ECE-320: Linear Control Systems
Homework 8

Due: **Wednesday** February 5 at 5 PM

Exam 2, Thursday February 6

1) For the z-transform

$$X(z) = \frac{3}{z-2}$$

a) Show that, by multiplying and dividing by z and then using partial fractions, the corresponding discrete-time sequence is

$$x(n) = -\frac{3}{2}\delta(n) + \frac{3}{2}2^k u(n)$$

b) By starting with the z-transform

$$G(z) = \frac{3z}{z-2}$$

where $X(z) = z^{-1}G(z)$, determine $g(n)$ and use the delay property to show that

$$x(n) = 3 \times 2^{n-1}u(n-1)$$

2) For impulse response $h(n) = \left(\frac{1}{3}\right)^{n-2} u(n-1)$ and input $x(n) = \left(\frac{1}{2}\right)^{n} u(n-1)$, use z-transforms of the input and impulse response to show the output is

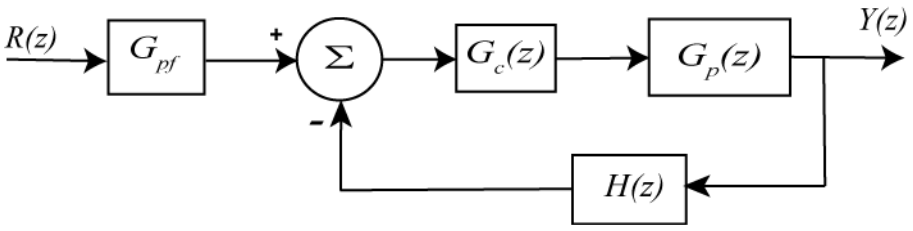
$$y(n) = 9\left[\left(\frac{1}{2}\right)^{n-1} - \left(\frac{1}{3}\right)^{n-1}\right]u(n-2) = 9\left[\left(\frac{1}{2}\right)^{n-1} - \left(\frac{1}{3}\right)^{n-1}\right]u(n-1)$$

Hint: Assume $Y(z) = z^{-1}G(z)$, determine $g(n)$ and then $y(n)$

3) For impulse response $h(n) = \left(\frac{1}{2}\right)^{n-3} u(n-1)$ and input $x(n) = \left(\frac{1}{4}\right)^{n+1} u(n-2)$, use z-transforms of the input and impulse response to show the system output is $y(n) = \left[\left(\frac{1}{2}\right)^{n} - \left(\frac{1}{4}\right)^{n-1}\right]u(n-3)$

Hint: Assume $Y(z) = z^{-2}G(z)$,

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

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

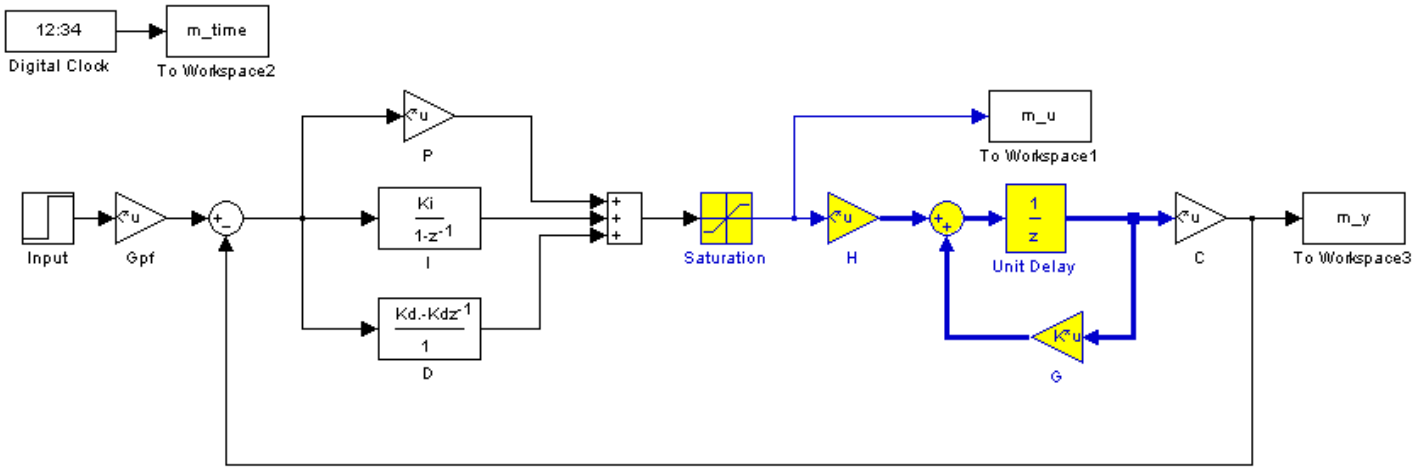
c) $G_p(z)=\dfrac{1}{z^2+0.4z+0.04}, \quad G_c(z)=\dfrac{0.2}{z+0.2}, \quad H(z)=\dfrac{1}{z+0.2}$

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

The remaining problem is the Prelab for Lab 6

5) Prelab (turn in as part of the homework!): In 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. We will use Matlab's *sisotool* to design discrete-time controllers for your systems in this prelab.

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***

A) *Run the Matlab program* ***DT_PID_driver.m****. This program is set up to read the data file bobs_1dof_210.mat, which is a continuous time state variable model for a one degree of freedom rectilinear 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.*

B) *Modify the program* ***DT_PID_driver.m*** *to read in the model for your system. Be sure Ts, the sampling interval, is set to 0.1 seconds. Run* ***DT_PID_dirver.m*** *so the model of your plant is in the Matlab workspace. Note: You and your partner should use different systems!*

C) *Start sisotool and load in your model of 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****.*

D) *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 plots and the step response using* ***DT_PID_driver.m*** *and print out the graph.*

E) *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 plots and the step response using* ***DT_PID_driver.m*** *and print out the graph*

You should have four graphs for this prelab, two root locus plots and two step response plots.