Lab 4: SFWR ENG 3DX4

Introduction to PD Compensator Design and Electromechanical Control

First week of lab: March 11th, 2024
Prelab Due Start of Lab Period, week of: March 18th, 2024
Demo Due End of Lab Period, week of: March 18th, 2024

Announcements

In this lab, you will be evaluated on prelab questions and demonstration of working laboratory exercises. Evaluation of lab exercises will be performed on a checkpoint system, with checkpoints indicated in your laboratory instructions. In order to receive full marks for a lab, you must show a TA your work/output at each checkpoint, and your work must be correct and complete. In addition, lab work must be completed in the allotted laboratory time. Prelab questions are to be completed *individually*. This lab consists of four checkpoints

Goals

- Introduction to computer based control of a voltage-controlled electromechanical system - a DC servo motor with ball and beam apparatus.
- Using an interactive system for a given plant model, learn how to design an ideal derivative compensator to satisfy specific requirements.
- Learn the difference between ideal PD compensator design and practical PD compensator design.
- Take empirical measurements of a physical system with student-designed controllers.

Pre-lab Background Reading

Before you go for your lab session, please read the following documents at your convenience, in addition to the class notes:

- Ball and Beam User Manual
  (download the `BallBeamUserManual.pdf` from the course webpage if the link does not work)
- Ball and Beam Student Workbook
  (download the `BallBeamStudentWorkbook.pdf` from the course webpage if the link does not work)

Equipment Arrangement

Our laboratory equipment for Lab 4 consists of the Quanser SRV02 rotary servo plant, with attached Quanser BB01 Ball and Beam plant. The Quanser SRV02 rotary servo plant consists of a DC motor, a gear box, a potentiometer, and a quadrature encoder. Some models of SRV02 also have a tachometer. The Quanser BB01 Ball and Beam plant consists of a steel ball, two rails along which the ball freely rolls and a sensor apparatus (collectively the *rail assembly*). The ball and beam has two posts. One is stationary, and the other is connected to the centre gear of the SRV02 rotary servo. This provides the ball and beam plant with one rotational degree of freedom. When the angular position of the servo changes, it causes a corresponding change in the angle of inclination of the rail. To measure the position of the ball, the two rails of the ball and beam act as a potentiometer, with the ball as the wiper. That is, the ball and one rail are low resistance, and the other rail is a resistor, where the resistance is distributed evenly along the length of the rail. We can therefore measure the position of the ball accurately, by measuring the voltage across this apparatus.

Unfortunately, this means that the accuracy of the ball and beam's position measurement is dependent on the electrical conductivity of the points of contact between the ball and both rails. Dirt, debris, oxidation, dust, or the fingerprint oil left by touching the ball or beams reduce the conductivity of the points of contact, and throw off our measurements. Therefore care must be exercised not to contaminate the laboratory equipment.

Mathematical Background

The ball and beam setup may be modelled mathematically using transfer functions, but it is also composed of two distinct systems, the BB01 ball and rail plant, and the SRV02 servo motor plant. Since these two systems are in series (or in cascade form), combining their transfer functions ($P_{bb}(s)$ and $P_s(s)$ respectively) is a simple multiplication:

$$P(s) = P_{bb}(s)P_s(s) \tag{1}$$

As we have seen in previous lab exercises, the Transfer function for the SRV02 rotary servo plant is given as follows:

$$P_s(s) = \frac{K}{s(\tau s + 1)} \tag{2}$$

where

$$K = 1.53\, rad/(V \cdot s) \tag{3}$$

and

$$\tau = 0.0248s \tag{4}$$

The transfer function for the Ball and Beam assembly without the SRV02 plant is derived from first principals in the Ball and Beam Student Workbook, and the result of that derivation is as follows:

$$P_{bb}(s) = \frac{K_{bb}}{s^2} \tag{5}$$

where

$$K_{bb} = 0.419 \frac{m}{s^2 rad} \tag{6}$$

The control system for the Ball and Beam and the SRV02 plant consists of two parts: an inner loop to control the SRV02's gear angle, and an outer loop to control the ball's position. (Figure 1)

For the system controller, we will assume that the servo controller is perfect (even though it isn't), and tune our PD compensator to compensate only for the Ball and Beam component of the plant. In other words, the transfer function we will be using in Part 1 of this lab is:

$$P_{bb}(s) = \frac{0.419}{s^2} \tag{7}$$

We will use MATLAB's rltool, an interactive root locus design tool to design our PD compensator.

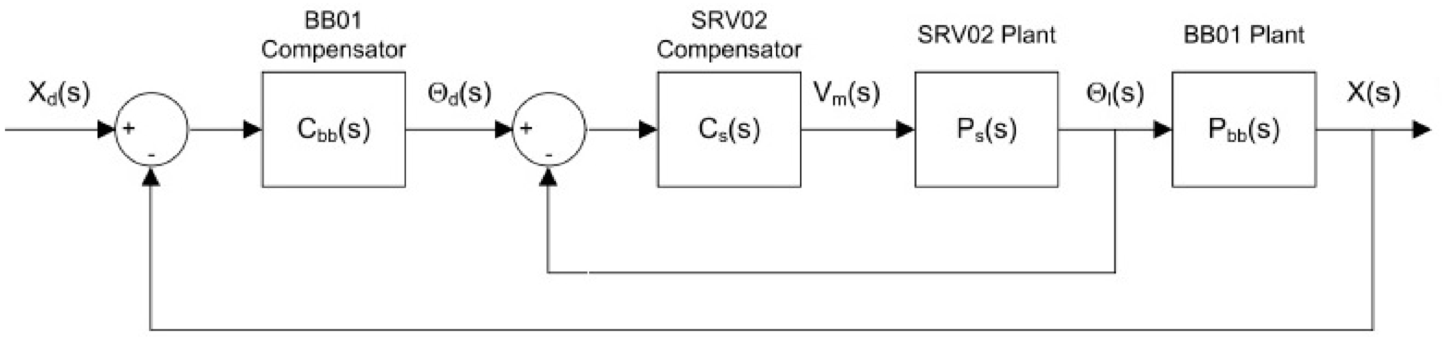

Figure 1: The cascade control system used to control the ball's position in the combined SRV02 and BB01 plant (Figure courtesy of Quanser's student workbook)

Transient response characteristics

In order to use the ω_n and ζ parameters later in the lab, you must calculate them from the settling time T_s and percent overshoot $\%OS$. Please use the following formulae to do so:

$$\zeta = \frac{-ln(\frac{\%OS}{100})}{\sqrt{\pi^2 + ln^2(\frac{\%OS}{100})}} \tag{8}$$

$$T_s = \frac{4}{\zeta\omega_n} \tag{9}$$

Practical PD Compensator Design

With reference to Figure 1 above, we will use a PD controller for $C_{bb}(s)$, and a proportional controller for $C_s(s)$.

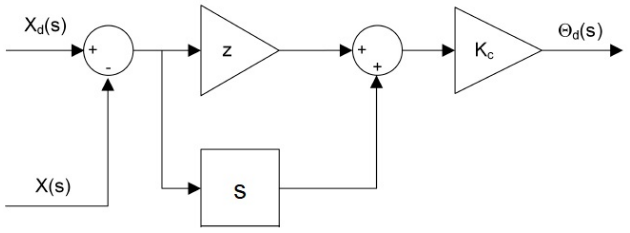

Figure 2: A traditional PD controller

For a standard PD controller, as shown in Figure 2, the derivative term operates on our measurements of the process variable (in this case, the ball's position). With the physical apparatus, the inconsistent presence of dirt on the electrical contacts, as well as vibration, misalignment, and many other factors, result in noise within the measurement signal. Derivatives tend to amplify noise, since the derivative is the rate of change of the signal, and noise causes rapid oscillations around the true value. Needless to say, this is undesirable in a control system, as such effects can push control systems into unstable responses.

To correct this effect, the natural solution would be to use a low-pass filter to filter out the high-frequency noise in the signal. However, this can also be problematic, because of the latency that will necessarily be introduced.

Instead, we will replace the derivative term with a high pass filter. High pass filters and differentiators behave very similarly, but we can set a high-pass filter up so that high frequencies will not be amplified.

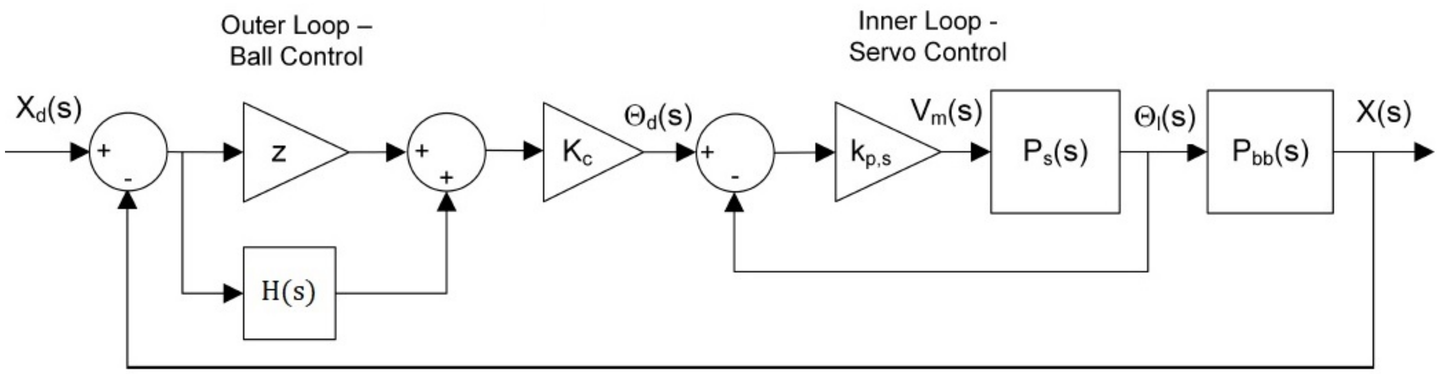

Figure 3: The control system for the lab

The first order filter that is used in the controller to replace the traditional derivative term is:

$$H(s) = \frac{\omega_f s}{s + \omega_f} \tag{10}$$

For adequate filtering of the noise in our lab set up, the cutoff frequency, ω_f, will be set to 5 Hz. That is

$$\omega_f = 31.4 rad/s \tag{11}$$

From Figure 3, we know:

$$\theta_d(s) = K_c(z + \frac{\omega_f s}{s + \omega_f})\Big(X_d(s) - X(s)\Big) \tag{12}$$

This follows as we assume the inner loop servo controller is perfect, i.e. $\theta_d(s) = \theta_l(s)$

Substitute Equation (12) into Equation (5) and solve for $X(s)/X_d(s)$:

$$\frac{X(s)}{X_d(s)} = \frac{K_{bb}K_c(z + \omega_f)s + K_{bb}K_c z\omega_f}{s^3 + \omega_f s^2 + K_{bb}K_c(z + \omega_f)s + K_{bb}K_c z\omega_f} \tag{13}$$

A standard third-order characteristic equation can be written as:

$$(s^2 + 2\zeta\omega_n s + \omega_n^2)(1 + T_{pole}s), \tag{14}$$

where T_{pole} is the pole decay in seconds.

Expanding equation (14), the standard third-order characteristic equation becomes:

$$s^3 + \frac{2\zeta\omega_n T_{pole} + 1}{T_{pole}}s^2 + \frac{\omega_n^2 T_{pole} + 2\zeta\omega_n}{T_{pole}}s + \frac{\omega_n^2}{T_{pole}} \tag{15}$$

From equation (13), we know that the characteristic equation of the control system with the practical PD controller is:

$$s^3 + \omega_f s^2 + K_{bb}K_c(z+\omega_f)s + K_{bb}K_c z\omega_f \tag{16}$$

By equating the coefficients of the s^2 terms in Equation (15) and Equation (16), we can get:

$$\omega_f = \frac{2\zeta\omega_n T_{pole} + 1}{T_{pole}} \tag{17}$$

Therefore the pole location T_{pole} is:

$$T_{pole} = \frac{1}{\omega_f - 2\zeta\omega_n} \tag{18}$$

where the cut-off frequency ω_f is 31.4 rad/s. The damping ratio ζ and the natural frequency ω_n are determined by the desired system response specifications.

In order to find the zero location z and the compensator gain K_c for the Practical PD controller, two equations can be obtained by equating the s^1 and s^0 coefficients of the prototype characteristic equation in Equation (15) and the BB01 characteristic equation in Equation (16):

$$K_{bb}K_c z\omega_f = \frac{\omega_n^2}{T_{pole}} \tag{19}$$

and

$$K_{bb}K_c\omega_f + K_{bb}K_c z = \frac{\omega_n^2 T_{pole} + 2\zeta\omega_n}{T_{pole}} \tag{20}$$

Solving for K_c, the compensator gain:

$$K_c = \frac{\omega_n^2}{T_{pole}K_{bb}z\omega_f} \tag{21}$$

Solving for z, the zero required to meet the specifications.

$$z = \frac{\omega_n\omega_f}{-\omega_n + \omega_f\omega_n T_{pole} + 2\omega_f\zeta} \tag{22}$$

where ω_n and ζ may be calculated with given $\%OS$ and T_s using Equation (8) and Equation (9), and

$$\omega_f = 31.4\frac{rad}{s} \tag{23}$$

$$T_s = 3.0s \tag{24}$$

$$PO \leq 5.0\% \tag{25}$$

$$K_{bb} = 0.419\frac{m}{s^2 rad} \tag{26}$$

In-Lab Activities

Design an ideal PD Compensator using MATLAB

In this section, we will be designing a PD compensator using the root locus method in MATLAB.

1. Launch MATLAB.
2. We will begin by plotting the root locus of the system model (Equation (7) in the Mathematical Background section above).
    - Create a continuous-time transfer function object using the MATLAB function `tf()`. For information on its usage, consult the MATLAB documentation. Check the displayed transfer function to ensure the function was called correctly.
    - Plot the function using `rlocus()` (for usage, consult the MATLAB documentation).
        - The lines you see indicate the paths the poles travel through as a gain K is applied to the transfer function.
    - We will take the following as design parameters for our system. Using these, as well as equations (8) and (9), calculate theoretical values of ζ and ω_n for our compensated system.
        - $\%OS = 5.0\%$
        - $T_s = 3s$
    - Using the `sgrid()` function, plot ζ and ω_n on your root locus plot. Resize the viewing area using the `axis()` function, so that the viewing range is from -3 to 3 for both the x and y axes. This gives some idea of what we're aiming for with our compensator design.
    - It is the purpose of PD compensator design to add a zero to the root locus graph such that the root locus passes through the intersection of ζ and ω_n. We then find the K corresponding to this pole location.
    - Start the interactive Control System Designer with the `rltool()` function. Again, for usage, take a look at the MATLAB documentation. You will see the Control System Designer window as in Figure 4.
    - It will be convenient to have both plots (both tabs in the Control System Designer) displayed side-by-side, which can be accomplished by pulling them to either side of the display pane.
    - Click and drag the pink squares on the Root Locus Editor plot, and see how the step response is affected. The pink square represents the real pole position for some gain K. Notice also that, since our poles are exactly along the y axis, our system is marginally stable.
    - We can add design guides to the Root Locus Editor graph to make the process of selecting a zero position more systematic. Right click the white area on the root locus editor graph and select **Design Requirements → New...**
        - From the drop-down menu, select damping ratio, and enter your value of ζ.
        - Repeat the process for your calculated value of natural frequency ω_n.

      Now your Control System Designer window should look something like Figure 5.
    - We can also add some features to the step response graph to make reading our values somewhat easier. Right click on the step response graph and select...
        - **Characteristics → Peak Response**

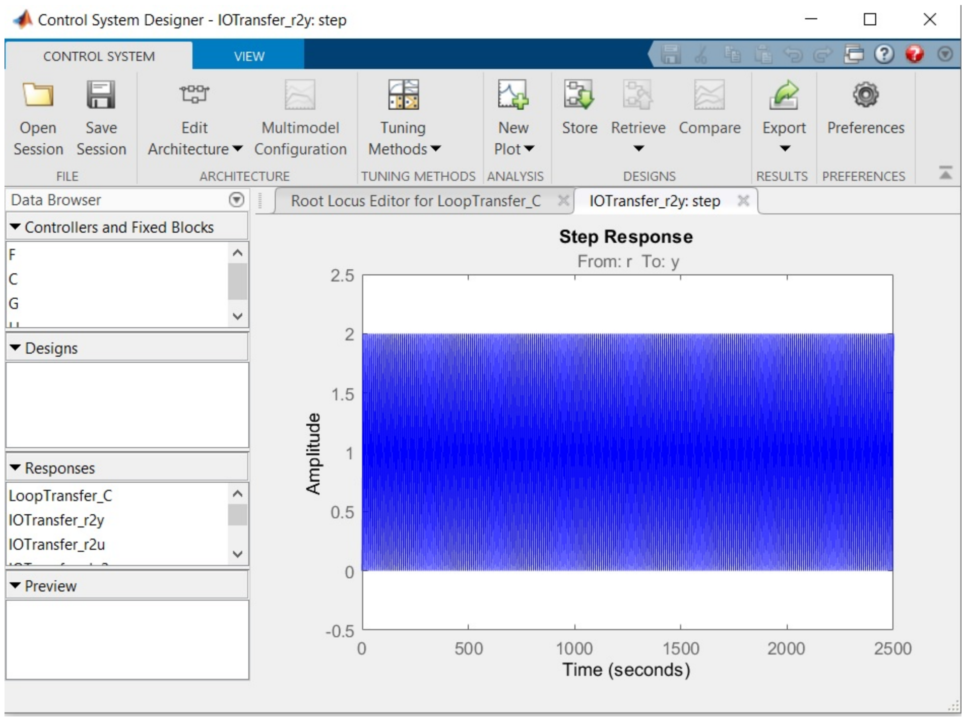

Figure 4: Control System Designer

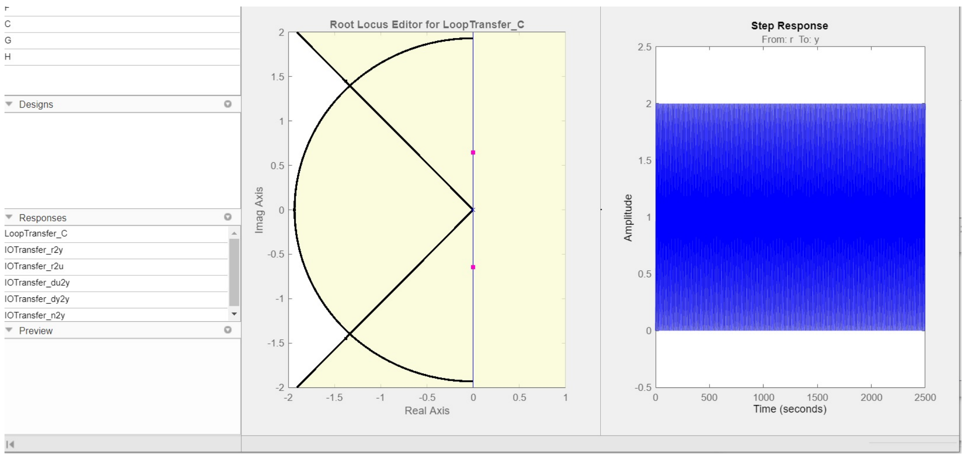

Figure 5: Control System Designer After Adding Design Requirements of ζ and ω_n

- **Characteristics $\rightarrow$ Settling Time**

- Hovering over the nodes added to the graph will give you a reading of other step response characteristics.
- Next, we need to add a zero. Right click the white area on the root locus editor graph, and select **Add Pole or Zero $\rightarrow$ Real Zero**. Add the zero somewhere to the left of the origin (*it seems that you can only add to the white area with MATLAB 2022b*). Your plot will look something like Figure 6.

3. Configure both the position of the new zero and the gain (represented by the pink rectangles) until the step response conforms to our design specifications.

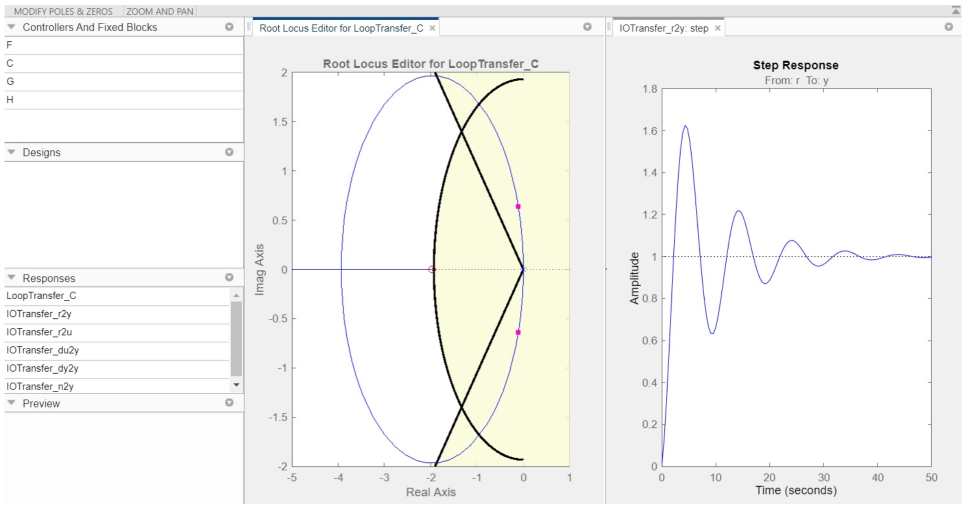

Figure 6: Root Locus Editor Plot After Adding a Real Zero

4. When you have it the way you want it, double click on "C" in the **Controllers and Fixed Blocks** box.

5. Record the equation and show it to your TA for **CHECKPOINT 1**.

Design a Practical PD Controller

1. Make a table in your notebook similar to the one below:

$\%OS$	20%	15%	10%	5%
T_s	4 s	3 s	2 s	1 s
ζ				
ω_n				
z				
K_c				

2. Using the information in the Mathematical Background section (in particular, equations (21) and (22)), fill in the table with the values you calculate. This process can be made simpler and easier by using MATLAB or a spreadsheet. The z and K_c values here will be used to implement practical PD compensators in the Simulink model for the Ball and Beam control.

3. Show your values to your TA for **CHECKPOINT 2**

Creating the Controller

1. Begin by downloading "ballbeam_start.slx" from Avenue. This model already has in place the Quarc components necessary for communicating with the virtual model.

2. Within your model workspace, create variables for K_c, z and ω_f. Initialize ω_f with the value found in the mathematical background section of this document.

3. In order to simulate the voltage restrictions of the physical apparatus, add a saturation block to the diagram, wire it to the **Vm(V))** port of the "Ball&Beam Core" subsystem, and configure it for operation between ±9.5V. (**Please note: "wire it to the port of the 'Ball&Beam Core' subsystem"! In other words, keep the subsystem unchanged, add all of your blocks outside the subsystem!**).

4. Add a scope which displays this input voltage.

5. Add a gain block and connect its output to the input of the saturation block. Give it a gain of 12. This will be our proportional controller for the servo ($C_s(s)$ in Figure 1, or $K_{p,s}$ in Figure 3).

6. Next we will provide the difference signal to the gain block. Create a subtract block, and feed the **theta_l** signal of the Ball&Beam Core subsystem into the negative terminal. The **theta_l** signal from the Ball&Beam Core subsystem is the measured SRV02 gear angle position. The positive terminal of the subtraction block will receive the desired SRV02 gear angle, which is the signal incoming from the ball and beam controller.

7. In the physical system, the servo's rotation is restricted to the range between ±56 degrees (0.9769 Rad). Therefore a saturation block is needed before the desired SRV02 gear angle is applied. Add a saturation block and feed it into the positive terminal of the subtraction block. To better protect the equipment, you can set the limits of this saturation block to ±0.942, which is ±54 degrees, which is slightly smaller than ±56 degrees.

8. Next, feed the "applied desired" and the "measured" servo gear angle positions to a scope.

9. Create a gain block, assign the variable you created for K_c to it, and connect its output to the saturation block for the SRV02 gear angle position.

10. Create a transfer function, and configure it such that it represents the following equation, which acts as both derivative and high pass filter, as discussed above.

$$G_D(s) = \frac{\omega_f \cdot s}{s + \omega_f}$$

11. Create a gain block and assign the variable z to it.

12. Create an addition block, and run the preceding two block outputs into its inputs. Connect the adder's output to the K_c gain.

13. Now we need to calculate the difference between the desired ball position and the actual measured ball position. Take the position output of the "Ball&Beam Core" subsystem (**X(cm)**), and feed it into the negative terminal of a newly created subtract block.

14. Feed the output of the subtract block into the input of the z gain and the transfer function.

15. Create a pulse generator and assign it an amplitude of 10, a period of $40s$, and a pulse width of 50%. Before connecting it as input to the system, subtract a constant 5 from the signal, so that it oscillates from positive to negative $5cm$. This is our desired ball position. Connect it to the positive terminal of the last subtract block. Also connect this signal to a scope displaying the actual ball position, so that they may be compared.

16. Name the signals fed to the scopes and configure the scopes to display legends.

17. Check your model with your TA to complete **CHECKPOINT 3**

Test your Controller on the Physical System

In this part of the lab you will use the controller parameters obtained in the previous part to calibrate a practical PD compensator controlling the Ball and Beam plant.

PLEASE READ THE FOLLOWING IMPORTANT NOTICES:

1. ***When starting up models to control the ball & beam plant, for calibration purposes, the initial position of the ball and beam assembly is with the servo as far clockwise as it can turn. This is the assembly's natural rest point, and will often acquire this position naturally when not powered.***
2. ***Do not force the motor if the motor is powered.***
3. ***Do not touch the motor gear if the motor is powered! This is a safety concern. Please note that the motor is only powered while the Simulink model is running.***
4. ***Please note that the ball and beam assembly is not fastened to the base plate, but resting in holes. It is unlikely that the assembly will come out of the holes through normal operation, but if this does occur, please do not panic. Stop the model, or turn off the MyRIO, or turn off the power to the power module to SRV02, and then reposition the components in their respective holes in the base plate.***
5. ***The electrical conductivity between the ball and the beam is crucial for measuring and controlling the ball's position. If the Ball and Beam set is not assembled properly, or if there is dust or oil film, which may result from touching the ball and beam, the conductivity may be lost in certain areas. This will make the ball's position reading not continuous when the ball is rolling on the beam, and this will make the ball's position control difficult or impossible. To ensure good electrical conductivity between the ball and beam, please clean the ball and beam using alcohol and the low-lint wipes provided in the lab. To test the conductivity between the ball and beam, you can log into your MyRIO using an SSH client (Bitvise on the lab computers. Hostname: your MyRIO IP address. Username: admin. Password: 4aa4) and run the program "testBallBeam". If there is no ball position reading within certain 0.5cm ranges on the beam, the testBallBeam program considers the ball's position reading is not continuous. Then you need to clean the ball and the beam, or ask a TA for help.***

1. Set up your physical Ball & Beam system. Power on the MyRIO and the power module for SRV02 and Ball & Beam.
2. Perform trials for the z and K_c values you found above using a table similar to the following:

$\%OS$	20%	15%	10%	5%
T_s	4 s	3 s	2 s	1 s
z				
K_c				
Empirical $\%OS$				
Empirical T_s				
Empirical Steady State Error				

3. The z and K_c values can be entered from the model workspace dialogue. Aren't variables handy?
4. When running the model, allow it to go through a few cycles, so that the step response becomes consistent.
5. Once the system's response is consistent, stop the model with two steps in view, one for the +5 step and one for the -5 step.
6. Use the pan and zoom tools of the scope to accurately read the settling time, $\%OS$ and steady-state error of multiple transient responses (any even number of responses will do, as long as there are the same number of upcycles and downcycles). Average the values and tabulate them.
7. If the ball does not achieve steady state before the next cycle begins, decrease the frequency of the square wave function.
8. Take the average of the up cycle and the down cycle step responses.
9. Save your model for use in Lab 5. Show your tabulated results to your TA to complete **CHECK-POINT 4**, along with answers to the following questions.
   - How well did the designed controllers work?
   - Did they meet their design parameters?
   - Do the $\%OS$ and T_s improve as you move right in the chart? What about steady-state error?
   - If any of your trials didn't conform to your design requirements, how would you explain it?