LAB 2 - Deliverables

Hand in all of the code necessary to compile your project. Please put sufficient comments in your code so that I can follow what you are doing. Also, if your code is not working in some aspect, please include that in your report. In addition, submit a report that addresses the following:

1. Implement the controller to maintain speed. The pieces to experiment with are the sampling rate and the range of reference speed. Note that you can only achieve about 5 degrees of accuracy due to the sensor, thus you will need to calculate speed using a sliding window of averaged measurements or slow your sampling rate. (This was the old motors. With the new you have better than 1 degree accuracy. But you still want to think about sampling rate and when and where to calculate speed.) Use the information from the warmup exercises to set the frequency of the controller. Tune your gains for an average speed.

1. Use only P. Increase the gain to get a fast response (i.e. get to your set point quickly). Then increase it a little more to oscillate.

|  |  |  |
| --- | --- | --- |
| Kp | Result | Comments |
| 0.01 | N/A | Too small to get the motor moving at all. |
| 0.05 | Slow to get to target | Need to increase to get to target quicker |
| 0.2 | Still slow |  |
| 0.4 | Quick with oscillation | This value seems to meet the described target above the best |
| 1 | Got there real quick and is oscillating | Seems that the higher the values, even though it is quick, the oscillation becomes less noticeable. |
| 2 | Super quick | Mainly steady state |

1. Add in I. Increase the gain to get a faster response and address the steady state error in which once the system achieves its speed, P falls back to 0 thus the speed cannot be steadily maintained. Add in more I so that it is oscillating.

|  |  |  |
| --- | --- | --- |
| Kp | Ki | Result |
| 0.4 | 0.05 | This might be even better |
| 0.4 | 0.1 | Looks like I got the described behavior already |
| 0.4 | 0.5 | Oscillation lasted longer at this rate |
| 0.4 | 1 | Looks like the Oscillation steady out again |

1. Add in D to dampen the response and eliminate oscillation.

|  |  |  |  |
| --- | --- | --- | --- |
| Kp | Ki | Kd | Result |
| 0.4 | 0.5 | .0001 | Seems like less oscillation |
| 0.4 | 0.5 | .0005 | Still oscillation |
| 0.4 | 0.05 | .001 | Closer to target, still some oscillation \* |
| 0.4 | 0.5 | .005 | Oscillation |
| 0.4 | 0.5 | .05 | Brown out |
| 0.4 | 0.5 | .1 | Brown out |

**Observations**: It was really interesting trying to figure out how the values each affected the motor when speeding up to reach the set-point. I got things going really well with P and I, in the sense of how I interpreted the described results in each gain adjustment. No matter what I could not completely eliminate the oscillation with D, but I was able to get really close.

# Report on your observations.

2. Use your gains above to maintain a very low speed, then a very high speed. Do these gains work for both? What if you change the frequency of the speed calculation (not derived inside the interrupt but rather in the controller) to a quarter of the frequency of the encoder interrupt, then what happens to the system? (Essentially, I'm asking you to play around with the system a bit and see what the different settings do and how the system changes. If you have other questions that you are more curious about, then you can experiment with that.) **Report on your observations and form some conclusions (e.g. "For the best results, the frequency of the controller should be set at XXX.").**

**Kp = 0.4, Ki = 0.05, Kd = .001 -> Normal Hz**

|  |  |
| --- | --- |
| Speed | Result |
| 5 | Does work well |
| 50 | Still working well |
| 70 | Still working well |

**Kp = 0.4, Ki = 0.05, Kd = .001 -> 0.25 Hz**

|  |  |
| --- | --- |
| Speed | Result |
| 5 | Does NOT work well |
| 50 | Works ok, it seems closer than at the low speed |
| 70 | This one still works pretty well |

**Observations**: I was impressed that the frequency change was as close as it was. Especially since I did not account for the time change in the calculation.

3. Implement the controller to maintain position. Using a good sampling rate, tune your gains in a similar manner above, and experiment with the speed and the range of reference position.

1. Start with very low gains so that the motor runs very slow and takes its time getting to the set position. Take out the I term, so that you are only using a PD controller. Do you observe any undershoot?

**Kp = 0.2, Ki = 0.03, Kd = .001 -> 0.25 Hz**

|  |  |
| --- | --- |
| Speed | Result |
| 5 | Way undershooting |
| 50 | Minor undershoot. Kind of interesting |
| 70 | This one choked the motor |

**Observations**: I set the Ki to .03 for the set up, but did not use it in the speed samples. The behavior was erratic as I expected. Though not quite as much as I would have expect.

1. Use high gains for positional control, so that the motor mostly runs at full speed. Observe the overshoot and see if you can adjust the gains to maintain both a high speed and not overshoot. For these "ideal" gains you determined, record the measured position, measured speed, reference position, and error. (You can do a screen dump and capture the screen output using PuTTY (or others) then import that data into Excel.) **Graph the variables and describe and explain your findings.**

4. Determine optimally tuned values for the PID positional controller (i.e. those that achieve *good* control while maintaining *good* speed) and the optimal frequency of the controller based on above experiment. Implement the interpolator and execute the trajectory: rotate the motor forward 90 degrees, hold for .5 seconds (have the system precisely measure this time period), then rotate backwards for 360 degrees, hold for .5 seconds, rotate forwards for 5 degrees. **Graph Pm, Pr and T while executing the trajectory. Be sure to graph the entire trajectory.**

5. Execute the same trajectory described above, except run your PD controller at a "slow" rate and at a "very slow" rate while **graphing the same variables. Discuss the results.**