Imperial College London

DE2 Electronics 2

Tutorial 3

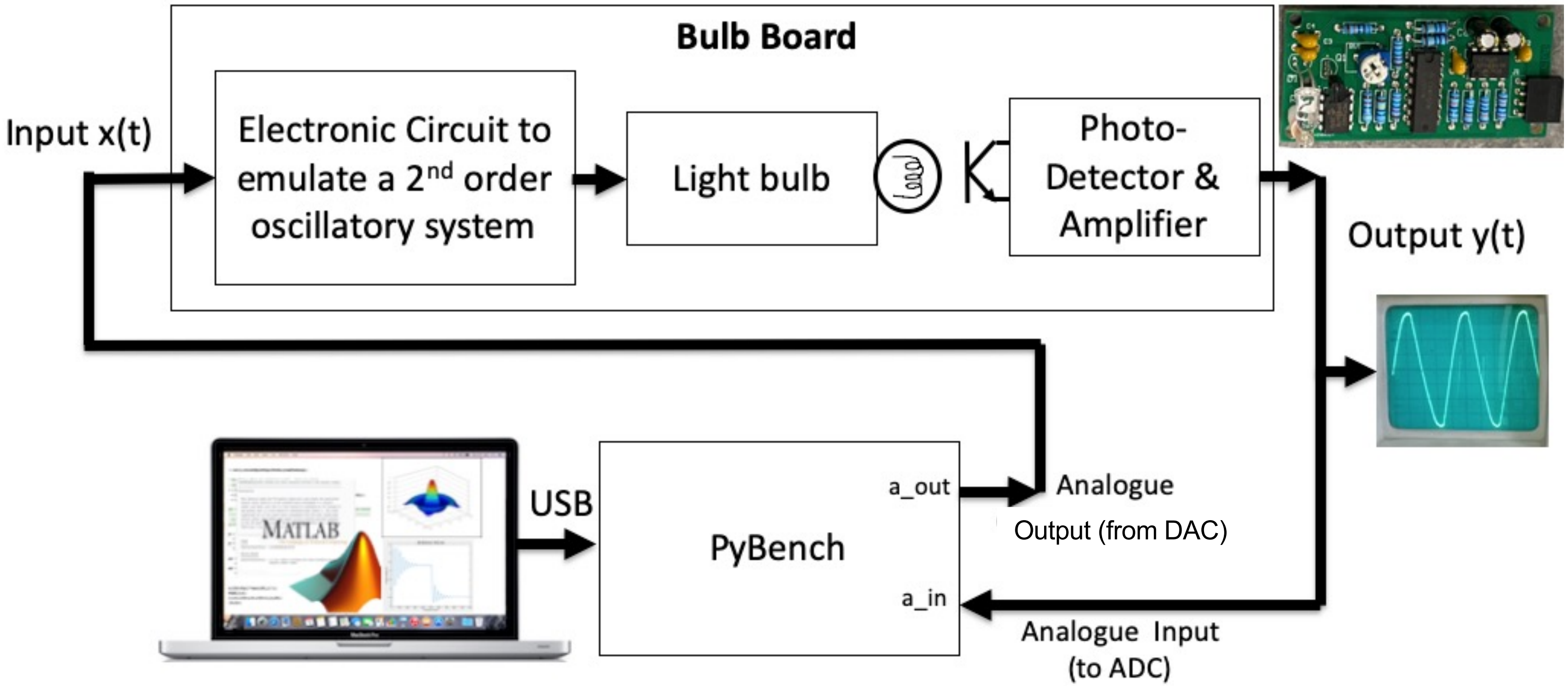
Lab 3 & 4 Explained

Prof Peter YK Cheung

Dyson School of Design Engineering

URL: www.ee.ic.ac.uk/pcheung/teaching/DE2_EE/
E-mail: p.cheung@imperial.ac.uk

Bulb Board

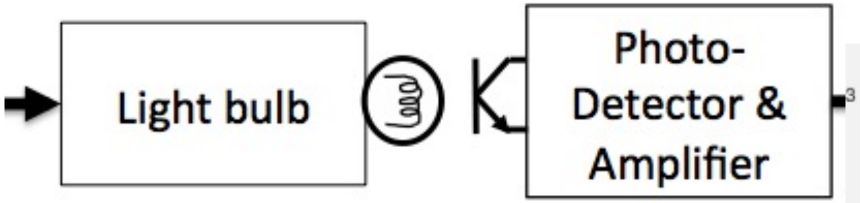


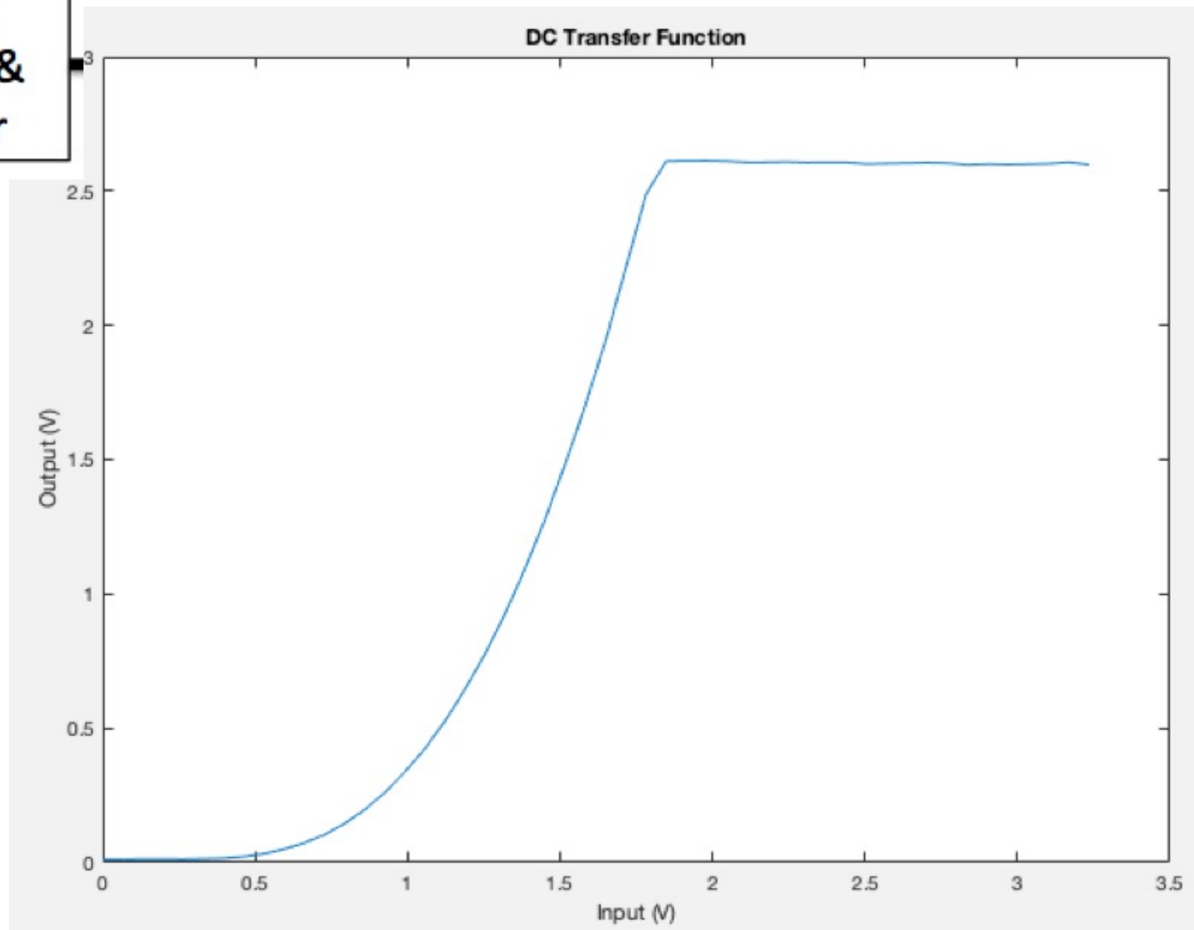
- We are interested in mathematical modelling system.
- Bulb Board is designed to behave like a 2nd order system + a non-linear system with some delay (the light bulb)
- We want to verify that the mathematical model is a good representation.
- We also want to explore the limitations of this model

Key aspects of Lab 3

1. DC characteristics – no time variation. Measure light intensities at different drive voltages.
2. Steady state response to sinusoidal signals at different frequencies – we call this **frequency response H(jω)**.
3. Use of Matlab for modelling and simulation using **transfer function** H(s) .
4. Transient behaviour of the system – we call this step response.
5. Impact of non-linearity in the system.

Task 1 – DC Characteristic



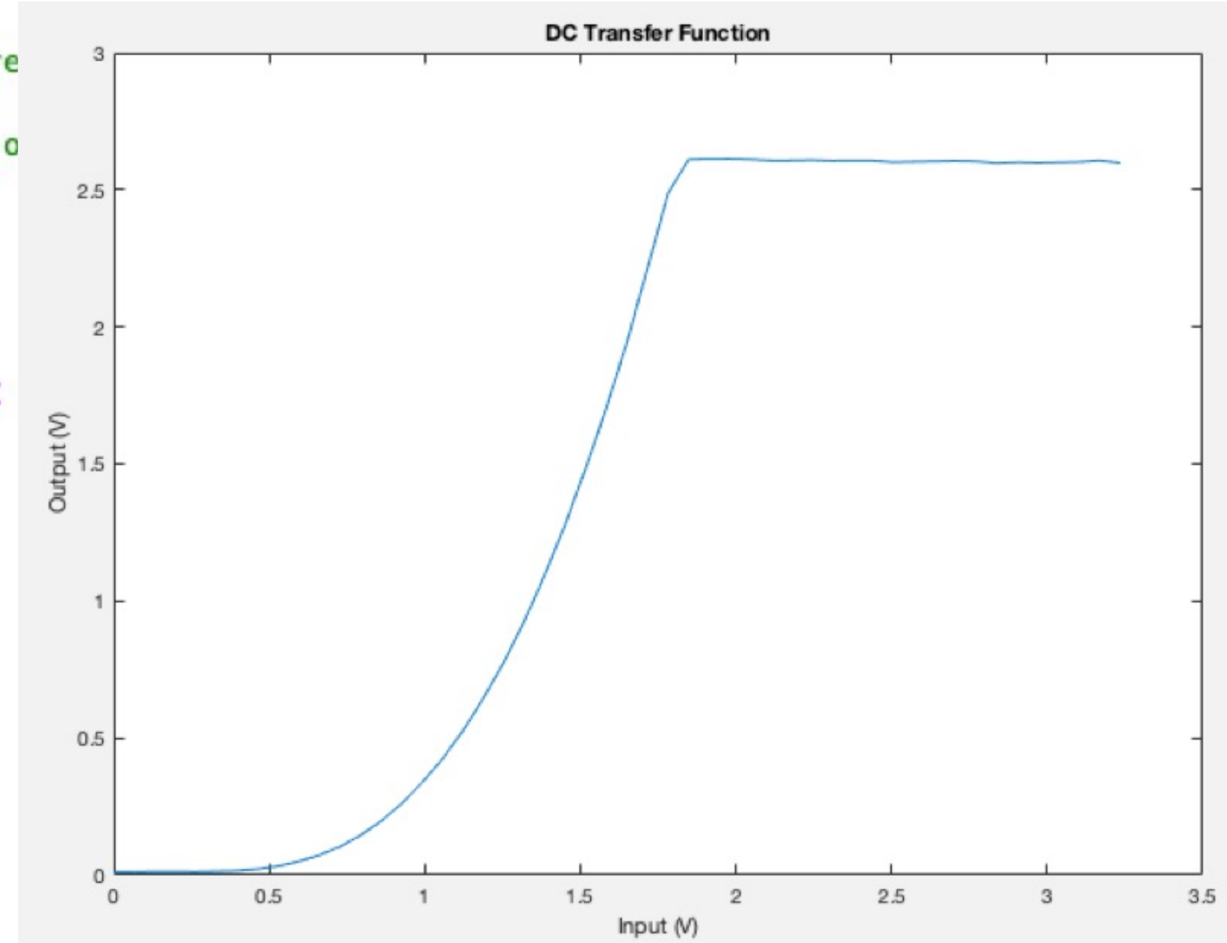


- $y = F(x)$
- F is a non-linear function.
- F is a quadratic function because:

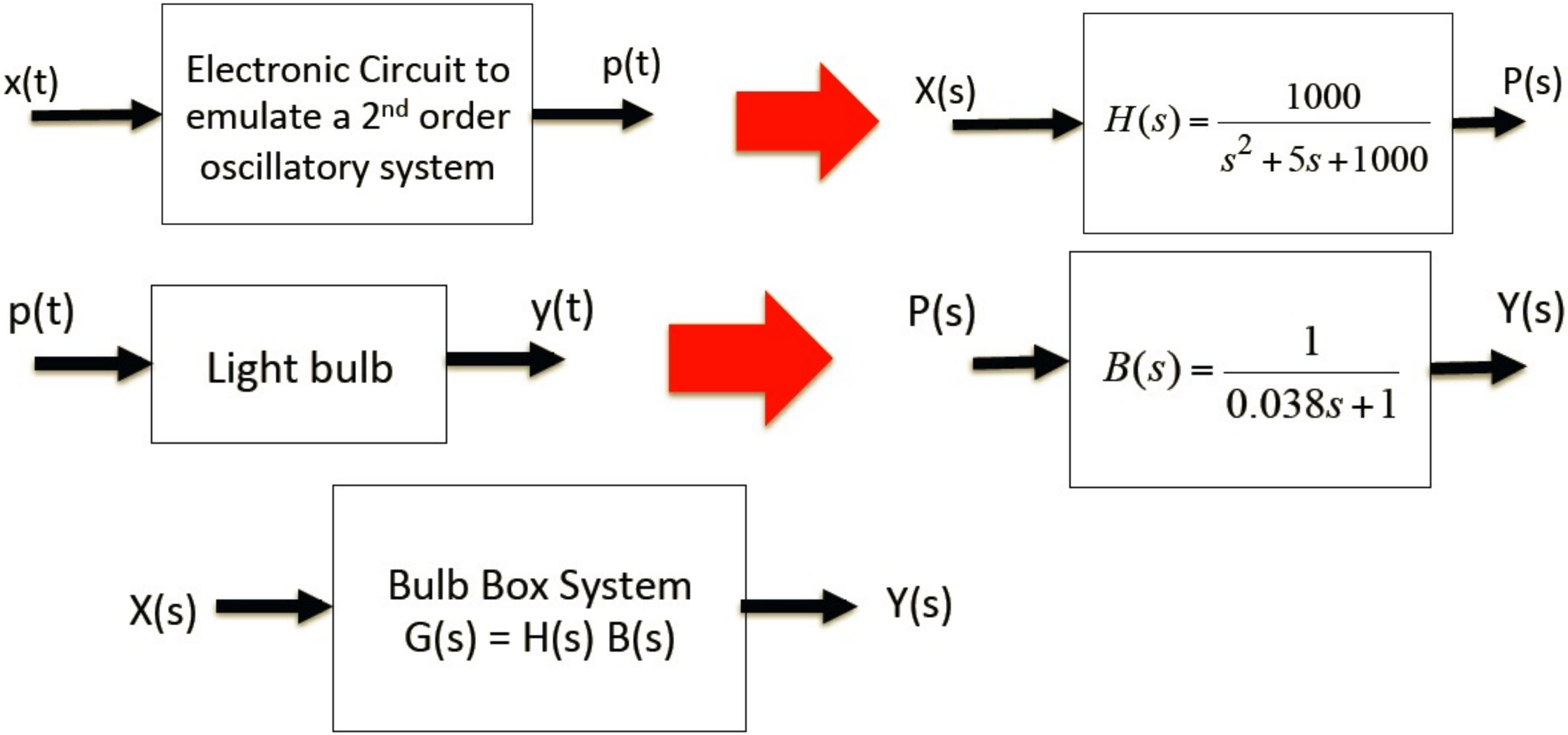
  light intensity $\propto x^2$
- Light is dependent temperature of filament in bulb
- Temperature is dependent on power to bulb
- Power is proportional to x^2.

Task 1 – Solution

```
% Lab 3 - Task 1 DC characteristics of the Bulb Boardl
clear all
ports = serialportlist;
pb = PyBench(ports(end));  % cre

%  measure the steady-state DC o
pb.samp_freq = 200;
NSTEPS=50;
input = zeros(NSTEPS,1);
output = zeros(NSTEPS,1);
tic
disp('Sweeping DC output for DC
for i = [1:NSTEPS]
    v = (i-1)*3.3/NSTEPS;
    input(i) = v;
    pb.dc(v);
    pause(0.5);
    data = pb.get_block(10);
    output(i) = mean(data);
end
pb.dc(0.0);
toc
plot(input,output)
xlabel('Input (V)');
ylabel('Output (V)');
title('DC Transfer Function');
fclose(instrfind());
```



Task 2 – Modeling dynamics in a system



$$G(s) = \frac{1000}{0.038s^3 + 1.19s^2 + 43s + 1000}$$

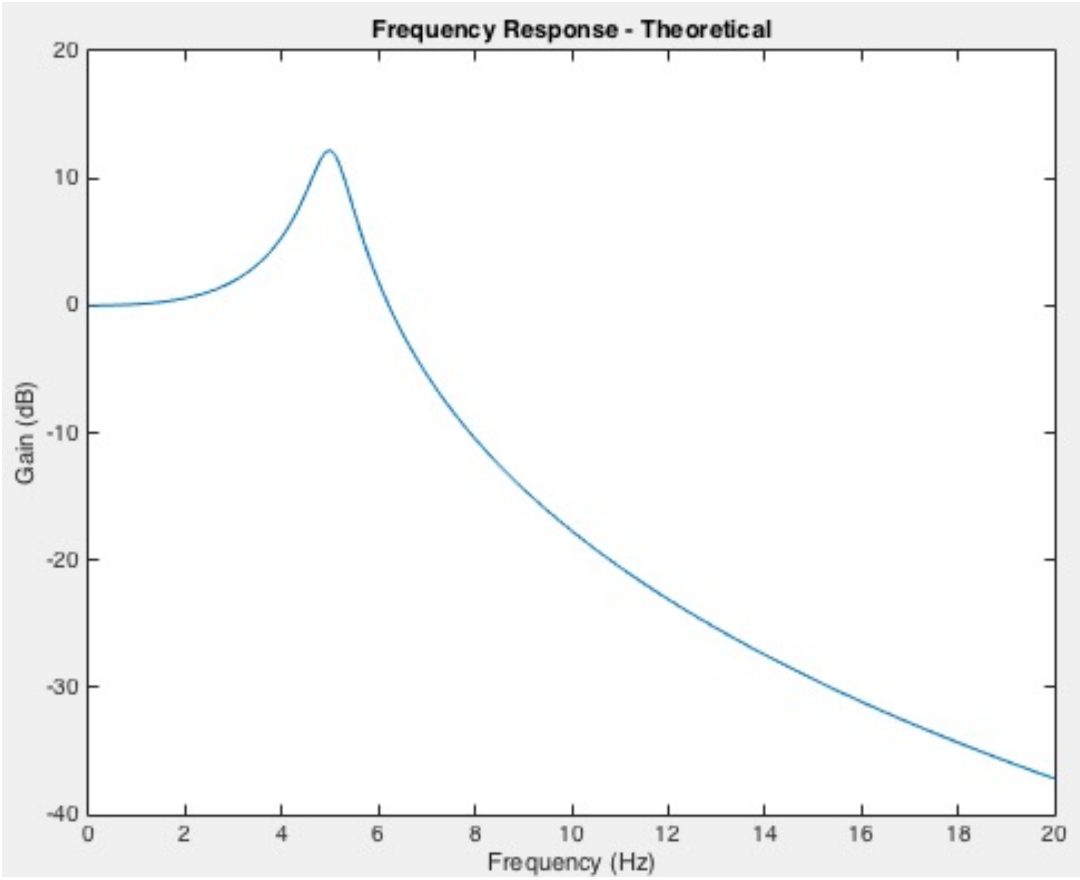
Task 2 – Predict the frequency response

```
% Lab 3 Task 2 - Plot theoretical freq. response of Bu
f = (0:0.1:20);
D = [0.038 1.19 43 1000];    % specify denominator
s = 1i*2*pi*f;                % s = jw (1i is sqrt(-1))
G = 1000./abs(polyval(D,s));  % polynomial evaluation
Gdb = 20*log10(G);            % Gain in dB
figure;
plot(f,Gdb);
xlabel('Frequency (Hz)');
ylabel('Gain (dB)');
title('Frequency Response - Theoretical');
```

$$G(s) = \frac{1000}{0.038s^3 + 1.19s^2 + 43s + 1000}$$

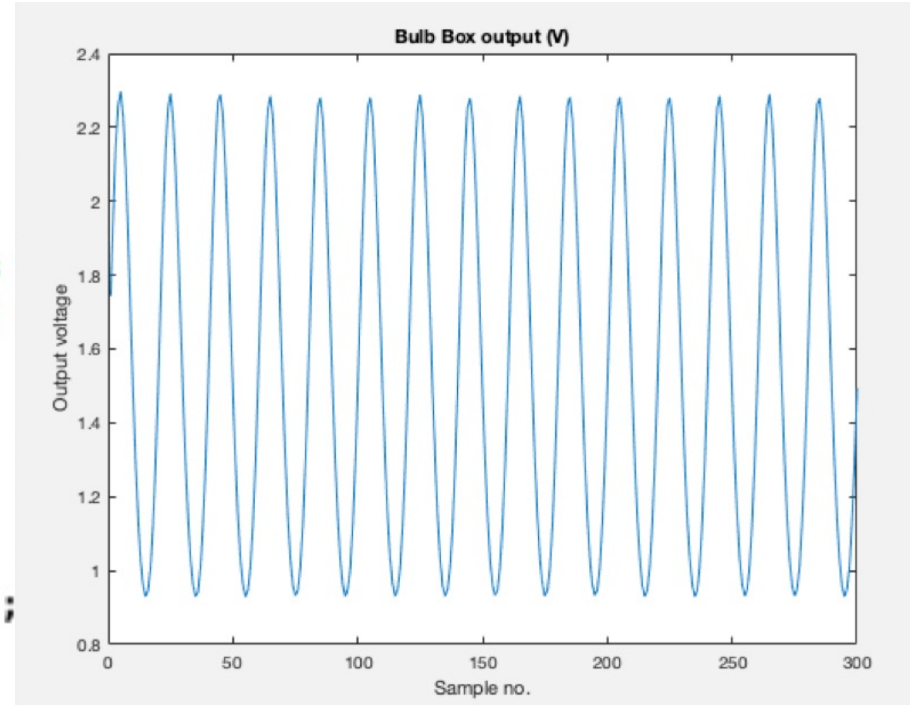
Task 2 – Predict the frequency response

$$G(s) = \frac{1000}{0.038s^3 + 1.19s^2 + 43s + 1000}$$



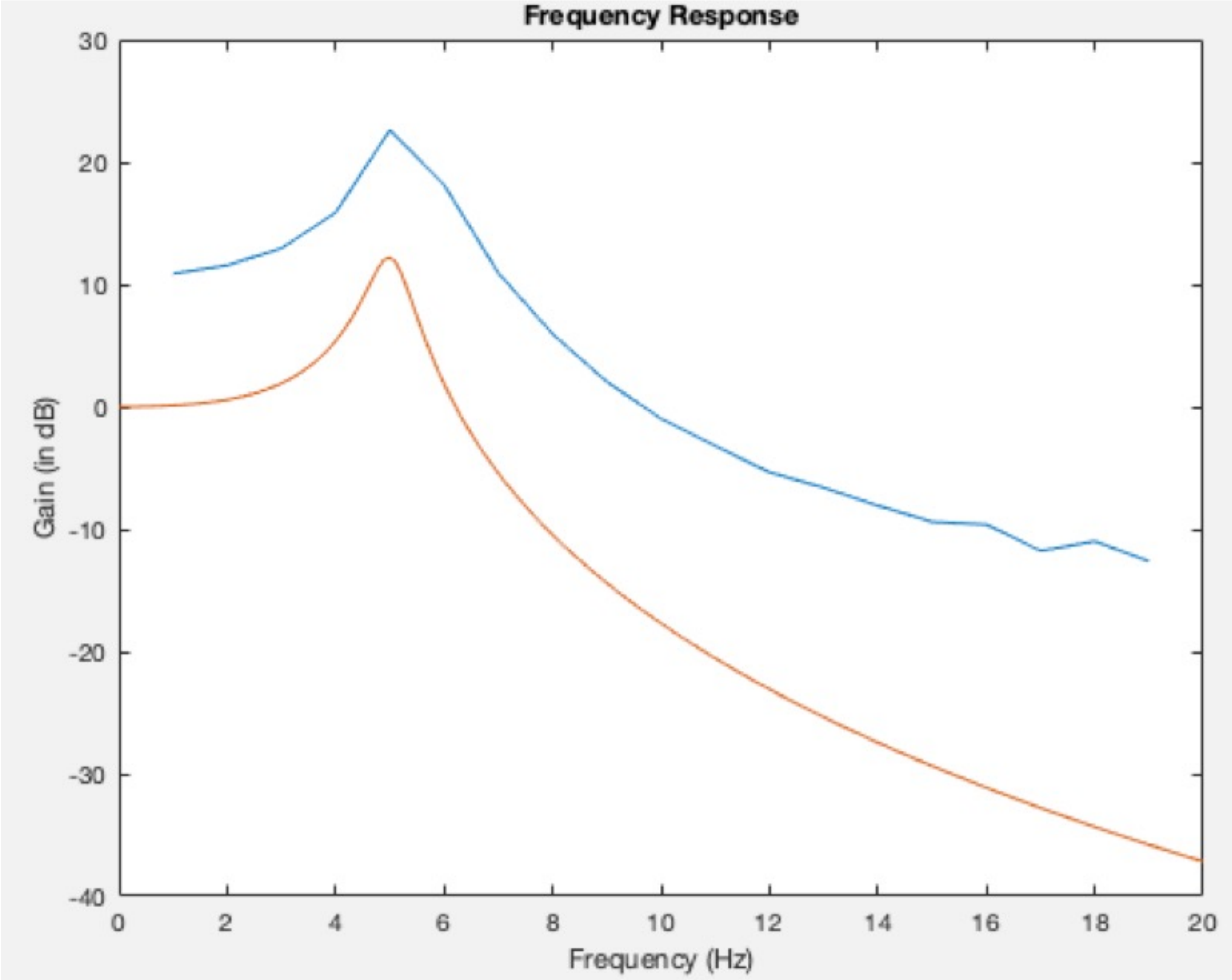
Task 3 – Measure Real Gain at 5Hz

```
% Generate a sine wave at sig_freq Hz
max_x = 1.55;
min_x = 1.45;
f_sig = 5.0;
pb=pb.set_sig_freq(f_sig);
pb=pb.set_max_v(max_x);
pb=pb.set_min_v(min_x);
pb.sine();
pause(2)
% Capture output y(t)
pb=pb.set_samp_freq(100);  %
N = 300;      % no of samples
y = pb.get_block(N);
% plot signal
plot(y);
xlabel('Sample no.');
ylabel('Output voltage');
title('Bulb Box output (V)');
% Compute Gain
x_pk2pk = max_x - min_x;
y_pk2pk = max(y) - min(y);
G = y_pk2pk/x_pk2pk
G_dB = 20*log10(y_pk2pk/x_pk2pk)
```

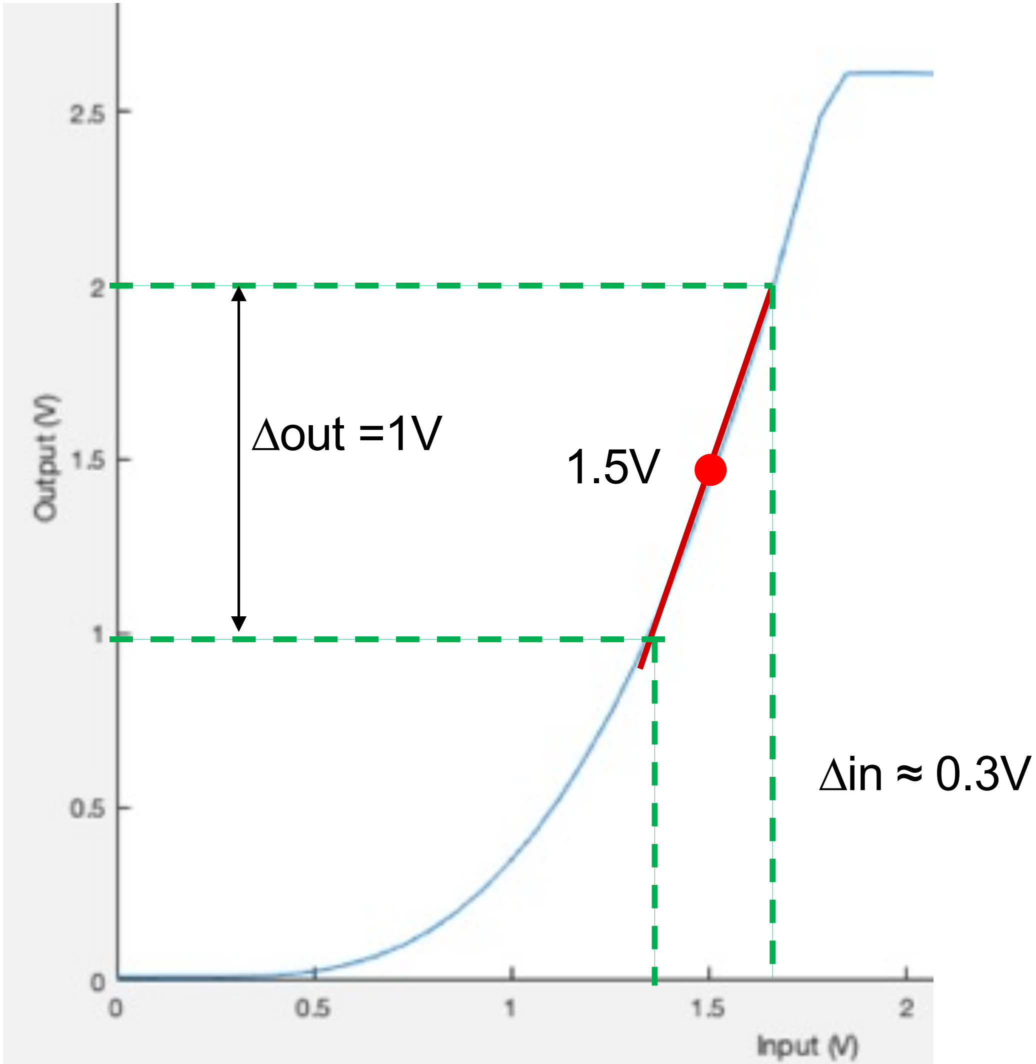



```
G =

    13.6802


G_dB =

    22.7218
```

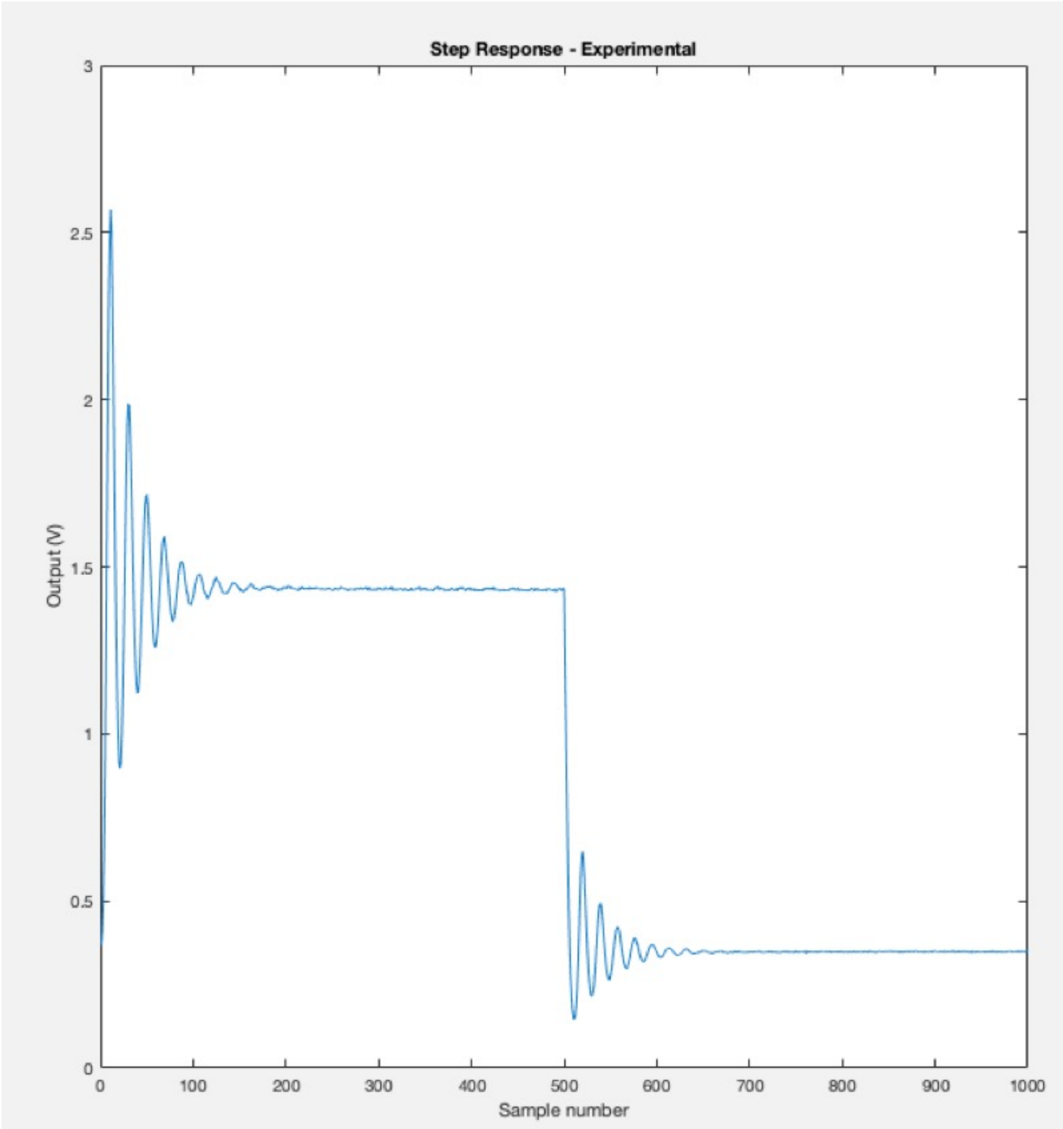
Task 3 – Theory vs Measurements



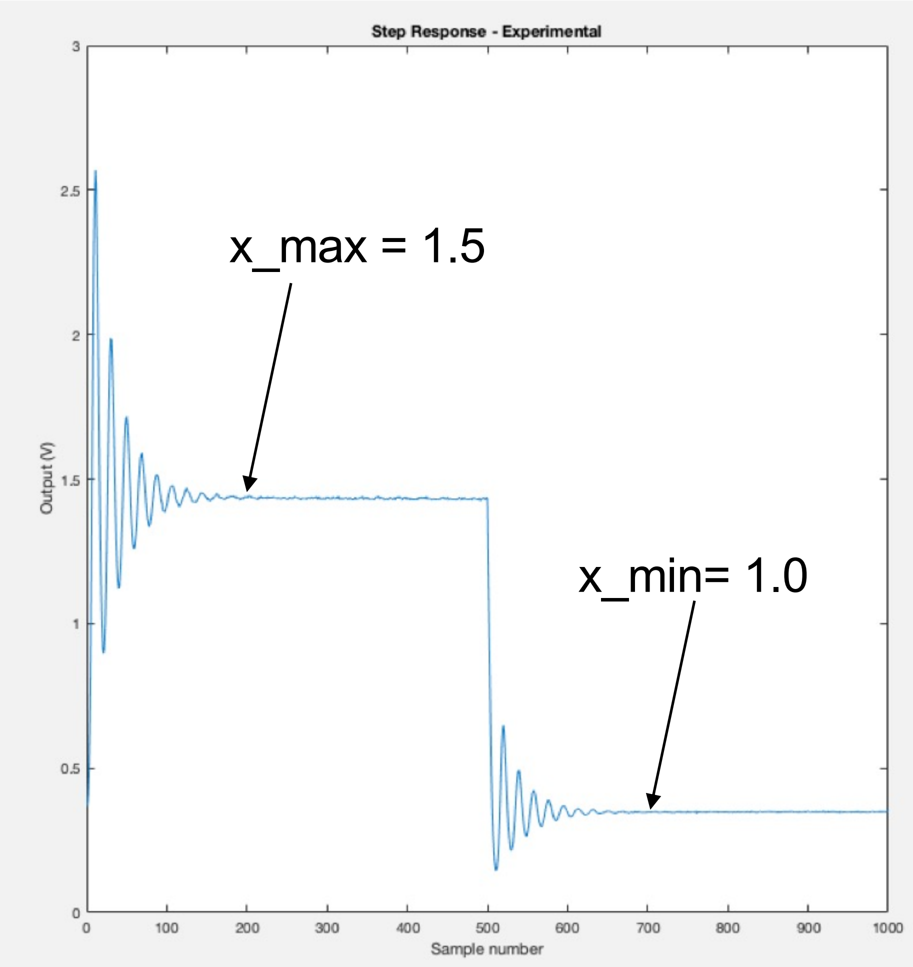
Task 3 – Explain theory vs practice

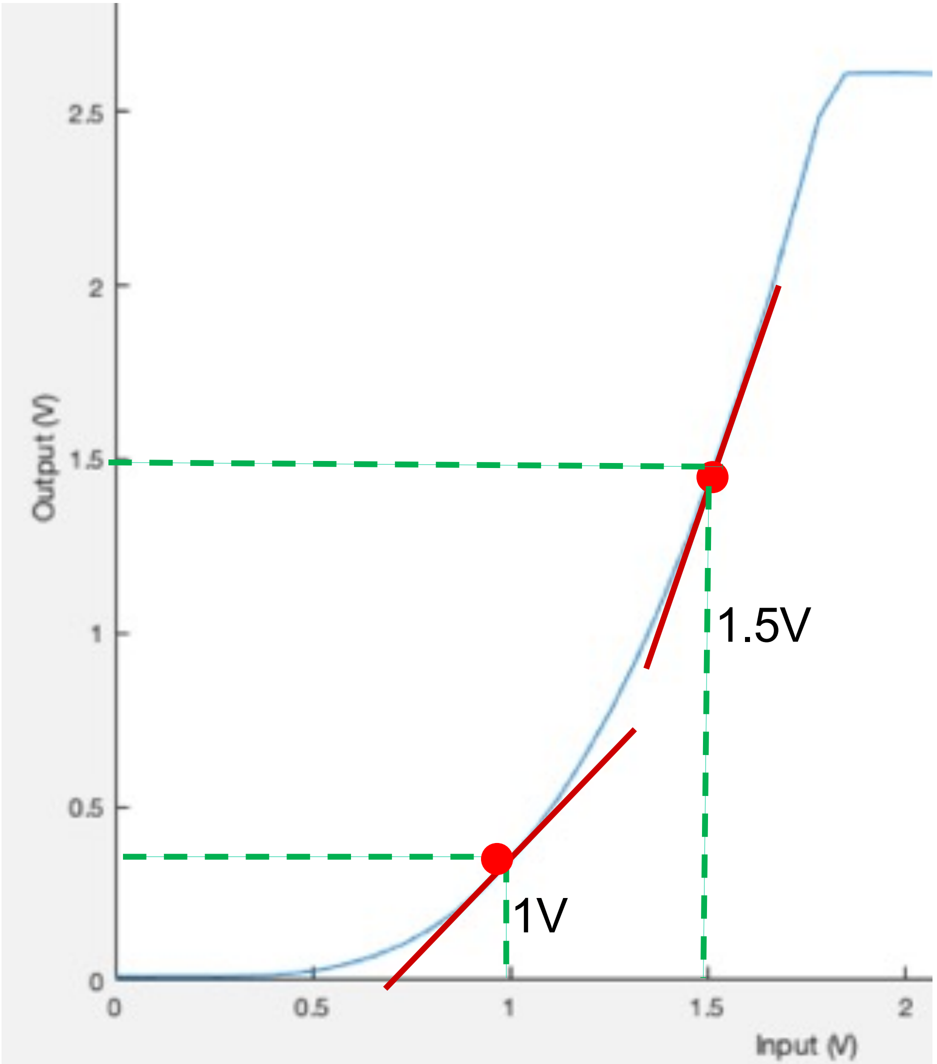


Task 4 – Step Response



Task 4 – Explained





Lab 4 – Task 1: Measuring Angel of tilt – the IMU

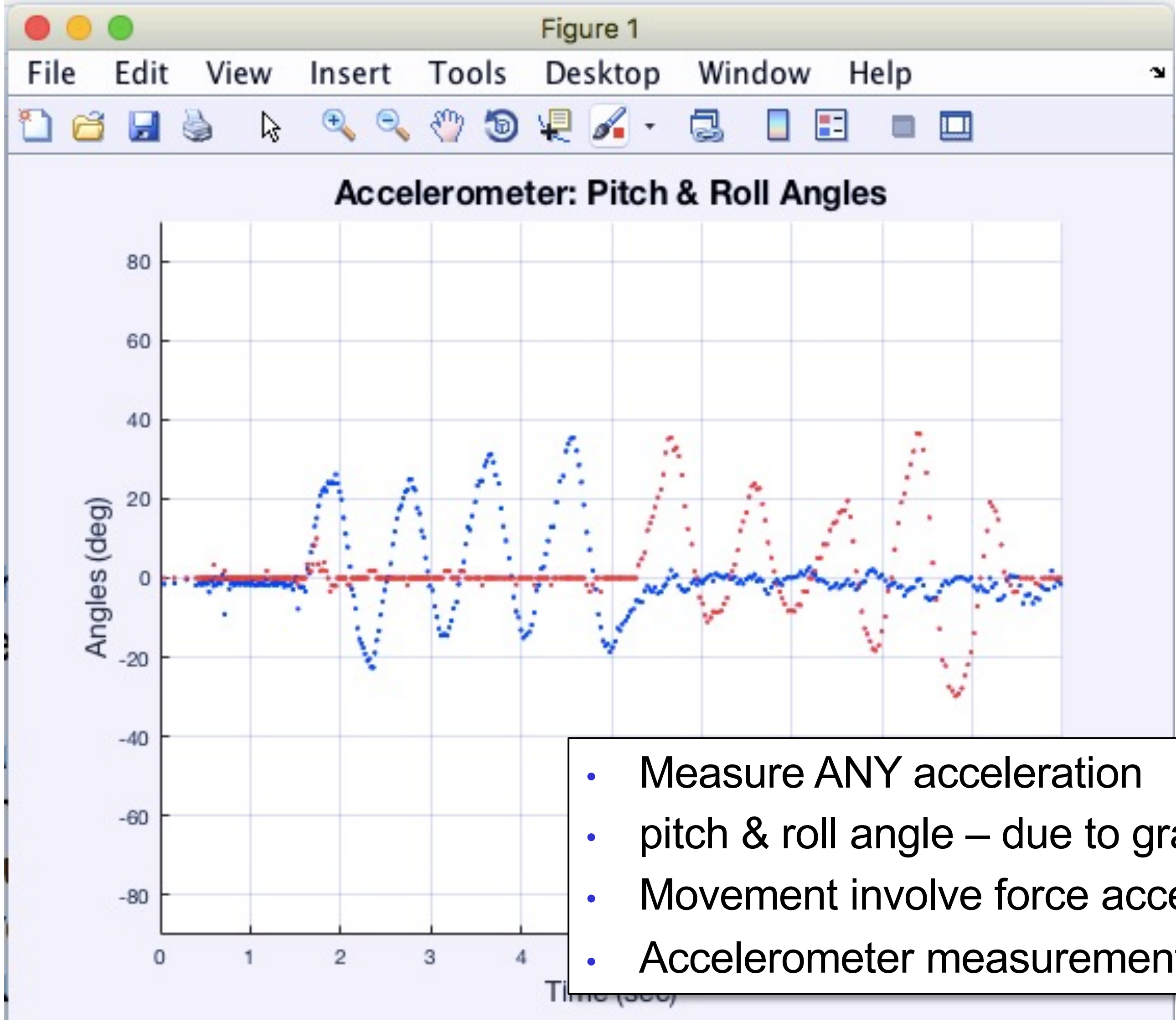
- The IMU – inertia measurement unit – has built in 3-axis accelerometer and 3-axis gyroscope
- Easy to access from Matlab using PyBench:.

```
[p, r] = pb.get_accel();        % p, r  = pitch & roll angle in radians
[x, y, z] = pb.get_gyro();      % x, y, z = rate of rotation in 3-axes in rad/sec
```

- Pitch angle – plane pointing up or down
- Roll angle – plane pointing left or right
- Angle can be in unit radian or degree: degrees = radians *180 / π
- Generally use radian for calculations; use degree for display

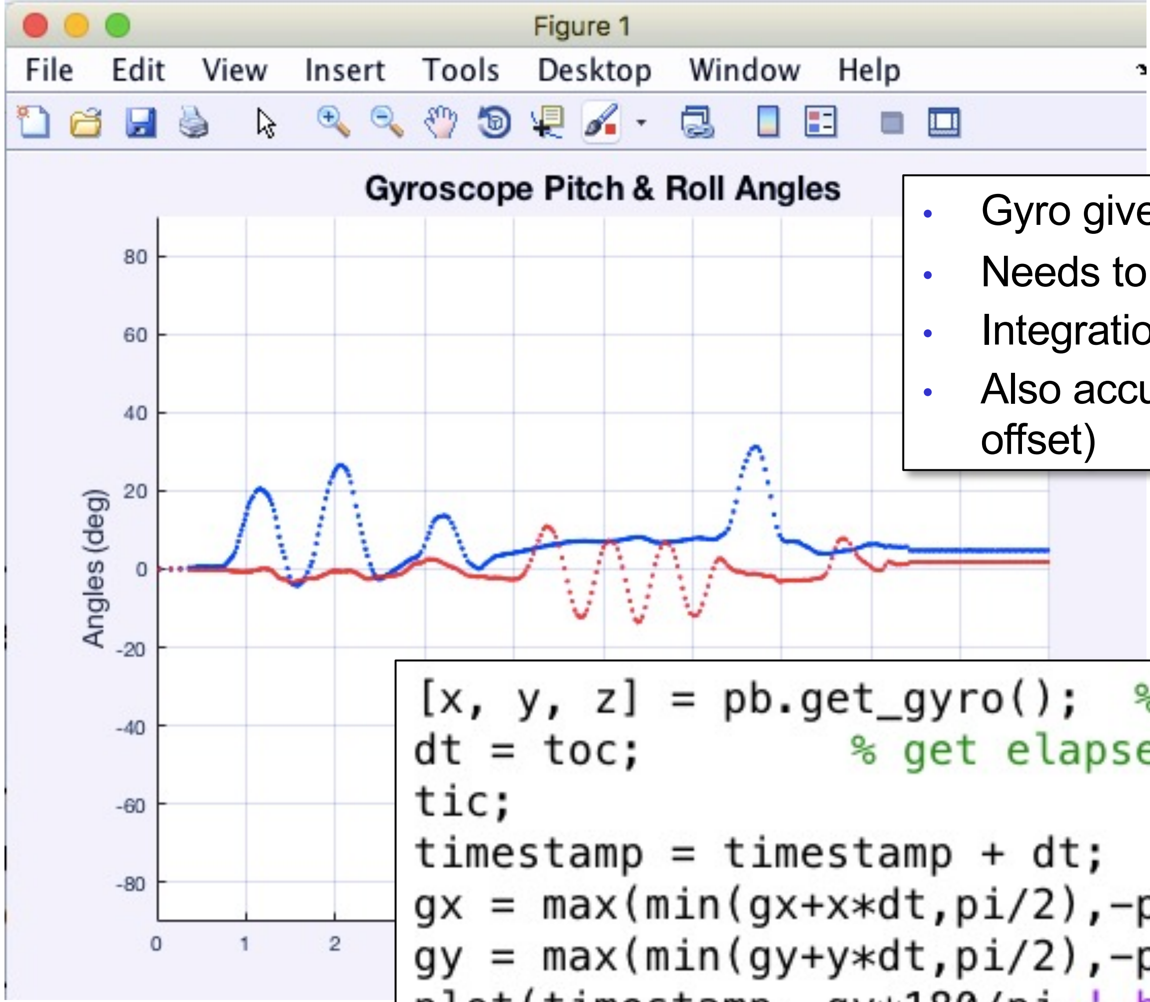
- Learn usefulness and limitations of accelerometer and gyroscope

Lab 4 – Task 1a: Accelerometer



- Measure ANY acceleration
- pitch & roll angle – due to gravity g
- Movement involve force acceleration, also measured
- Accelerometer measurement of tilt angle is NOISY

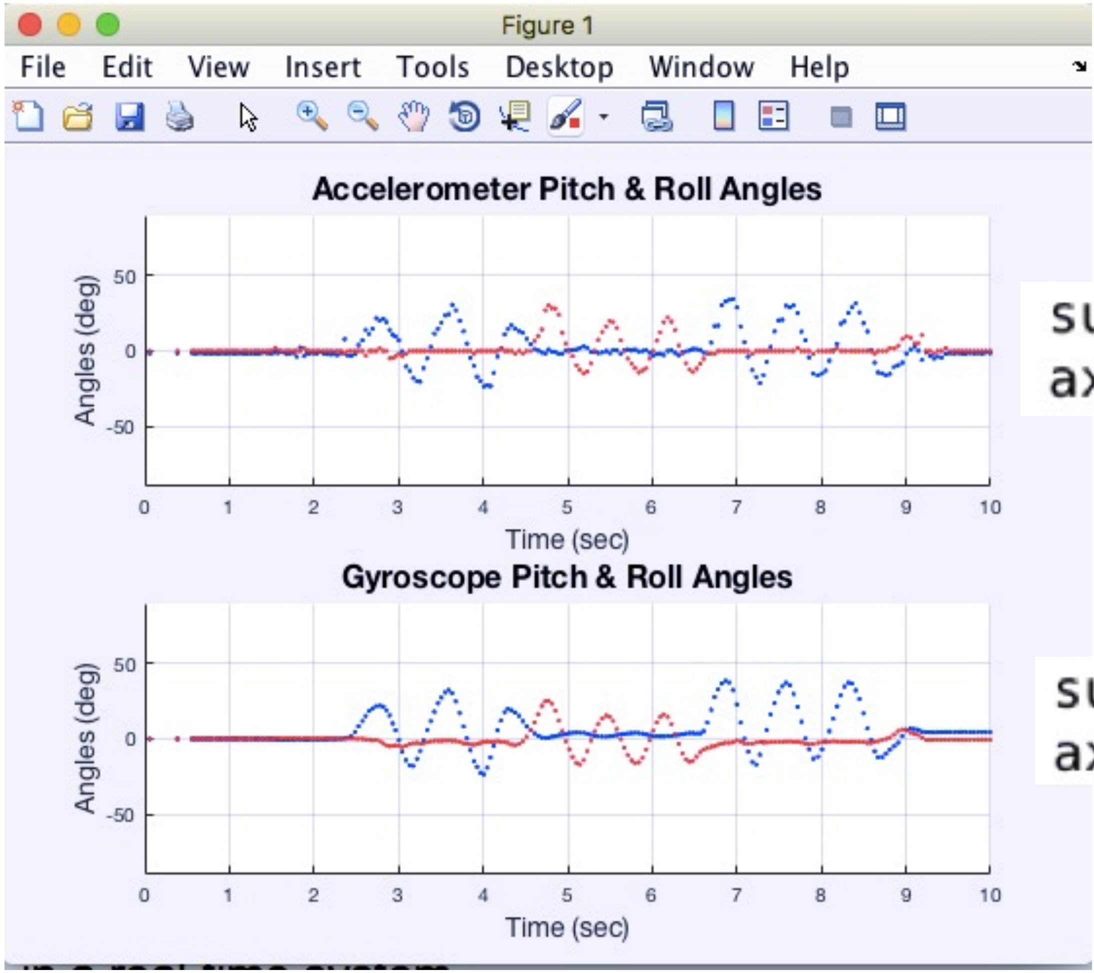
Lab 4 – Task 1b: Gyroscope



- Gyro gives angular velocity, not angle
- Needs to integrate to get angle
- Integration = accumulation
- Also accumulate errors – causing drift (or dc offset)

```
[x, y, z] = pb.get_gyro();  % angular rate in rad/sec
dt = toc;          % get elapsed time
tic;
timestamp = timestamp + dt;
gx = max(min(gx+x*dt,pi/2),-pi/2);  % limit to +/- pi/2
gy = max(min(gy+y*dt,pi/2),-pi/2);
plot(timestamp, gy*180/pi,'.b');    % plot pitch in blue
plot(timestamp, gx*180/pi,'.r');    % plot roll in red
pause(0.001);           % delay for 1 ms, needed for plot
```

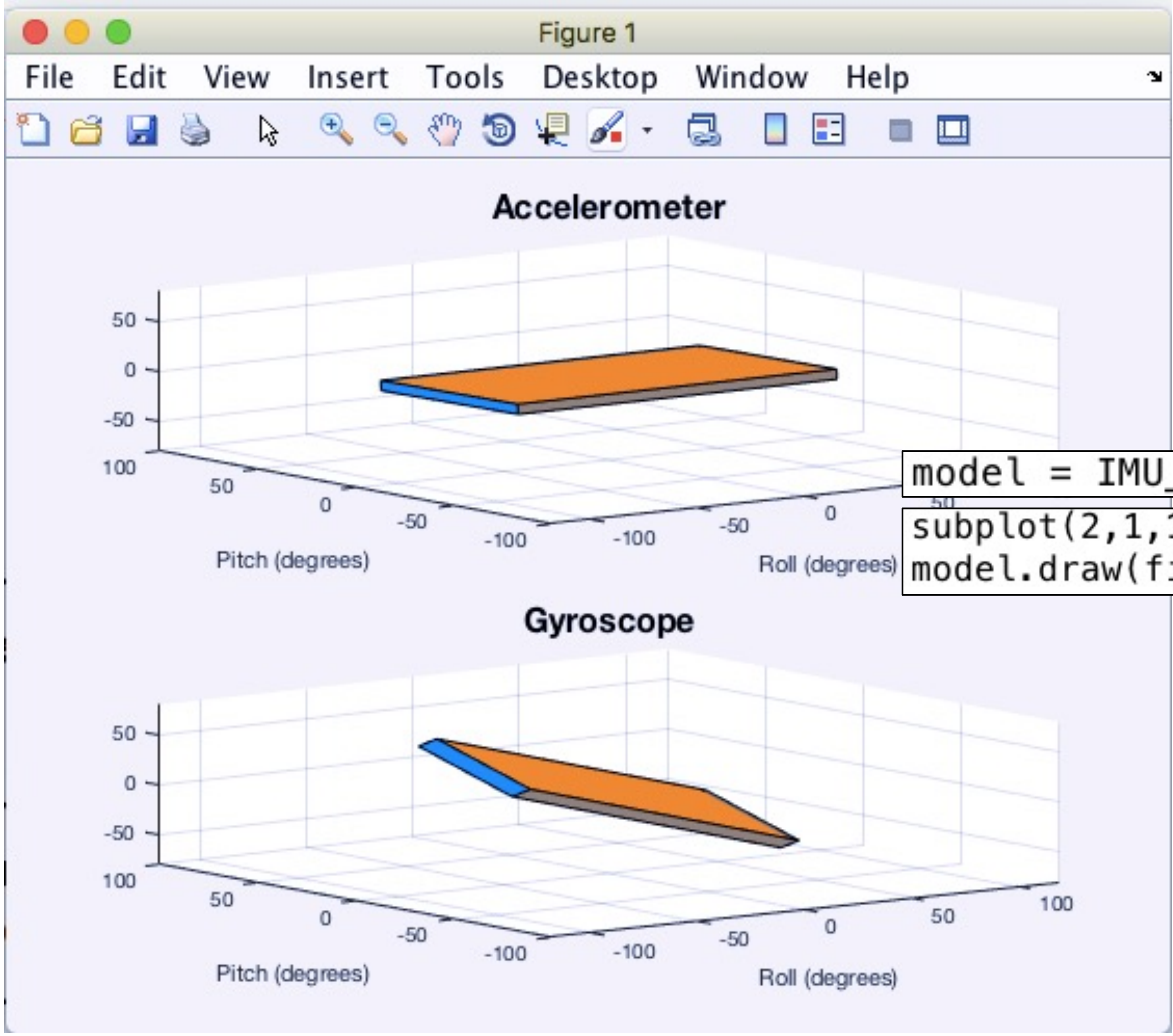
Lab 4 – Task 1c: Gyroscope




```
subplot(2,1,1)
axis([0 end_time -90 90]);
```

```
subplot(2,1,2)
axis([0 end_time -90 90]);
```

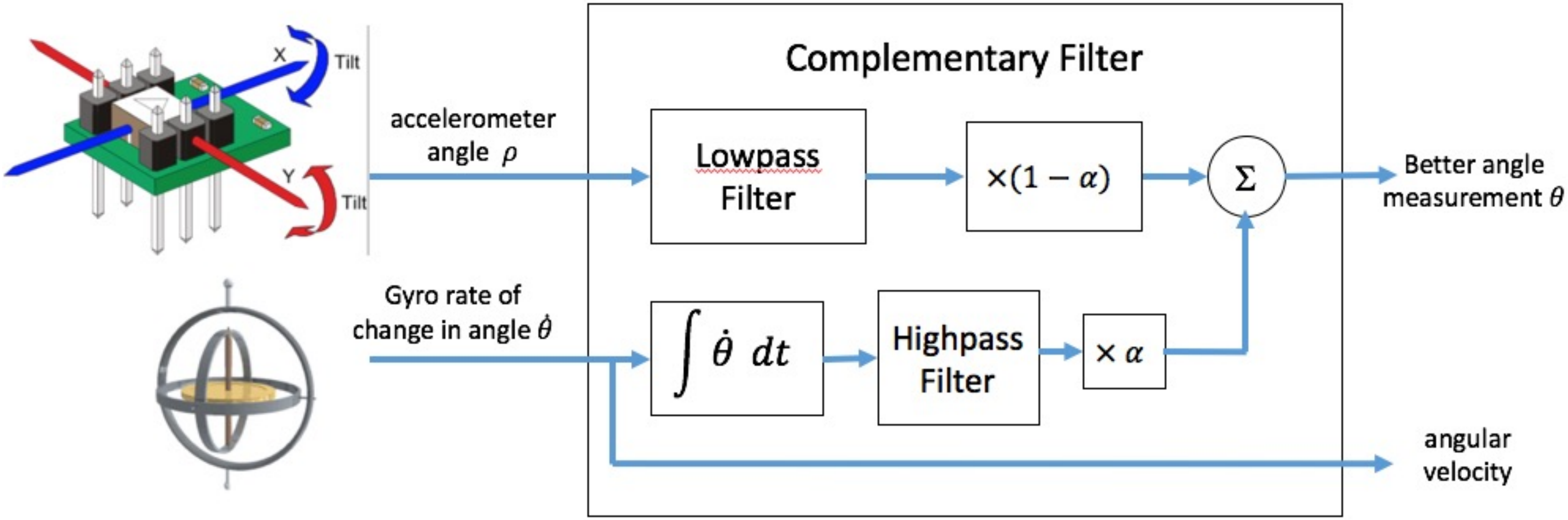
Lab 4 – Task 2: 3D visualization




```
model = IMU_3D();
```

```
subplot(2,1,1);
model.draw(fig1, p, r, 'Accelerometer');
```

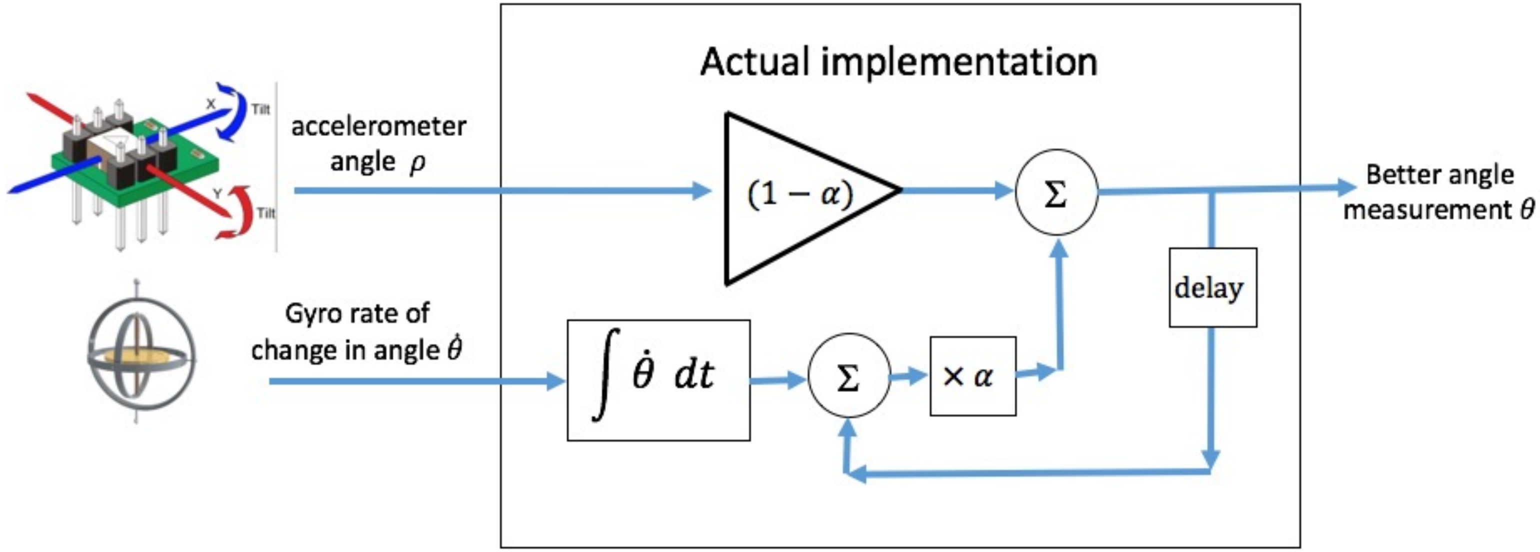
Lab 4 – Task 3: Complementary Filter - Concept



angle $\theta = \alpha \times \left(\theta + \dot{\theta}\, dt \right) + (1-\alpha) \times \rho$

where
- α = scaling factor chosen by users and is typically between 0.7 and 0.98
- ρ = accelerometer angle
- θ = output angle computed
- $\dot{\theta}$ = gyroscope reading of the rate of change in angle
- dt = time interval between gyro readings

Lab 4 – Task 3: Complementary Filter - Implementation



$$\text{angle } \theta = \alpha \times \left(\theta + \dot{\theta}\ dt \right) + (1-\alpha) \times \rho$$

- What happens if $\dot{\theta}$ is zero? Effectively average out the value of ρ
- What happens if $\dot{\theta}$ has a small error? Effectively reduce this error over time

Lab 4 – Task 4: Untethered – OLED Display

```
# Create peripheral objects
b_LED = LED(4)                  # blue LED
pot = ADC(Pin('X11'))           # 5k ohm potentiometer to ADC input on pin X11

# I2C connected to Y9, Y10 (I2C bus 2) and Y11 is reset low active
oled = OLED_938(pinout={'sda': 'Y10', 'scl': 'Y9', 'res': 'Y8'}, height=64,
                   external_vcc=False, i2c_devid=61)
oled.poweron()
oled.init_display()

#  Simple Hello world message
oled.draw_text(0,0,'Hello World!')   # each character is 6x8 pixels

tic = pyb.millis()              # store start time
while True:
    b_LED.toggle()
    toc = pyb.millis()          # read elapsed time
    oled.draw_text(0,20,'Delay time:{:6.3f}sec'.format((toc-tic)*0.001))
    oled.draw_text(0,40,'POT5K reading:{:5d}'.format(pot.read()))
    tic = pyb.millis()          # start time
    oled.display()
    delay = pyb.rng()%1000  # Generate random number btw 0 and 999
    pyb.delay(delay)            # delay in milliseconds
```