ECE-520: Discrete-Time Control Systems
Homework 3

Due: Tuesday December18 in class

1) Prove or disprove the following claims: if $u, v,$ and w are linearly independent vectors, then so are

a) $u, u+v, u+v+w$
b) $u+2v-w, u-2v-w, 4v$
c) $u-v, v-w, w-u$
d) $-u+v+w, u-v+w, -u+v-w$

Note: You must do this for arbitrary vectors. ***Do Not*** assume $u, v,$ and w are specific vectors.

2) For the following matrix

$$A = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 1 & 1 & 0 & 2 \end{bmatrix}$$

a) Find a set of vectors that form a basis for the null space of A.

b) Is the vector $\underline{n} = \begin{bmatrix} 2 & 2 & -2 & 2 \end{bmatrix}^T$ in the null space of A? That is, can you represent this vector as a linear combination of your basis vectors?

c) Is the vector $\underline{a} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$ in the range (column) space of A?

3) For the following matrix

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 2 & 2 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

a) Find a set of vectors that form a basis for the null space of A.

b) Is the vector $\underline{n} = \begin{bmatrix} 2 & 6 & -2 & -1 \end{bmatrix}^T$ in the null space of A? That is, can you represent this vector as a linear combination of your basis vectors?

c) Is the vector $\underline{a} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$ in the range (column) space of A?

4) For the following matrix

$$A = \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 1 \\ 2 & 1 & 3 & 5 \end{bmatrix}$$

a) Find the rank of A (the number of linearly independent rows or columns).

b) Determine two vectors that span the null space of A.

c) Determine two vectors that span the row space of A.

d) Show that any vector in the row space of A is orthogonal to any vector in the null space of A.

e) Determine two vectors that span the column space of A.

5) Suppose we want to minimize a function while satisfying a constraint. For example, find the point in the plane $x + y = 5$ closest to the origin. We want to write this as a minimization problem with a constraint, such as

$$\begin{aligned} &\text{minimize} \quad x^2 + y^2 \qquad \text{(distance from origin)} \\ &\text{subject to} \quad \text{x+y-5 = 0} \quad \text{(constraint)} \end{aligned}$$

We do this with Lagrange multipliers (λ) and form the minimization problem

$$\text{minimize } L(x, y, \lambda) = x^2 + y^2 + \lambda(x + y - 5)$$

To solve the problem we now set $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} = \frac{\partial L}{\partial \lambda} = 0$. Show that the optimal point is $x = y = \frac{5}{2}$.

6) Consider the discrete-time state variable system

$$\underline{x}(k+1) = G\underline{x}(k) + Hu(k)$$

with initial state $\underline{x}(0) = \underline{0}$.

a) Show that after three steps $(k = 0,1,2)$ we have the system of equations

$$\underline{x}(3) = \begin{bmatrix} G^2H & GH & H \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \end{bmatrix}$$

b) Assume we want to go from the origin to the final state $\underline{x}_f$ in three time steps with a penalty on the amount of input (i.e., the signal energy) We can formulate this problem as

$$\begin{aligned} &\text{minimize} \quad \underline{u}^T R\underline{u} \qquad \text{(minimize engergy)} \\ &\text{subject to} \quad \underline{x}_f - Q\underline{u} \quad \text{(constraint: must reach final state)} \end{aligned}$$

R is a *symmetric* weighting matrix, indicating how much to penalize the input energy at each time step. What are Q and $\underline{u}$?

c) We can again solve this problem using Lagrange multipliers. The form of the Lagrange multiplier is chosen so the L function makes mathematical sense. For example, for this problem, the function to be minimized is a scalar, so the Lagrange multiplier must be chosen to make the problem a scalar problem. Specifically, we form

$$L(\underline{u}, \underline{\lambda}) = \underline{u}^T R\underline{u} + \underline{\lambda}^T (Q\underline{u} - \underline{x}_f)$$

Assuming R^{-1} exists but Q^{-1} does not exist, show that the optimal control signal is

$$\underline{u} = R^{-1}Q^T (QR^{-1}Q^T)^{-1}\underline{x}_f$$

d) How would your answer change if $\underline{x}(0) \neq \underline{0}$?

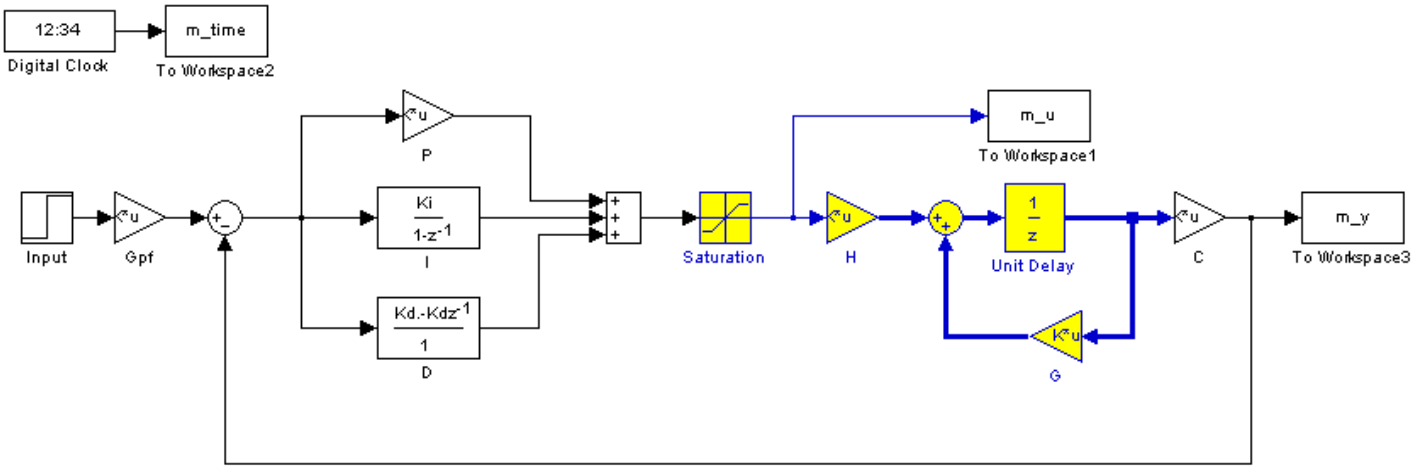
e) For $G = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}$ and $H = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, determine the matrix Q and then determine vector(s) that span the null space of Q

f) For $\underline{x}_f = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$, find a control vector $\underline{u}$ that takes the system from the origin to $\underline{x}_f$ that minimizes the energy (is a minimum norm solution). Assume and $R = I$.

g) Show that your control vector $\underline{u}$ to part **e** is orthogonal to the vector(s) that span the null space of Q, hence $\underline{u}$ has no components in the null space of Q.

7) 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 6.5.1 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 (Breif) 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.*

- Type **sisotool** in the command window
- Click **close** when the help window comes up
- Click on **view → open loop bode** to turn off the bode plot. (Whatever is checked will be shown, we only want to see the root locus.)

B) *Loading the Transfer Function*

- Click on **file → import**
- A window on the left will show you the transfer functions in your workspace, while the window in the right will let you choose the control system configuration.
- 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.
- 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.

C) *Odds and Ends :*

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

- Click **Edit → Root Locus → Properties**
- Click on **Limits**
- Set the limits

You may also want to put on a grid, as another method of checking your answers. Type **Edit → Root Locus → 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**.

You need to 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****.*

D) *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 and see how the step response changes.

- 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.

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

- Type **Compensators → Edit → C**
- Click on **Add Real Zero** or **Add Real Pole** 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.
- Click **OK** to exit 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 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.

F) *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.**

G) *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.

H) 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

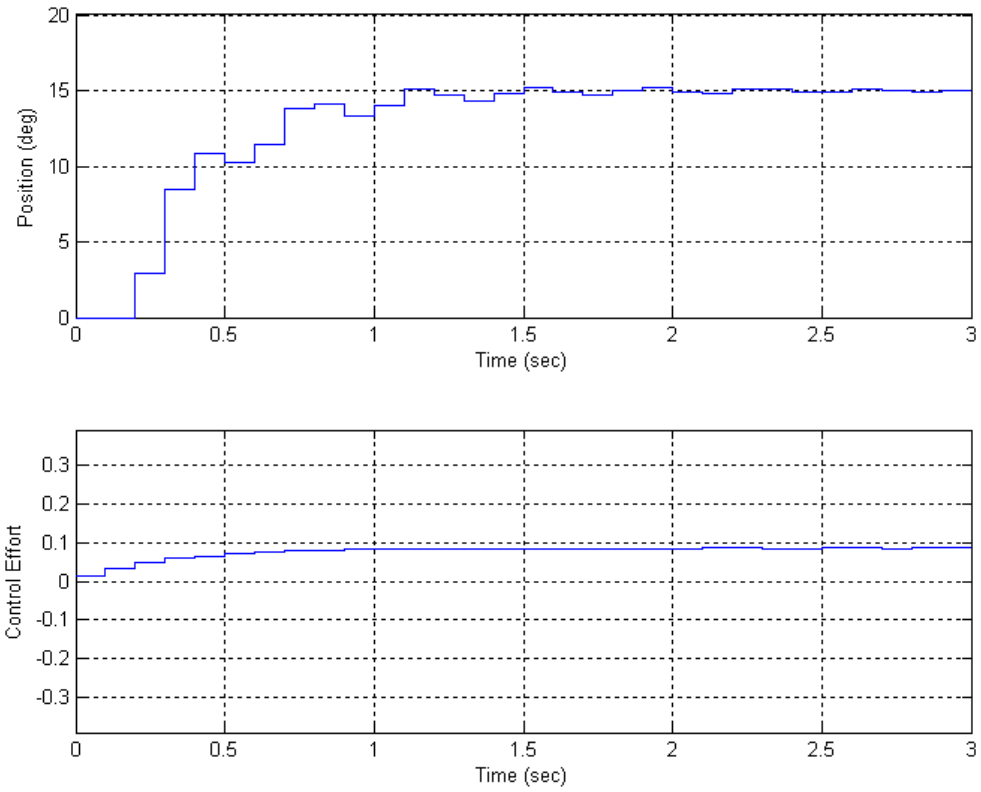


Figure 1: Matlab/Simulink results for PI controller.