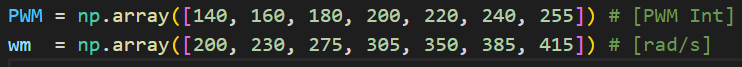
Part 1: Open Loop Control

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

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.

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



Plotting these gave us the following graph:

A graph with a line and a blue line

Description automatically generated with medium confidence

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

A black background with white text

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Open Loop Control Performance** | | | | |
| Ref [rad/s] | Exp [rad/s] | PWM Out [Integer] | 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. Step Reference:

A graph with green and orange lines

Description automatically generated

* 1. Sinusoidal Reference:

A graph of a motor speed and pwm output

Description automatically generatedA graph of a graph

Description automatically generated

|  |  |  |
| --- | --- | --- |
| **Closed Loop Control (PID) Gains** | | |
| Proportional |  | 0.9 |
| Integral |  | 1.0 |
| Derivative |  | 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.