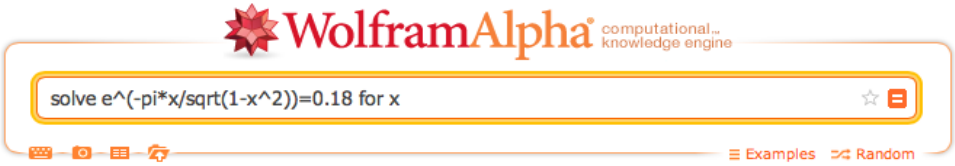
LAB 7: POSITION CONTROL USING P, PD AND LEAD CONTROL

7.1. **Start-Up Procedure.**

(1) Your TA will power up the DCMCT motor unit.
(2) Download the Labview zip file. Unzip the file and place ALL of the contents in a folder on the desktop. Then, within one of the subfolders you will find the exe file. Open it.
(3) Once the system powers on, you will notice that the motor does not move with both $k_p = 0$ and $k_i = 0$. To check to make sure the program is working properly, provide a small value offset value and a value for k_p to make sure the motor responds.
(4) This lab will use several methods, as described in the Prelab, in order to obtain a $PO \leq 18\%$, a $T_p \leq 0.2$ seconds and $T_s \leq 2$ seconds.
(5) Recall that when we discuss voltage saturation, the largest voltage that the motor can receive is ± 15V, so any control voltage that is larger than 15V will be cut off at 15V.
(6) Note that when starting the LABVIEW program, the encoder that measures the angle will always start at 0 degrees, regardless of what is labeled on the mass of the DC Motor.

7.2. **Proportional Position Control.** In this section, you will first examine the theoretical expected performance, then you will use MATLAB to verify the performance, and then run the experiment to see how well the model predicts the actual dynamics.

(1) From the pre-lab, **RECALCULATE** the damping ζ and K_p that will satisfy the PO design specification. Also obtain T_p and ω_d for that K_p. (Hint: In order to solve transcendental equations is often useful to use either the Wolfram Alpha engine (https://www.wolframalpha.com) or the symbolic MATLAB tools. The scripts for each environment, shown here calculating ζ, is




```
>> sym x

ans =

x

>> solve('exp(-pi*x/(1-x^2)^(1/2))=0.18')

ans =

  0.47911103277774525244222637304617
 -0.47911103277774525244222637304617
```

(2) You will now use a MATLAB tool, "SISO (single-in-single-out) tool", in order to view the time simulation of the closed loop. Start MATLAB. In the command terminal, type in the open loop transfer function:

>> tau = 0.09; K = 19; sys = tf([K],[tau 1 0])

Press Enter. You will see the transfer function displayed in the command terminal. Now, in order to open the SISO tool by typing in:

>> sisotool(sys)

There will be several windows that pop-up. See the tutorial about how to use the SISO tool.

(3) Type in the compensator value C = K_p obtained in step (1). Obtain a step response associated with the closed loop transfer function. Place a marker on the peak. **PRINT** the plot, label it 7.2.3.SIM and attach it to the end of the report.

(4) From plot 7.2.3.SIM, **FIND** the simulated percent overshoot, PO_{sim} and $T_{p,sim}$. **COMPARE** the simulated results to the theoretical result from step (1).

(5) Open up a root locus diagram of the roots. Drag the red dots (ie. changing the K_p) and observe changes in the time trace.

(6) Make sure the gain is the calculated K_p and **RECORD** the calculated damping and frequency, ζ_{sim} and $\omega_{d,sim}$, respectively. These values can be found from the root locus plot. **COMPARE** these values to the theoretical result from step (1) (ie. calculate the relative error and qualitatively describe whether or not the difference is acceptable.)

(7) Now that you have an expectation of what should happen, return to the LABVIEW program. For the reference signal, set the amplitude to 180 degrees, the frequency to 0.6 (Hz), the offset to 0 degrees. You will likely want to vary the frequency from 0.1 to 1, depending on the time scale of the dynamics. The frequency is only a suggestion; adjust it so that it is long enough between jumps that you observe a steady state behavior. Set the K_p to the above value and set $K_d = 0$. Make sure that the PD loop is selected.

(8) For a given step response, **PRINT** out the screenshot, label it 7.2.8.EXP.

(9) From the plot 7.2.8.EXP, **OBTAIN** the PO_{exp} and the $T_{p,exp}$ and **COMPARE** these results with the above simulation and theoretical results.

(10) Now, look at the steady state error. You will notice that it may not be zero even though the theory and simulation both suggest it should be zero. **HYPOTHESIZE** why the steady state may not zero.

7.3. **PD Position Control,** $b_{sd} = 0$**.** You will extend the above methods and apply it to the PD control, where $b_{sd} = 0$.

(1) **CALCULATE** the damping ζ and ω_n that will satisfy the PO and T_p design specifications. First solve the PO for ζ and then use that ζ and the T_p to **OBTAIN** ω_n. Also **CALCULATE** the ω_d.

(2) Then, **WRITE** ω_n in terms of K_p and the other known parameters, **SOLVE** for K_p using the results from step (1). **WRITE** ζ in terms of K_p and K_d and solve for K_d. These will be the design parameters you will use for both the simulation and the experiment.

(3) Now, for the SISO tool, you can only have one gain term. As such, we will put the K_d in the transfer function and consider K_p as the variable. In the command terminal, type in the open loop transfer function:

>> tau = 0.09; K = 19; Kd = K_d; sys = tf([K],[tau 1+K*Kd 0])

where K_d is your calculated value. Press Enter. You will see the transfer function displayed in the command terminal. Now, in order to open the SISO tool by typing in:

>> sisotool(sys)

(4) Open up a root locus and a time trace of a step response. Drag the red dots (ie. the K_p value) and observe changes in the response.
(5) Type in the compensator value C = K_p obtained in step (1). Place a marker on the peak of the step response. **PRINT** the plot, label it 7.3.5.SIM and attach it to the end of the report.
(6) From plot 7.3.5.SIM, **FIND** the simulated percent overshoot, PO_{sim} and $T_{p,sim}$. **COMPARE** the simulated results to the theoretical result from step (1).
(7) Make sure the gain is the calculated K_p and **RECORD** the calculated damping and frequency, ζ_{sim} and $\omega_{d,sim}$, respectively. **COMPARE** these values to the theoretical result from step (1) (ie. calculate the relative error and qualitatively describe whether or not the difference is acceptable.)
(8) Now, in LABVIEW, set the K_p and K_d to the above values. Make sure that the PD loop is selected and the reference signal is the same as in the above section. For a given step response, **PRINT** out the screenshot, label it 7.3.8.EXP.
(9) From the plot 7.3.8.EXP, **OBTAIN** the PO_{exp} and the $T_{p,exp}$ and **COMPARE** these results with the above simulation and theoretical results.
(10) Manually vary the K_d value. **DESCRIBE** for an increase/decrease of K_d what happens to the overshoot and the peak time. If you were to design the system to meet the design specs, what would be your ultimate suggestion for the K_p and K_d values?

7.4. **Lead Compensator Position Control.** In this section we do not have many analytical methods so we will largely rely on the SISO tool in MATLAB to help design a suitable compensator.

(1) **IDENTIFY** the desired coordinate in the root locus for the design specification. This coordinate can be found in the Prelab.
(2) In the command terminal, type in the open loop transfer function:

>> tau = 0.09; K = 19; sys = tf([K],[tau 1 0])

Press Enter. Open the SISO tool by typing in:

>> sisotool(sys)

(3) Open up a root locus and a time trace of a step response. Drag the red dots (ie. changing the C value) and observe changes in the response. You will notice that this root locus is exactly what you observed with the proportional control system in the first experiment above.
(4) In order to put in a Lead Compensator you will need to add a zero and pole on the negative real axis of the root locus and $|z| < |p|$, so that the zero should be closer to the origin than the added pole. For example, add a pole at -6 and a zero at -2. **SKETCH** the new root locus

plot and **DISCUSS** what happens as the gain increases (ie. **SKETCH** several time traces corresponding to different gains that illustrate the effect of increasing the gain.)

(5) Adjust the pole and zero placement to obtain a root locus path that will pass through the desired coordinate. Adjusting the C so that the imaginary roots are located at the desired coordinate, **PRINT** both the root locus and the time trace (with a marker at the first peak) labeling them 7.4.5RL and 7.4.5TT and attaching them to the end of the report.

(6) **COMMENT** as to whether the 7.4.5TT response satisfies the PO and T_p and T_s design specifications.

(7) If you were to design this controller, **EXPLAIN** whether you would be limited to only passing through the "desired coordinate" or if you had more than one option for a satisfactory compensator.

(8) Pick a zero, pole, and gain K_c that meets ALL of the specifications. **WRITE** down the full compensator as

$$C(s) = K_c \frac{s+z}{s+p}$$

(Note that $K_c = C * p/z$, where C is the compensator value given in MATLAB and p and z is the pole and zero that you added).

(9) **PLOT** the time trace from the simulation and also obtain a time trace from a LABVIEW experiment (be sure to switch to the Lead-Lag Controller), labeling them 7.4.8.SIM and 7.4.8.EXP. **COMMENT** on their similarities/differences.

(10) What would be your recommendation for implementing a Lead Compensator Controller for meeting the design specifications.