AME 30315; Spring 2015; Homework 12 Due April 29th, 2015

Read: Section 10.1 – 10.2.

Problems:

Homework 12 is the implementation of the controller you designed in Homework 11. If your controller design is similar to the design given in the solution to Homework 11 you should use your own design. If your design is grossly off from the solution design, follow the solution procedure and synthesize a new design for your plant model.

Remember that your design criteria will be a rise time of less than 1.5 seconds and an overshoot of less than 25%.

Your controller was designed in continuous time as you were using the theoretical framework explored in AME30315. However, the inverted pendulum, and most modern control systems for that manner, use digital controllers and thus your continuous-time controller must be converted to a digital difference equation. This paradigm of designing in continuous-time and then converting to digital right before implementation is a common practice in industry. You will be using the discrete-time transfer function, which is in terms of variable \( z \), instead of the continuous-time transfer function that you are used to, which is terms of variable \( s \).

(A) Matlab will perform the continuous-time to discrete-time conversion for you. Enter in your controller design as a transfer function in the Matlab workspace. Next type in \( \text{c2d}(K, ts) \) where \( K \) is the continuous-time transfer function and \( ts \) is the sampling time (\( ts = 1/20 \) seconds for the inverted pendulum). Report \( K(z) \).

(B) You next need to convert the discrete-time transfer function to a difference equation (discrete-time version of a differential equation). The following provides an example of how to write the difference equation.
\[ K(z) = \frac{U(z)}{E(z)} = \frac{\beta_1 z^1 + \beta_0}{z^2 + \alpha_1 z^1 + \alpha_0} = \frac{\beta_1 z^{-1} + \beta_0 z^{-2}}{1 + \alpha_1 z^{-1} + \alpha_0 z^{-2} } \]  

(1)

\( U(z) \) is the discrete-time frequency representation of the controller output / plant input (called the z-transform instead of the Laplace transform) and \( E(z) \) is the discrete-time frequency representation of your error. The difference equation can be easily found from \( \frac{U(z)}{E(z)} \). If \( U(z) \) is the frequency-domain representation of \( u[k] \), where \( k \) is your discrete time-step \((k = 0, 1, \ldots)\), then \( z^{-1}U(z) \) is the frequency-domain representation of \( u[k-1] \); in summary, \( z^{-1} \) performs a one time-step time shift, \( z^{-2} \) performs a two time-step time shift, etc. Using this basic relationship you can write the difference equation from Eqn. (1):

\[ u[k] = -\alpha_1 u[k-1] - \alpha_0 u[k-2] + \beta_1 e[k-1] + \beta_0 e[k-2] \].  

(2)

Your difference equation will require variables to hold the previous two values of the applied torque and measured error. These variables are initialized in line 82 in StudentControlCode.c; if you want more variables to be held you will have to initialize more variables. The control loop runs at 20 Hz. The only place you have to edit the code, besides if you are initializing more variables, is between lines 108 and 168. Please note this is between lines 108 and 168 in the code you will download, these lines will change as you add lines to your code. The current position, in degrees*100, is calculated in line 107. The current error is calculated in line 111, in degrees*100. HINT: you will need to change this line of code in some way to work with your difference equation. You will implement your difference equation at line 117. Set the variable \( u \) to equal the output torque calculated by the difference equation. The next several lines involve checking motor limits and setting the torque to the pendulum.
Starting at line 134 you may reset your variables indicating the calculated torques and errors of the last two steps.

PLEASE NOTE: The processor CANNOT support floating point variables (anything with a decimal point). To reduce truncation error, you should multiply the coefficients in your difference equations by a high enough power so that their values are fairly represented by the processor. Once the torque is calculated with these difference calculations, divide that torque by whatever constant your coefficients were multiplied by. (e.g. if you have a coefficient of 0.18 in your difference equation you would have to multiply your coefficients by AT LEAST 100, using 18 as the coefficient. Then when the calculated torque is determined, set the variable u to torque/100). HINT: This will require changing the variable for the current error as well.

(C) Run code StudentControlCode.c to evaluate how well your controller tracks the reference signal. Does your inverted pendulum work as expected? If it does not work as expected, revisit your work and make sure that everything is correctly performed; receive assistance from the TA if required. Once working properly, plot both the reference trajectory and the output position on the same plot, clearly labeling each signal. Measure and report the rise time, settling time, overshoot and steady-state error to see if it matches your predictions; note any discrepancies.

(D) Try four different reference positions within the range ±15° by modifying line 87 of code StudentControlCode.c. You may alternatively have the desired position change as a function of time by resetting the desired position at line 166. Report an average value for the rise time, settling time, overshoot and steady-state error for all final reference positions tested.

(E) A successfully designed feedback controller is capable of compensating for unforeseen disturbances to the system. With your successful controller design running, bump the pen-
dulum with your hand and record and observe the response. Plot the data from this event, demonstrating that the system recovered from the unexpected disturbance. Clearly indicate on the plot when the bump happened and measure how long it takes your system to recover from this bump; the recovery time should be approximately the same as the settling time for the system.

(F) **Extra Credit 1:** The extra credit is for those who are interested in robotic control design and want to get more involved with the project. As such, the extra credit portion is fairly unstructured as to not stifle creativity. Extra credit will be given on a case-by-case basis in proportion to the difficulty of the project. Please discuss your project idea with Prof. Hoelzle to get a ‘quote’ on how much the project will be worth if completed successfully. Some examples are given below.

- Design and implement a lead-lag controller to attain a smaller $e_{ss} < 1$ deg (+10pts). The large $e_{ss}$ is a major deficiency of the lead controller you designed during this project.
- Design and implement a more elaborate reference signal and demonstrate that your controlled system can accurately track this reference signal (+20pts).
- Design a switched control system that changes controller parameters as a function of time or pendulum position. (+25pts).

**Team Effort Allocation.** You will receive a team grade, but there could be some team members that simply did not pull their weight. If you feel like this is the case you can report effort allocations and we will adjust the grades accordingly. Report via this poll: [https://www.surveymonkey.com/s/QXW6NRG](https://www.surveymonkey.com/s/QXW6NRG)

Report for each student in your team. Column assignments refer to the columns in the AME30315_Pendulum_Teams.xls file on the course website. For instance, if a member on Team 1 thought that the allocations should be Kevin Andres = 40%, Daniel Lynch = 20%, and Matthew
Williams = 40% they should enter Column 1 = 40, Column 2 = 20, Column 3 = 40. If you do not submit a survey response it will be assumed that you are requesting a 33%, 33%, 33% split.

**Extra Credit 2.** You will receive 5 points extra credit for showing proof of filling out your CIF report. Save a screenshot of your CIF *confirmation* and submit to the course Dropbox on Sakai. Do not take a screenshot of your actual evaluation values for Prof. Hoelzle.