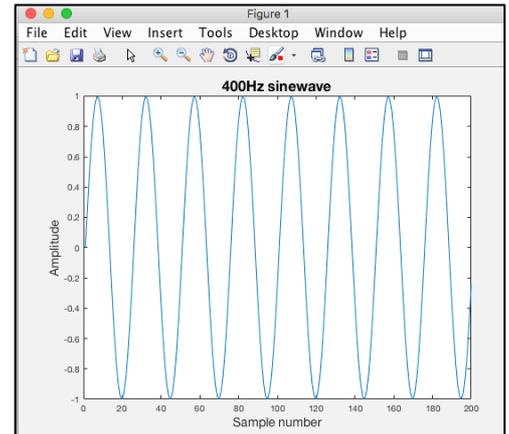
function [sig] = sine_gen(amp, f, fs, T)
% Function to generate a sinewave of amplitude amp, frequency f
% .... with a sampling frequency fs for a duration T
%
% usage:    signal = sine_gen(1.0, 440, 8800, 1)
%
% author: Peter YK Cheung, 9 Jan 2019

    dt = 1/fs;
    t = 0:dt:T;
    sig = amp*sin(2*pi*f*t);
  
```

Test this function in the interactive mode of Matlab:

```
>> s1 = sine_gen(1.0, 400, 10000, 1);
>> plot(s1(1:200));
>> xlabel('\fontsize{14}Sample number');
>> ylabel('\fontsize{14}Amplitude');
>> title('\fontsize{16}400Hz sinewave');
```

Make notes in the electronic logbook about what you have learn, and the results you get.



Exercise 2: Spectrum of the signal

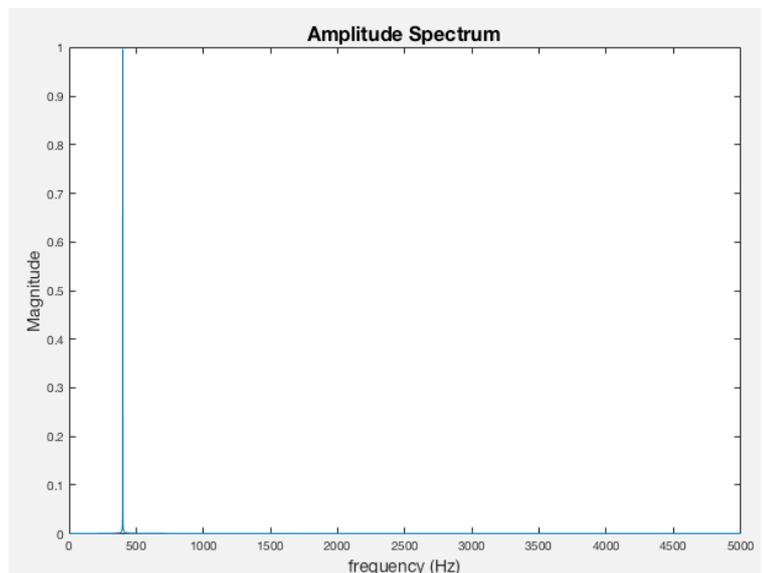
Enter the following function using the filename: **plot_spec.m**. This function uses the Matlab built-in function “fft” to compute the frequency spectrum of the signal. Don't worry about exactly how this works for now. (I deliberately want you to type the code into Matlab instead of doing cut-and-paste. In this way, there is a higher chance that you will remember some of the syntax of Matlab.)

Test this function in the interactive mode of Matlab and you should see the following frequency spectrum plot. You can

zoom onto the peak frequency at 400Hz using the  button.

```
function plot_spec(sig, fs)
% Function to plot frequency spectrum of sig
% usage:
%       plot_spectrum(sig, 1000)
%
% author: Peter YK Cheung, 9 Jan 2019
magnitude = abs(fft(sig));
N = length(sig);
df = fs/N;
f = 0:df:fs/2;
Y = magnitude(1:length(f));
plot(f, 2*Y/N)
xlabel('\fontsize{14}frequency (Hz)')
ylabel('\fontsize{14}Magnitude');
title('\fontsize{16}Spectrum');
```

```
>> s1 = sine_gen(1.0, 400, 10000, 1);
>> plot_spec(s1,10000);
>> title('\fontsize{16}Amplitude Spectrum');
```



Exercise 3: Two tones

Now generate two sinewaves, `s1` at 400Hz (amplitude 1.0V) and `s2` at 1000Hz (amplitude 0.5V), using a sampling frequency of 10kHz and each having a duration of 1.0 second. Add these together as “`sig`” and plot the waveform.

Plot the spectrum of the combined signal.

Exercise 4: Two tones + noise

Assume that your two-tone signal is called “`sig`”, create a noisy version of this signal using:

```
noisy = sig + randn(size(sig));
```

The function `randn(.)` produces a set of random numbers. How many samples? That is decided by the number of data samples in “`sig`” with the function `size(.)`. Now, plot `noisy` and its spectrum.

What have you learned from this exercise?

Exercise 5: Projection using dot product

Let us now treat the two sinusoid signals `s1` (400Hz) and `s2` (1000Hz) as two vectors. You can find the projection of `s1` on `s2` by computing their **inner product** in Matlab:

```
dot_product = s1*s2';
```

(Note that `s2'` is the transpose of `s2`.)

What value do you get? Now create another sinewave `s3` at 401Hz, and find the dot product (or inner product) of `s1` on `s3`. What do you get?

Sometime, dot product is called projection of `s1` onto `s2`. It is a measure of how much of the `s1` signal can be found in `s2` signal.

Finally, find the **dot product** of `(s1 + s2)` on `s1`. What do you get?

Exercise 6: Using PyBench board as a spectrum analyser

The PyBench board is designed to support this module. It runs MicroPython natively on an ARM processor on the Pyboard module. The microSD card contains various Python code that I have prepared for you. Which MicroPython (mPy) script is run depends on the setting of the 3 DIP switches at the top left corner as described below.

SW[2:0]	Function
000	Run user.py
001	Not used
010	Not used
011	Wifi module Test
100	Spectrum of mic signal
101	Bulb board test
110	Pybench board Test
111	Run pybench.py



Connect the PyBench board to a USB power source using a micro USB cable.

Make sure that RED switches on the top left corner is in the 100 position (SW=4):

Press the left button (RST) on the Pyboard (the main processor board) to run the FFT program running on the Pyboard.

The OLED display should show the spectrum of the audio signal capture by the microphone amplifier. The X-axis is from 0Hz to 5kHz. The Y-axis is in dB and shows the range of 0dB to -40dB.

Go to the online tuning fork website on:

https://toolster.net/tuning_fork

or, download a free tuning app onto your phone. Generate a pure tone signal and watch the spectrum being displayed on the PyBench board.

Suggestions to Explore for Yourself

1. Modify Exercise 3 by changing the phase angle of the two tone signals and sum them. Explore how the shape of the time domain signal varies with phase.
2. Below is from Slide 20 of Lecture 3, showing the Fourier series coefficient of a square signal. Assume $A = 1$. Write a Matlab program that gradually increases the number of harmonics components added together and plot the resultant signal. This will show how increasing the number of harmonics result in better and better quality square wave.

If you want to be fancy, you can even animate this in Matlab! In previous year, a student even made a movie of this!

