Kyle English

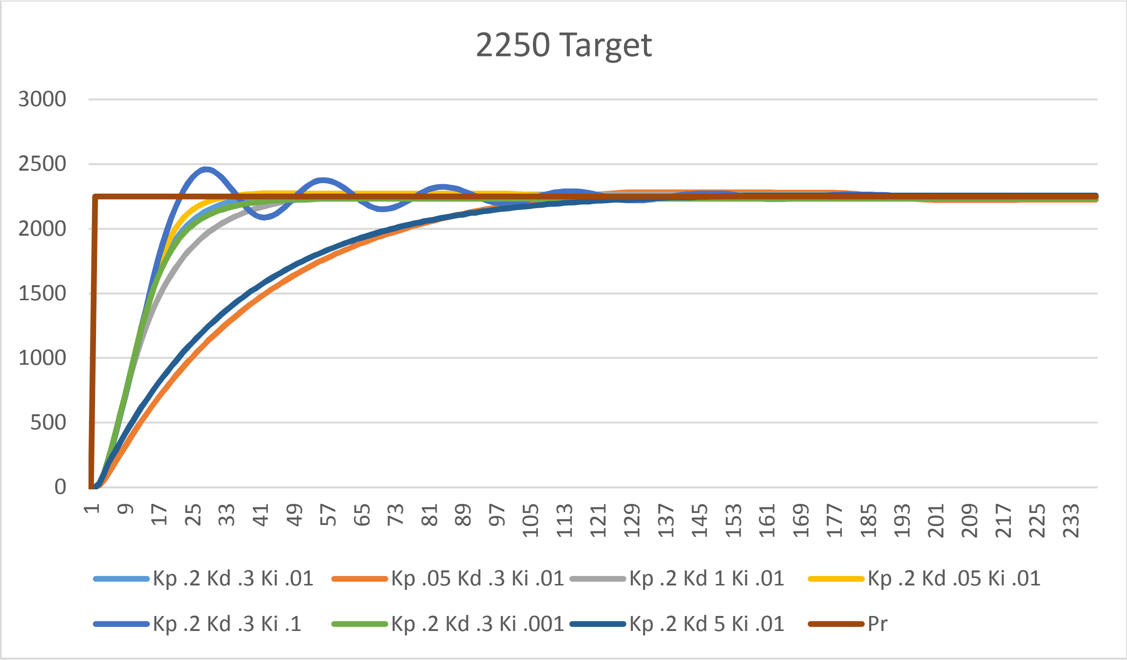
Lab 2: Motors

1. I'm sampling the speed every 10ms (averaging it over 100ms) and updating my Pid controller every 20ms. I had a little trouble getting the torque going in the right direction and converting into torque from counts / sec, but once I was able to do some of those things everything fell into place. My P and D values seems to like to be around the .3 range with the I gain value being around .01. This gets me up to speed reasonably quickly with only a little overshoot/oscillation depending on the speed that is desired (see #2 for more info).

2. Started at 20ms as described above (slow speed = 200, high speed = 6000). It appears to work well at both very low and high speeds. A little overshoot and oscillation on the high speeds, but overall fairly well. If I went to 10ms PID update, then there was some oscillation on the very slow speeds, and significant oscillation on the high speed. If I went to 50ms, then there was minimal overshoot/oscillation for both speeds, but it took longer to get there. I'll stick with the 20ms PID Update.

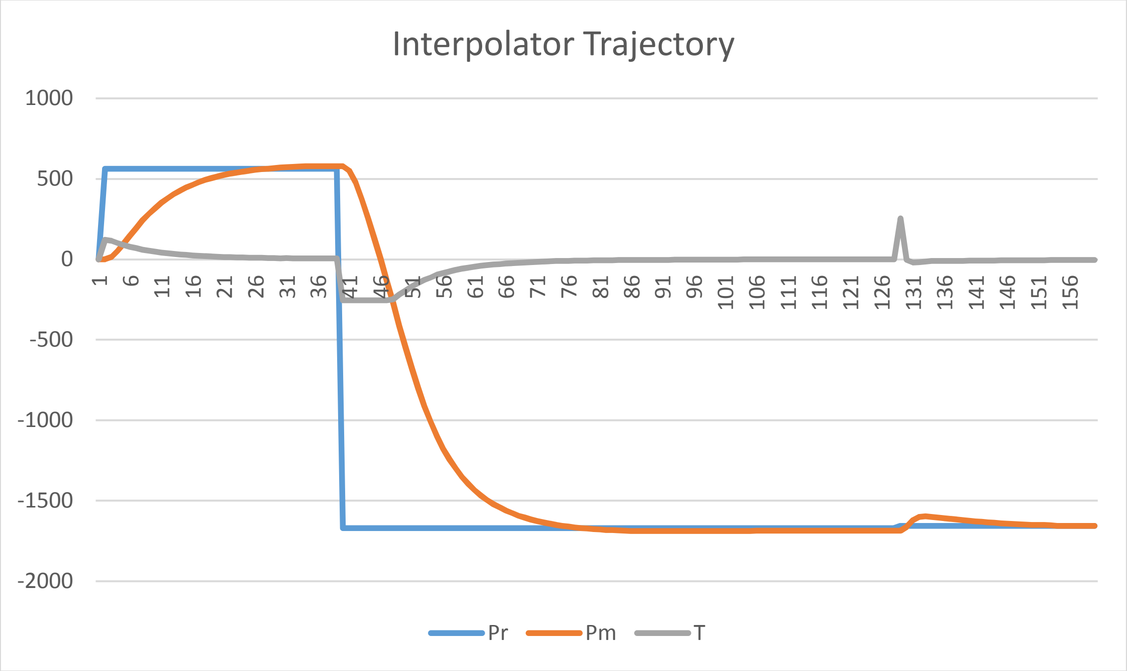
3. Ran into some weird things here. At encode 2000, slightly modified gains work fine, but at 5000 they appear to do something the board/motor doesn't like. Tuned my gains using 2250 (one full rotation, figured that should work for now) as the encode value and went from there. I'm assuming that this is because my torque is changing too rapidly, but I don't have time to debug this fully.

   With the I term out and very low gains, there was some undershoot by the controller. With high gains, there was overshoot, and too high, it even turns the micro off. I ran it at my gain values for speed and it mostly looked good. Ended up lowering my p gain to .2 but that was it. This was because a pGain of .3 killed the micro for an encode value of 2250. In the graph below is the results as I changed my gains around.

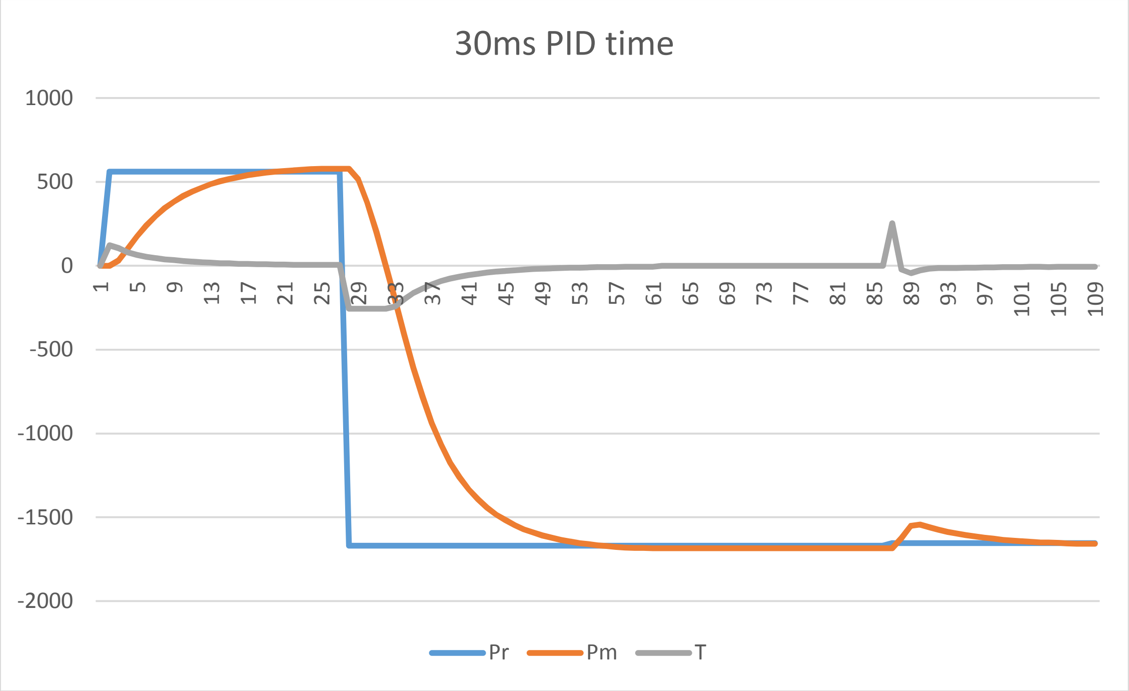


Most of my values are undershooting. This is because I can't really raise my p value any higher without the board turning off (really annoying).

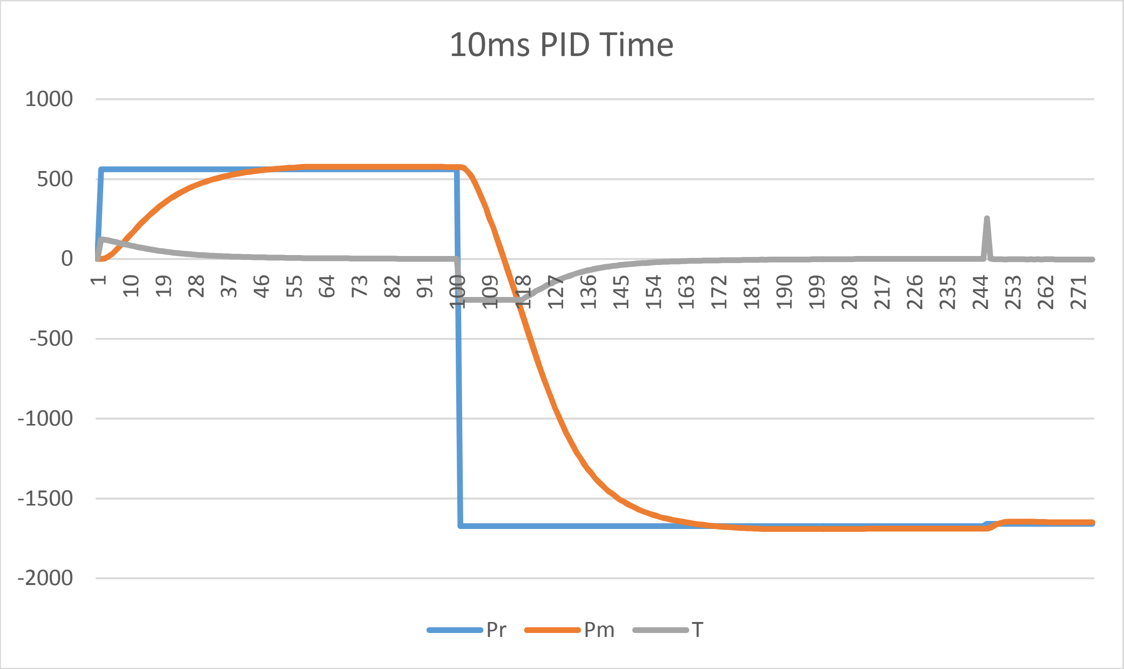
4. I ended up with the below graph for my trajectory using Kp = .2, Kd = .3, and Ki = .01. I think some of these may be a bit high, as seen in the spike in torque for the 5 degree turn and the subsequent overshoot. Having said that, for the other two larger rotations, it gets there quick with little to no overshoot. I noticed earlier that my torque had an inverse relation with the encode increment. I had compensated by simply multiplying the torque by -1 to make this function through the first couple questions. I realized here that my motor was plugged in the opposite direction (+ torque would decrease the encode value) than it was probably supposed to be. I reversed that and got rid of my * -1 to have both values going the same direction. Makes the below graph look much better.



5. I’ve been running my PID update every 20ms intervals for all of the above so I started out at 50ms to see what that response was like. It promptly killed the micro on the backwards 360 command. I could get it to work at 30ms but that’s the best for the “slow” pid portion I could do given the situation. The graph is below. Not much difference between the two.



I then decided to try to run it quicker since I couldn’t go any slower. I ran it at 10ms and got the below graph. It was interesting, because with it running faster I would have expected better speed and accuracy to the target position, but it actually caused it to oscillate ever so slightly around the target so it couldn't move on to the next trajectory.



I think I can tell you what I should have seen just based on my understanding of how the PID works and how frequency should affect it. I believe with a slower response time, you would see varying levels of oscillations (depending on just how much you slowed it down) around the target position since the PID would be slow to respond. These oscillations would be in both the torque and Pm values as it goes around the target, similar to my 10ms only much, much larger. This would cause it to take longer to complete each of the trajectories as it overshoots the target mark.

Looking back on everything, I probably should have sacrificed some speed with lower gain values so (I think) the PID controller could run at slower speeds and wouldn't kill/turn off/reset the micro. I wanted to see good oscillation, but that was tough given the state of things. Having said that, I think the frequencies/gains that I did implement work really well for the given lab assignment. I just wouldn't push it too far beyond that.