Lab 0 - Ex 2: The plot_spec function

```
function plot_spec(sig, fs)
    % Function to plot frequency spectrum of sig
    % usage:
    %      plot_spec(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('Frequency (Hz)')
    ylabel('Magnitude');

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

Lab 0 - Ex 3: Two tones (fs = 8192, T = 1)

- s1 = 440Hz sine at 1V
- s2 = 1kHz sine at 0.5V

\[ \text{sig} = \text{s1} + \text{s2} \]
\[ \text{plot_spec}(\text{sig}, 8192) \]
Lab 0 - Ex 4: Two tones + noisy

- Noisy = sig + randn(size(sig));

- plot_spec(noisy, 8192)

What's on the Board

<table>
<thead>
<tr>
<th>Pyboard</th>
<th>Motor Driver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic LED</td>
<td>128 x 64 display</td>
</tr>
<tr>
<td>Potentiometer</td>
<td>Inertial Measurement Unit (IMU)</td>
</tr>
<tr>
<td>12V to 5V converter</td>
<td>Bluetooth to UART</td>
</tr>
</tbody>
</table>

What are stored in the MicroSD card?

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

How PyBench works?

- Create an object pb for the PyBench Board:
  ```
  pb = PyBench('/dev/tty.usbmodem1422');
  ```
- Control the Board via "methods", e.g. `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 \(\text{int}(4096 \times \frac{v}{3.3}) / 256\), and second byte is \(\text{int}(4096 \times \frac{v}{3.3}) \mod 256\).
- All along, PyBoard is running a Python program (`pybench_main.py`) listening for a command. The BLUE LED is ON in this state. Waiting for a event such as a character to arrive is known as "polling".
- When it receives the command (3 bytes), it call the `pybench class method` to set the maximum voltage.
Lab 1 Exercise 1 – Generate and Capture Signals

% Lab 2 – Ex 1 – Signal generation and capture with PyBench
% clear all
pb = PyBench('/dev/tty.usbmodem1422');
if pb.ok()
    display('PyBench Board working')
else
    display('PyBench Board NOT working')
end
% Set the various parameters
f = 440;
fs = 8000;
pb = pb.set_sig_freq(f);
pb = pb.set_samp_freq(fs);
pb = pb.set_max_v(3.0);
pb = pb.set_min_v(0.5);
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 1 Exercise 1 – Spectrum of square wave (440Hz)

440Hz
A=2.5V

\[ a_1 = 1.6 \]
\[ a_3 = 0.53 \]
\[ a_5 = 0.32 \]
\[ a_7 = 0.23 \]

Spurious components
(why?)

Lab 1 Exercise 1 – Spectrum of square wave (400Hz)

400Hz
A=2.5V

Spurious components disappeared

Lab 1 Exercise 1 – Fourier Coefficients of square wave

Take fundamental \( f_0 \)
Add \( 3f_0 \)
Add \( 5f_0 \)

\[ \text{... carry on indefinitely ...} \]

Add \( 7f_0 \)
Lab 1 Exercise 1 – Fourier Coefficients of common waveforms

Time Domain | Frequency Domain
---|---
a. Pulse | \(a_0 = d\)
| \(a_n = \frac{2A}{\pi} \sin\left(\frac{n\pi f}{2}\right)\)
| \(b_n = 0\)
| (All even harmonics are zero)
b. Square | \(a_0 = 0\)
| \(a_n = \frac{4A}{\pi^n}\)
| \(b_n = 0\)
| (All even harmonics are zero)
c. Triangle | \(a_0 = 0\)
| \(a_n = \frac{2A}{\pi^n}\)
| \(b_n = 0\)
| (All even harmonics are zero)

Lab 1 Exercise 1 – plot_spec_full ()

```matlab
function plot_spec_full(sig, fs)
    % Function to plot frequency spectrum of sig
    N = length(sig);
    df = fs/N;
    f = 0:df:fs-df;
    Y = magnitude(1:length(f));
    plot(f, 2*pi*Y)
    xlabel('Frequency (Hz)');
    ylabel('Magnitude');
    title('Frequency Spectrum');
end
```

Lab 1 Exercise 2 – Microphone signal

```matlab
fs = 8000;
pb = pb.set_samp_freq(fs);
% Capture N samples
N = 1000;
samples = pb.get_mic(N);
data = samples - mean(samples);
```

Lab 1 Exercise 2 – Demonstrate spectral folding (aliasing)

Increasing frequency

- 3000Hz folded to 3700Hz
- 4300Hz folded to 3000Hz
- 5000Hz folded to 3000Hz
Lab 1 Exercise 2 – Effect of changing N – no of samples to analyse

- Increase N improves the frequency resolution ($\Delta f$) of the spectrum
- $\Delta f = \frac{1}{N \cdot T_s}$
- $T_s =$ sampling period $= \frac{1}{f_S}$

Lab 1 Exercise 3 – Magnitude in dB

```matlab
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('\textit{frequency (Hz)}')
ylabel('\textit{Magnitude (dB)}')
title('\textit{Spectrum}')
```

Lab 1 Exercise 3 – Windowing effect

- 1000 Hz sampled at 8kHz
  - $N = 1000$ or 125 exact cycles

- 1100 Hz sampled at 8kHz
  - $N = 1000$ or 137.5 cycles

Lab 1 Exercise 3 – Rectangular vs Hamming Window

- Rectangular window
- Hamming window

```matlab
% find spectrum
figure();
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 1 Exercise 4 – Analyse beat of drum beats (1)

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

Energy of signal

Signal \( x(t) \)

Lab 1 Exercise 4 – Analyse beat of drum beats (2)

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

Energy vs time

Spectrum of energy

FFT