Due: Monday January 4 in class
Exam 1, Thursday January 7

1) For each of the following transfer functions, determine if the system is asymptotically stable, and if so, the estimated 2% settling time for the system. Assume the sampling interval is $T = 0.1$ seconds.
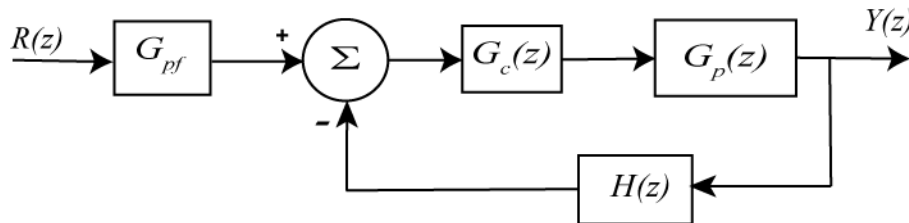
a) $H(z) = \dfrac{z+2}{(z-0.1)(z+0.2)}$ **d)** $H(z) = \dfrac{1}{z^2 + z + 0.5}$

b) $H(z) = \dfrac{1}{(z-2)(z+0.5)}$ **e)** $H(z) = \dfrac{z-1}{z^2 + 0.5z + 0.2}$

c) $H(z) = \dfrac{1}{(z-0.1)(z-0.5)}$ **f)** $H(z) = \dfrac{1}{z^2 + z + 5}$

Scambled Answers: 0.497, 0.58 , 1.15, 0.24, two unstable systems

2) For the following system, assuming the closed loop systems are stable, determine the prefilter gain G_{pf} that will result in zero steady state error for a unit step input. Are any of these systems type one systems?

a) $G_p(z) = \dfrac{0.2}{z^2 + 0.1z + 0.2}$, $G_c(z) = \dfrac{z}{z-1}$, $H(z) = 1$

b) $G_p(z) = \dfrac{0.2}{z^2 + 0.1z + 0.2}$, $G_c(z) = \dfrac{1}{z}$, $H(z) = 1$

c) $G_p(z) = \dfrac{1}{z^2 + 0.4z + 0.04}$, $G_c(z) = \dfrac{0.2}{z+0.2}$, $H(z) = \dfrac{1}{z+0.2}$

Answers: 7.5, 9.47, one is type one (so the prefilter has value 1)

3) Consider the continuous-time plant with transfer function

$$G_p(s) = \frac{1}{(s+1)(s+2)}$$

We want to determine the discrete-time equivalent to this plant, $G_p(z)$, by assuming a zero order hold is placed before the continuous-time plant to convert the discrete-time control signal to a continuous time control signal.

a) Show that if we assume a sampling interval of T, the equivalent discrete-time plant is

$$G_p(z) = \frac{z(0.5 - e^{-T} + 0.5e^{-2T}) + (0.5e^{-T} - e^{-2T} + 0.5e^{-3T})}{(z - e^{-T})(z - e^{-2T})}$$

Note that we have poles were we expect them to be, but we have introduced a zero in going from the continuous time system to the discrete-time system.

b) The Matlab script **continuous_discrete_ft.m** shows how to use Matlab to convert a discrete-time plant to a continuous-time plant, just as we have done by hand here. It also shows how to simulate and plot the two responses and the effect of a zero order hold on our system. Run this code and turn in your plot.

c) Modify the code to for three different continuous-time transfer functions and run the simulations, and turn in your plots.

4) Prelab: In this (and the next) lab you will be using Matlab's sisotool to simulate and implement discrete-time PI and PID controllers for your one degree of freedom systems. This prelab presents a brief review of Matlab's *sisotool* (the R2008 version) and some of the things you will need to know to apply this to our problem.

The file **DT_PID.mdl** is a Simulink model that implements a discrete-time PID controller. It is somewhat unusual in that the plant is represented in state-variable form, but this is the usual form we will using in this class. The Simulink model looks like the following:
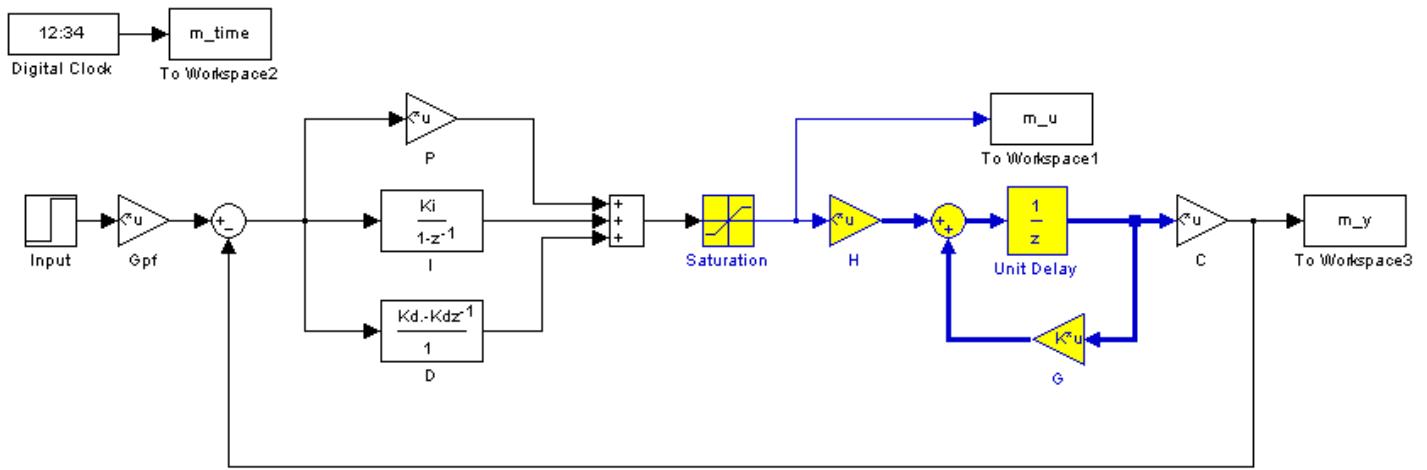
The file **DT_PID_driver.m** is the Matlab file that runs this code. We will be utilizing Matlab's *sisotool* for determining the pole placement and the values of the gains.

Before we go on, we need to remember the following two things about discrete-time systems:

- For stability, **all poles of the system must be within the unit circle.** However, zeros can be outside of the unit circle.

- The closer to the origin your dominant poles are, the faster your system will respond. However, the control effort will generally be larger.

The basic transfer function form of the components of a discrete-time PID controller are as follows:

Proportional (P) term : $C(z) = K_p E(z)$

Integral (I) term: $C(z) = \dfrac{K_i}{1-z^{-1}} = \dfrac{K_i z}{z-1}$

Derivative (D) term : $C(z) = K_d(1-z^{-1}) = \dfrac{K_d(z-1)}{z}$

PI Controller: To construct a PI controller, we add the P and I controllers together to get the overall transfer function:

$$C(z) = K_p + \frac{K_i z}{z-1} = \frac{(K_p + K_i)z - K_p}{z-1}$$

In *sisotool* this will be represented as

$$C(z) = \frac{K(z^2 + az)}{z(z-1)} = \frac{K(z+a)}{(z-1)}$$

In order to get the coefficients we need out of the *sisotool* format we equate coefficients to get:

$$K_p = -Ka, \quad K_i = K - K_p$$

PID Controller: To construct a PID controller, we add the P, I, and D controllers together to get the overall transfer function:

$$C(z) = K_p + \frac{K_i z}{z-1} + \frac{K_d(z-1)}{z} = \frac{K_p z(z-1) + K_i z^2 - K_d(z-1)^2}{z(z-1)} = \frac{(K_p + K_i + K_d)z^2 + (-K_p - 2K_d)z + K_d}{z(z-1)}$$

In *sisotool* this will be represented as

$$C(z) = \frac{K(z^2 + az + b)}{z(z-1)}$$

In order to get the coefficients we need out of the *sisotool* format we equate coefficients to get:

$$K_d = Kb, \quad K_p = -Ka - 2K_d, \quad K_i = K - K_p - K_d$$

For the PID controller, we can have either two complex conjugate zeros or two real zeros.

Sisotool (Brief) Example

A) *Run the Matlab program **DT_PID_driver.m**. This program is set up to read the data file bobs_1dof_205.mat, which is a continuous time state variable model for a one degree of freedom torsional system, and implement a P controller with gain 0.0116. It will put the value of the transfer function for your system, $G_p(z)$, in your workspace.*

> *In **sisotool**, you will see the transfer functions form of the controllers. To determine the parameters k_p, k_i, and k_d you need to equate powers of s of the transfer functions with the forms above. It is easies t if you use the zero/pole/gain format for the compensators. To do this click on **Edit → SISO Tool Preferences → Options** and click on **zero/pole/gain**.*

B) Read the Appendix to review *sisotool*

C) *Entering a Compensator (Controller). We will implement a PI controller here*

- Click on **Designs**, then **Edit Compensators**. Right click in the **Dynamics** window to enter real poles and zeros. You will be able to changes these values very easily later. Since we want a PI controller, we need a pole to be a 1 and we need to be able to change the value of the zero. For now assume the zero is at -1.
- Look at the form of *C* to be sure it's what you intended, and then look at the root locus with the compensator.
- You can again see how the step response changes with the compensator by moving the locations of the zero (grab the pink dot and slide it) and moving the gain of the system (grab the squares and drag them). Remember we need all poles and zeros to be inside the unit circle for stability!
- Move the pole an zero around until the zero is approximately -0.295 and the gain is approximately 0.0563.

D) *Printing/Saving the Figures:*

To save a figure **sisotool** has created, click **File → Print to Figure. Print out this figure and attach it to the homework.**

E) *Back to Matlab.*

- Determine the correct values of a and K
- Enter these in the Matlab code **DT_PID_driver.m**
- Modify **DT_PID_driver.m** to compute the proportional and integral gains
- Run **DT_PID_driver.m** and print out the picture and attach it to this homework. It should look like Figure 1.

F) Now your one degree of freedom system

Choose one of your one degree of freedom systems (if there are two partners, each should choose a different system) and use a sampling interval of 0.1 seconds. For torsional (model 205) systems, assume a 15 degree step, for rectilinear (model 210) systems assume a 1 cm step. Use sisotool to determine a PI controller so you system has a settling time less than 1.5 seconds and a percent overshoot less than 25%. The control effort must also be within the allowed bounds, though this may be different than that output by *sisotool* since *sisotool* always assumes a step of value 1.

- Print out the root locus plot
- Determine the step response using sisotool and print out the graph
- Determine the step response using **DT_PID_driver.m** and print out the graph
- Use sisotool to determine a PID controller so you system has a settling time less than 1.75 seconds and a percent overshoot less than 25%. The control effort must also be within the allowed bounds.
- Print out the root locus plot
- Determine the step response using sisotool and print out the graph
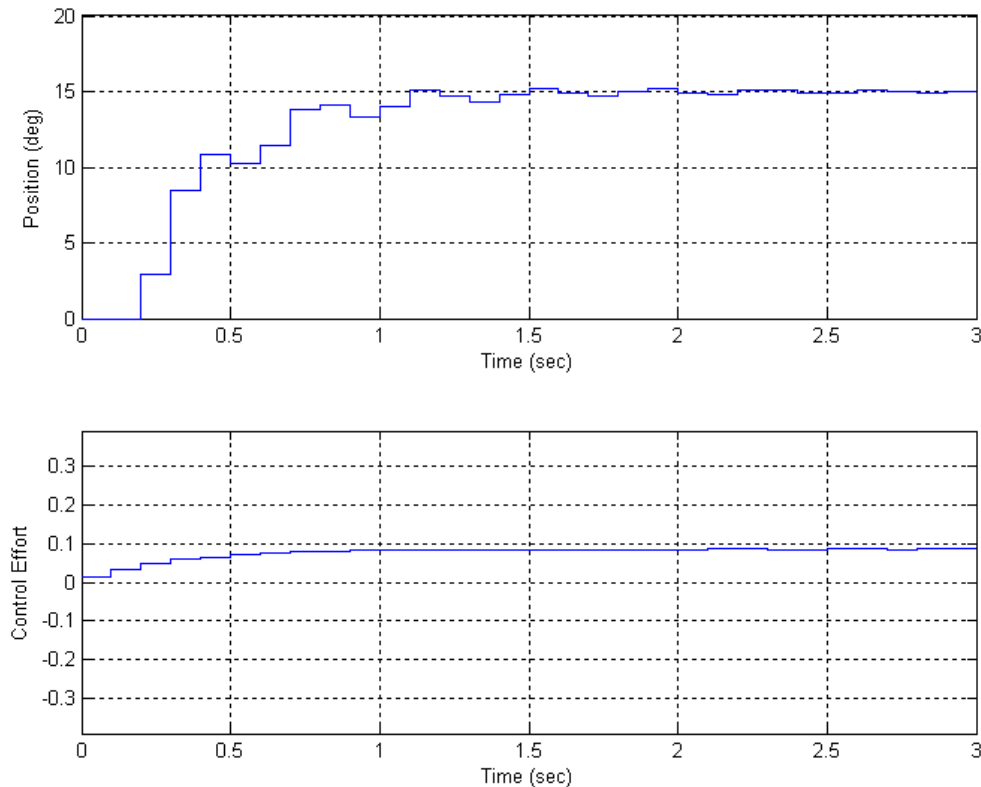- Determine the step response using **DT_PID_driver.m** and print out the graph

Figure 1: Matlab/Simulink results for PI controller.

Getting Started

- Enter the transfer function for the plant, *G(s)*, in your workspace.
- Type **sisotool** in the command window
- Click **close** when the help window comes up
- Click on **View,** then **Design Plots Configuration** and turn off all plots except the **Root Locus** plot

Loading the Transfer Function

- Click on **file → import.**
- We will usually be assigning $G_p(z)$ to block *G* (the plant), so type your transfer function name next to G and then **enter**. You must hit enter or nothing will happen (you can also use the Browse function).
- Once you hit enter, you should be able to click on the **OK** at the bottom of the window. The window will then vanish.
- Once the transfer function has been entered, the root locus is displayed. Make sure the poles and zeros of your plant are where you think they should be.

Generating the Step Response

- Click on **Analysis → Response to Step Command**
- You will probably have two curves on your step response plot. To just get the output, type **Analysis → Other Loop Responses**. If you only want the output, then only *r* to *y* is checked, and then click **OK**. However, sometimes you will also want the *r* to *u* output, since it shows the control effort for P, I, and PI controllers.
- You can move the location of the pole in the root locus plot by putting the cursor over the pink button and holding the left button down as you move the pole locations. You should note that the step response changes as the pole locations change.
- The bottom of the root locus window will show you the closed loop poles corresponding to the cursor location. However, if you need all of the closed loop poles you have to look at all of the branches.

Entering a Compensator (Controller)

- Type **Designs →Edit Compensator**
- Right click in the **Dynamics** window to enter real poles and zeros. You will be able to changes these values very easily later.

- You can either edit the pole/zero locations in this window, or by grabbing the poles and zeros in the root locus plot and moving them. However, sometimes you just have to go back to this window.
- Look at the form of *C* to be sure it's what you intended, and then look at the root locus with the compensator.
- You can again see how the step response changes with the compensator by moving the locations of the poles (grab the pink dot and slide it).
- You can also change the location of the pole and zeros of the compensator by grabbing them and sliding them. Be careful not to change the poles and zeros of the plant though!

Adding Constraints
- Right Click on the Root Locus plot, and choose **Design Constraints** then either **New** to add new constraints, or **Edit** to edit existing constraints.
- At this point you have a choice of various types of constraints.

Printing/Saving the Figures:

To save a figure **sisotool** has created, click **File → Print to Figure**

Odds and Ends :

You may want to fix the axes. To do this,

- Right click on the Root Locus Plot
- Choose **Properties**
- Choose **Limits**
- Set the limits and turn the **Auto Scale** off

You may also want to put on a grid, as another method of checking your answers. To do this, right click on the Root Locus plot, then choose **Grid**

It is easiest if you use the zero/pole/gain format for the compensators. To do this click on **Edit → SISO Tool Preferences → Options** and click on **zero/pole/gain**.