



# **DE2 Electronics 2**

## **Tutorial 1**

### **Lab 1 & 2 Explained**

Peter Cheung  
Dyson School of Design Engineering

URL: [www.ee.ic.ac.uk/pcheung/teaching/DE2\\_EE/](http://www.ee.ic.ac.uk/pcheung/teaching/DE2_EE/)  
E-mail: [p.cheung@imperial.ac.uk](mailto:p.cheung@imperial.ac.uk)

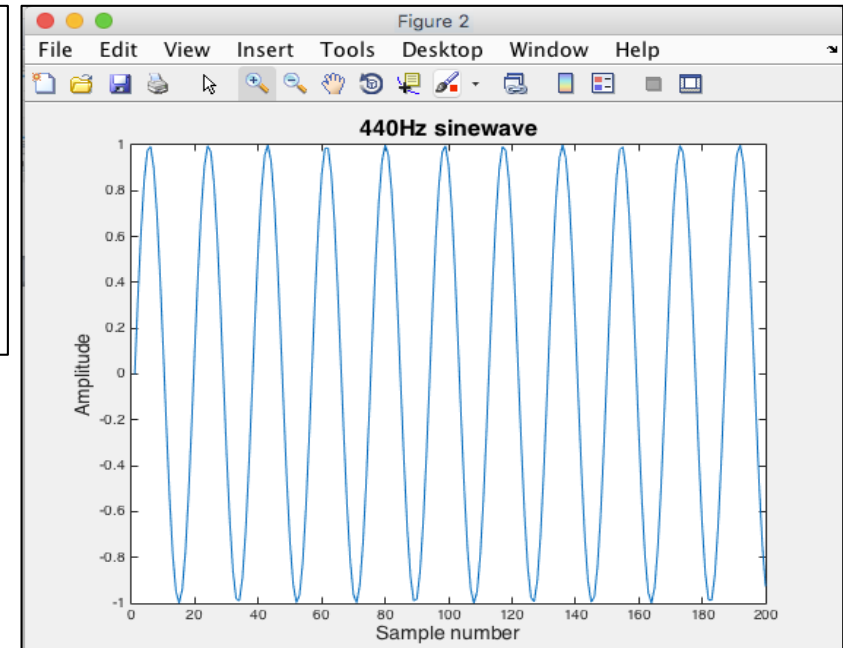
# Lab 1 - Ex 1: The sine\_gen function



```
function [sig] = sine_gen(amp, f, fs, T)% Function to generate a sinewave
%
% .... with a sampling frequency fs for a duration T
%
% usage:      signal = sine_gen(1.0, 440, 8192, 1)
%
% author: Peter YK Cheung, 17 Jan 2017

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

```
>> s1 = sine_gen(1.0, 440, 8192, 1);
>> plot(s1(1:200));
>> xlabel('\fontsize{14}Sample number');
>> ylabel('\fontsize{14}Amplitude');
>> title('\fontsize{16}440Hz sinewave');
```



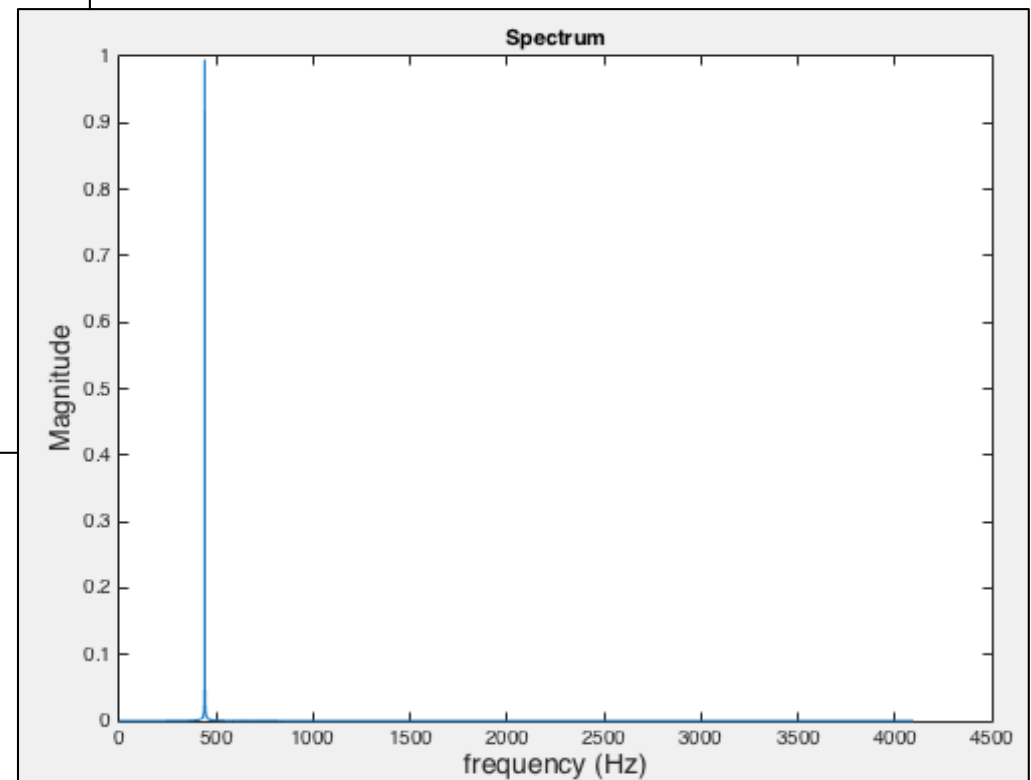
# Lab 1 - Ex 2: The plot\_spec function



```
function plot_spec(sig, fs)

% Function to plot frequency spectrum of sig
% usage:
%       plot_spectrum(sig, 8192)
%
% author: Peter YK Cheung, 17 Jan 2017
    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');
```

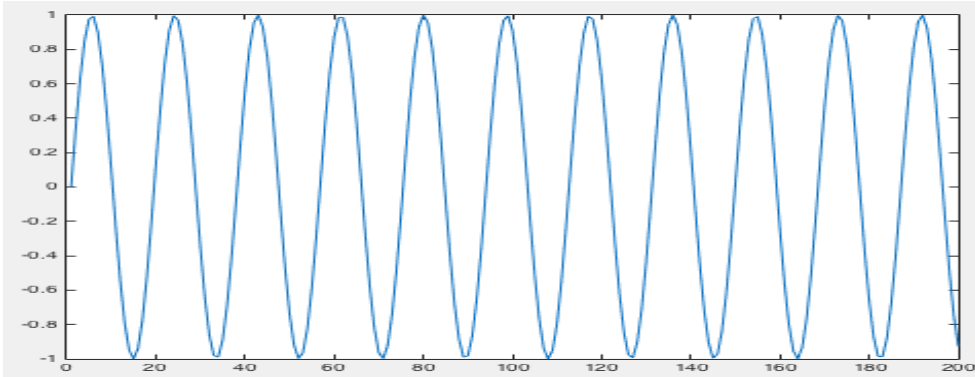
```
>> s1 = sine_gen(1.0, 440, 8192, 1);
>> plot_spec(s1,8192);
>> title('Spectrum')
```



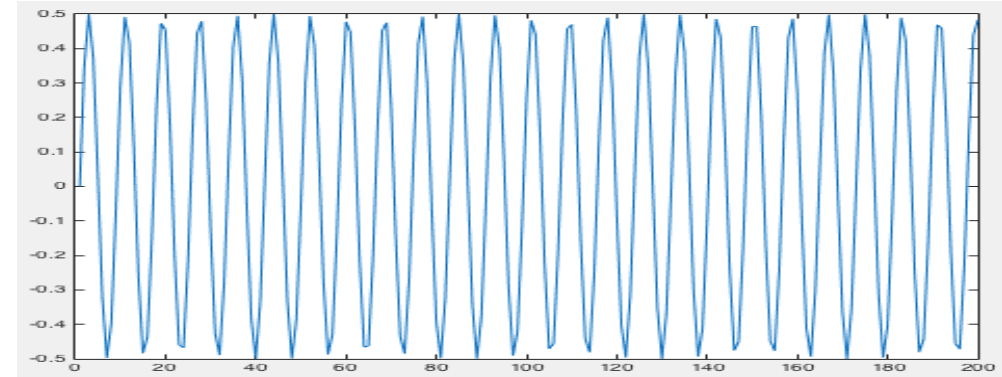
# Lab 1 - Ex 3: Two tones ( $f_s = 8192$ , $T = 1$ )



◆  $s1 = 440\text{Hz}$  sine at 1V

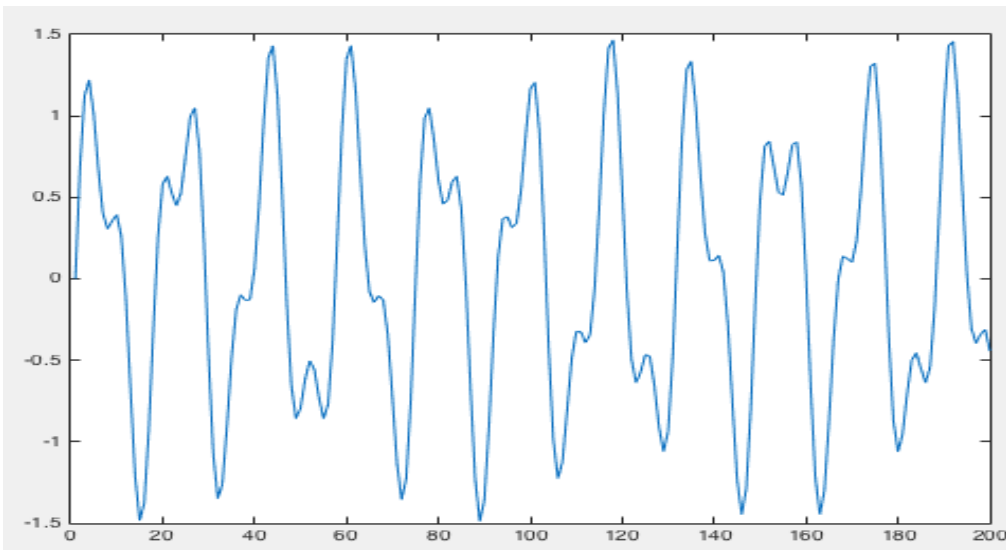


◆  $s2 = 1\text{kHz}$  sine at

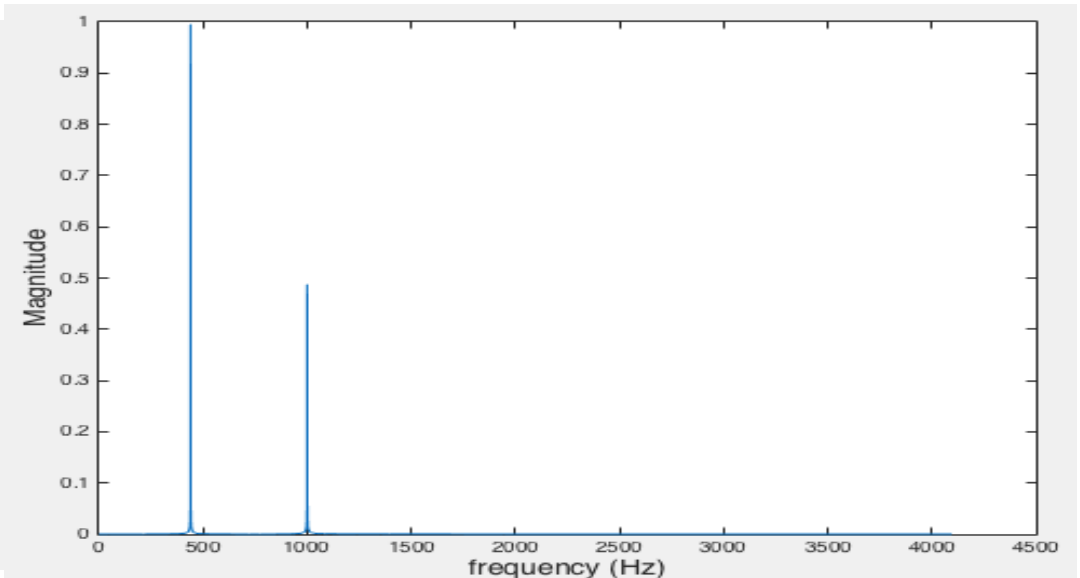


+

◆  $\text{sig} = s1 + s2$



◆  $\text{plot\_spec}(\text{sig}, 8192)$

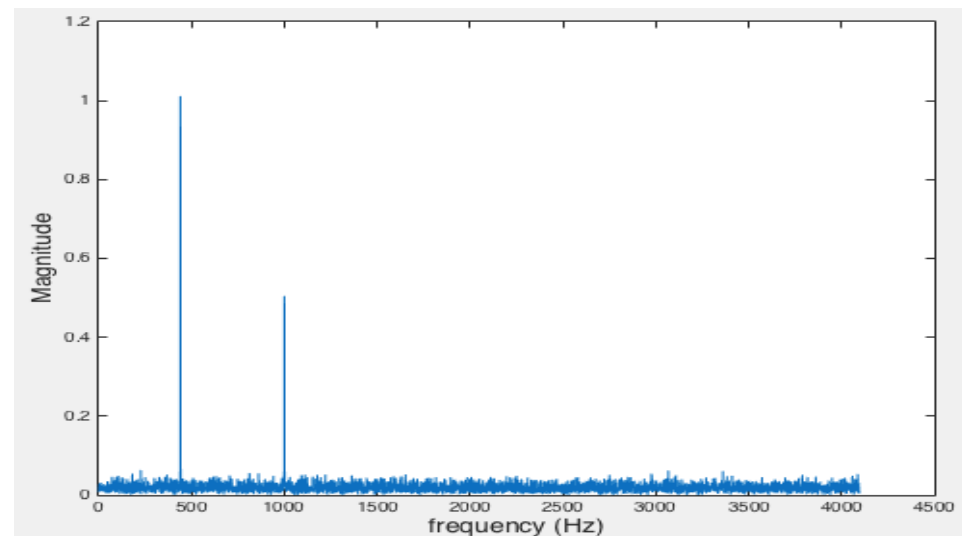
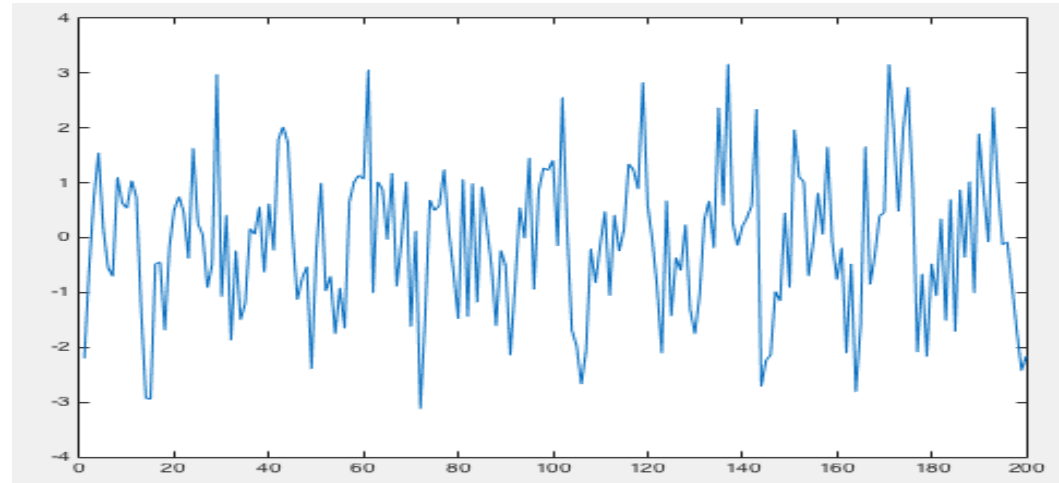


# Lab 1 - Ex 4: Two tones + noisy



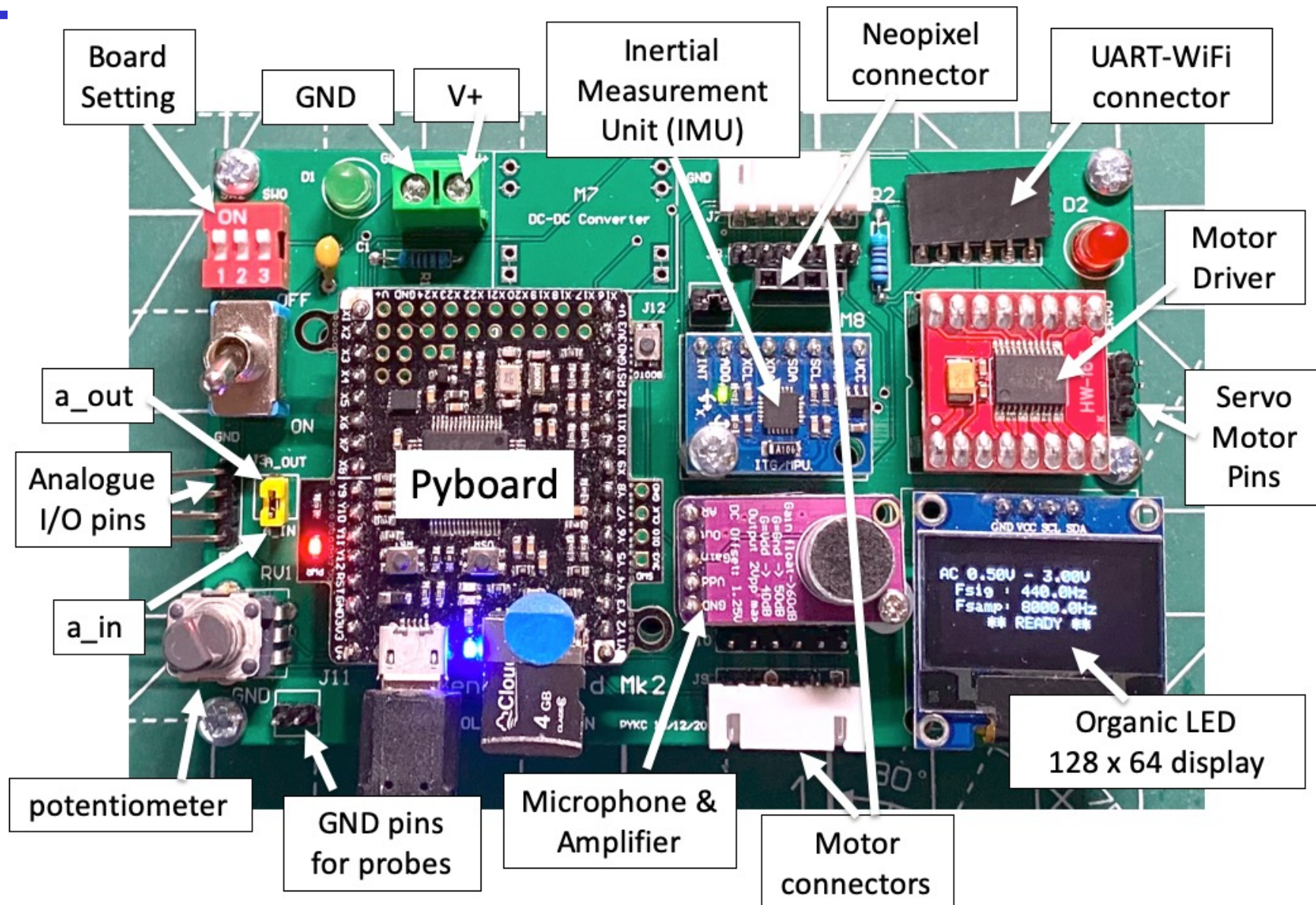
◆ Noisy = sig + randn(size(sig));

◆ plot\_spec (noisy, 8192)

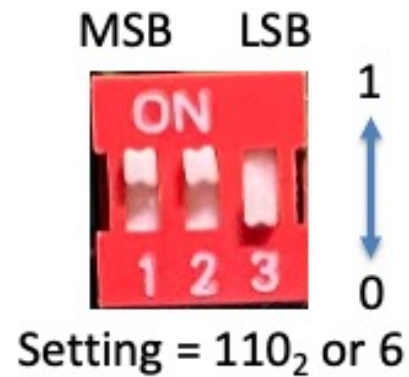




# What's on the Board



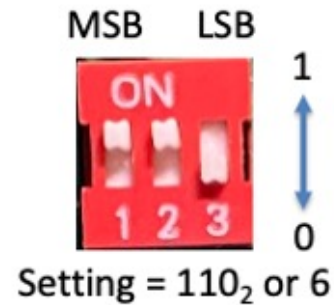
# Board setting



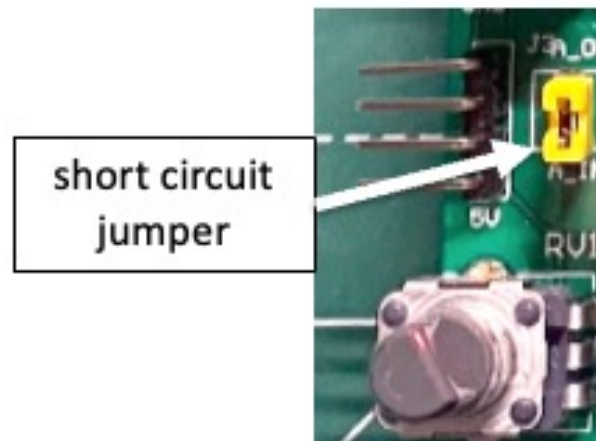
SW[2:0]	Function
000	Run user.py
001	Not used
010	Not used
011	<u>Wifi</u> module Test
100	Spectrum of mic signal
101	Bulb board test
110	Pybench board Test
111	Run pybench.py



# Self-test – Verify PyBench system works

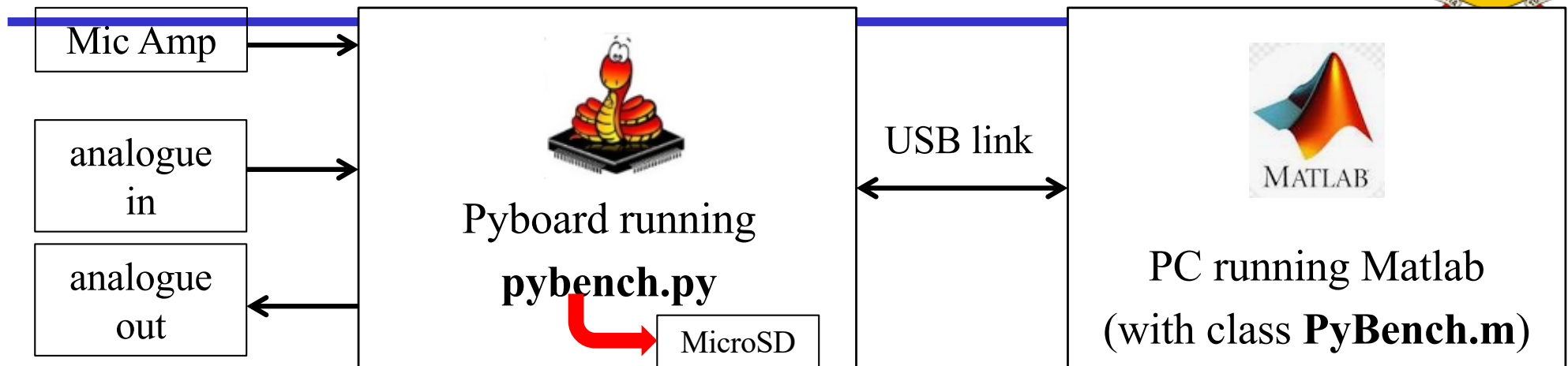


SW[2:0]	Function
000	Run user.py
001	Not used
010	Not used
011	<u>Wifi</u> module Test
100	Spectrum of mic signal
101	Bulb board test
110	Pybench board Test
111	Run pybench.py





# How PyBench works?



- ◆ Look for a serial link on computer:

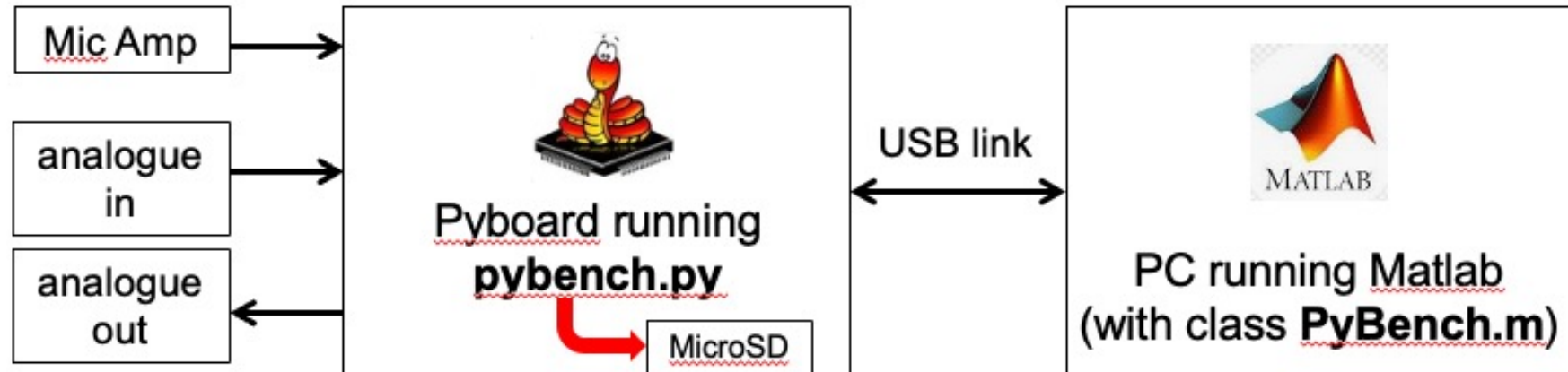
```
ports = serialportlist;    % find all serial port
```

- ◆ Last one is usually the one we want to use. The last port is given by: **ports(end)**.
- ◆ Create an object **pb** for the PyBench Board:

```
pb = PyBench(ports(end));  % create a PyBench object
```

- ◆ Control the Board via “methods”, e.g. **pb.set\_max\_v (2.5)**.

# pb.set\_max\_v(2.5) explained



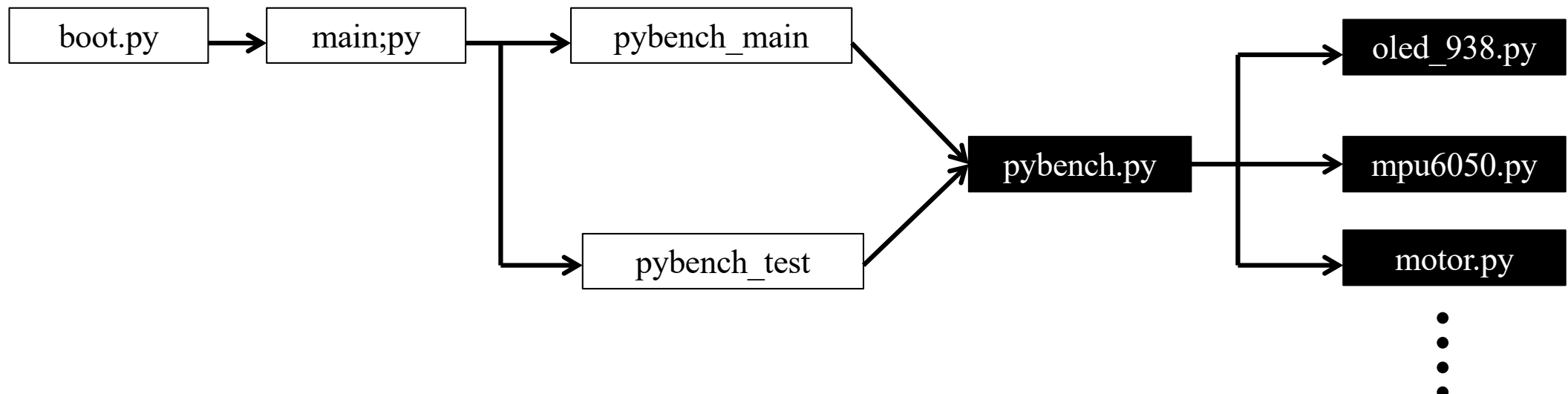
Here is what happens when you used this Matlab command: **pb = pb.set\_max\_v (2.5).**

- ◆ PC sends three bytes to PyBench board via USB link as serial data. First byte is a command character. In this case, 'X', followed by the value of voltage as two bytes. First byte is  $\text{int}(4096 * (v/3.3) / 256)$ , and second byte is  $\text{int}(4096 * (v/3.3)) \bmod 256$ .
- ◆ All along, PyBoard is running a Python program (pybench.py) listening for a command. The BLUE LED is ON in this state. Waiting for an event such as a character to arrive is known as "**polling**".
- ◆ When it receives the command (3 bytes), the pybench.py code sets the maximum voltage of the ADC to 2.5V.

# What are stored in the MicroSD card?



Program	Purpose
<a href="#">boot.py</a>	Boot file <u>specifying which</u> is the main program.
<a href="#">main.py</a>	Test the <u>DIP switch</u> setting and execute the corresponding <u>.py</u> file.
<a href="#">pybench_main.py</a>	The controlling program for <u>pybench</u> to <u>interpret</u> commands. Run if SW = 00.
<a href="#">pybench.py</a>	The <u>pybench</u> class library. Can be used in your own application program later.
<a href="#">pybench_test.py</a>	Self-test program for the <u>pybench</u> board to verify the hardware. Run if SW = 11.
<a href="#">oled_938.py</a>	OLED display driver class library.
<a href="#">font.py</a>	Character fonts used by <a href="#">oled_938.py</a> .
<a href="#">mpu6050.py</a>	IMU driver class library – to communicate with the accelerometer and gyroscope.
<a href="#">drive.py</a>	Drive class for the motor driver chip TB6612.



# PyBench Methods



PyBench.m must be in the Matlab search path.

```
clear all
ports = serialportlist;    % find all serial port
pb = PyBench(ports(end)); % create a PyBench object with last port
```

Methods	Purpose
<b>pb.set_sig_freq (f)</b>	Set signal frequency to f. $0.1 \text{ Hz} \leq f \leq 3000 \text{ Hz}$
<b>pb.set_samp_freq (f)</b>	Set sampling frequency to f. $1 \text{ Hz} \leq f \leq 30,000 \text{ Hz}$
<b>pb.set_max_v (v)</b>	Set maximum amplitude to v. $0 \leq v \leq 3.3$
<b>pb.set_min_v (v)</b>	Set minimum amplitude to v. $0 \leq v \leq 3.3$
<b>pb.set_duty_cycle (d)</b>	Set duty cycle of a square signal to d. $0 \leq d \leq 100$
<b>pb.dc (v)</b>	Output a dc voltage v. $0 \leq v \leq 3.3$
<b>pb.sine ( )</b>	Output a sinusoidal signal at set signal frequency between max_v and min_v.
<b>pb.triangle ( )</b>	Output a triangular signal at set signal frequency between max_v and min_v.
<b>pb.square ( )</b>	Output a square signal at set signal frequency between max_v and min_v, with the set duty cycle.
<b>v = pb.get_one ( )</b>	Capture one sample v from analogue input. $0 \leq v \leq 3.3$
<b>data = pb.get_block (n)</b>	Capture n samples from analogue input. $0 \leq \text{data} \leq 3.3$
<b>data = pb.get_mic (n)</b>	Capture n samples from microphone. $0 \leq \text{data} \leq 3.3$

# Lab 2 Task 2 – Generate and Capture Signals



```
% Lab 2 – Task 2 – Signal generation and capture with PyBench
```

```
%
```

```
clear all
```

```
ports = serialportlist; % find all serial ports
```

```
pb = PyBench(ports(end)); % create a PyBench object
```

```
% Set the various parameters
```

```
f = 440; % signal frequency (Hz)
```

```
fs = 8000; % sampling frequency (Hz)
```

```
pb = pb.set_sig_freq(f);
```

```
pb = pb.set_samp_freq(fs);
```

```
pb = pb.set_max_v(3.0); % set maximum voltage
```

```
pb = pb.set_min_v(0.5); % set minimum voltage
```

```
pb = pb.set_duty_cycle(50);
```

```
% Generate a signal
```

```
pb.sine();
```

```
% Capture N samples
```

```
N = 1000;
```

```
samples = pb.get_block(N);
```

```
data = samples - mean(samples);
```

```
% plot data
```

```
figure(1);
```

```
plot(data(1:200), 'o');
```

```
hold on
```

```
plot(data(1:200));
```

```
xlabel('Sample no');
```

```
ylabel('Signal voltage (V)');
```

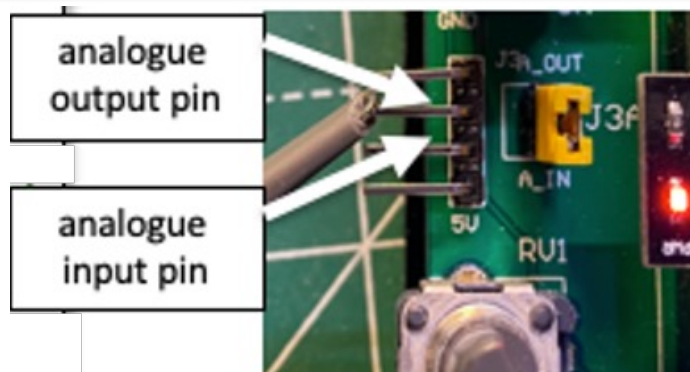
```
title('Captured signal');
```

```
hold off
```

```
% find spectrum
```

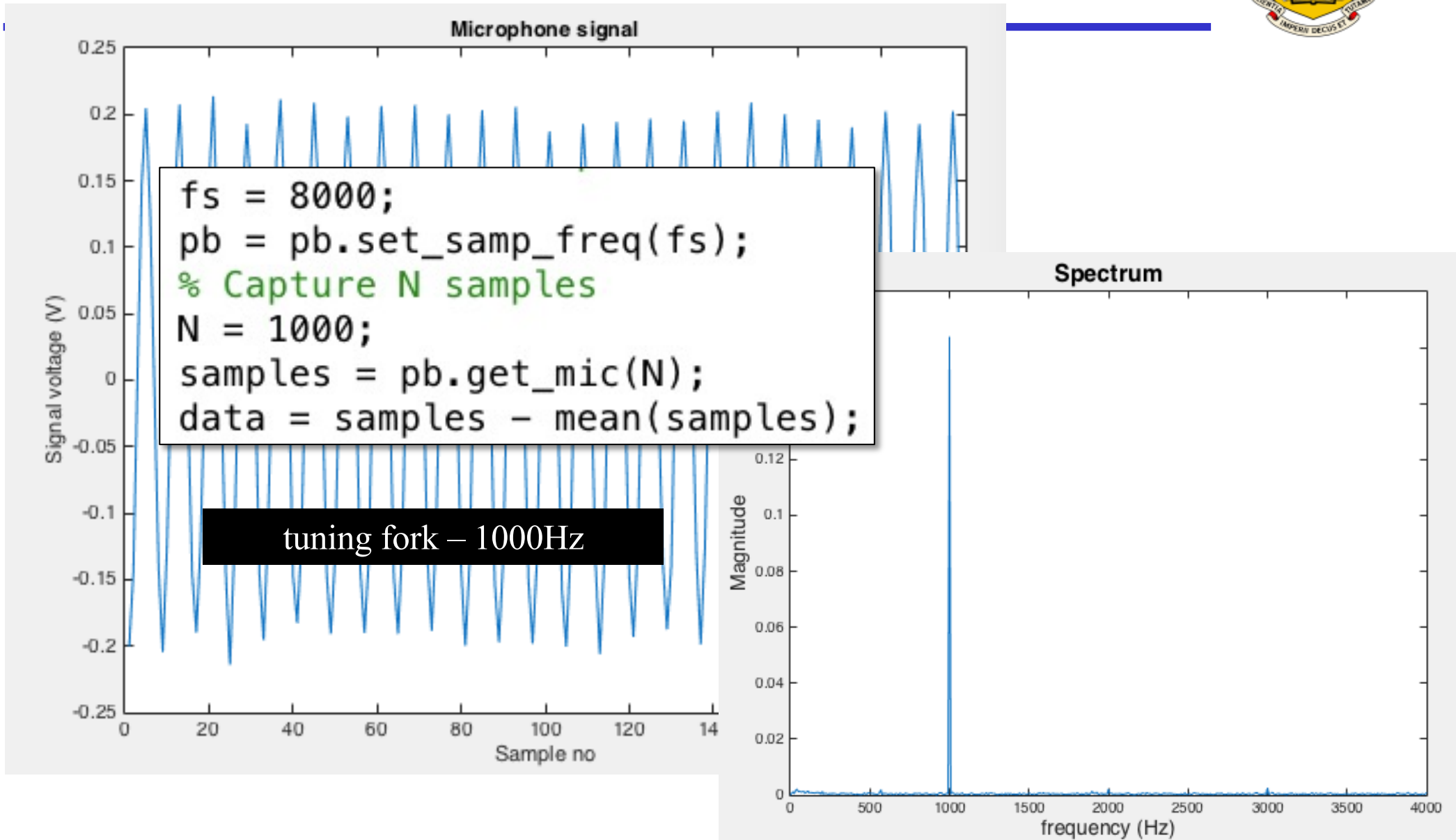
```
figure(2);
```

```
plot_spec(data, fs);
```





## Lab 2 Task 3 – Microphone signal



## Lab 2 Task 3 – Repeated capture & plot spectrum



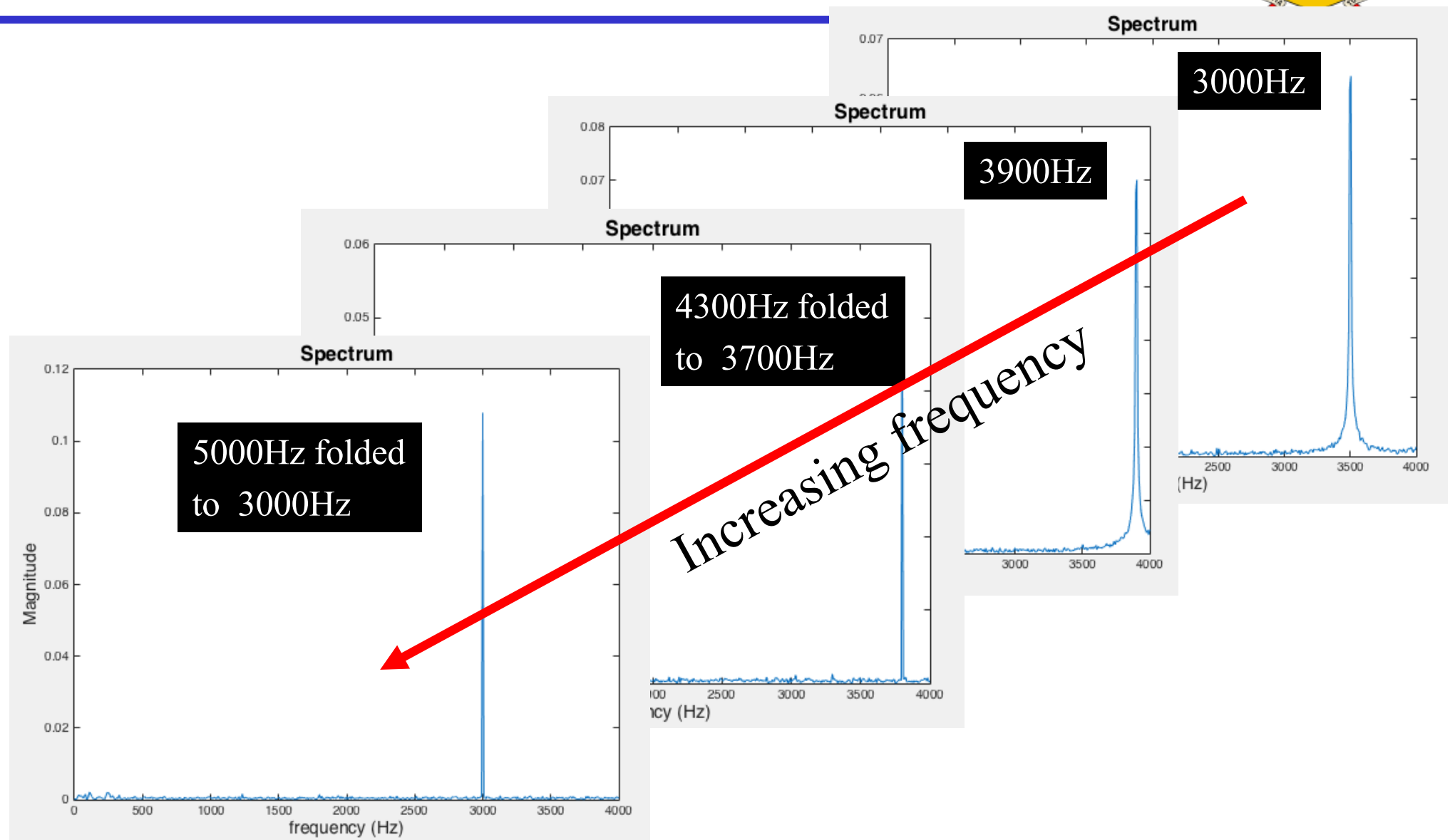
```
% repeat capture and plot spectrum
while true
    samples = pb.get_mic(N);
    data = samples - mean(samples);
    figure(2)
    clf;
    plot_spec(data,fs);
end
```

**Warning:** Running Matlab in an infinite loop may prevent you from re-gaining control over Matlab or even your computer. There are two things you may try if you want to get back control: 1) Type CTRL+C in the Command Window to interrupt Matlab; 2) kill the Matlab process and restart it again.

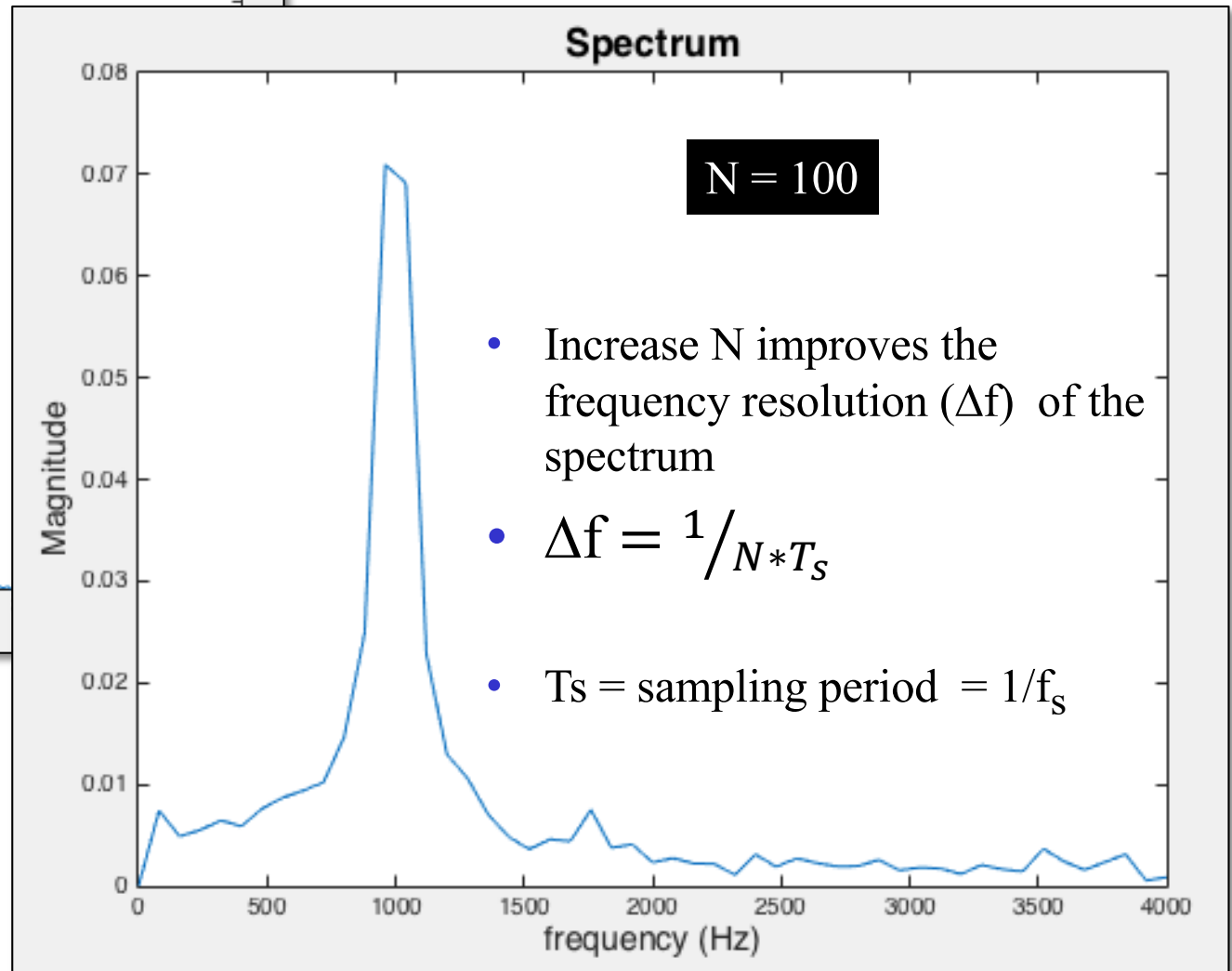
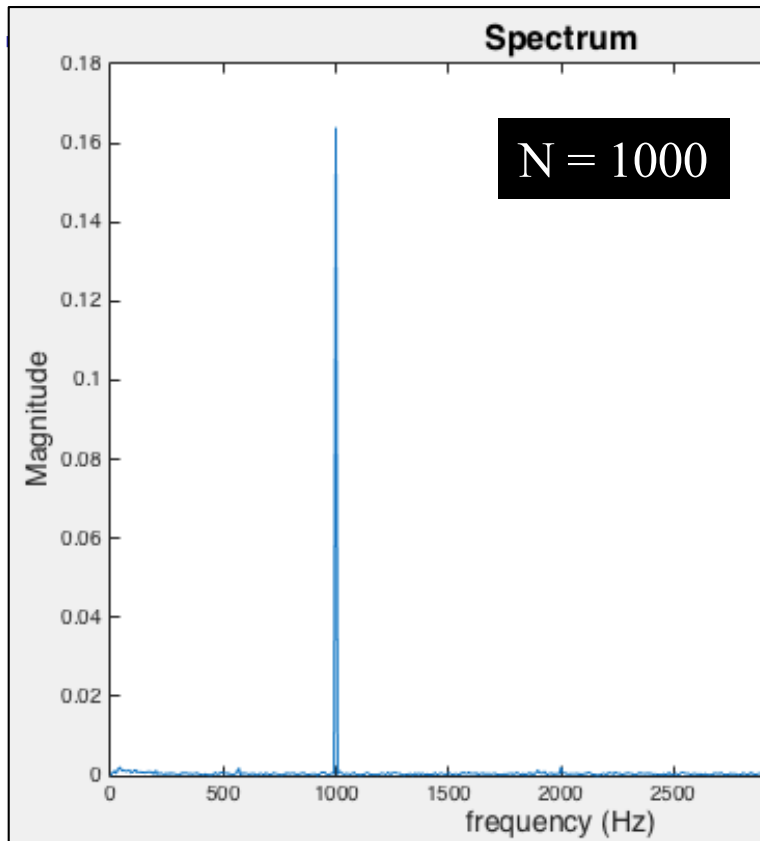
Recover from lost of serial communication:

1. Disconnect/reconnect USB; kill & restart Matlab
2. CTRL+C in command window, then type `fclose(pb.usb)` to shut down usb communication port

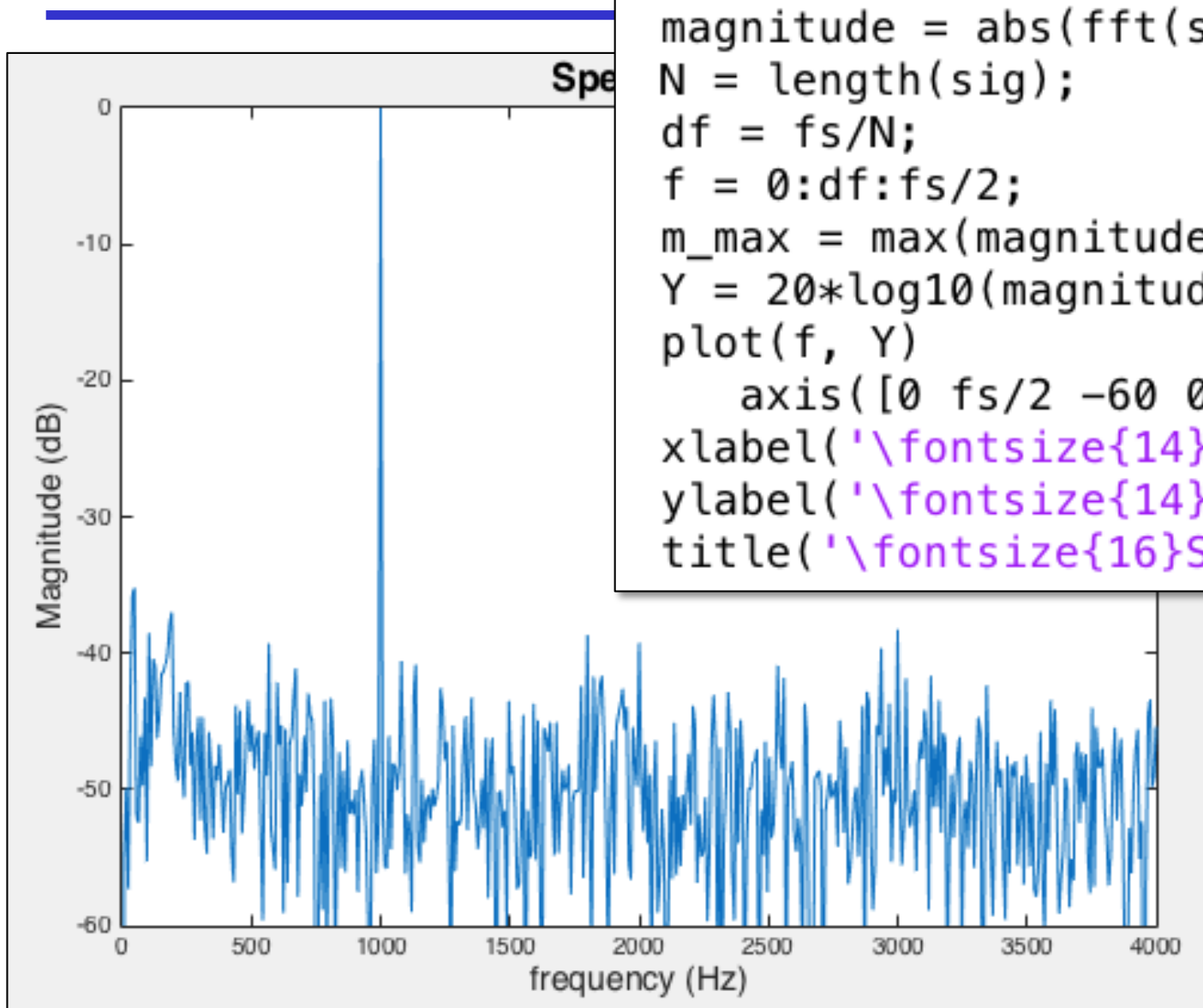
# Lab 2 Task 3 – Demonstrate spectral folding (aliasing)



## Lab 2 Task 3 – Effect of changing N – no of samples to analyse



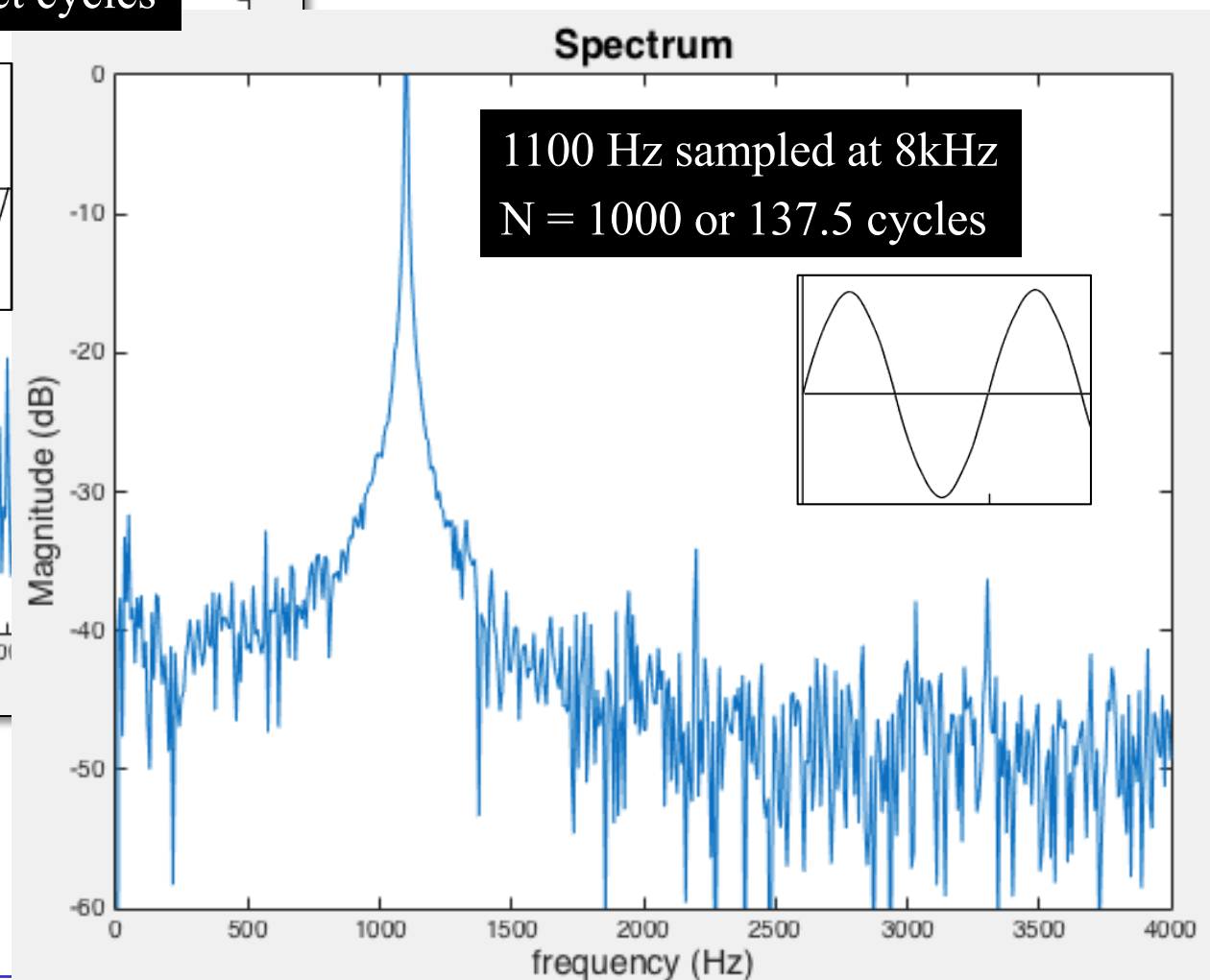
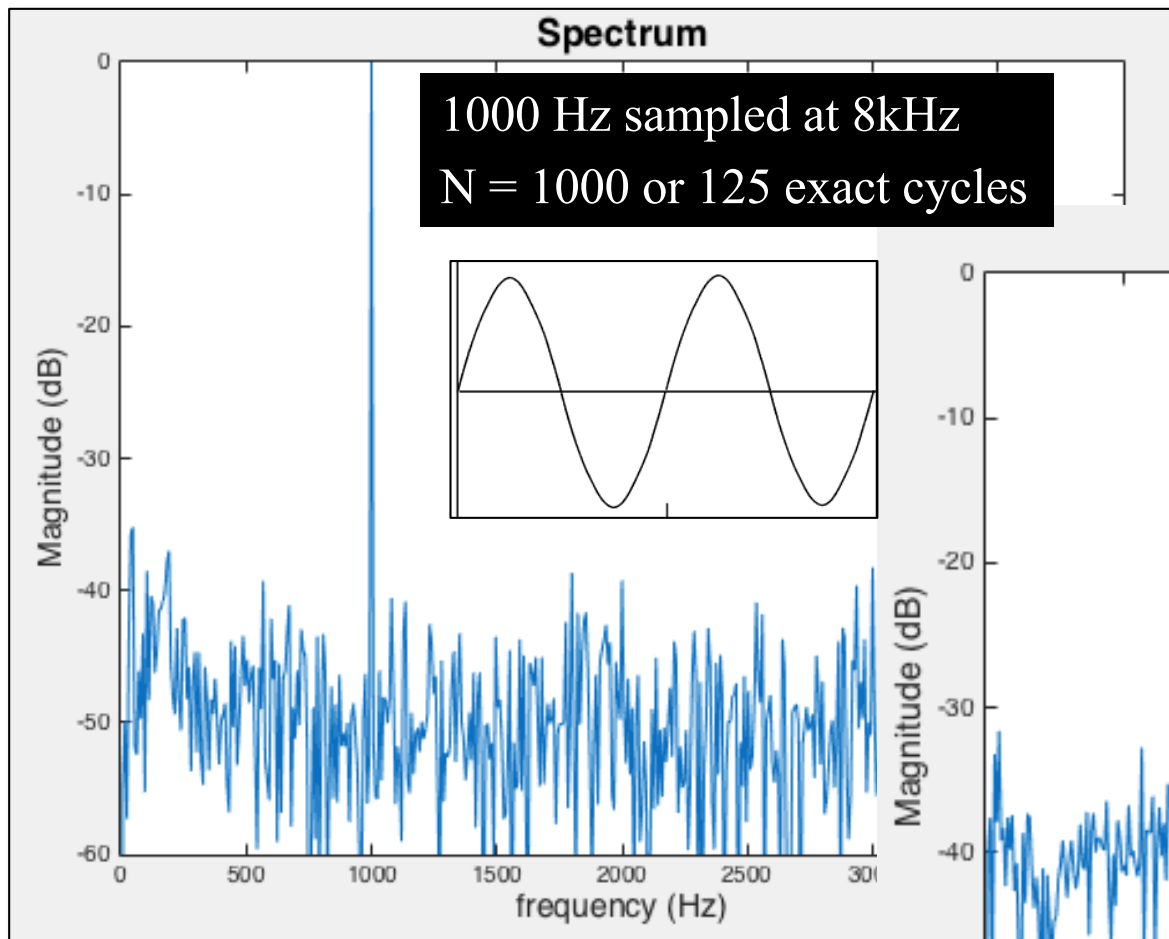
## Lab 2 Task 4 – Magnitude in dB



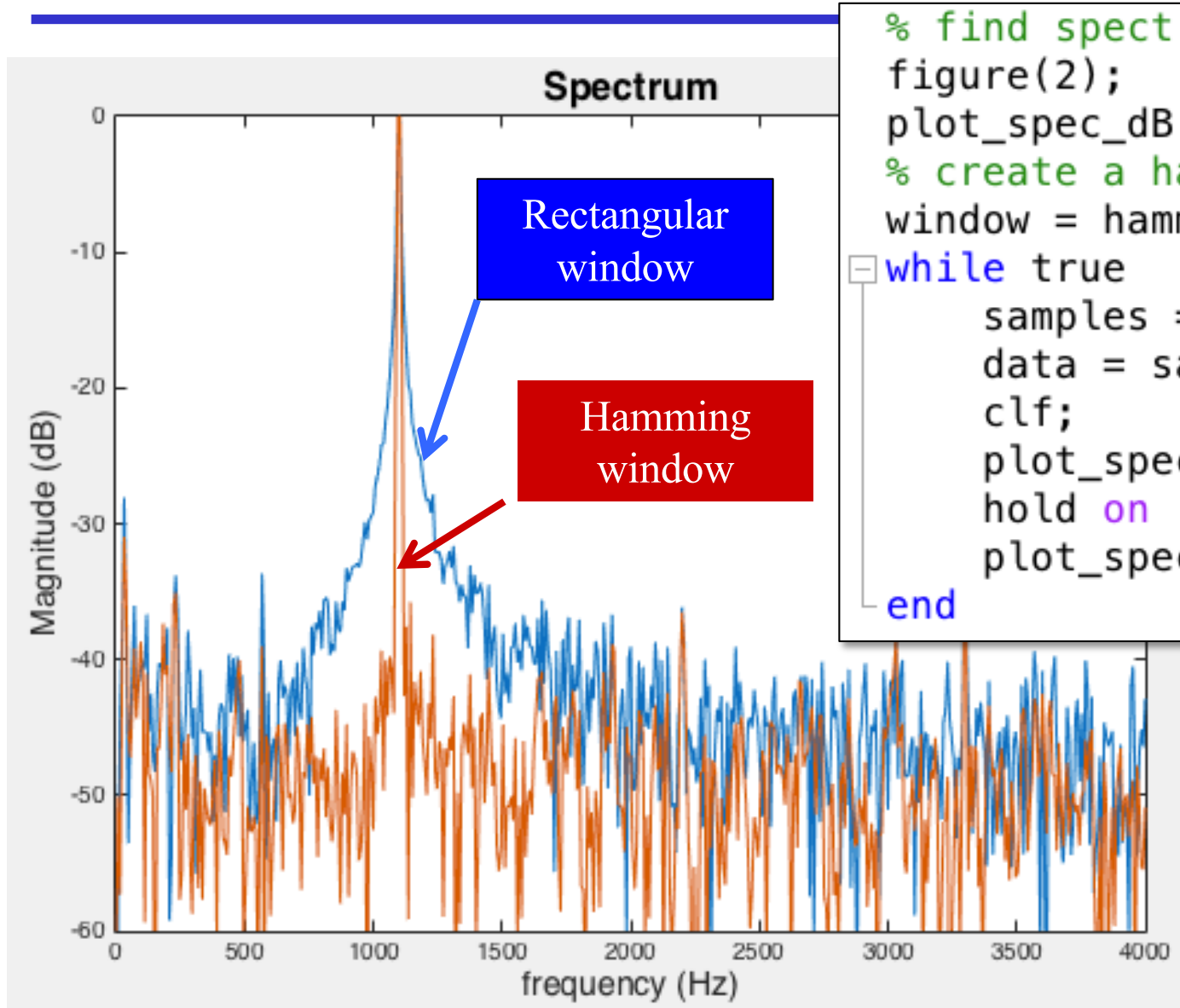
```
magnitude = abs(fft(sig));  
N = length(sig);  
df = fs/N;  
f = 0:df:fs/2;  
m_max = max(magnitude);  
Y = 20*log10(magnitude(1:length(f))/m_max);  
plot(f, Y)  
    axis([0 fs/2 -60 0]);  
xlabel('\fontsize{14}frequency (Hz)')  
ylabel('\fontsize{14}Magnitude (dB)');  
title('\fontsize{16}Spectrum');
```



## Lab 2 Exercise 4 – Windowing effect



# Lab 2 Task 4 – Rectangular vs Hamming Window



```
% find spectrum
figure(2);
plot_spec_dB(data,fs);
% create a hamming window
window = hamming(length(data));
while true
    samples = pb.get_mic(N);
    data = samples - mean(samples);
    clf;
    plot_spec_dB(data,fs);
    hold on
    plot_spec_dB(data.*window,fs);
end
```

## Lab 2 Task 5 – Calculate energy in 20ms segment



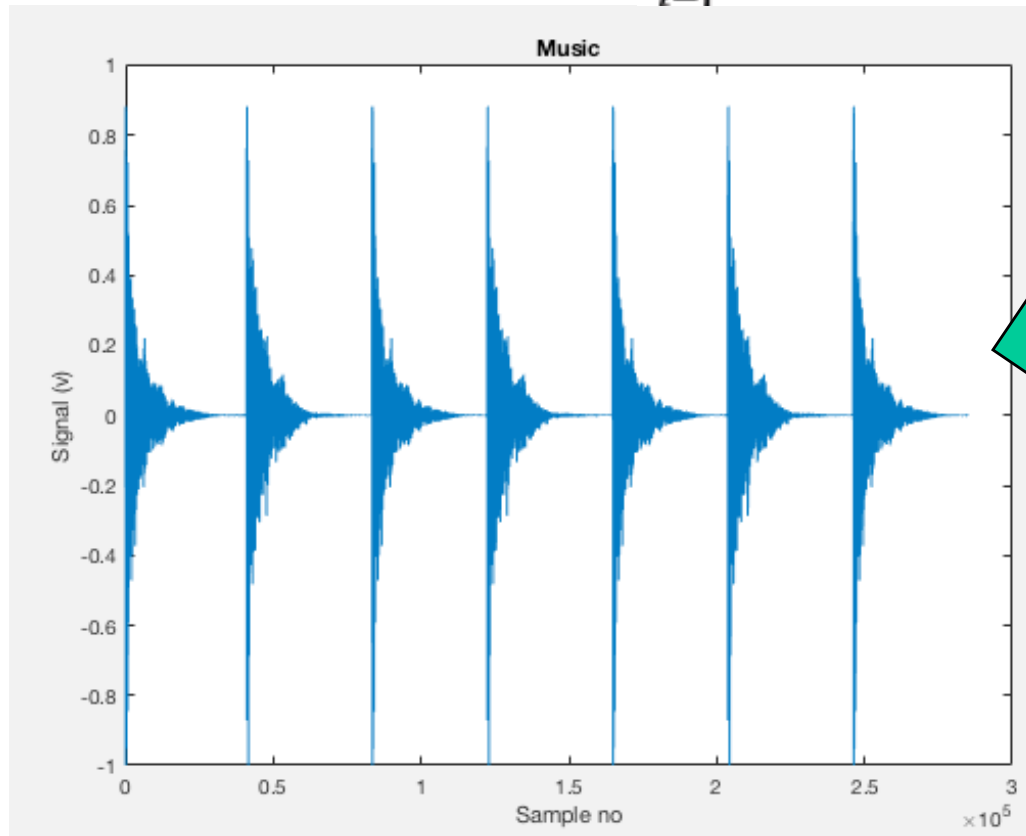
$$\sum_{i=1}^N x^2(t) \quad \text{where } N \text{ is the number of samples in 20ms}$$

```
% Divide signal into segments
T = 0.02;           % duration of segment in seconds
N = fs*T;           % number of samples in segment
E = [];
for i=1:N:length(sig)-N+1
    seg = sig(i:i+N-1);
    E = [E seg'*seg];
end
% plot the energy graph and the peak values
figure(2);
clf;
x = 1:length(E);
plot(x, E)
xlabel('Segment number');
ylabel('Energy');
hold on
% Find local maxima
[pks locs] = findpeaks(E);
plot(locs, pks, 'o');
hold off
% plot spectrum of energy
figure(3)
plot_spec(E - mean(E), 1/T);
```

## Lab 2 Task 5 – Analyse beat of drum beats (1)

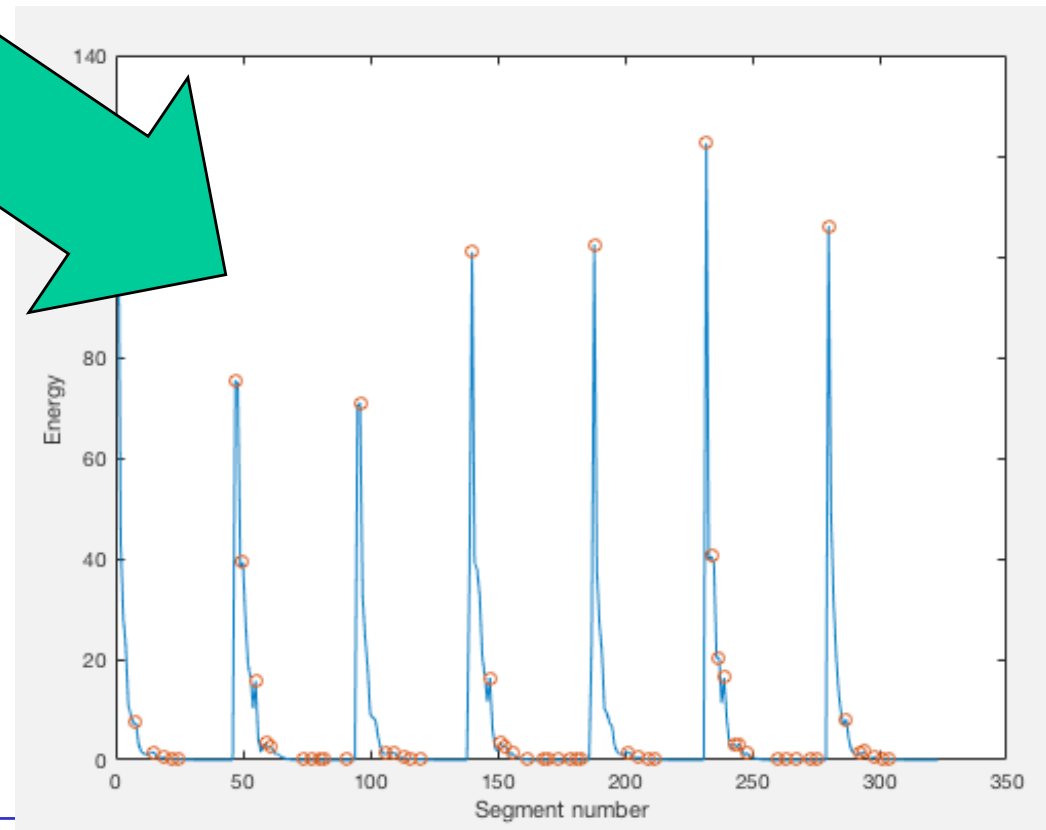


$$\sum_{i=1}^N x^2(t) \quad \text{where } N \text{ is the number of samples in 20ms}$$



Signal  $x(t)$

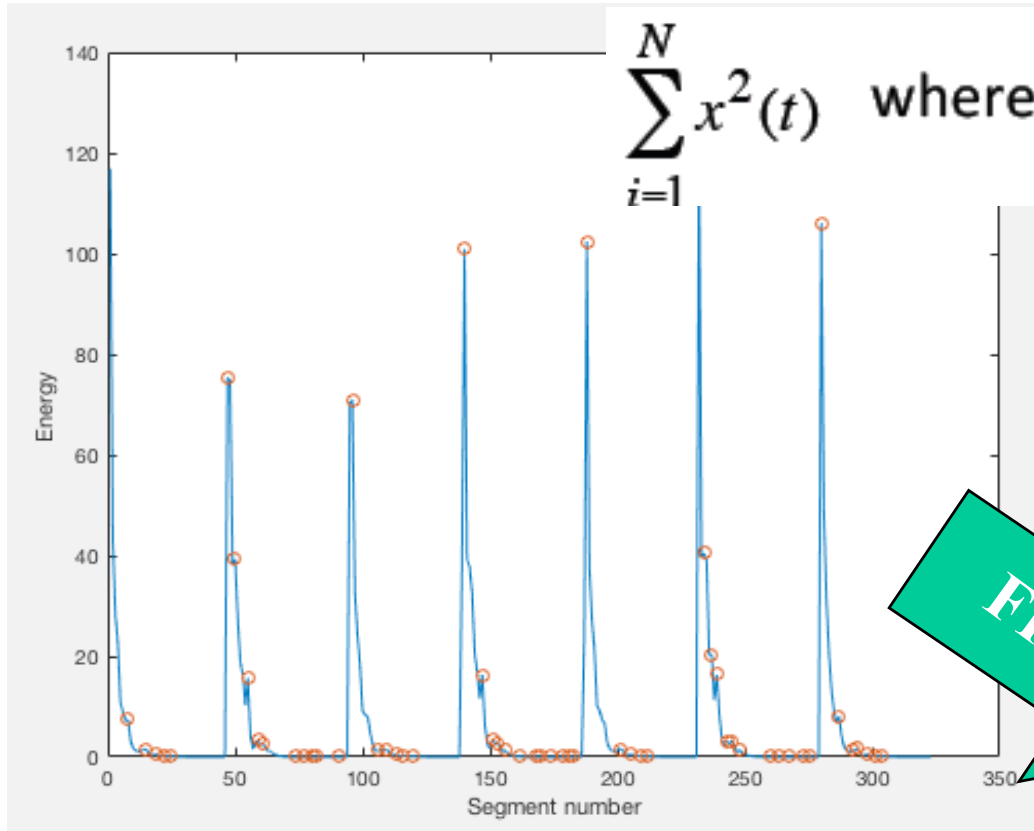
Energy of signal



## Lab 2 Task 5 – Analyse beat of drum beats (2)



$$\sum_{i=1}^N x^2(t) \quad \text{where } N \text{ is the number of samples in 20ms}$$



Energy vs time

Spectrum of energy

