ECE-420: Discrete-Time Control Systems
Homework 3

Due: **Thursday** September 25 at 5 PM
Exam 1, Friday September 26

1) Consider the continuous-time plant with transfer function

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

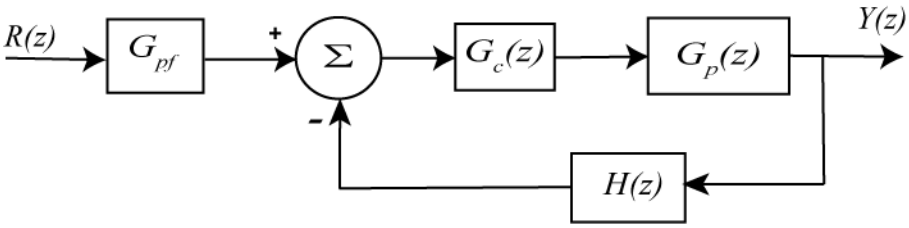
We want to determine the discrete-time equivalent to this plant, $G_p(z)$, by assuming a zero order hold is placed before the continuous-time plant to convert the discrete-time control signal to a continuous time control signal.

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.

2) In this problem assume the feedback configuration shown below.



a) Assume

$$H(z) = z^{-1}, G_p(z) = b_0^p, G_c(z) = \frac{b_0^c z + b_1^c}{z-1}$$

If the plant is equal to 3 and we want all of the closed loop poles at -0.5, show that the controller and closed loop transfer functions are

$$G_c(z) = \frac{\frac{2}{3}z + \frac{1}{12}}{z-1}, G_o(z) = \frac{z(2z+0.25)}{z^2 + z + 0.25}$$

b) Assume

$$H(z) = 1, G_c(z) = \frac{b_0^c z + b_1^c}{z-1}, G_p(z) = \frac{b_0^p}{z + a_1^P}$$

If the plant is equal to $\frac{2}{z-0.5}$ and the closed loop poles are at -0.5 and -0.333, show that the controller and the closed loop transfer function are

$$G_c(z)=\frac{1.167z-0.167}{z-1}, G_o(z)=\frac{2.333z-0.333}{z^2+0.833z+0.167}$$

3) The file **zoh_files.slx**and **zoh_driver.m** illustrate how to model discrete-times systems from continuous-time systems in both Matlab and Simulink.

- The first system in **zoh_files.slx** illustrates how to utilize a sample and hold and a zero order hold to allow modelling a continuous-time system as a discrete-time system.
- The second system is the equivalent discrete-time constructed in Matlab. If you run the file **zoh_driver.m** you should see the results of the two simulations lie on top of each other.
- The third system is a continuous-time state variable model of an open loop plant.
- The fourth system is the equivalent discrete-time state variable system with state variable feedback modelled in Matlab.

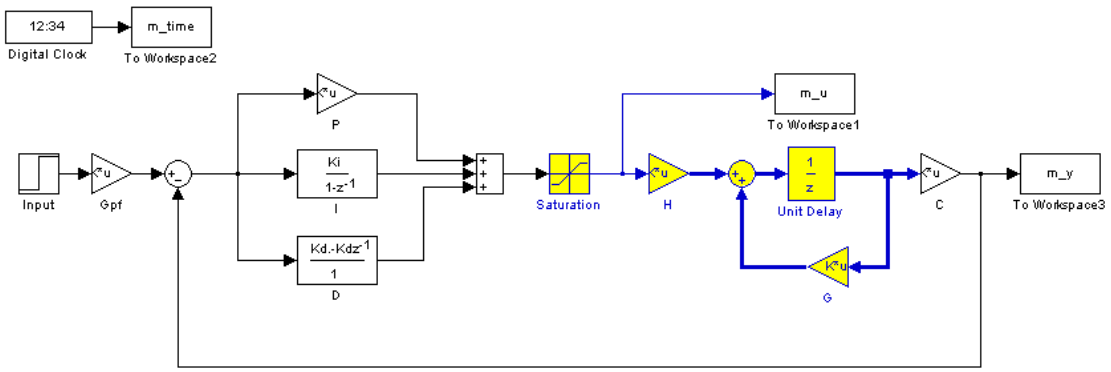
For this problem, you need to modify the third system using sample and hold(s) and zero order hold(s) so you can model the system as a discrete-time system and include state variable feedback. For this problem turn in your plot of the outputs (the Matlab and Simulink should be the same) and print your **zoh_files.slx** file (so I can see it). This problem should be very simple, but I want you to have to see how to do some of these things in Matlab and Simulink.

Note that we can write the discrete-time state equations as

$$x(n+1)=Gx(n)+Hu(n)$$
$$y(n)=Cx(n)+Du(n)$$

For this system D = 0. One way to implement state variable feedback is to let $u(n)=-Kx(n)$

4) (sisotool) 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 be 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 $C(z) = \dfrac{K(z^2 + az)}{z(z-1)} = \dfrac{K(z + a)}{(z - 1)}$

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

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.

Note that these calculations are pretty much done for you in the driver file ***DT_PID_driver.m***

The file DT_PID_driver implements a controller for the default plant $G_p(z) = \dfrac{9.348z + 8.879}{z^3 + 1.495z^2 + 0.8849z}$

assuming a sampling interval of $T_s = 0.1$ *seconds.*

You may want to read the review of sisotool at the end of this homework before going on.

For the remainder of this problem, you are to design two controllers for the plant $G_p(z) = \dfrac{12z + 5}{z^3 + 1.9z^2 + 0.5z}$ **assuming a sampling interval of 0.1 seconds.**

A) *Modify the default plant in the Matlab program* ***DT_PID_driver.m*** *and run the program. This program is set up to implement a P controller for the default plant with gain 0.0116. It will put the value of the transfer function for the system,* $G_p(z)$*, in your workspace.*

B) *Start sisotool and load in the transfer function. It is easiest to get the parameters you need if you use the pole-zero form of the controller. To do this type*

Edit → SISO Tool Preferences → Options *and click on* ***zero/pole/gain****.*

C) *Use sisotool to determine a PI controller so the system has a settling time less than 2.5 seconds and a percent overshoot less than 15%. 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 plots and the step response using* ***DT_PID_driver.m*** *and print out the graph. Include the values of* $k_p, k_i,$ *and* k_d *with your graph.*

D) *Use sisotool to determine a PID controller (I would suggest real zeros, but it is up to you!) so the system has a settling time less than 2.5 seconds and a percent overshoot less than 15%. The control effort must also be within the allowed bounds. Print out the root locus plots and the step response using* ***DT_PID_driver.m*** *and print out the graph. Include the values of* $k_p, k_i,$ *and* k_d *with your graph.*

You should have four graphs for this problem, two root locus plots and two step response plots. *The values of* $k_p, k_i,$ *and* k_d *should be included with each graph.*

Sisotool (Brief) Example

*Run the Matlab program **DT_PID_driver.m**. This program is set up to 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. Now we are ready for sisotool.*

Getting Started

- 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 (set the **Plot Type** to **Root Locus** for **Plot 1**, and set the **Plot Type** to **None** for all other Plots)

Loading the Transfer Function

- In the **SISO Design** window, Click on **file → import.**
- We will usually be assigning *Gp(z)* to block *G* (the plant). Under the **System** heading, click on the line that indicates **G**, then click on **Browse**.
- Choose the available Model that you want assigned to G (Click on the appropriate line) and then click on **Import**, and then on **Close**.
- Click **OK** on the System Data (Import Model) window
- 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** (the system is unstable at this point)
- 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 mouse 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 if you hold down the left mouse button. However, if you need all of the closed loop poles you have to look at all of the branches.

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 and zero around until the zero is approximately -0.295 and the gain is approximately 0.0563. You should get a figure like that shown in Figure 1.

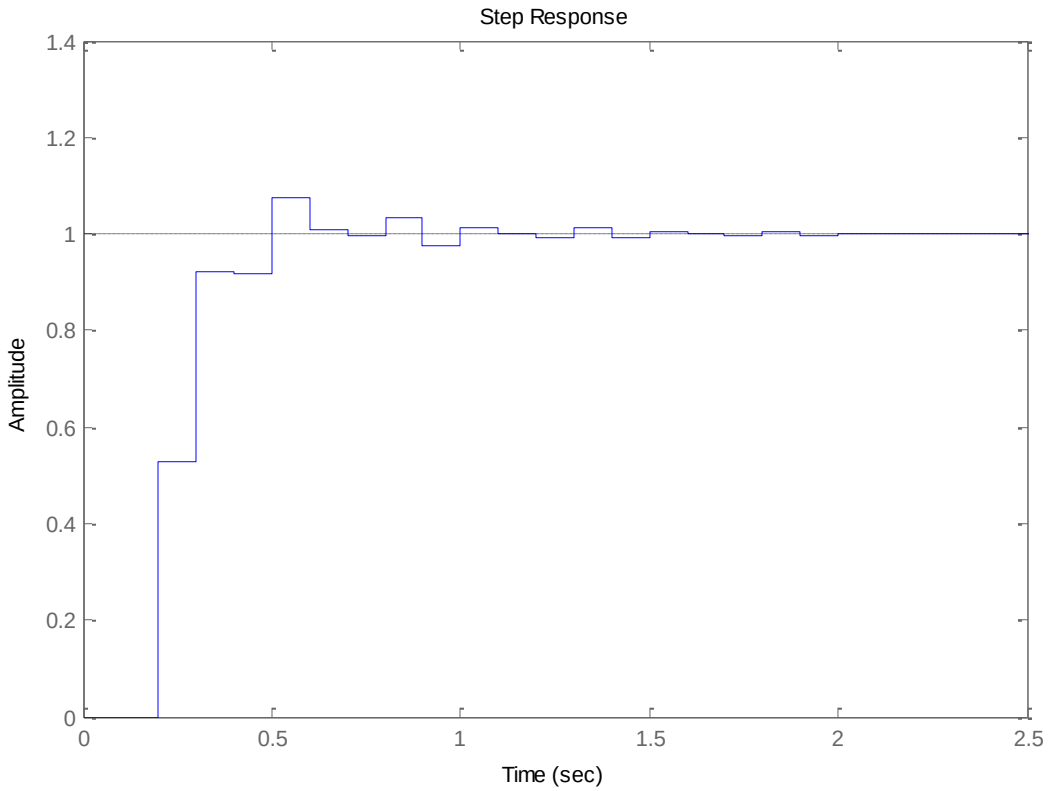


Figure 1. Discrete-time example with w PI controller.

Adding Constraints

- Right Click on the Root Locus plot, and choose **Design Requirements** 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.
- *Remember these constraints are only exact for ideal second order systems!!!!!*

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**.