LE 6.2 – DA32435 – Dyab Asdi – 19305

TA: Mobina Tavangarifard

Part 1: Open Loop Control

1.

In Lab 6.1, we were able to characterize our PMDC motor with the following equation:

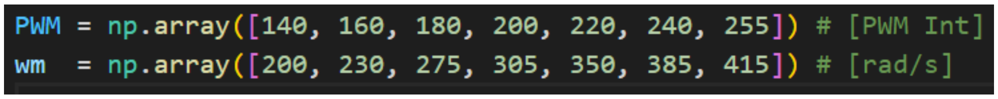
$$T_0 = \frac{r_m}{R_m} \times v_{in,rated} - \left[\frac{r_m^2}{R_m} + B_m\right] \times \omega_m$$

Here, for different voltages, we input speeds that give us the torque value at that speed and voltage. However, for open loop gain, we need to solve for the speed while voltage is being input. Therefore, we must invert the function, and for a given speed, solve for the PWM required to achieve that speed.

Making substitutions with some of our known values of the system, we get the following equation to solve for mcc, which is a ratio ranging from 0 to 1, essentially representing the percent of the bus voltage that is being used for open loop gain.

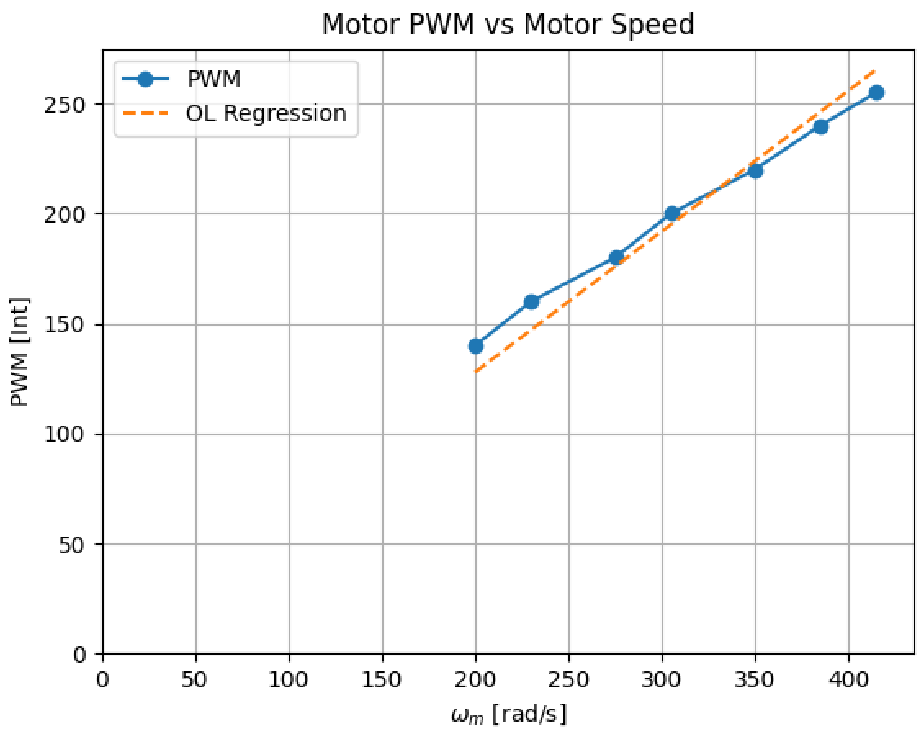
$$mcc = [R_m(T_0 sgn(\omega_{md}) + B_m\omega_{md})\,/r_m + r_m\omega_{md}]\,/v_b$$

In this lab, we used last week's PWM and respective speed data to create a plot of PWM vs Wm.

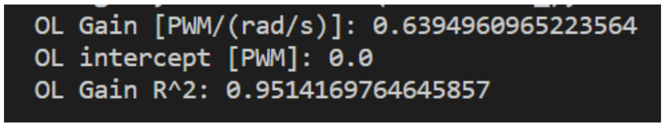


```
PWM = np.array([140, 160, 180, 200, 220, 240, 255]) # [PWM Int]
wm  = np.array([200, 230, 275, 305, 350, 385, 415]) # [rad/s]
```

Plotting these gave us the following graph:



We created a linear regression model, where the slope would give us the following Open Loop Gain, intercept, and R^2 values:



```
OL Gain [PWM/(rad/s)]: 0.6394960965223564
OL intercept [PWM]: 0.0
OL Gain R^2: 0.9514169764645857
```

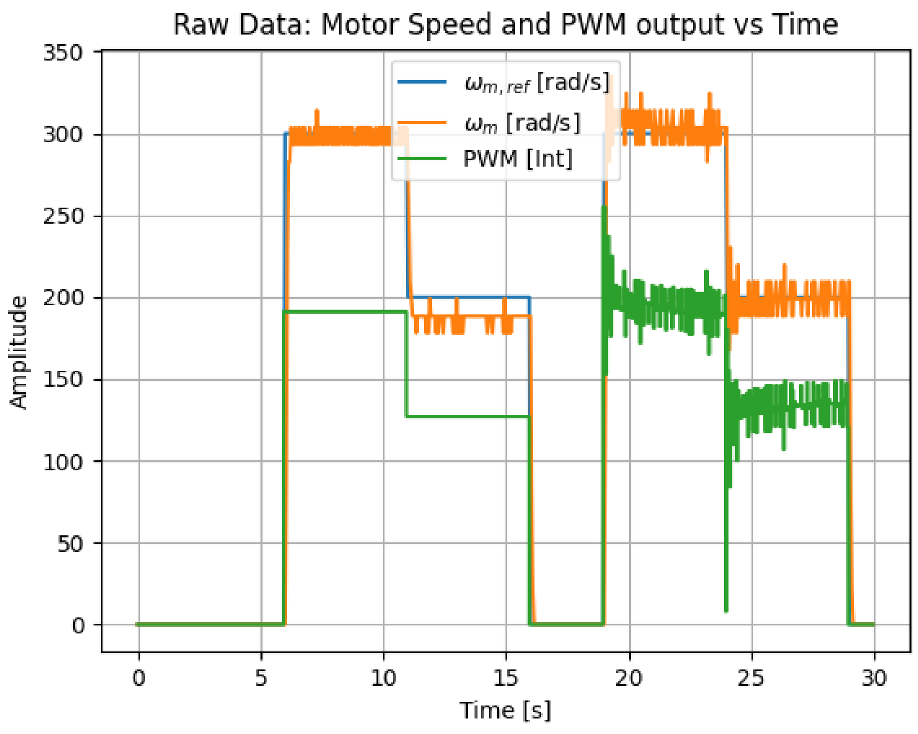
2.

Open Loop Control Performance				
ω_m Ref [rad/s]	ω_m Exp [rad/s]	PWM Out [Integer]	ω_m Error [rad/s]	Percent Error [%]
205	188.5	131	16.5	8.05%
275	261.80	176	13.20	4.80%
325	314.16	208	10.84	3.34%
400	408.41	255	-8.41	2.10%

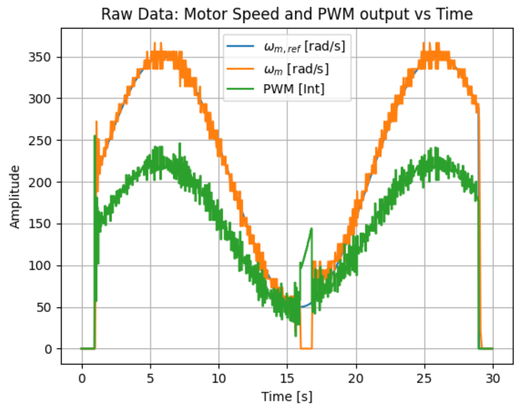
Part 2: Closed Loop Control

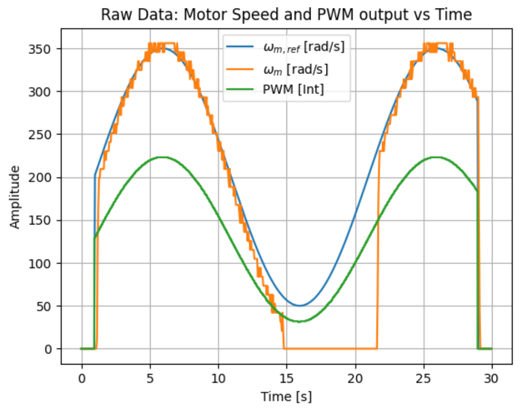
1.

a. Step Reference:



b. Sinusoidal Reference:





2.

Closed Loop Control (PID) Gains		
Proportional	$K_p \left[\frac{PWM}{rad/s}\right]$	0.9
Integral	$K_i \left[\frac{PWM}{rad}\right]$	1.0
Derivative	$K_d \left[\frac{PWM}{rad/s^2}\right]$	0.01

To determine these values, we first researched how each of the components affect the output of a closed loop system. Then, specifying a goal (reaching the target as quick as possible, or having the smoothest transition to steady-state), we adjusted each of the PID gain values in the Arduino software until we achieved a system with the lowest percent error at steady-state.