Rotary Motion Servo Plant: SRV02

Rotary Experiment #17: 2D Ball Balancer

2D Ball Balancer Control using QUARC

Instructor Manual

Table of Contents

1. Introduction

The objective of the 2D Ball Balancer experiment, i.e. 2DBB, is to stabilize the ball to a desired position on the balance plate. Using the proportional-integral-derivative (PID) family, a control system is designed to meet a set of specifications.

The following topics are covered in this laboratory:

- Model the dynamics of the ball from first-principles.
- Obtain a transfer function representation of the system.
- Design a proportional-velocity (PV) compensator to control the position of the servo load shaft according to certain time-domain requirements.
- Design a proportional-integral-derivative (PID) compensator that regulates the position of the ball on the beam and meets certain specifications. This together with the servo control is the complete Ball and Beam cascade control system.
- Control techniques such as set-point weight and integrator anti-windup are explored.
- Simulate the Ball and Beam control using the model of the plant and ensure the specifications are met without any actuator saturation.
- Implement the controllers on the Quanser 2DBB device and evaluate its performance.

Regarding Gray Boxes:

Gray boxes present in the **instructor manual** are not intended for the students as they provide solutions to the pre-lab assignments and contain typical experimental results from the laboratory procedure.

2. Prerequisites

In order to successfully carry out this laboratory, the user should be familiar with the following:

- Data acquisition card (e.g. Q2-USB), the power amplifier (e.g. VoltPAQ), and the main components of the SRV02 (e.g. actuator, sensors), as described in References [1], [4], and [5], respectively.
- Wiring and operating procedure of the SRV02 plant with the amplifier and DAQ device, as discussed in Reference [5].
- Laboratory described in Reference [6] in order to be familiar using QUARC with the SRV02.
- Designing a PV position control for the SRV02 as dictated in Reference [8].

3. Overview of Files

Table 1 below lists and describes the various files supplied with the SRV02 Ball and Beam Position Control laboratory.

File Name	*Description*
55 – 2D Ball Balancer User Manual.pdf	This manual describes the hardware of the 2D Ball Balancer and explains how to setup and wire the system for the experiments.
57 – 2D Ball Balancer Control – Student Manual.pdf	This laboratory guide contains pre-lab and in-lab exercises demonstrating how to design and implement a position controller on the Quanser 2D Ball Balancer plant using QUARC.
setup_srv02_exp17_2dbb.m	The main Matlab script that sets the SRV02 motor and sensor parameters, the SRV02 configuration-dependent model parameters, and the associated 2DBB parameters. **Run this file only to setup the laboratory.**
setup_srv02_configuration.m	Returns the configuration-based SRV02 model specifications *Rm*, *kt*, *km*, *Kg*, *eta_g*, *Beq*, *Jeq*, and *eta_m*, the sensor calibration constants K_POT, K_ENC, and K_TACH, and the amplifier limits VMAX_AMP and IMAX_AMP.
setup_2dbb_configuration.m	Returns the 2DBB model specifications *L_tbl, r_arm, r_b, m_b, J_b,* and *g,* and the min/max servo limits: THETA_MIN and THETA_MAX.
d_model_param.m	Calculates the SRV02 model parameters *K* and *tau* based on the device specifications *Rm*, *kt*, *km*, *Kg*, *eta_g*, *Beq*, *Jeq*, and *eta_m*.
calc_conversion_constants.m	Returns various conversions factors.
s_2dbb_pos.mdl	Simulink file that simulates the cascade ball position controller. Both the outer-loop ball position control and the inner-loop servo position control are used in this file.
q_2dbb_pos.mdl	Simulink file that implements a closed-loop cascade position controller on the actual 2DBB system using QUARC.
56 – 2D Ball Balancer Control – Instructor Manual.pdf	Same as the student version except the gray boxes are no longer shaded to reveal the solution to the pre-lab and in-lab exercises.
srv02_exp17_2dbb.mws	Maple worksheet used to develop the model and design the position position controller for the 2DBB experiment. Waterloo Maple 9, or a later release, is required to open, modify, and execute this file.

File Name	*Description*
srv02_exp17_2dbb.html	HTML presentation of the Maple Worksheet. It allows users to view the content of the Maple file without having Maple 9 installed. No modifications to the equations can be performed when in this format.
d_pv_design.m	Matlab script file that calculates the control gains *kp* and *kv* based on the SRV02 model parameters *K* and *tau* as well as the peak time an overshoot specifications *tp* and *PO*.
d_2dbb_model_param.m	Calculates the 2DBB model parameter *K_bb* based on the device specifications *r_arm, L_tbl, r_b, m_b,* and *J_b*.
d_2dbb_pid_design.m	Matlab script file that calculates the *PID* gains *kp_bb*, *ki_bb*, and *kd_bb*, based on the Ball Balancer model parameters *K_bb*, desired overshoot, settling time, and pole decay time specifications.
meas_2dbb_specs.m	Plots the response found in the variables *data_x, data_theta_l_x,* and *data_vm* in the Matlab workspace and measures the corresponding peak time, settling time, percentage overshoot, and steady-state error. Users can also use the saved responses contained in MAT files.
plot_2dbb_rsp.m	Plots the x and y ball position response along with the corresponding servo input voltages found in the variables *data_x, data_y, data_theta_l_x*, *data_theta_l_y,* and *data_vm* in the Matlab workspace. Users can also use the saved responses contained in MAT files.
meas_step_rsp_specs.m	Measures the peak time, settling time, steady-state error, and percentage overshoot of a given step response.
sample_rsp_*.mat	These various MAT files contain saved measured closed-loop ball position step, ramp, and sinusoidal responses when using both the PD and PID control systems with different configurations, e.g. with and without velocity set-point weight, with a tuned integrator, and so on. The solution blocks in the particular exercise below explain which MAT files to use and what response it represents. For example, *sample_rsp_pd_step_x.mat* contains the measured closed-loop PD step response.

Table 1: Files supplied with the SRV02 2D Ball Balancer Position Control experiment.

4. Pre-Lab Assignments

4.1. Modeling from First-Principles

The complete open-loop system of the 2D Ball Balancer is represented by the block diagram shown in Figure 1. The SRV02 transfer function $P_s(s)$ represents the dynamics between the servo input motor voltage and the resulting load angle. The dynamics between the angle of the servo load gear and the position of the ball is described by transfer function $P_{bb}(s)$. Notice that this is a decoupled system (i.e. the *x*-axis actuator does not affect the *y*-axis response).

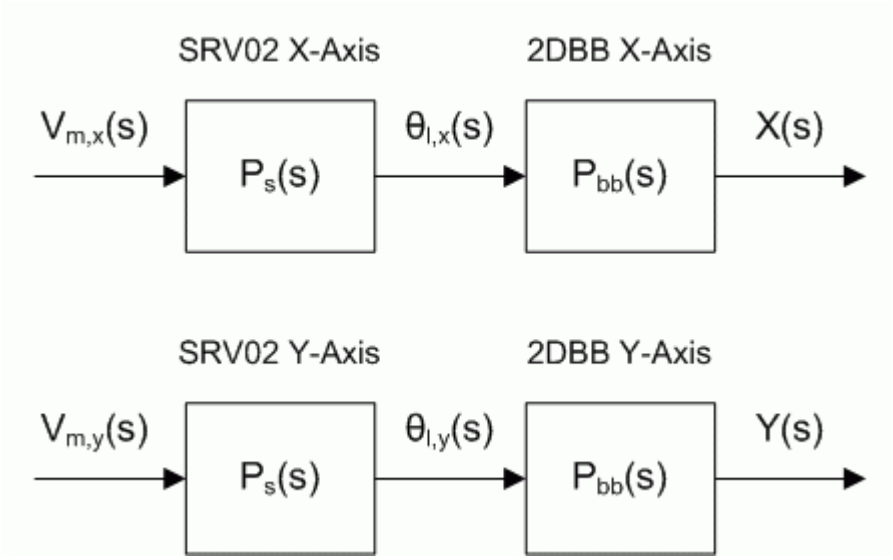

Figure 1: 2D Ball Balancer open-loop block diagram.

Since the both SRV02 devices have the same hardware and the table is symmetrical, it is assumed that the dynamics of each axis is the same. Therefore the modeling is done for only one axis. The block diagram for a single-axis of the 2D Ball Balancer, denoted 1DBB, is shown in Figure 2.

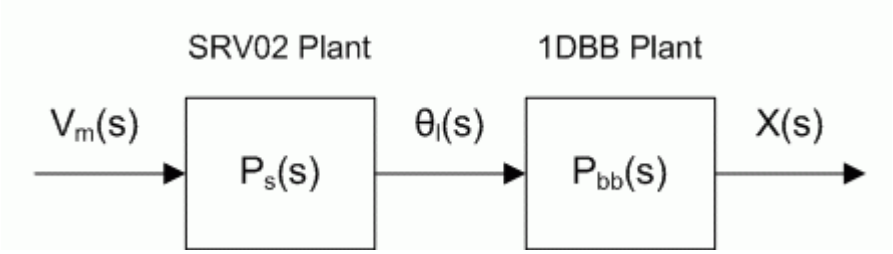

Figure 2: 1D Ball Balancer open-loop block diagram.

The main objective in this section is to obtain the complete SRV02+1DBB transfer function

$$P(s) = P_{bb}(s)\,P_s(s) \qquad [1]$$

where the 1DBB transfer function is

$$P_{bb}(s) = \frac{X(s)}{\theta_l(s)} \quad [2]$$

and the SRV02 transfer function is

$$P_s(s) = \frac{\theta_l(s)}{V_m(s)} \quad [3]$$

.

The 1DBB transfer function describes the displacement of the ball with respect to the load angle of the servo. In the next few sections, the time-based motion equations are developed and, from these equations of motion, its transfer function is obtained. Recall from Reference [7], that the SRV02 voltage-to-load angle transfer function is

$$P_s(s) = \frac{K}{(\tau s + 1)s} \quad [4]$$

.

This can be added to the system to get the full one-axis SRV02+1DBB model.

4.1.1. Nonlinear Equation of Motion

In this section, the equation describing the motions of the ball, *x*, relative to the angle of the beam, α, is derived. Thus the equation of motion, or *eom* for short, will be of the form

$$\frac{d^2}{dt^2}x(t) = f(\alpha(t)) \quad [5]$$

,

where f(α(t)) is a nonlinear function. The incomplete free-body diagram of the ball on a beam is illustrated in Figure 3. Applying Newton's Law of Motion, the sum of the forces acting on the ball alongside the beam equals

$$m_b\left(\frac{d^2}{dt^2}x(t)\right) = \sum F \quad [6]$$

,

where m_b is the mass of the ball.

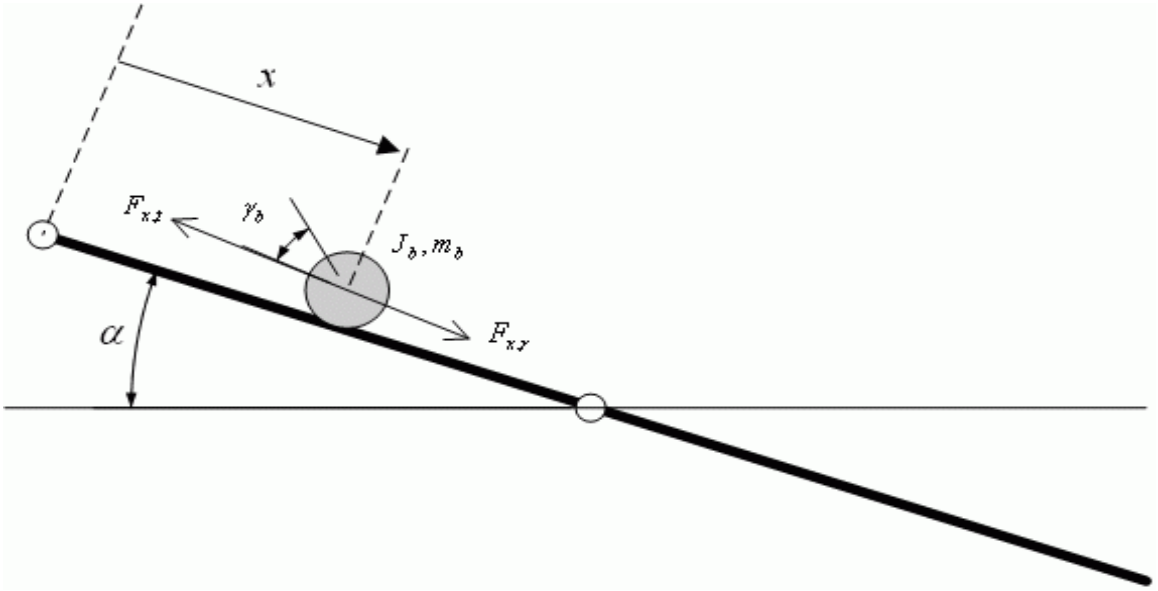

Figure 3: Free-body diagram.

Neglecting friction and viscous damping, the ball forces can be represented by

$$m_b \left(\frac{d^2}{dt^2} \mathrm{x}(t) \right) = F_{x,t} - F_{x,r} \qquad [7]$$

where $F_{x,r}$ is the force from the ball's inertia and $F_{x,t}$ is the translational force generated by gravity. For the ball to be stationary at a certain moment, i.e. be in equilibrium, the force from the ball's momentum must be equivalent to the force produced by gravity.

1. Find the force in the *x* direction, i.e. along the beam, that is caused by gravity, $F_{x,t}$.

Solution:
The complete free-body diagram of the ball on the beam is pictured in Figure 4.

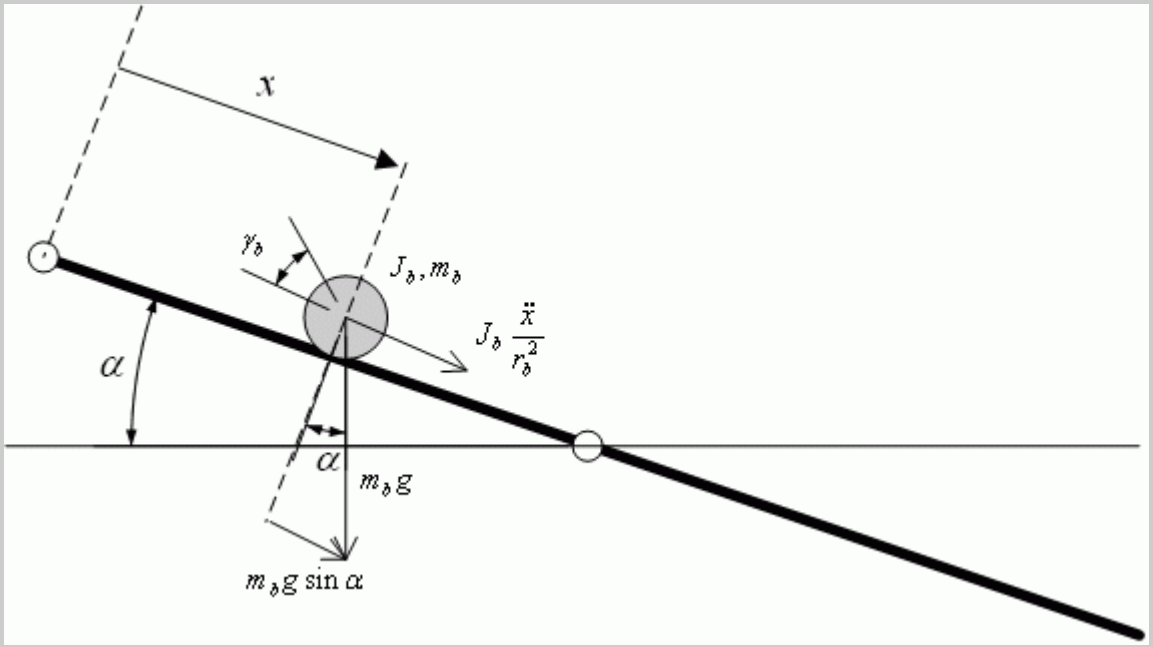

Figure 4: Completed Ball and Beam free-body diagram.

As illustrated above in Figure 4, the force acting in the positive *x* direction when the incline is positive is

$$F_{x,t} = m_b g \sin(\alpha(t)) \quad \text{[s1]}$$

0	1	2

2. Find the force that is caused by the rotational inertia of the ball in the *x* direction, $F_{x,r}$.
   Hint: Use the sector formula to convert between linear and angular displacement (or velocity and acceleration)

$$x(t) = \gamma_b(t)\, r_b \quad \text{[8]}$$

   where γ_b is the angle of the ball and r_b is the ball radius.

Solution:
The force caused by the rotation spin of the ball is

$$F_{x,r} = \frac{\tau_b}{r_b} \quad [s2]$$

where r_b is the radius of the sphere and τ_b is the torque. Given that the torque of the ball equals

$$\tau_b = J_b \left(\frac{d^2}{dt^2} \gamma_b(t) \right) \quad [s3]$$

where γ_b the ball angle, and relationship [8] the force acting on the ball in the *x* direction from its momentum becomes

$$F_{x,r} = \frac{J_b \left(\frac{d^2}{dt^2} x(t) \right)}{r_b^2} \quad [s4]$$

0	1	2

3. Give the nonlinear equation of motion of the ball and beam. It should be in the form shown in [5].

Solution:
Substitute the translational and rotational forces derived in [s1] and [s4], respectively, into Equation [7] to get

$$m_b \left(\frac{d^2}{dt^2} x(t) \right) = m_b g \sin(\alpha(t)) - \frac{J_b \left(\frac{d^2}{dt^2} x(t) \right)}{r_b^2} \quad [s5]$$

Solving for the linear acceleration of the ball this becomes

$$\frac{d^2}{dt^2} x(t) = \frac{m_b g \sin(\alpha(t)) r_b^2}{m_b r_b^2 + J_b} \quad [s6]$$

0	1	2

4.1.2. Adding SRV02 Dynamics

In this section the equation of motion representing the position of the ball relative the angle of the SRV02 load gear is found. The obtained equation is nonlinear, i.e. it includes a trigonometric term, and will have to be linearized so the model can be used for a Laplace-based PID control design.

1. Using the schematic given in Reference [10, find the relationship between the SRV02 load gear angle, θ_l, and the beam angle, α.

Solution:
Consider the servo angle needed to change the height of the beam by *h* given the length of the plate. Taking the sine of the balance table angle gives the expression

$$\sin(\alpha(t)) = \frac{2h}{L_{tbl}} \quad [s7]$$

and taking the sine of the servo load angle results in the equation

$$\sin(\theta_l(t)) = \frac{h}{r_{arm}} \quad [s8]$$

From these we can obtain the following relationship between the table angle and servo angle

$$\sin(\alpha(t)) = \frac{2\sin(\theta_l(t))\, r_{arm}}{L_{tbl}} \quad [s9]$$

0	1	2

2. Find the equation of motion that represent the ball's motion with respect to the SRV02 angle θ_l. Linearize the equation of motion about servo angle $\theta_l(t) = 0$.

Solution:
Insert the servo and beam angle relationship, Equation [s9], into the nonlinear equation of motion (i.e. denoted eom) found in [s6]

$$\frac{d^2}{dt^2}x(t) = \frac{2 m_b g \sin(\theta_l(t))\, r_{arm} r_b^2}{L_{tbl}(m_b r_b^2 + J_b)} \quad [s10]$$

About angle zero, the sine function can be approximated by

$$\sin(\theta_l(t)) = \theta_l(t) \quad [s11]$$

Applying this to the nonlinear eom results in the linear equation of motion of the 1DBB plant

$$\frac{d^2}{dt^2}x(t) = \frac{2 m_b g \theta_l(t)\, r_{arm} r_b^2}{L_{tbl}(m_b r_b^2 + J_b)} \quad [s12]$$

0	1	2

3. Simplify the expression by lumping the coefficient parameters of $\theta_l(t)$ into parameter K_{bb}. This is the model gain of the 1DBB system. Show the new simplified equation of motion. Then, evaluate the model gain numerically the parameters given in Reference [10.
   Hint: Recall that the moment of inertia of a solid sphere is

$$J = \frac{2\,m\,r^2}{5} \qquad [9]$$

   where *m* is the mass of the ball and *r* is its radius.

Solution:

The model gain is the coefficient of $\theta_l(t)$ and is

$$K_{bb} = \frac{2\,m_b\,g\,r_{arm}\,r_b^2}{L_{tbl}(m_b\,r_b^2 + J_b)} \qquad [s13]$$

The linear equation of motion becomes

$$\frac{d^2}{dt^2}x(t) = K_{bb}\,\theta_l(t) \qquad [s14]$$

Given the moment of inertia of a ball in [9] and substituting the parameters listed in Reference [10 into the model gain results in

$$K_{bb} = 1.30\left[\frac{m}{s^2\,rad}\right] \qquad [s15]$$

0	1	2

4.1.3. Obtaining Transfer Function

In this section the transfer function describing the servo voltage to ball position displacement is found for a single axis of the 2D Ball Balancer system.

1. Find the transfer function $P_{bb}(s)$ of the 1DBB. Assume all initial conditions are zero.

Solution:
The Laplace transform of the linear equation of motion in [s14] is

$$s^2 \mathrm{X}(s) - \mathrm{D}(x)(0) - s\,\mathrm{x}(0) = K_{bb}\,\theta_l(s) \quad [s16]$$

Given that the initial conditions are all zero and solving for X(s) we obtain the transfer function

$$\mathrm{X}(s) = \frac{K_{bb}\,\theta_l(s)}{s^2} \quad [s17]$$

From Equation [2], the 1DBB plant transfer function is

$$P_{bb}(s) = \frac{K_{bb}}{s^2} \quad [s18]$$

0	1	2

2. Give the complete SRV02+1DBB process transfer function P(s).

Solution:
As illustrated in Figure 2, both system are in series. Inserting the 1DBB process [s18] and the SRV02 process [4] into Equation [1] gives the complete process

$$\mathrm{P}(s) = \frac{K_{bb}\,K}{(\tau\, s + 1)\, s^3} \quad [s19]$$

Also the servo voltage to *x*-axis ball displacement transfer function is

$$\mathrm{X}(s) = \frac{K_{bb}\,K\,V_m(s)}{(\tau\, s + 1)\, s^3} \quad [s20]$$

0	1	2

4.2. Desired Control Response

The desired closed-loop response specifications are listed in Section 4.2.1 and various equations required in the control design to attain these requirements are given in Section 4.2.2.

4.2.1. Time-Domain Specifications

The time-domain specifications for controlling the position of the SRV02 load shaft are:

$$e_{ss} = 0 \quad , \qquad [10]$$

$$t_p \leq 0.15\,[s] \quad , \text{ and} \qquad [11]$$

$$PO \leq 5.0\,[\text{"\%"}] \quad . \qquad [12]$$

Thus when tracking the load shaft step reference, the transient response should have a peak time less than or equal to 0.15 seconds, an overshoot less than or equal to 5 %, and no steady-state error.

The specifications for controlling the position of the ball for both the *x* and *y* axes of the 2D Ball Balancer are:

$$|e_{ss}| \leq 0.001\,[m] \quad , \qquad [13]$$

$$t_s \leq 2.5\,[s] \qquad [14]$$

$$c_{ts} = 0.04 \quad , \text{ and} \qquad [15]$$

$$PO \leq 7.5\,[\text{"\%"}] \quad . \qquad [16]$$

Given a step reference, the peak position of the ball should not overshoot over 7.5%. After 2.5 seconds, the ball should settled within 4% of its steady-state value (i.e. not the reference) and the steady-state should be within 1 mm of the desired position.

4.2.2. Percentage Overshoot, Peak Time, and Settling Time

Recall from Reference [8] that the peak time and percentage overshoot equations are

$$t_p = \frac{\pi}{\omega_n \sqrt{1 - \zeta^2}} \qquad [17]$$

and

$$PO = 100\, \mathbf{e}^{\left(- \frac{\pi \zeta}{\sqrt{1 - \zeta^2}} \right)} \qquad [18]$$

where ω_n is the natural frequency and ζ is the damping ratio of the system.

The settling time equation, derived in Reference [9], is expressed

$$t_s = -\frac{\ln(c_{ts}\sqrt{1-\zeta^2})}{\zeta \omega_n} \qquad [19]$$

where c_{ts} is the settling time percentage.

4.3. Ball and Beam Cascade Control Design

The cascade control that will be used for the x-axis of the SRV02+2DBB system is illustrated by the block diagram given in Figure 5. Based on the measured ball position, X(s), the ball balancer compensator, $C_{bb}(s)$, in the outer-loop computes the servo load angle needed, $\Theta_d(s)$, to attain the desired ball position, $X_d(s)$. The inner loop is the servo position control system described in Reference [8]. Thus the servo compensator $C_s(s)$ calculates the motor voltage required for the load angle to track the given desired load angle.

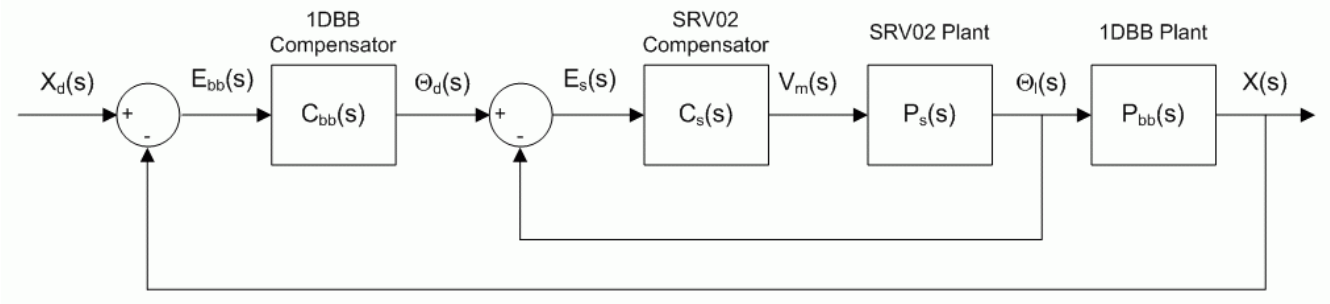

Figure 5: Cascade control system used to control ball position in the x-axis of the SRV02+2DBB plant.

In Section 4.3.1, the inner-loop position controller for the SRV02 is designed, similarly as explained in Reference [8]. The outer-loop compensator is then designed in Section 4.3.2 according to the given specifications

4.3.1. Inner Loop Design: SRV02 PV Position Controller

In this section, the proportional-velocity (PV) controller gains are computed for the SRV02 when it is in the high-gear configuration and based on the specifications given in Section 4.2.1. The internal control loop is depicted in the block diagram shown in Figure 6.

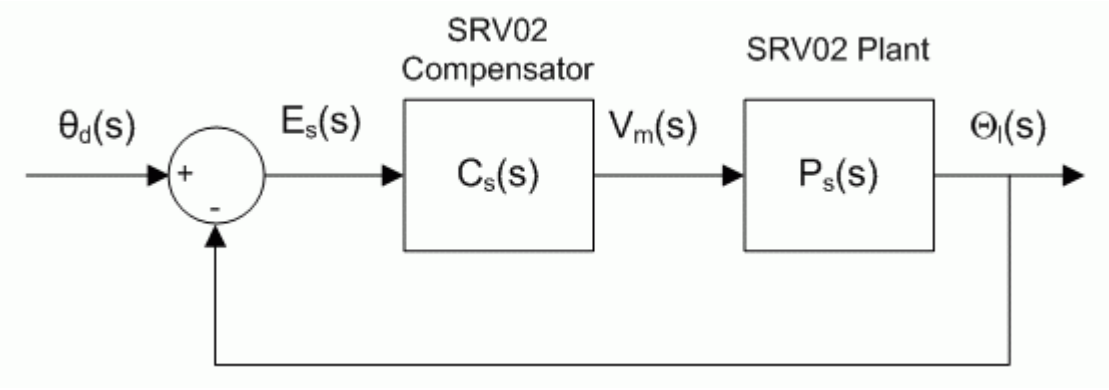

Figure 6: SRV02 closed-loop system.

1. Compute the nominal SRV02 model parameters, K and τ, when it is in high-gear configuration. See Reference [7] for more information on this.

   Solution:
   How to compute the steady-state gain and time constant SRV02 transfer function parameters based on the device parameters is explained in Reference [7]. Going through this for the SRV02 in high-gear configuration results in a steady-state gain of

   $$K = 1.76\left[\frac{rad}{s\,V}\right] \qquad [s21]$$

   and a time constant of

   $$\tau = 0.0285[s] \qquad [s22]$$

   Note: This exercise is optional and is at the instructor's discretion. There are considerable calculations involved in this exercise and the instructor may wish to give these model parameters to the student.

0	1	2

2. Calculate the minimum damping ratio and natural frequency required to meet the SRV02 specifications given in Section 4.2.1.

Solution:
The damping ratio needed to obtain the desired percentage overshoot specifications given in [12] is found using Equation [18],

$$\zeta = 0.690 \qquad \text{[s23]}$$

Using Equation [17], the minimum natural frequency required to meet the desired peak time, given in [11], is

$$\omega_n = 28.9\left[\frac{rad}{s}\right]. \qquad \text{[s24]}$$

3. Using the derivations in Experiment #2: SRV02 Position Control (Reference[8]), calculate the control gains needed to satisfy the time-domain response requirements.

Solution:
Using the model parameters in [s21] and [s22] as well as the desired natural frequency found in [s24] generates the proportional control gain

$$k_p = 13.5\left[\frac{V}{rad}\right]. \qquad \text{[s25]}$$

The velocity control gain is obtained using the model parameters and the minimum damping ratio specification in [s23] to get

$$k_v = 0.078\left[\frac{V\,s}{rad}\right]. \qquad \text{[s26]}$$

Thus the position response of the load gear on an SRV02 in the high-gear configuration using the PV controller with the gains above will satisfy the specifications listed in Section 4.2.1.

4.3.2. Outer Loop Controller Design

The inner loop that controls the position of the SRV02 load shaft is complete. The dynamics of the servo dynamics can now be considered negligible. Thus it is assumed that the desired load angle equals the actual load angle

$$\theta_l(t) = \theta_d(t). \qquad \text{[20]}$$

The outer-loop shown in Figure 7 will be used to control the position of the ball along the x-axis of the plate.

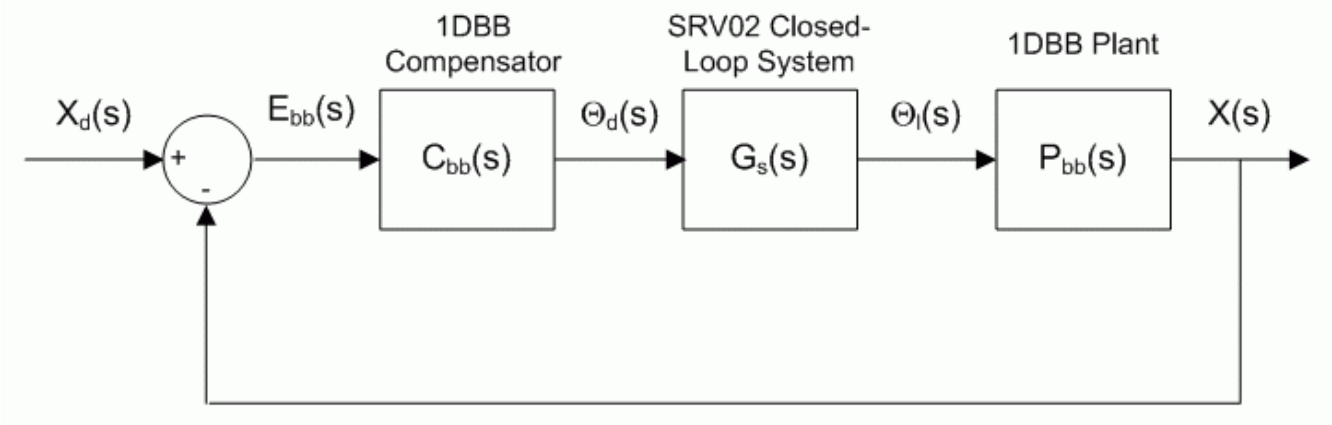

Figure 7: X-Axis 2DBB closed-loop system.

A proportional-integral-derivative (PID) compensator is designed in Section 4.3.2.1. Both the PD and PID gains are then computed and the steady-state error of each is assessed.

4.3.2.1. PID Control Design

The PID compensator shown in Figure 8 has the following form in the time-domain

$$\theta_d(t) = k_{p,bb}\left(x_d(t) - \mathrm{x}(t)\right) + k_{d,bb}\left(b_{sd}\left(\frac{d}{dt}x_d(t)\right) - \left(\frac{d}{dt}\mathrm{x}(t)\right)\right) + k_{i,bb}\int x_d(t) - \mathrm{x}(t)\,dt \quad [21]$$

In this section the required proportional gain $k_{p,bb}$, integral gain $k_{i,bb}$, and derivative gain $k_{d,bb}$ are found to meet the specifications in Section 4.2.1. The controller includes a velocity set-point weight parameter, denoted by b_{sd}. This controls the amount of influence the setpoint has in the derivative error computation. Its effect on the response will be investigated in the in-lab exercises.

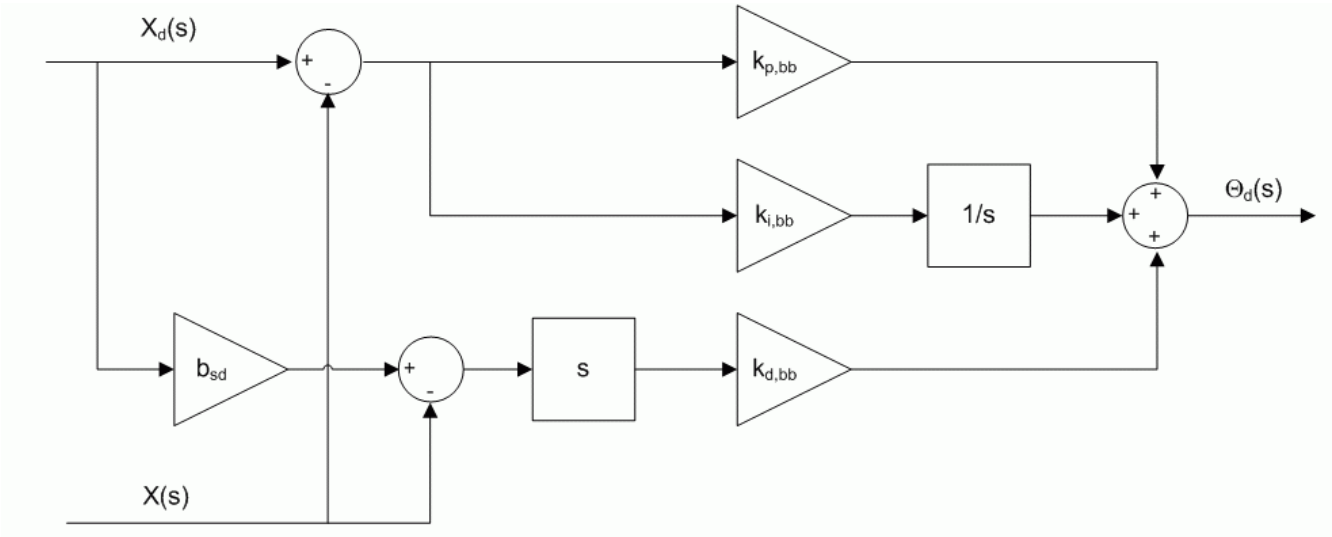

Figure 8: PID Compensator with derivative set-point weighting.

1. Using the block diagram in Figure 8 or the time-domain controller in Equation [21], find the closed-loop equation of the 2DBB *x*-axis system (only the outer-loop, no servo dynamics).

Solution:

From Figure 8 or by taking the Laplace of Equation [21], the outer-loop controller is

$$\Theta_d(s) = \left(k_{p,bb} + \frac{k_{i,bb}}{s} \right) (X_d(s) - X(s)) + k_{d,bb}\, s\, (b_{sd} X_d(s) - X(s)) \quad \text{[s27]}$$

Given that there are no servo dynamics, i.e. $\Theta_d(s) = \Theta_l(s)$, the closed-loop equation can be found by substituting the above controller into the 1DBB open-loop transfer function in [s17] and solving for $X(s)/X_d(s)$,

$$\frac{X(s)}{X_d(s)} = \frac{K_{bb}(k_{p,bb}\, s + k_{i,bb} + k_{d,bb}\, s^2\, b_{sd})}{s^3 + K_{bb}\, k_{p,bb}\, s + K_{bb}\, k_{i,bb} + K_{bb}\, k_{d,bb}\, s^2} \quad \text{[s28]}$$

0	1	2

2. The third-order prototype characteristic equation is

$$(s^2 + 2\zeta \omega_n s + \omega_n^2)(s + p_0) \quad \text{[22]}$$

where ω_n and ζ are the natural frequency and the damping ratio of the system and p_0 is the pole location. These parameters determine the response of the third-order system and can be set such that the specifications in Section 4.2.1 are satisfied. Find the PID gains needed to meet the ω_n, ζ, and p_0 parameters.

Solution:

When expanded, the third-order characteristic equation becomes

$$s^3 + (2\zeta\omega_n + p_0)s^2 + (\omega_n^2 + 2\zeta\omega_n p_0)s + \omega_n^2 p_0 \quad \text{[s29]}$$

The characteristic equation of the closed-loop PID system in [s28] is

$$s^3 + K_{bb}k_{p,bb}s + K_{bb}k_{i,bb} + K_{bb}k_{d,bb}s^2 \quad \text{[s30]}$$

The following expressions are obtained when equating the coefficients of s^0, s^1, and s^2 in each characteristic equation

$$K_{bb}k_{i,bb} = \omega_n^2 p_0, \quad \text{[s31]}$$

$$K_{bb}k_{p,bb} = \omega_n^2 + 2\zeta\omega_n p_0 \quad \text{[s32]}$$

and

$$K_{bb}k_{d,bb} = 2\zeta\omega_n + p_0 \quad \text{[s33]}$$

Solving for the PID control gains gives

$$k_{p,bb} = \frac{\omega_n(\omega_n + 2\zeta p_0)}{K_{bb}}, \quad \text{[s34]}$$

$$k_{i,bb} = \frac{\omega_n^2 p_0}{K_{bb}}, \quad \text{[s35]}$$

and

$$k_{d,bb} = \frac{2\zeta\omega_n + p_0}{K_{bb}} \quad \text{[s36]}$$

0	1	2

3. Using the equations described in Section 4.2.2, express the natural frequency and the damping ratio in terms of percentage overshoot and settling time specifications.

Solution:
To find the damping ratio given a percentage overshoot specification, solve for ζ in Equation [18] and get the expression

$$\zeta = -\ln\left(\frac{PO}{100}\right)\sqrt{\frac{1}{\ln\left(\frac{PO}{100}\right)^2 + \pi^2}} \quad \text{[s37]}$$

Solving for ω_n in Equation [19] expresses the natural frequency in terms of the settling time

$$\omega_n = -\frac{\ln(c_{ts}\sqrt{1-\zeta^2})}{\zeta t_s} \quad \text{[s38]}$$

0	1	2

4. Calculate the minimum damping ratio and natural frequency required to meet the 2DBB overshoot and settling time specifications given in Section 4.2.1.

Solution:
Substitute the percentage overshoot specifications given in [12] into Equation [s37] to get the required damping ratio

$$\zeta = 0.635 \quad \text{[s39]}$$

Using this result and the desired settling time, given in [14], with Equation [s38] gives the minimum natural frequency needed

$$\omega_n = 2.19\left[\frac{rad}{s}\right] \quad \text{[s40]}$$

0	1	2

4.3.2.2. PD Control Gains

The required proportional-derivative control gains needed to meet the specifications will first be investigated. By setting the integral gain to zero, or setting the pole location to

$$p_0 = 0 \quad \text{[23]}$$

and the velocity set-point weight to zero, $b_{sd} = 0$, the closed-loop PD transfer function fits the prototype second-order system

$$\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad [24]$$

1. Find the proportional and derivative gains needed to obtain a closed-loop ball position response that will yield to the specifications in Section 4.2.1.

Solution:
Substituting the pole location $p_0 = 0$, the model parameters

$$K = 8.82\left[\frac{rad}{s\,V}\right] \quad [s41]$$

and

$$\tau = 0.0507\,[s], \quad [s42]$$

as well as the desired damping ratio and natural frequency computed in [s39] and [s40], respectively, into Equation [s34], generates the proportional control gain

$$k_{p,bb} = 3.69\left[\frac{rad}{m}\right]. \quad [s43]$$

Similarly, the derivative control gain is obtained by inserting these parameters into Equation [s36]

$$k_{d,bb} = 2.14\left[\frac{rad\,s}{m}\right]. \quad [s44]$$

Since $p_0 = 0$ the integral gain is zero, i.e. $k_{i,bb} = 0$.

0	1	2

4.3.2.3. PID Control Gains

To compute the PID gains, the pole location, p_0, is specified according to the pole decay time constant T_p with the expression

$$p_0 = \frac{1}{T_p}. \quad [25]$$

1. For the pole decay time constant

$$T_p = 1.0\,[s] \quad [26]$$

find the proportional, integral, and derivative gains needed to obtain a closed-loop ball position response that will yield to the specifications in Section 4.2.1.

Solution:
To obtain the specified time constant, the pole is placed at

$$p_0 = 1.00\left[\frac{rad}{s}\right] \quad [s45]$$

This is found by substituting the decay time given in [25] into Equation [26].

Inserting the pole location along with the model parameters

$$K = 8.82\left[\frac{rad}{s\,V}\right] \quad [s46]$$

and

$$\tau = 0.0507\,[s] \quad [s47]$$

as well as the desired damping ratio and natural frequency computed in [s39] and [s40], respectively, in equations [s34], [s35], [s36], generates the PID control gains

$$k_{p,\,bb} = 5.83\left[\frac{rad}{m}\right] \quad [s48]$$

$$k_{d,\,bb} = 2.91\left[\frac{rad\,s}{m}\right] \quad [s49]$$

and

$$k_{i,\,bb} = 3.69\left[\frac{rad}{m\,s}\right] \quad [s50]$$

0	1	2

4.3.3. Steady-State Error

In this section, the steady-state error of the ball position is evaluated using the PD controller when tracking a step and ramp reference.

1. The error of the x-axis of the 2DBB system is defined

$$E(s) = X_d(s) - X(s) \quad [27]$$

Find the error transfer function when using the PD controller.

Solution:

The closed-loop PID transfer function found in Equation [s28] can be re-written as

$$X(s) = \frac{K_{bb} X_d(s) (k_{p,bb} s + k_{i,bb} + k_{d,bb} s^2 b_{sd})}{s^3 + K_{bb} k_{p,bb} s + K_{bb} k_{i,bb} + K_{bb} k_{d,bb} s^2} \quad [s51]$$

Substituting this into the error expression E(s) above results in the PID error transfer function

$$E(s) = \frac{X_d(s) s^2 (s + K_{bb} k_{d,bb} - K_{bb} k_{d,bb} b_{sd})}{s^3 + K_{bb} k_{p,bb} s + K_{bb} k_{i,bb} + K_{bb} k_{d,bb} s^2} \quad [s52]$$

Setting $k_i = 0$ gives the PD error transfer function

$$E(s) = \frac{X_d(s) s (s + K_{bb} k_{d,bb} - K_{bb} k_{d,bb} b_{sd})}{s^2 + K_{bb} k_{p,bb} + K_{bb} k_{d,bb} s} \quad [s53]$$

0	1	2

2. Find the steady-state error of the PD system when a step reference with an amplitude of R_0

$$X_d(s) = \frac{R_0}{s} \quad [28]$$

is applied. Is the steady-state error specification in Section 4.2.1 satisfied?
Hint: Use the final-value theorem.

Solution:
Substitute the setpoint into the PD error transfer function

$$E(s) = \frac{R_0 (s + K_{bb} k_{d,bb} - K_{bb} k_{d,bb} b_{sd})}{s^2 + K_{bb} k_{p,bb} + K_{bb} k_{d,bb} s} \quad [s54]$$

Recall the final-value theorem

$$e_{ss} = \lim_{s \to 0} s E(s) \quad [s55]$$

Applying this to the PD step error transfer function above gives

$$e_{ss} = 0 \quad [s56]$$

The steady-state error specification is satisfied since the resulting error is less than 1 mm.

0	1	2

3. Find the steady-state error of the PD system when subject to the ramp setpoint

$$X_d(s) = \frac{R_0}{s^2} \quad [29]$$

Solution:
Substitute the ramp setpoint into the PD error transfer function

$$E(s) = \frac{R_0 (s + K_{bb} k_{d,bb} - K_{bb} k_{d,bb} b_{sd})}{s (s^2 + K_{bb} k_{p,bb} + K_{bb} k_{d,bb} s)} \quad [s57]$$

Using the final-value theorem, the steady-state error of the system is

$$e_{ss} = - \frac{R_0 k_{d,bb} (-1 + b_{sd})}{k_{p,bb}} \quad [s58]$$

0	1	2

4. Calculate the steady-state error when the ramp setpoint amplitude (i.e. the slope) is

$$R_0 = 0.0120 \left[\frac{m}{s} \right] \quad [30]$$

and when the velocity set-point weight is $b_{sd} = 0$. Is the steady-state error specification satisfied?

Solution:
To obtain the PD ramp steady-state error when $b_{sd} = 0$, insert the gains found in Section 4.3.2.2 and the amplitude in Equation [s58],

$$e_{ss} = 0.696\,[cm] \quad \text{[s59]}$$

This exceeds 0.1 mm and therefore does not meet the desired the steady-state error requirement.

0	1	2

5. Compute the velocity set-point weight needed in order for the steady-state error specification to be met.

Solution:
Solve for the set-point weight in the [s58] to get

$$b_{sd} = \frac{-e_{ss}\,k_{p,bb} + R_0\,k_{d,bb}}{R_0\,k_{d,bb}} \quad \text{[s60]}$$

Substitute steady-state error specification [13] in Section 4.2.1 and the PD gains found in Section 4.3.2.2 in the above expression to find the needed set-point weight

$$b_{sd} = 0.86 \quad \text{[s61]}$$

This result in a steady-state error that is 0.1 mm.

0	1	2

4.3.4. Integrator Windup

The actuator limits sometimes have to be taken into consideration when the control system includes an integrator. Usually the limitation is due to the inherent capability of the actuator, e.g. maximum supply voltage of amplifier. When the controller saturates the actuator, due to a large setpoint for example, the integrator keeps increasing to try and compensate for the error. This called integrator windup and it can result in the system going unstable or having bad performance, e.g. large overshoot.

The PID system used to control the 2D Ball Balancer is depicted in Figure 9 and it has an anti-windup feedback loop. Compared to the PID controller in Figure 8, this includes another loop around the integrator. When the integrator saturates the additional loop decreases the integrator input and prevents

the it from "building up" too much energy. Commonly in other windup protection schemes, the anti-windup loop feeds back the entire output of the control system, i.e. the output of the PID controller and not only the integral control. However for the 2D Ball Balancer, anti-windup is primarily geared to improve the response performance. There are no hardware-bound actuator limitations. Instead the user sets the INT_MAX parameter to a value at which the integrator should saturate.

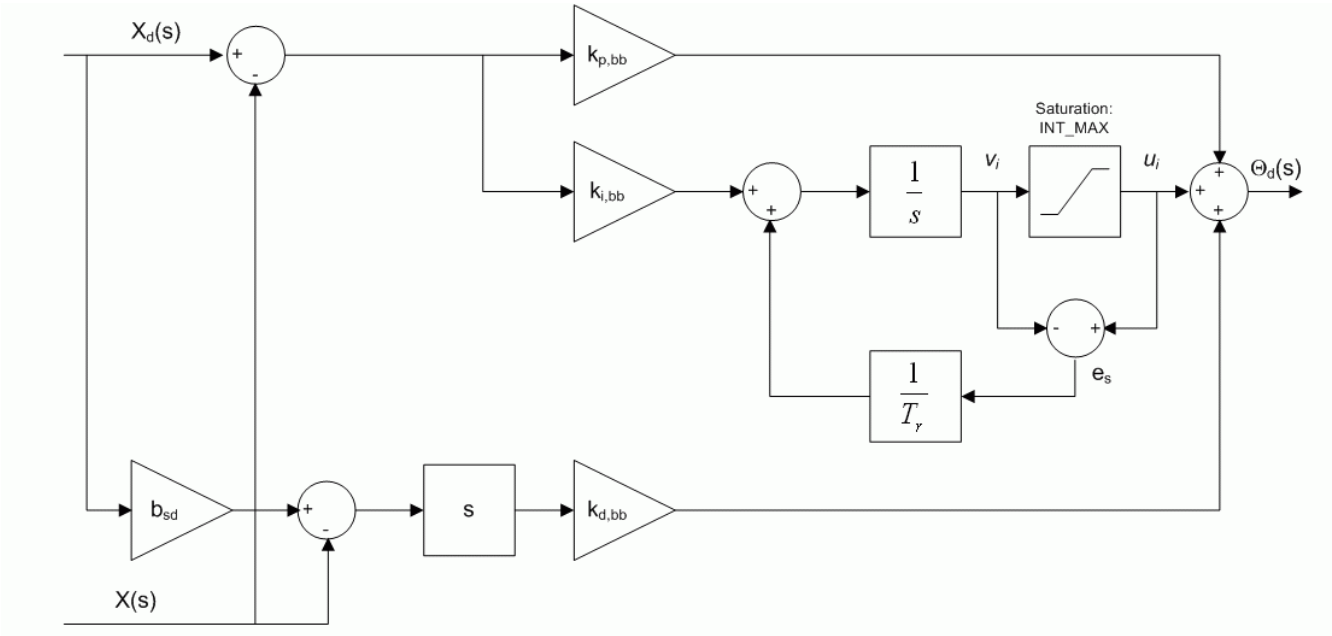

Figure 9: Block diagram of PID controller with integrator anti-windup.

Basically the integral control equals

$$u_i = \begin{cases} -INT\_MAX & v_i < -INT\_MAX \\ v_i & v_i - INT\_MAX < 0 \ \textbf{and} \ -INT\_MAX - v_i < 0 \\ INT\_MAX & INT\_MAX < v_i \end{cases} \quad [31]$$

where INT_MAX is the maximum limit of the integrator set in the *Saturation* block. The anti-windup loop has a time constant of T_r and is active only when the integrator saturates. When the integrator does not saturate it functions as a normal integral control, thus $u_i = v_i$. When the integrator saturates, the anti-windup controller drives the saturation error, e_s, to zero. This effectively makes the integrator output equal to the saturation limit, i.e. either INT_MAX or -INT_MAX.

How fast the integrator is reset depends on the tracking time constant T_r. A long time constant gives a slow reset and a short time constant results in a fast reset time.

4.4. Reading Ball Position from Camera

The position of the ball is measured using the reading from the overhead camera. The controllers provided can detect the position of the ball and output the planar position of the ball. In the laboratory, the camera is adjusted so the plate fits inside its viewing area. The width and height resolutions of the camera are both set to the same value, denoted by the variable *res*, so that the viewing area is a square As illustrated in Figure 10, the raw ball position measurement along the x-axis and the y-axis, labeled as the $[X_c,Y_c]$ coordinate system, are in pixels and are called p_x and p_y, respectively. For example, if the camera is set to resolution of 320 by 320 and the ball travels from the bottom-left to the top-left corner of the plate, then p_x goes from 0 points to 320. Remark that the camera-based axis is not intuitive, the y-axis goes horizontal and the x-axis goes vertical.

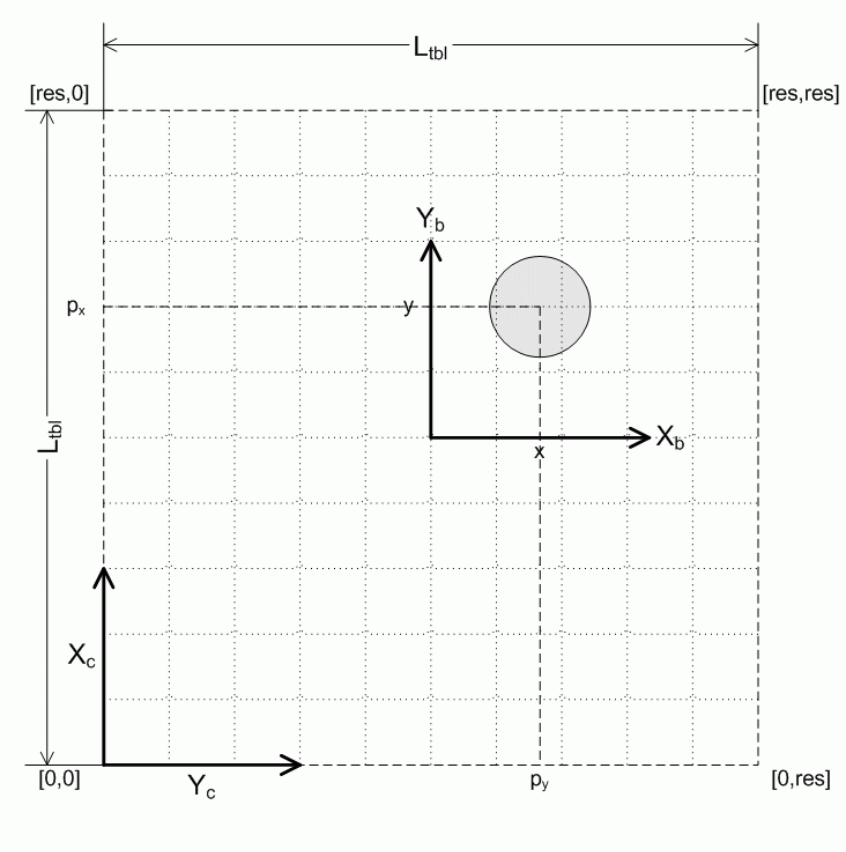

Figure 10: Ball position in terms of raw camera measurement and $[X_b,Y_b]$ Cartesian coordinate.

1. The ball is to be controlled relative to the coordinate axis $[X_b,Y_b]$ and in metric units instead of pixels. Using Figure 10, find the functions

   $$x = f(p_y) \qquad [32]$$

   and

   $$y = f(p_x) \qquad [33]$$

that describe the position of the ball relative to the $[X_b, Y_b]$ coordinate system from the raw camera point measurement p_x and p_y.

Solution:

From Figure 10, the *x* and *y* ball positions can be expressed

$$x = L_{tbl}\left(\frac{p_y}{res} - \frac{1}{2}\right) \quad [s62]$$

and

$$y = L_{tbl}\left(\frac{p_x}{res} - \frac{1}{2}\right) \quad [s63]$$

.

Remark that the X and Y axes of the camera-based, O_c, and the ball-based, O_b ,coordinate systems are reverse.

0	1	2

2. As discussed in Reference [10, the image viewed by the camera has to be customized in order to view the entire plate. However, the minimum increment at which the *Left* and *Top* image position can be changed is 20 pixels. How does this affect the accuracy of the metric position measurement? Assuming the camera has been setup with a resolution of 440 pixels (i.e. width and height of image is 440), calculate in centimeters how much the 20 pixel resolution translate to.

Solution:

The left and top image position increment resolution is

$$e_p = 20\,[pixels] \quad [s64]$$

.

Using the equation

$$e_x = \frac{e_p L_{tbl}}{res} \quad [s65]$$

,

the metric amount that the resolution translates to can be calculated. For a camera resolution of 440 pixels and the length of the table given in Reference [9], the position resolution is

$$e_x = 1.31\,[cm] \quad [s66]$$

.

0	1	2

5. In-Lab Procedures

The closed-loop response of the 2D Ball Balancer is simulated in Section 5.1 using the PD and PID controllers developed. The designed feedback systems are then used in Section 5.2 to control the position of the ball on the actual 2DBB device.

5.1. Position Control Simulation

The *s_2dbb_pos* Simulink diagram shown in Figure 11 is used to simulate the closed-loop position response of the 2DBB system using the nonlinear model developed in Section 4.1 and the cascade control designed in Section 4.3.

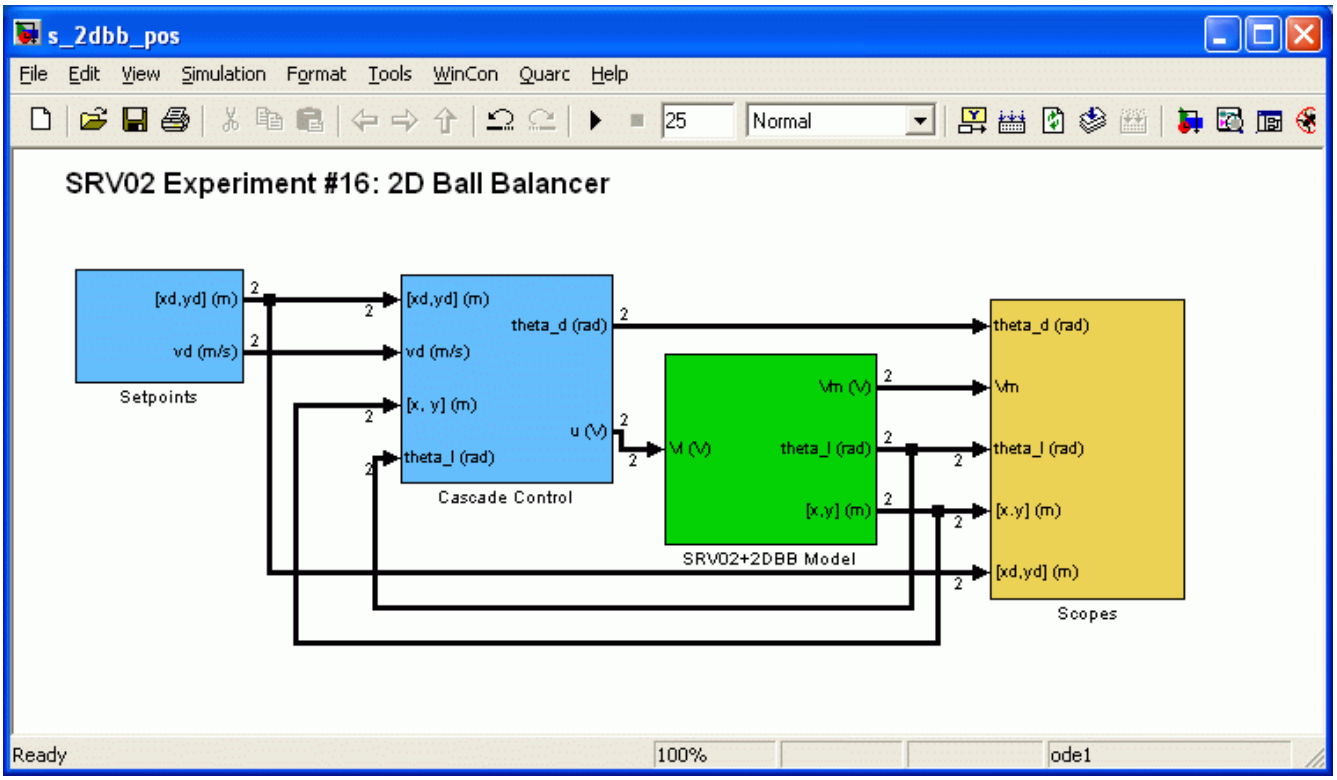

Figure 11: Simulink diagram used to simulated the closed-loop position control of the Quanser 2DBB system.

The *SRV02+2DBB Model* subsystem includes the $P_{bb}(s)$ transfer function that was derived in Section 4.1.3 for both axes. Recall in Section 4.1.2 that the model had to be linearized in order to obtain the $P_{bb}(s)$ transfer function. This nonlinearity is re-introduced in order to represent the plant more accurately and ensure the control specifications can still be satisfied. The *Cascade Control: X and Y* subsystem implements the PID control with the anti-windup protection detailed in Section 4.3. Thus it contains the inner-PV SRV02 control designed in Section 4.3.1 and the the outer PID control developed in Section 4.3.2. Remark that it includes a *Saturation* block that limits the SRV02 angle between ±30 degrees. The *Setpoints* block can generate a step, ramp, or sinusoidal setpoint for both x and y axes. It

includes *Continuous Sigmoid* blocks to generate the reference velocity trajectory.

In both the simulated and implemented controllers, the direct derivative is not taken to compute the velocity of the ball. The ball position measurement has some inherent noise and taking its derivative would result in an amplified high-frequency signal that is eventually fed back into the motor. This causes a grinding noise and may eventually damage the actuator. Instead, the measured ball velocity is computed using a first-order high-pass filter of the form

$$H(s) = \frac{\omega_f s}{s + \omega_f}, \qquad [34]$$

where ω_f is the cutoff frequency.

Go through Section 5.1.1 to simulate the 2DBB system using the PD controller and ensure it meets the specifications under a step, ramp, and sinusoidal reference. The full PID control is then simulated in Section 5.1.2 under the same reference inputs.

5.1.1. PD Simulation

Go through the steps in Section 5.1.1.1 to setup the simulation for PD control. The procedure to simulate the step and ramp responses are given in sections 5.1.1.2 and 5.1.1.3.

5.1.1.1. Setup for PD Simulation

Follow these steps to configure the lab properly:

1. Load the Matlab software.
2. Browse through the *Current Directory* window in Matlab and find the folder that contains the 2DBB controller files.
3. Double-click on the *s_2dbb_pos.mdl* file to open the Simulink diagram shown in Figure 11.
4. Double-click on the *setup_srv02_exp17_2dbb.m* file to open the setup script for the 2DBB Simulink models.
5. **Configure setup script**: When used with the 2D Ball Balancer, the SRV02 must be in the high-gear configuration and no load is to be specified. Make sure the script is setup to match this configuration, i.e. the EXT_GEAR_CONFIG should be set to 'HIGH' and the LOAD_TYPE should be set to 'NONE'. Also, ensure the ENCODER_TYPE, TACH_OPTION, K_CABLE, AMP_TYPE, and VMAX_DAC parameters are set according to the SRV02 system that is to be used in the laboratory. Next, set CONTROL_TYPE to 'MANUAL'.

Solution:
Set CONTROL_TYPE = 'AUTO' and "Tp = Inf" (in the *Control Specifications* section) to automatically calculate the PD gains according to the specifications.

The students should not have access to the scripts d_pv_design.m, d_2dbb_model_param.m, d_2dbb_specs.m, plot_2dbb_rsp.m, and d_2dbb_pid_design.m described in Table 1. However, exactly what should be given to the students is at the discretion of the instructor. For example, it may be desired to supply d_pv_design to automatically calculate the SRV02 PV gains.

6. Run the script by selecting the Debug | Run item from the menu bar or clicking on the *Run* button in the tool bar. The messages shown in Text 1, below, should be generated in the Matlab Command Window. The specifications are loaded but the SRV02 PV gains and the 2DBB PD gains are all set to zero. These gains are found in the pre-lab exercises and must be entered in Matlab.

```
SRV02 model parameters:
   K   = 1.76 rad/s/V
   tau = 0.0285 s
SRV02 Specifications:
   tp = 0.15 s
   PO = 5 %
Balance Table model parameter:
   K_bb = 0 m/s^2/rad
Balance Table Specifications:
   ts = 2.5 s
   PO = 7.5 %
SRV02 PV control gains:
   kp = 0 V/rad
   kv = 0 V.s/rad
Balance Table PID Gains:
   kp_bb = 0 rad/m
   ki_bb = 0 rad/m/s
   kd_bb = 0 rad.s/m
```

Text 1: Display message shown in Matlab Command Window after running setup_srv02_exp17_2dbb.m.

7. Enter the 2DBB model gain found in Section 4.1.2 in Matlab as variable *Kbb*.
8. Enter the 2DBB PD gains that were found in Section 4.3.2.2 as variables *kp_bb* and *kd_bb*.
9. Set the 2DBB integral gain to zero, thus *ki_bb* = 0.
10. Enter the low-gear SRV02 model gain, *K*, and the model time constant, *tau*, in Matlab found in Section 4.3.1.
11. Enter the SRV02 PV gains: called variables *kp* and *kv* in Matlab found in Section 4.3.1.
12. For the x-axis, open the ball position scope *x (m)*, the load shaft position scope *theta_l_x (deg)*, and the SRV02 motor input voltage scope *Vm(V)*. The *Vm (V)* scope displays the voltage of both the x-axis and y-axis servo units.
13. For the y-axis, open the ball position scope *y (m)* and the load shaft position scope *theta_l_y (deg)*.

14. Open the *X-Y Figure* scope. This displays both the desired and simulated *x* and *y* positions.

5.1.1.2. Step PD Response

In this section, the step response using the proportional-derivative controller is simulated to verify that the specifications are met. The setpoints are configured such that the ball tracks the shape of square when viewed in the 2D plane.

Follow these steps to simulate the PD step response:

1. Go through the steps in Section 5.1.1.1 to setup the Matlab workspace.
2. In the *Setpoints* subsystem, select square in the *Signal Type* field of both the *SRV02 Signal Generator X* and *SRV02 Signal Generator Y* in order to generate a step reference.
3. Set both the *Amplitude X (cm)* and *Amplitude Y (cm)* slider gain blocks to 5 to generate a step with an amplitude of 5.0 centimeters. Notice that the y-axis setpoint is delayed in order for the ball to make a square shape when viewed in the *xy* plane. Thus the trajectory in *X-Y Figure* should be a 5.0 by 5.0 cm^2 square.
4. In the first simulation and as designed in Section 4.3.2.2, the velocity set-point weight will be set to zero. To do this set the *bsd* Slider Gain block in both the *Cascade Control\2DBB PID Position Control: X* and *Cascade Control\2DBB PID Position Control: Y* subsystems to 0.
5. Start the simulation. By default, the simulation runs for 25.0 seconds. The scopes should be displaying responses similar to figures 12, 13, and 14. The yellow and purple plots in the *x (m)* scope is the ball position setpoint and its simulated response. Similarly in the *theta_l_x (deg)* scope, the yellow trace is the desired servo angle position, which is generated by the outer-loop control, and the the purple plot is the simulated servo response. The yellow and purple plots in the *Vm (V)* scope are the voltages applied to the servo motor for the x-axis and y-axis controllers, respectively.

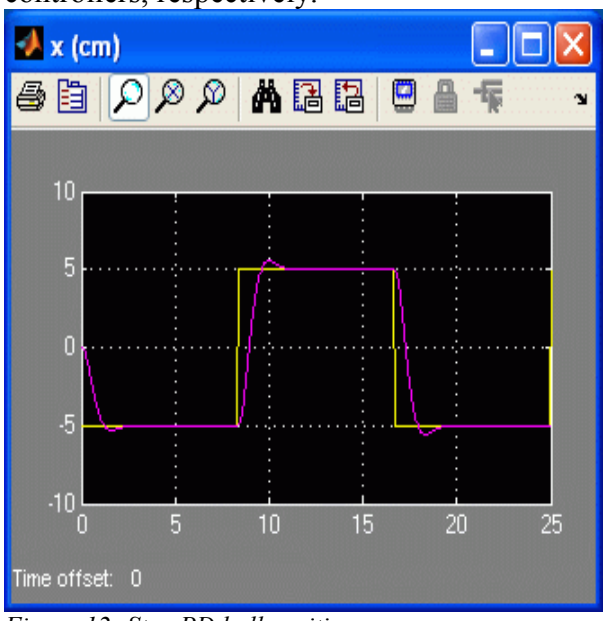

Figure 12: Step PD ball position response.

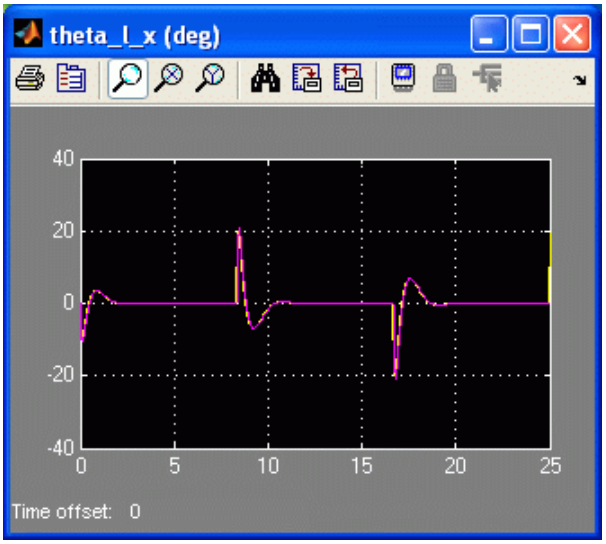

Figure 13: Step PD servo angle response.

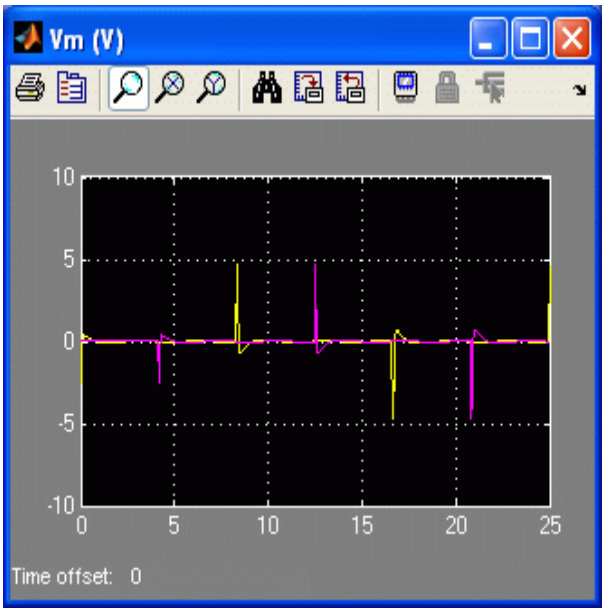

Figure 14: Step PD input voltage (both x- and y-axes).

6. Generate a Matlab figure showing the x-axis PD step ball position, servo angle, and servo input voltage response and attach it to your report. The response from each scope is saved to a Matlab variable after each simulation run. The *x (m)* scope saves its response to the variable called *data_x* and the *theta_l_x (deg)* scope saves its response to the variable *data_theta_l_x*. The *data_x* variable has the following structure: *data_x(:,1)* is the time vector, *data_x(:,2)* is the setpoint, and *data_x(:,3)* is the simulated ball position. For the *data_theta_l* variable, *data_theta_l (:,1)* is the time and *data_theta_l (:,2)* is the servo angle. The SRV02 motor input voltage is saved in the *data_vm* variable where *data_vm (:,1)* is the time, *data_vm (:,2)* is the x-axis servo voltage, and *data_vm (:,2)* is the y-axis servo voltage.

Solution:
The closed-loop step position response when using the PD control is depicted in Figure 15. This is generated using the *meas_2dbb_specs.m* script. To use this script, do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, and *Tp* = Inf.
2. Run the *s_srv02_2dbb* Simulink model with *bsd* = 0 in both PID X and Y control systems.
3. Run the *meas_2dbb_specs.m* script.

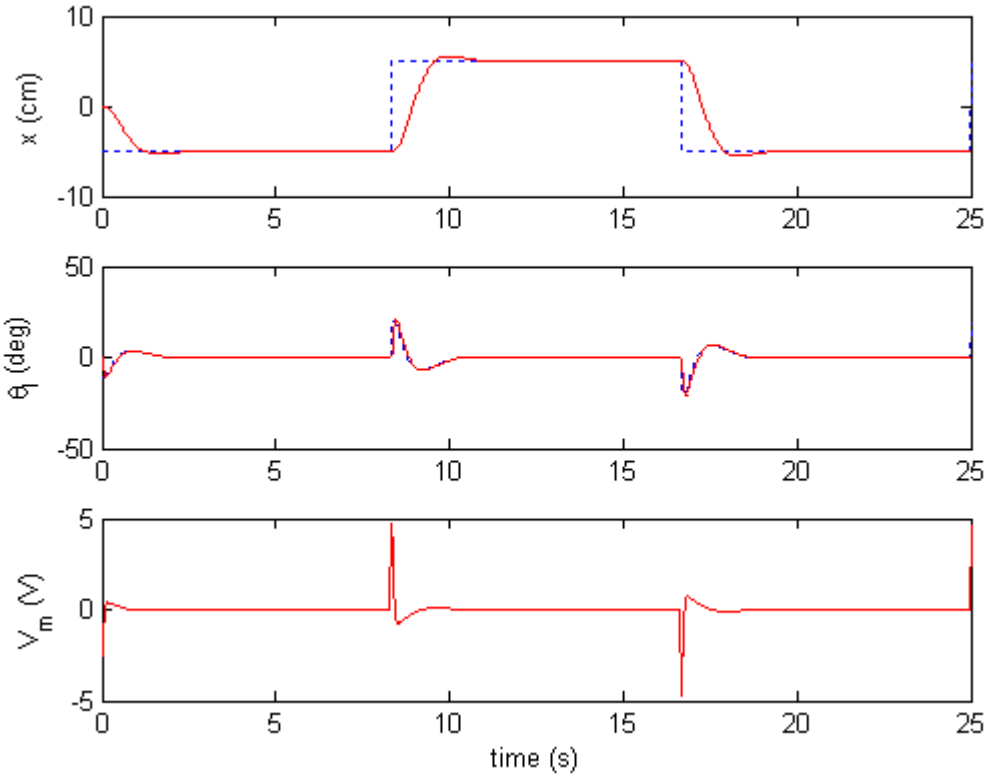

Figure 15: Simulated PD step response with b_{sd} = 0.

0	1	2

7. Measure the steady-state error, the settling time, and the percentage overshoot.

Solution:
The steady-state error, settling time, and percentage overshoot measured in the response shown in Figure 15 is

$$e_{ss} = -0.376\,10^{-5}\,[cm], \quad [s67]$$

$$t_s = 2.31\,[s] \quad [s68]$$

and

$$PO = 5.48\,["\%"]. \quad [s69]$$

Run the *meas_2dbb_specs.m* script after running *s_2dbb_pos* Simulink diagram to measure these specifications from a step response saved in *data_x* Matlab variable automatically.

0	1	2

8. Are the specifications in Section 4.2.1 satisfied? Also, make sure the servo angle is within ±30.0 degrees and the servo voltage is between ±10.0 V.

Solution:
The steady-state error, settling time, and percentage overshoot all meet the requirements without saturating the actuators. That is, by keeping the SRV02 angle between ±30 degrees and the voltage within ±10.0 V.

0	1	2

9. Take a look at the *X-Y Figure* scope. It should be a square as depicted in Figure 16. The blue plot is the desired position and the green trace is the simulated position. For the most part, because there is not steady-state error, both the desired and simulated traces are directly on top

of each other. However, the overshoot in the transient response is apparent.

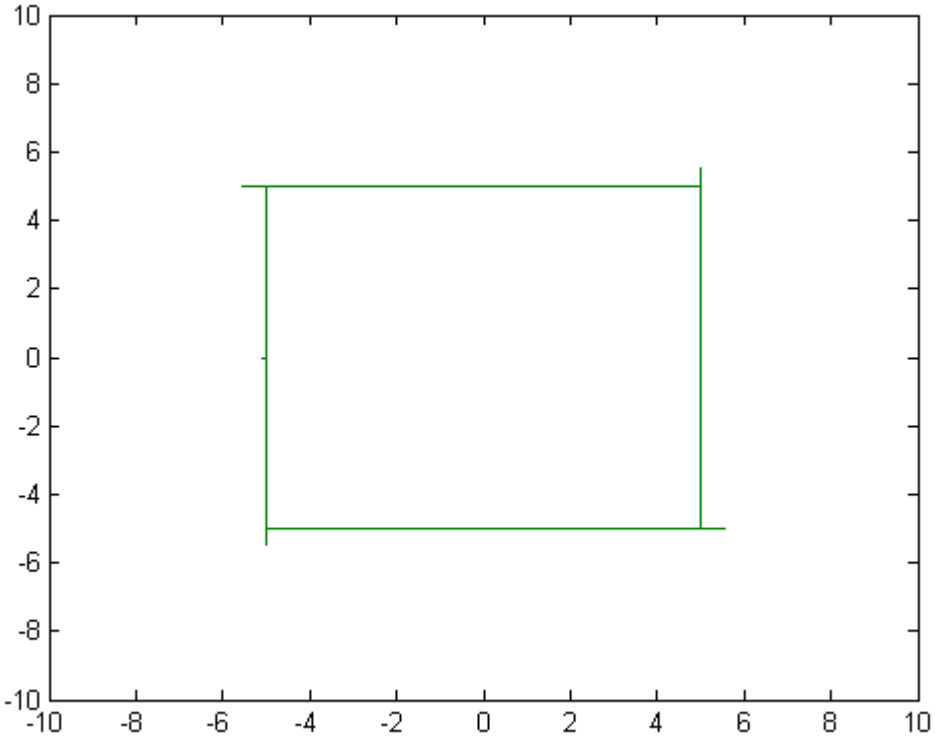

Figure 16: Sample PD square response when $b_{sd} = 0$.

10. For the x-axis only, vary the velocity set-point weight parameter, b_{sd}, between 0 and 1. Attach a Matlab figure of the response that represents the effect of the velocity set-point weight and comment on its effect.

Solution:
The closed-loop step position response when using the PD control with b_{sd} =1 is depicted in Figure 17. This is generated using the *meas_2dbb_specs.m* script. To use this script, do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, and *Tp* = Inf.
2. Run the *s_srv02_2dbb* Simulink model with *bsd* = 1 in both PID X and Y control systems.
3. Run the *meas_2dbb_specs.m* script.

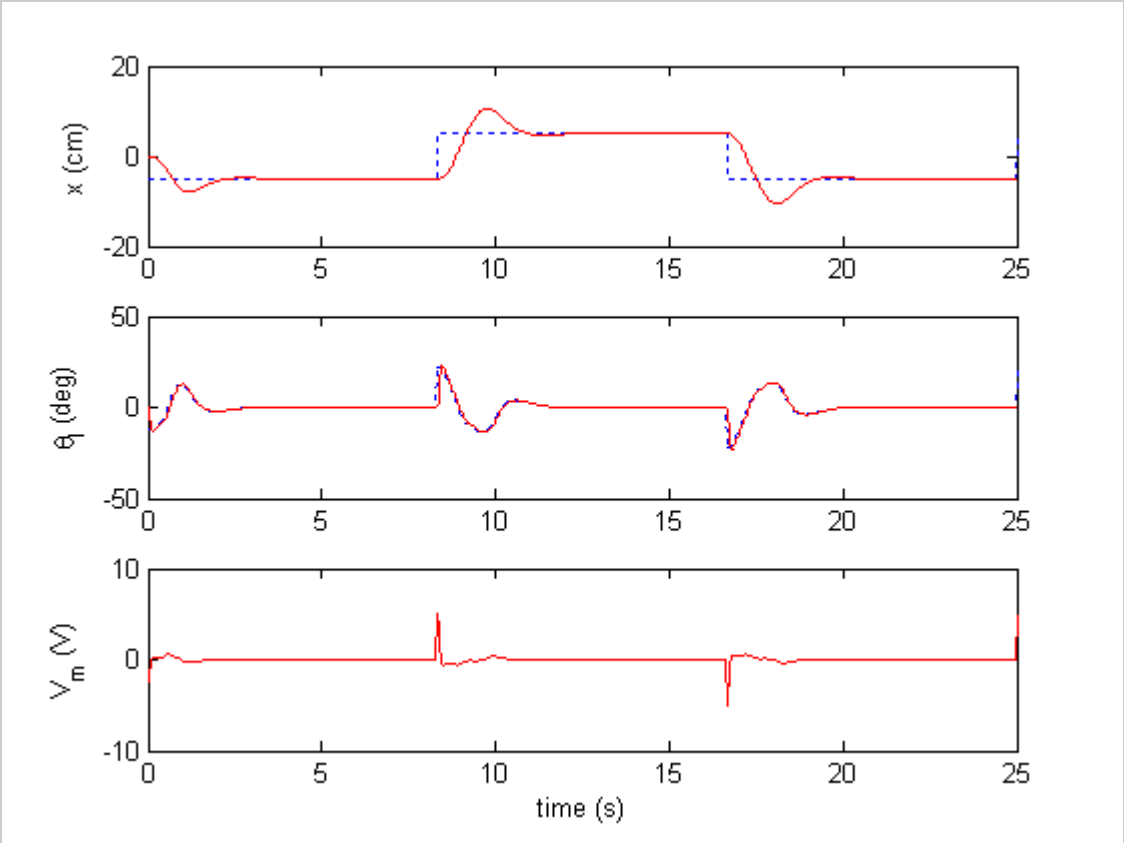

Figure 17: Simulated PD step response with b_{sd} = 1.

Compare the response with b_{sd}=1 in Figure 17 with the when b_{sd} = 0 in Figure 15. As the velocity set-point weight parameter is increased, the overshoot of the response also increases. In this case, the percentage overshoot becomes 55.4%. The settling time is increased slightly to 2.55 seconds, and the steady-state error is still zero.

0	1	2

5.1.1.3. Ramp PD Response

The response when the closed-loop PD system is subject to a ramp setpoint is now simulated. For the

ramp response, the steady-state error specification is the main concern. The desired trajectory in the *x-y* plane is a diamond.

Follow these steps to simulate the PD ramp response:

1. Go through the steps in Section 5.1.1.1 to setup the Matlab workspace for PD control.
2. In the *Setpoints* subsystem, select *triangle* in the *Signal Type* field of both the *SRV02 Signal Generator X* and *SRV02 Signal Generator Y* in order to generate a ramp reference.
3. Set both the *Amplitude X (cm)* and *Amplitude Y (cm)* slider gain blocks to 5 to generate a ramp with an amplitude of 5.0 centimeters.
4. Set the velocity set-point weight to zero, i.e. set the *bsd* Slider Gain block in both the *Cascade Control\2DBB PID Position Control: X* and *Cascade Control\2DBB PID Position Control: Y* subsystems to 0.
5. Start the simulation. By default, the simulation runs for 25.0 seconds. The scopes should be displaying responses similar to figures 18, 19, and 20.

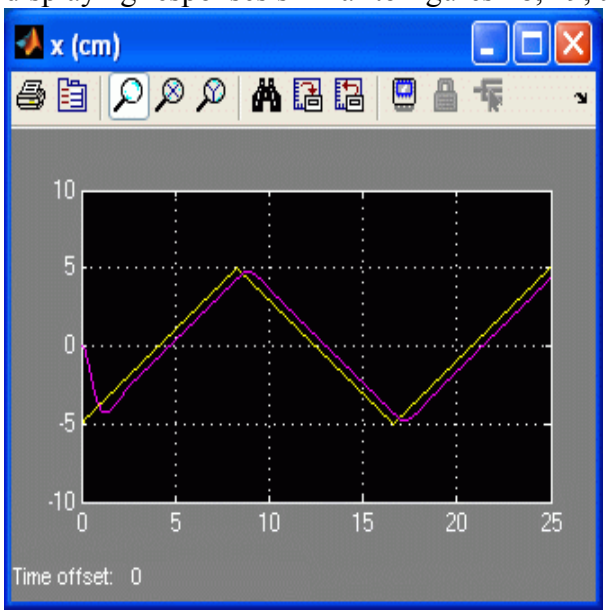

Figure 18: Ramp PD ball position response.

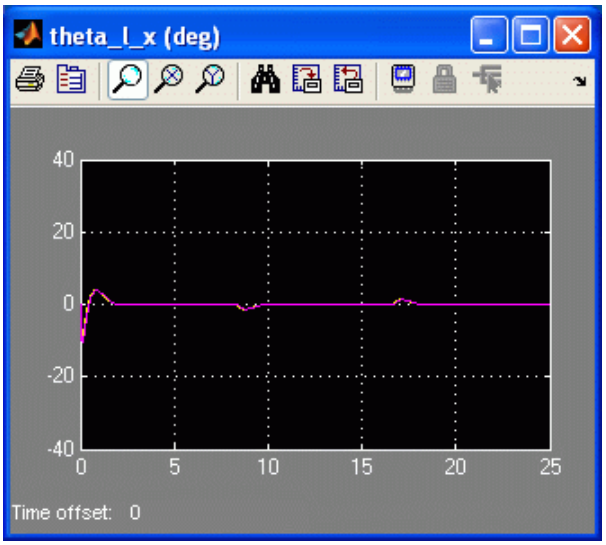

Figure 19: Ramp PD servo angle response.

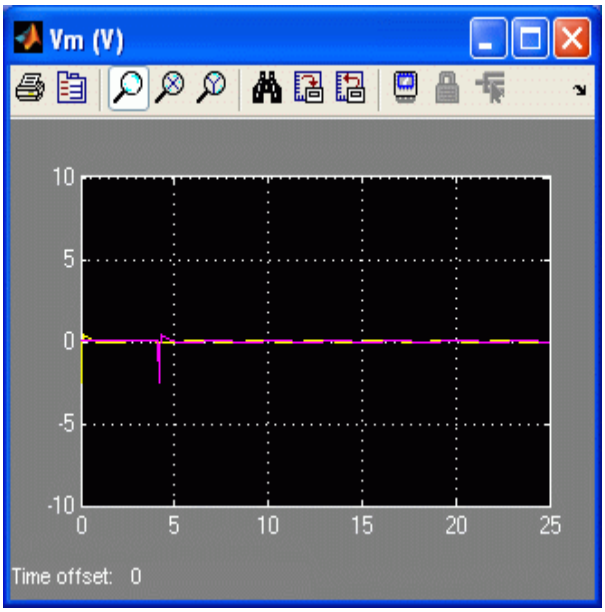

Figure 20: Ramp PD input voltage (both x- and y-axes).

6. Generate a Matlab figure showing the ramp x-axis PD ball position, servo angle, and servo input voltage response and attach it to your report. As explained earlier, the response from each scope is saved to a Matlab variable after each simulation run.

Solution:
The closed-loop ramp position response when using the PD control is depicted in Figure 21. This is generated using the *meas_2dbb_specs.m* script. To use this script, do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, and *Tp* = Inf.
2. Run the *s_srv02_2dbb* Simulink model *bsd* = 0 in both PID X and Y control systems.
3. Run the *meas_2dbb_specs.m* script.

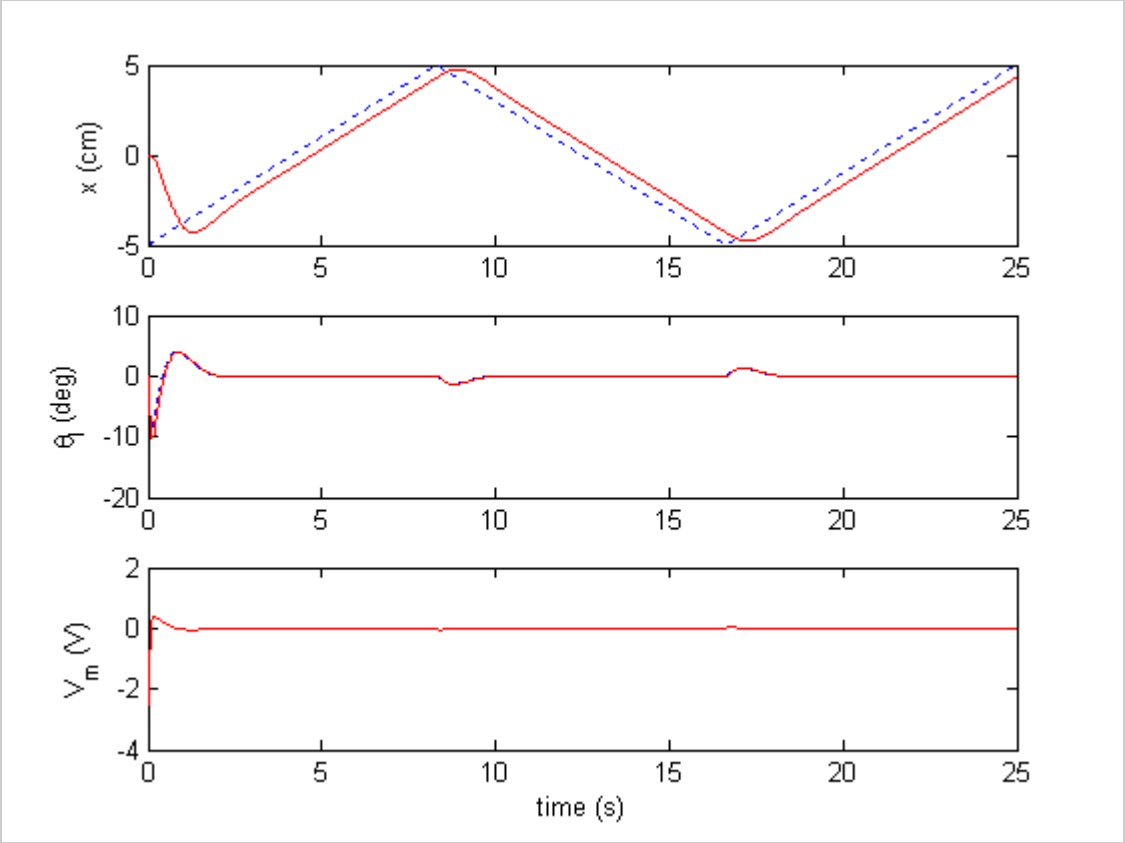

Figure 21: Simulated PD ramp response with b_{sd} = 0.

0	1	2

7. Measure the PD ramp steady-state error and compare it with the error calculated in pre-lab Section 4.3.3.
   Hint: Use the *ginput* Matlab command to take measurements from a Matlab figure.

Solution:
The steady-state error measured in ramp response shown in Figure 21 is

$$e_{ss} = 0.698\,[cm]$$ [s70]

The steady-state error computed in the pre-lab is 0.696 cm. Thus the theoretical and simulated values are very close to each other.

Run the *meas_2dbb_specs.m* script after running *s_2dbb_pos* Simulink diagram to measure the steady-state error specification from a ramp response saved in *data_x* Matlab variable automatically (ignore the peak time, settling time, and overshoot measurements).

0	1	2

8. Now run the simulation with the velocity set-point weight calculated in Exercise 5 of Section 4.3.3. Attach the Matlab figure of the response to your report.

Solution:

The closed-loop PD ramp position response when using the b_{sd} = 0.86 is depicted in Figure 21. This is generated using the *meas_2dbb_specs.m* script. To use this script, do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, and *Tp* = Inf.
2. Run the *s_srv02_2dbb* Simulink model with *bsd* = 0.86 in both PID X and Y control systems.
3. Run the *meas_2dbb_specs.m* script.

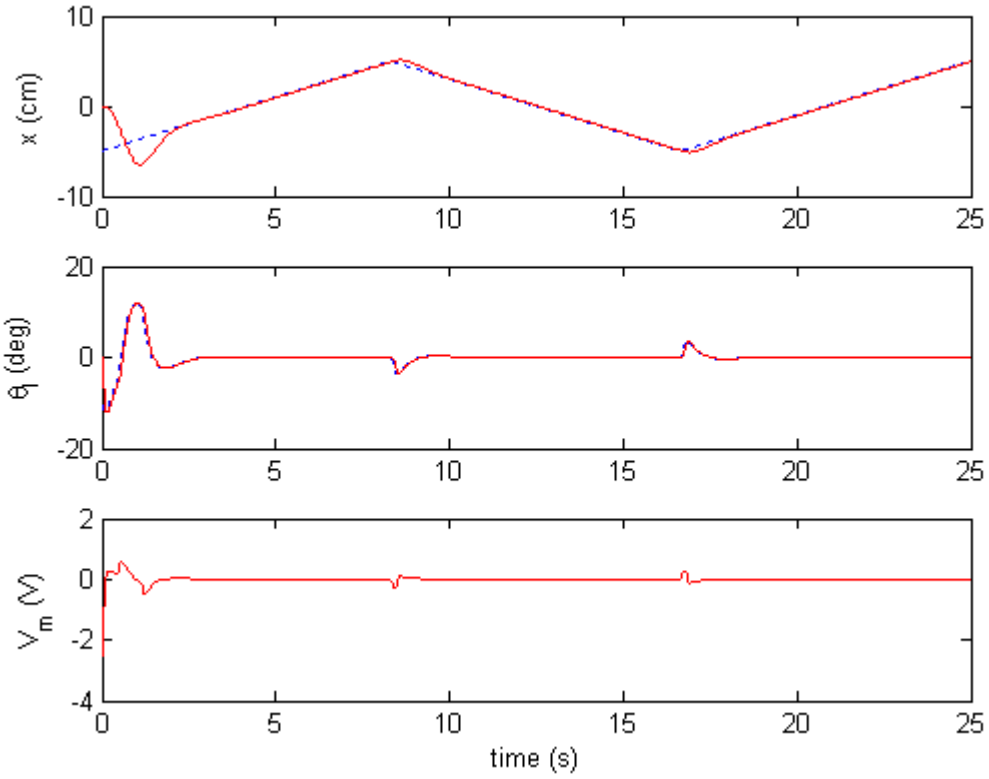

Figure 22: Simulated PD ramp response with b_{sd} = 0.86.

0	1	2

9. Are the specifications in Section 4.2.1 satisfied? Also, make sure the servo angle is within ±30.0 degrees and the servo voltage is between ±10.0 V.

Solution:
The steady-state error of the response is 0.1 cm when using the set-point weight calculated in the pre-lab. Thus the specification is satisfied while keeping both the SRV02 angle between ±30 degrees and the voltage between ±10.0 V.

0	1	2

10. Open the *X-Y Figure* scope and run the simulation. The desired blue trace and simulated green plot should be similar to Figure 23.

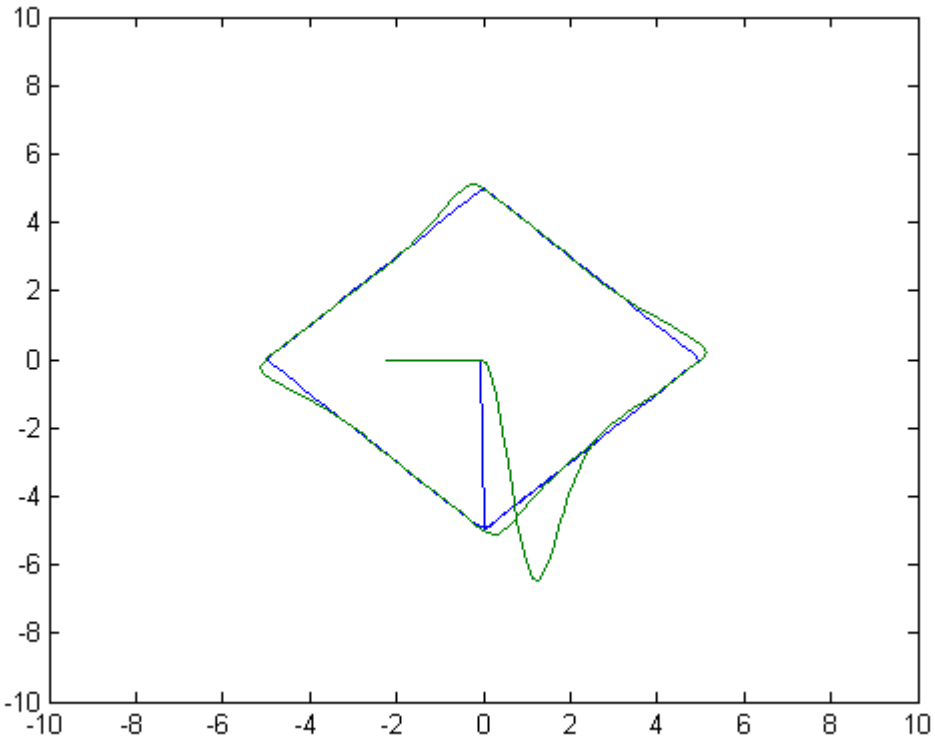

Figure 23: PD diamond response in X-Y Figure with adjusted b_{sd}.

5.1.2. PID Simulation

The use of integration with this system is tricky because it can lead to instability. This begins with undesirable oscillations in the response. However, as will be found integral control is necessary in certain cases to obtain adequate tracking performance when controlling the ball on the actual 2D Ball Balancer. When used properly, it is a simple and effective method to compensate for the un-modeled aspects of the system, such as friction.

Follow the procedure in Section 5.1.2.1 to setup the Matlab workspace for the PID control simulation. Then, go through the steps in Section 5.1.2.2 to simulate the step PID response.

5.1.2.1. Setup for PID Simulation

Follow these steps to configure the lab:

1. Setup the simulation as dictated in steps 1-7 in Section 5.1.1.1.

   Solution:
   Set CONTROL_TYPE = 'AUTO', Tp = 1.0, Tr = 5.0, and INT_MAX = 2.5 *pi / 180 to automatically calculate the PID gains according to the specifications.

   The students should not have access to the scripts d_pv_design.m, d_2dbb_model_param.m, d_2dbb_specs.m, plot_2dbb_rsp.m, and d_2dbb_pid_design.m described in Table 1. However, exactly what should be given to the students is at the discretion of the instructor. For example, it may be desired to supply d_pv_design to automatically calculate the SRV02 PV gains.

2. Enter the 2DBB PID gains that were found in Section 4.3.2.3 as variables *kp_bb*, *ki_bb*, and *kd_bb*.
3. Enter the low-gear SRV02 model gain, *K*, and the model time constant, *tau*, in Matlab found in Section 4.3.1.
4. Enter the SRV02 PV gains: called variables *kp* and *kv* in Matlab found in Section 4.3.1.
5. For the x-axis, open the ball position scope *x (m)*, the load shaft position scope *theta_l_x (deg)*, and the SRV02 motor input voltage scope *Vm(V)*.
6. For the y-axis, open the ball position scope *y (m)* and the load shaft position scope *theta_l_y (deg)*.
7. Open the *X-Y Figure* scope. This displays both the desired and simulated *x* and *y* positions.

5.1.2.2. Step PID Response

Recall that in the PID control design of Section 4.3.2, the 2DBB closed-loop system does not match the prototype third-order system (unlike the closed-loop PD transfer function when $b_{sd} = 0$, which matched its second-order prototype system). As a result, the response obtained using the designed gains may not yield a response that satisfies the specifications. Some tuning may be required.

Follow these steps to simulate the PID step response:

1. Go through the steps in Section 5.1.2.1 to setup the Matlab workspace.
2. Setup the signal generators to generate a 5.0 by 5.0 cm^2 square setpoint, as described by steps 2-3 in Section 5.1.1.2.
3. When performing position control, adding an integrator increases the speed of the response. Using a velocity set-point weight such as $b_{sd} = 1$ when a step is applied will cause too much overshoot. Therefore set $b_{sd} = 0$ for both the X and Y controllers (as explained in Step 4 of Section 5.1.1.2).
4. In the script, set INT_MAX = 2.5*pi/180. The default maximum integral windup is 2.5 degrees. In effect, the integral control cannot rotate the load gear of the servos by more then this amount.
5. Ensure the integrator reset time is set to the default value of Tr = 5.0.
6. Save and run the setup_srv02_exp17_2dbb.m script again to update the Matlab workspace with any parameter changes.
7. Start the simulation.
8. Generate a Matlab figure showing the PID step ball position, servo angle, and servo input voltage response and attach it to your report. As previously explained, the response from each scope is saved to a Matlab variable after each simulation run.

Solution:
The closed-loop step position response when using the PID control with the default integral windup limit and the default reset time is depicted in Figure 24. This is generated using the *meas_2dbb_specs.m* script. To obtain this response, do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, *Tp* = 1.0, INT_MAX = 2.5*pi/180, *Tr* = 5.0.
2. Run the *s_srv02_2dbb* Simulink model with the bsd = 0 in both PID X and Y control systems.
3. Run the *meas_2dbb_specs.m* script.

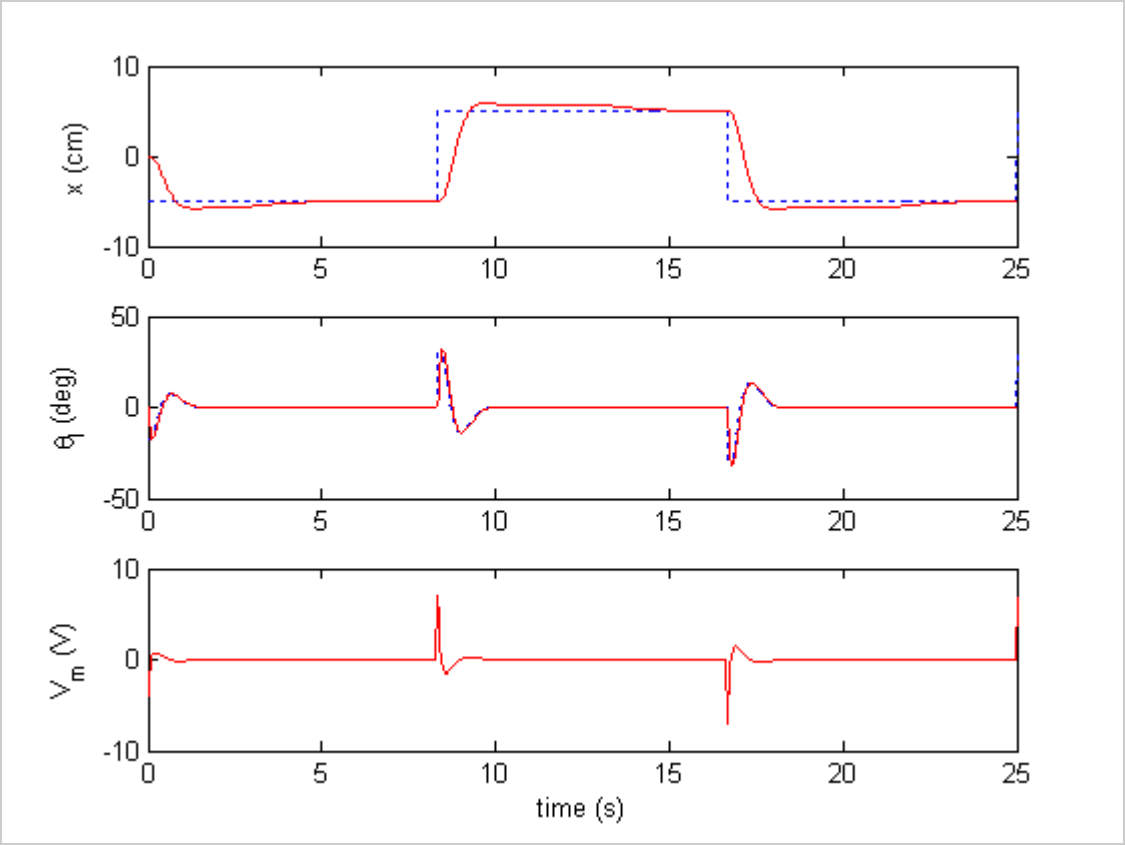

Figure 24: Simulated PID step response with INT_MAX = 2.5 deg and Tr = 5.0 s.

0	1	2

9. Measure the steady-state error, the settling time, and the percentage overshoot.

Solution:
The steady-state error, settling time, and percentage overshoot measured in the response shown in Figure 24 is

$$e_{ss} = -0.0152\,[cm], \quad \text{[s71]}$$

$$t_s = 6.02\,[s], \quad \text{[s72]}$$

and

$$PO = 8.28\,[\text{"\%"}]. \quad \text{[s73]}$$

Run the *meas_2dbb_specs.m* script after running *s_2dbb_pos* Simulink diagram to measure these specifications from a step response saved in *data_x* Matlab variable automatically.

0	1	2

10. Are the specifications in Section 4.2.1 satisfied? Is the servo angle kept within ±30.0 degrees and the servo voltage between ±10.0 V? Provide an explanation if an element in the response is not satisfied.

Solution:
The steady-state error meets the specification. The settling time and the percentage overshoot specifications are, however, not satisfied with these controller settings.

The servo angle does get slightly saturated when the step rises and causes the integrator to become saturated. Due to the slow reset time, the integrator has a chance to windup and causes the response to take longer to settle about the setpoint (until the integrator input is decreased enough).

0	1	2

11. Run the simulation with different integrator reset times and examine the changes in the response. Attach a Matlab figure showing the response when $T_r = 0.001$ seconds. Comment on

why this response is different then when T_r = 5.0 seconds.

Solution:

The closed-loop step position response when using the PID control with the default integral windup limit and with a reset time of Tr = 0.001 seconds is depicted in Figure 24. This is generated using the *meas_2dbb_specs.m* script. To use this script, do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, *Tp* = 1.0, INT_MAX = 2.5*pi/180, *Tr* = 0.001.
2. Run the *s_srv02_2dbb* Simulink model with *bsd* = 0 in both PID X and Y control systems.
3. Run the *meas_2dbb_specs.m* script.

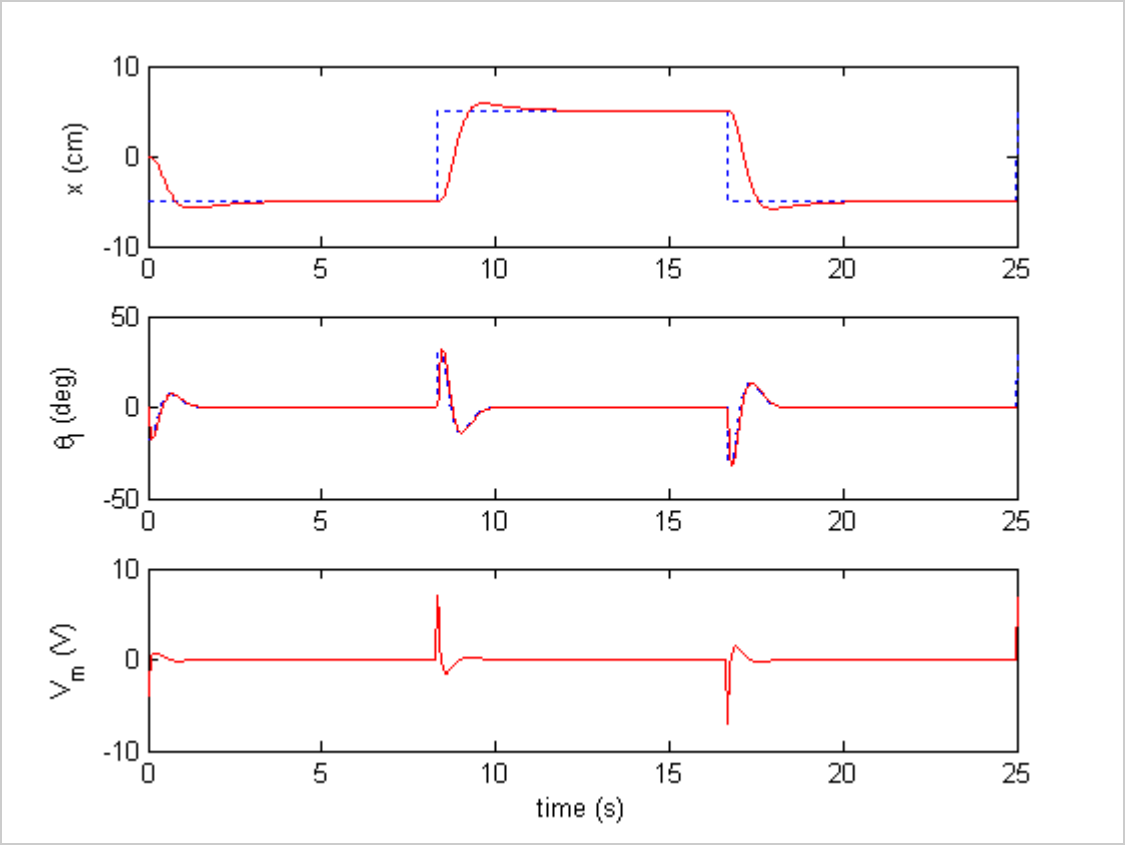

Figure 25: Simulated PID step response with INT_MAX = 2.5 deg and Tr = 0.001 s.

Decreasing the anti-windup loop time constant makes the integrator reset much faster compared to when T_r = 5.0 seconds. The integrator does not have a chance to windup and the resulting response has a lower settling time, i.e. the response settles to its steady-state value more quickly.

0	1	2

12. Vary INT_MAX in the script between 0 and 10.0 degrees and examine the changes in the response. Tune the INT_MAX parameter to obtain a response that satisfies the specifications. Generate a Matlab figure showing the *tuned windup PID step* ball position, servo angle, and

servo input voltage response and attach it to your report.

Solution:

The closed-loop step position response when using the PID control with a tuned integral saturation parameter is depicted in Figure 26. This is generated using the *meas_2dbb_specs.m* script. To use this script, do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, *Tp* = Inf, INT_MAX = 1.6*pi/180, and Tr = 0.001 seconds.
2. Run the *s_srv02_2dbb* Simulink model with the bsd = 0 in both PID X and Y control systems.
3. Run the *meas_2dbb_specs.m* script.

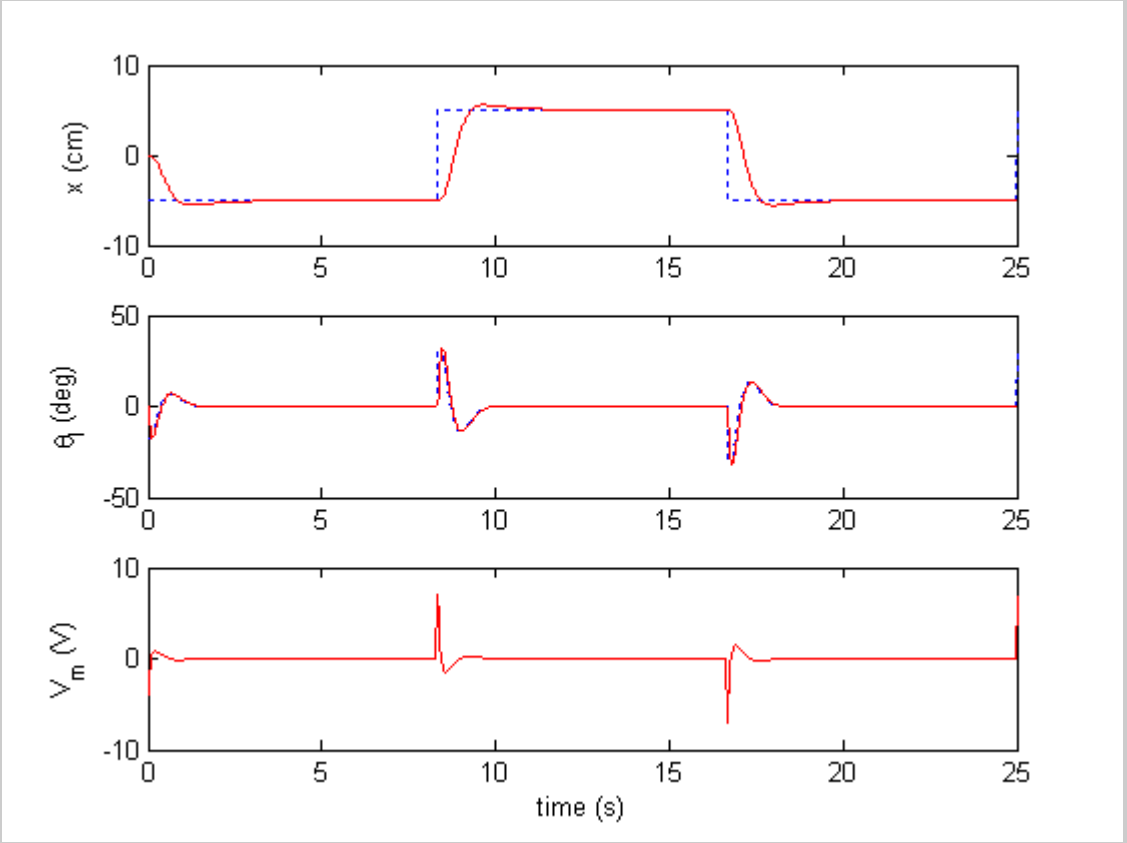

Figure 26: Simulated PID step response with INT_MAX = 1.6 deg and Tr = 0.001 s.

13. List the steady-state error, settling time, and percentage overshoot of this response using the newly tuned PID compensator.

Solution:
The steady-state error, settling time, and percentage overshoot measured in the response shown in Figure 26 is

$$e_{ss} = -0.000223[cm], \quad [s74]$$

$$t_s = 2.47[s] \quad [s75]$$

and

$$PO = 5.66["\%"]. \quad [s76]$$

Run the *meas_2dbb_specs.m* script after running *s_2dbb_pos* Simulink diagram to measure these specifications from a step response saved in *data_x* Matlab variable automatically.

0	1	2

14. Are the specifications in Section 4.2.1 satisfied? Is the servo angle is within ±30.0 degrees and the servo voltage is between ±10.0 V? Recall that the servo angle is an imposed limit.

Solution:
The steady-state error, settling time, and the percentage overshoot specifications are satisfied by properly tuning the windup parameters. The servo angle does, however, become saturated at the imposed 30.0 degree limit but this does not cause any instability or adversely effect since the windup is properly tuned. The servo motor is not saturated.

0	1	2

15. Take a look at the X-Y Figure scope. It should be a square as depicted in Figure 27. The blue plot is the desired X-Y position and the green trace is the simulated position. For the most part because there is no steady-state error, the desired and simulated traces are directly on top of

each other but the overshoot in the transient response is captured.

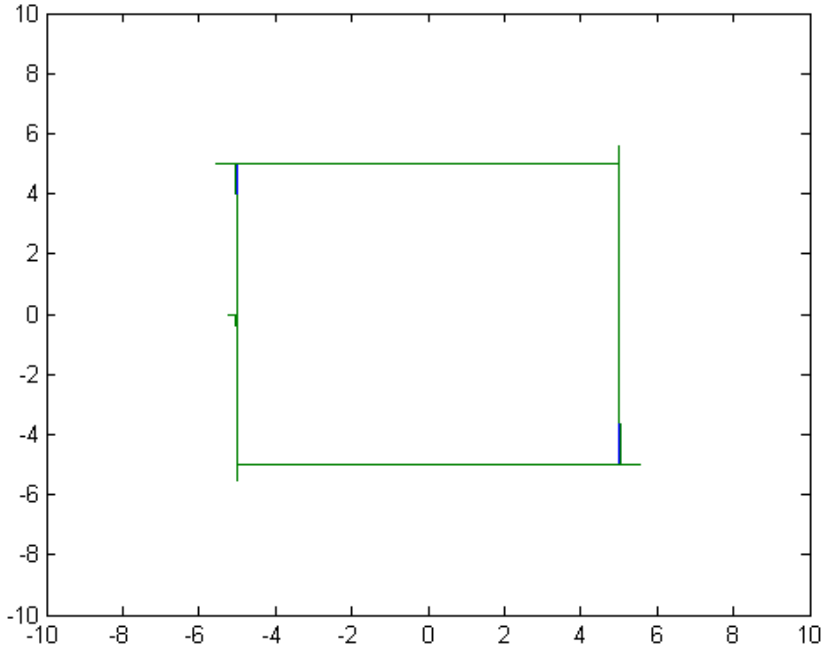

Figure 27: PID square response with tuned windup parameters.

5.2. Position Control Implementation

The *q_2dbb_pos* Simulink diagram shown in Figure 28 is used with QUARC to control the ball position on the Quanser 2D Ball Balancer device. The *SRV02-ET+2DBB* subsystem contains QUARC blocks that interface with the DC motor and sensors of the 2D Ball Balancer system. The *Setpoints* and *Cascade Control: X and Y* subsystems are the same as used in the Simulink diagram *s_2dbb_pos.mdl* for the simulations.

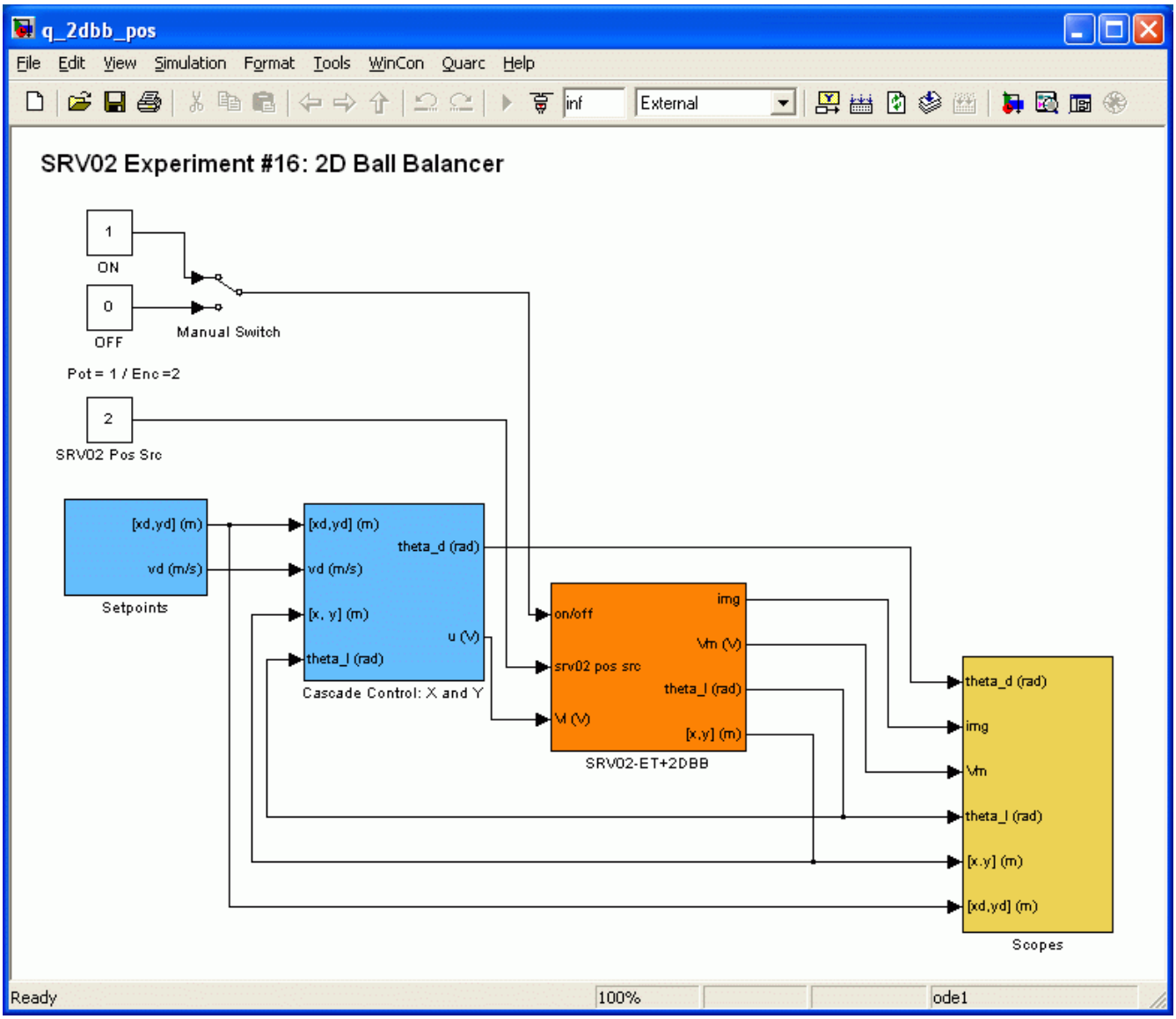

Figure 28: Simulink model used with QUARC to run the PID controller on the 2D Ball Balancer system.

5.2.1. Setup for Position Control Implementation

Before beginning the in-lab exercises on the 2D Ball Balancer, the q_2dbb_pos Simulink diagram and the setup_srv02_exp17_2dbb.m script must be configured.

Follow these steps to get the system ready for this lab:

1. Setup the SRV02 with the 2DBB module as detailed in Reference [10.
2. Load the Matlab software.
3. Browse through the *Current Directory* window in Matlab and find the folder that contains the QUARC 2DBB control file *q_2dbb_pos.mdl*.
4. Double-click on the *q_2dbb_pos.mdl* file to open the 2D Ball Balancer Position Control

Simulink diagram shown in Figure 28.

Solution:
Open the *q_2dbb_pos_soln.mdl* Simulink diagram which includes the completed *Camera Calibration* subsystem. This block is empty in *q_2dbb_pos.mdl*.

5. **Configure DAQ**: Ensure the *HIL Initialize* block in the *SRV02-ET+2DBB* subsystem is configured for the DAQ device that is installed in your system. By default, the block is setup for the Quanser Q8 hardware-in-the-loop board. See Reference [6] for more information on configuring the *HIL Initialize* block.
6. **Configure Sensor**: The position of the load shaft can be measured using various sensors. Set the *Pos Src* Source block in q_2dbb_pos.mdl, as shown in Figure 28, as follows:
   - 1 to use the potentiometer
   - 2 to use to the encoder

   Note that when using the potentiometer, there will be a discontinuity.
7. **Configure setup script**: Set the parameters in the setup_srv02_exp17_2dbb.m script according to your system setup. See Section 5.1.1.1 for more details.

5.2.2. QUARC Camera Block Settings

Before doing any control, the QUARC camera blocks must be properly configured according to how it was calibrated in Reference [10. Tests will then be made to ensure the *x* and *y* ball planar positions are being measured as discussed in Section 4.4.

Follow the steps below:

1. Make sure the 2DBB system is setup and the camera is calibrated as explained in Reference [10.

2. **Ensure the FlyCap software is NOT running. Otherwise an error will occur.**
3. Follow steps 1-6 and 11-13 in Section 5.1.1.1 for *q_2dbb_pos* to setup the Matlab workspace and open the appropriate scopes.
4. The *Image Pos Left*, *Image Pos Top*, *Image Width*, and *Image Height* properties of the *PGR Gab Image* block in the *SRV02-ET+2DBB* subsystem must be configured according to how the camera was calibrated. Go to the *Camera Configuration* section of the *setup_srv02_exp17_2dbb* script, shown in Text 2, and set the *cam_res*, *img_pos_left*, and *img_pos_top* parameters to the values attained in the camera calibration procedure in Reference [10.

```
% Resolution (pixels): determines width and height of image
cam_res = 440;
% Image Pos Left (pixels 0-752)
img_pos_left = 100;
% Image Pos Left (pixels 0-752)
img_pos_top = 40;
% RGB Values of ball
img_R = 250;
img_G = 125;
img_B = 50;
% Threshold (%)
img_threshold = 50;
% Minimum object size
min_object_size = 500;
```

Text 2: "Camera Configuration" section in the setup_srv02_exp17_2dbb.m script.

5. The RGB parameters of the object must be set in the *Find Object* block. Go to the *Camera Configuration* section of the *setup_srv02_exp17_ 2dbb* script, as shown in Text 2, above, and set the *img_R*, *img_G*, and *img_B* variables according to the object RGB values found in the camera calibration procedure in Reference [10.
6. Run the script again to load the *PGR Grab Image* and *Find Object* block settings.
7. By default the *Threshold* and *Minimum Object Size* parameters in the *Find Object* block are set to 50 and 500, respectively. This is done through the *threshold* and *min_object_size* variables found in *Camera Configuration* section of the *setup_srv02_exp17_ 2dbb* script.

CAUTION: These may have to change according to the object being used and the lighting of the environment. For example, if an orange ball is not being used then change the RGB setting accordingly or if there is a lot of noise in the measured signal then decrease the *Threshold* parameter.

8. The *Camera Calibration* subsystem is left blank. Add blocks to this system in order to implement the equations found in Section 4.4 that extrapolate the (x,y) ball position from the raw camera pixel output (p_x, p_y). Copy and paste the contents of the subsystem to your report.

Solution:
The subsystem depicted in Figure 29 implements equations [s62] and [s63] in Section 4.4 such that the ball position is read relative to the $[X_b, Y_b]$ coordinate system shown in Figure 10.

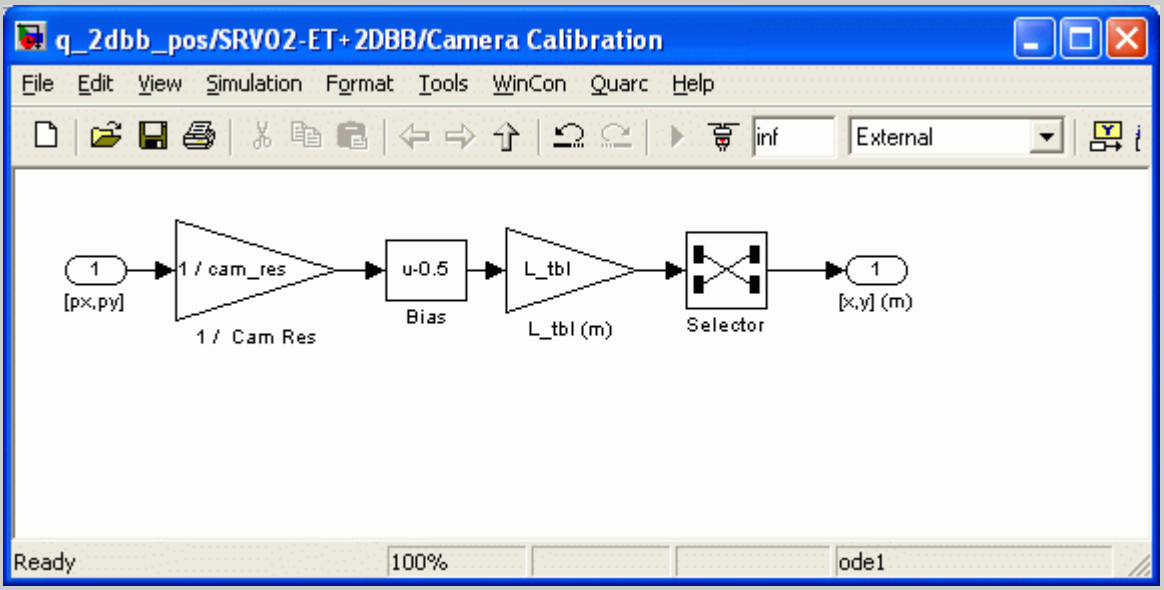

Figure 29: Camera calibration subsystem.

This subsystem is given in the *q_2dbb_pos_soln.mdl* Simulink diagram.

0	1	2

9. Click on QUARC | Build to compile the Simulink diagram.
10. In the *q_2dbb_pos* Simulink model, set the Manual Switch to the OFF position. This deactivates the PID control by feeding zero voltage to each servo.
11. Select QUARC | Start to begin running the controller.
12. Place the ball on the plate. The scopes should begin displaying position measurements.
13. Manually move the ball horizontally from left to right and examine the response in the *x (m)* scope. Is the measurement direction correct? Measure the distance the ball is moved with a ruler and compare it with the measurement in the scope. Attach a Matlab figure showing this response and comment on the accuracy of the measurement. Keep in mind the image position resolution computed in Exercise 2 of Section 4.4.

Solution:
The response when moving the ping pong ball along the x-axis in the positive direction (i.e. left to right when facing the experiment) is shown in Figure 30.

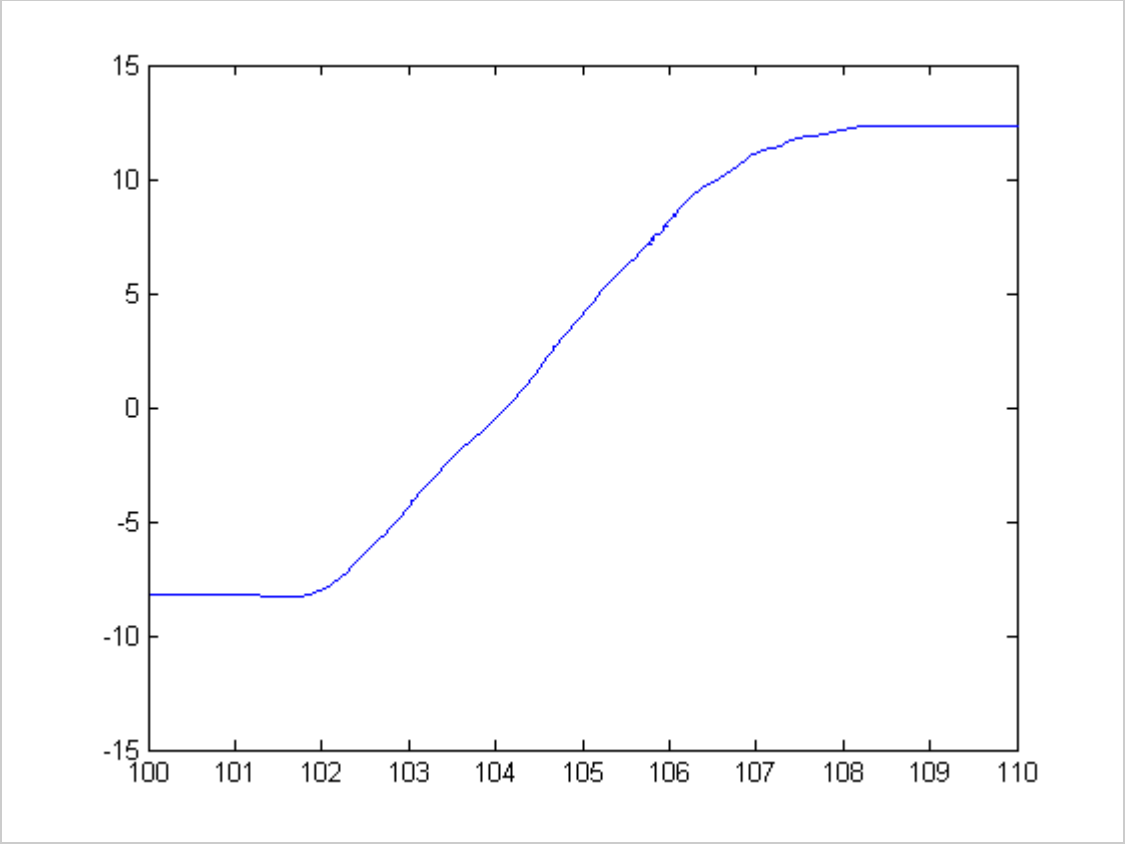

Figure 30: Sample measurement when moving the ball along the x-axis.

The ball was moved along a ruler a distance of

$$\Delta x_{ruler} = 20.32\,[cm] \qquad \text{[s77]}$$

and the displacement measured from the obtained response using the camera is

$$\Delta x = 20.52\,[cm] \qquad \text{[s78]}$$

The difference between the ruler measurement and the camera reading is

$$e_x = 0.20\,[cm] \qquad \text{[s79]}$$

This is good given the image position resolution in [s66].

0	1	2

14. Do the same in the vertical direction. Attach a Matlab figure showing the response and comment on the measurement accuracy.

Solution:
The response when moving the ping pong ball along the y-axis in the negative direction (i.e. top to bottom when facing the experiment) is shown in Figure 30.

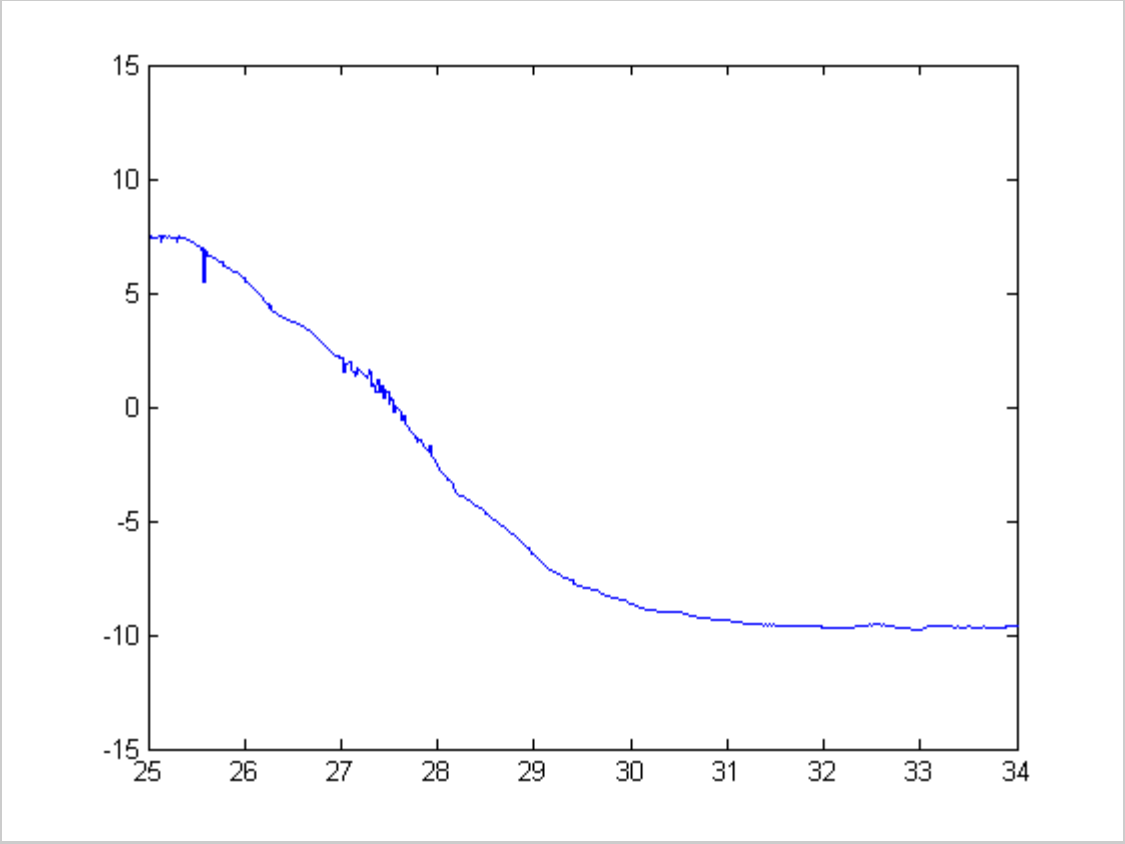

Figure 31: Sample measurement when moving the ball along the y-axis.

The ball was moved along a ruler a distance of

$$\Delta y_{ruler} = 17.78\,[cm] \qquad [s80]$$

and the displacement measured from the obtained response using the camera is

$$\Delta y = 16.81\,[cm] \qquad [s81]$$

The measurement discrepancy along the y-axis is

$$e_y = 0.97\,[cm] \qquad [s82]$$

This is still under the image position resolution calculated in [s66]. The camera measurement is therefore reasonably accurate.

0	1	2

15. If the ball position is being measured correctly then click on the Stop button to cease running the QUARC controller.

5.2.3. Step Response using PD Control

In this lab, the ball on the 2DBB device is to track a step reference using the developed proportional-derivative controller. Measurements will be taken to see if the specifications are satisfied.

Follow the steps below:

1. Go through the steps in Section 5.1.1.1 for *q_2dbb_pos* to setup the Matlab workspace for the PD controller.
2. Make sure the camera parameters *cam_res*, *img_pos_left*, and *img_pos_top* in the *setup_srv02_exp17_ 2dbb* script are set properly as dictated in Section 5.2.2.
3. In the *Setpoints* subsystem, select *square* in the *Signal Type* field of both the *SRV02 Signal Generator X* and *SRV02 Signal Generator Y* in order to generate a step reference.
4. Place the ball approximately in the center of the plate and adjust the servo load gears so the ball remains in place.
5. Power up the amplifiers.
6. Make sure the *Manual Switch* in *q_2dbb_pos* is set to the ON position in order to enable the controller.
7. Set the setpoint to 0 for both axes. That is, set the *Amplitude X (cm)* and *Amplitude Y (cm)* slider gain blocks to 0.
8. Click on QUARC | Build to compile the Simulink diagram.
9. Select QUARC | Start to begin running the controller. The servos should adjust to stabilize the ball to the center of the plate.
10. Set the *Amplitude X (cm)* slider gain block to 5 to generate a step with an amplitude of 5.0 centimeters in the x-axis. The ball should begin going back-and-forth along the x-direction (i.e. moving horizontally when facing the experiment) and the scopes should be displaying responses similar to figures 32, 33, 34, 35, and 36.

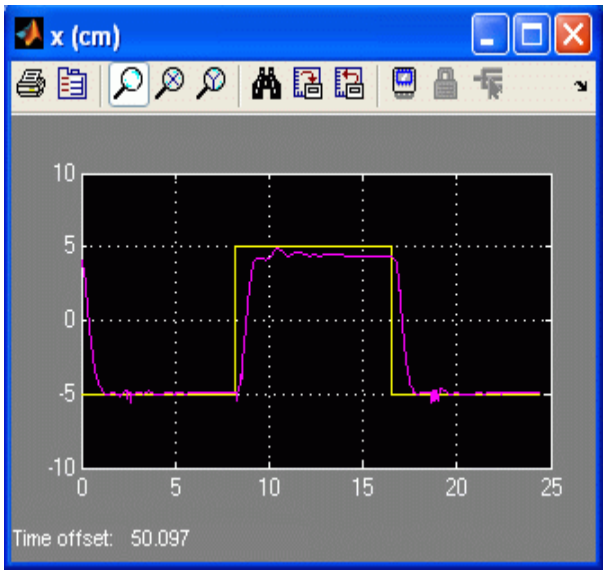

Figure 32: PD step x position response.

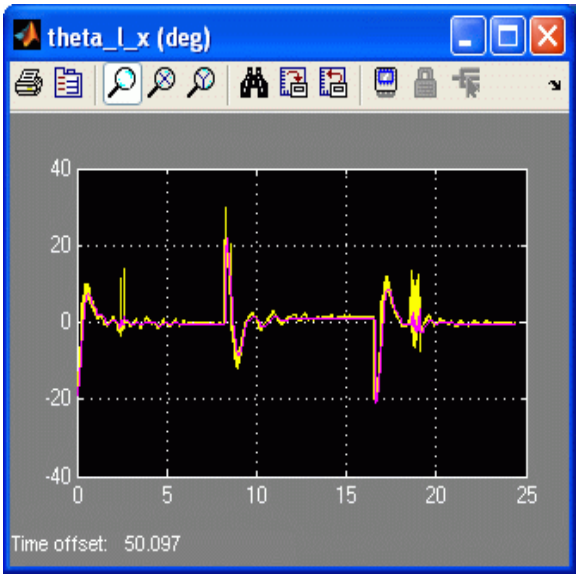

Figure 33: PD step load gear x angle response.

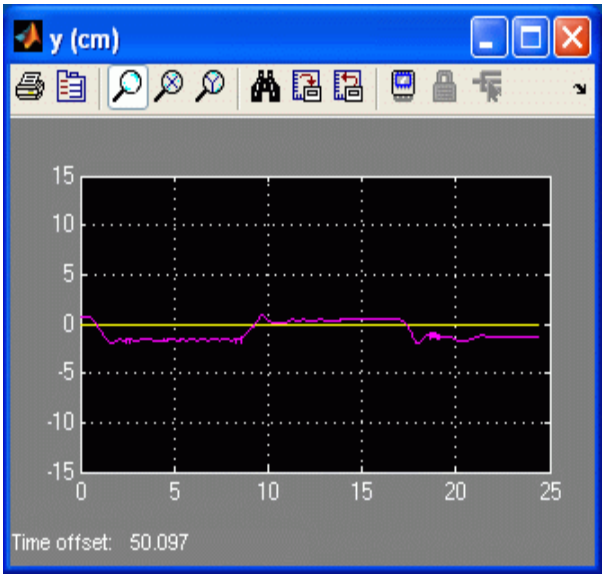

Figure 34: PD step y position response.

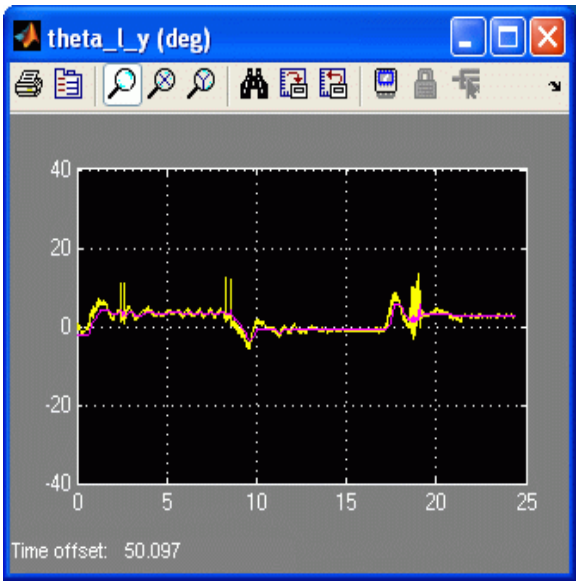

Figure 35: PD step load gear Y angle response.

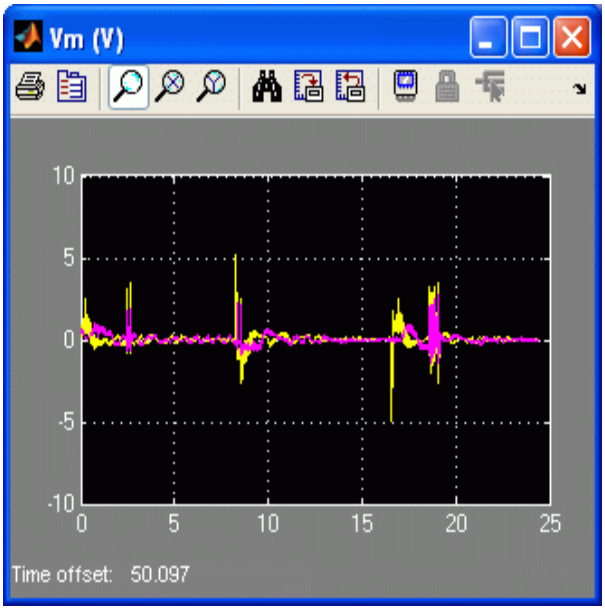

Figure 36: PD step input motor voltage.

11. When a suitable response is obtained, click on the *Stop* button in the Simulink diagram tool bar (or select QUARC | Stop from the menu) to stop running the code. Generate a Matlab figure showing the x-axis ball position, servo angle response, and input voltage. Attach it to your report. Recall that each scope automatically saves their response to a variable in the Matlab workspace when the controller is stopped.

Solution:
The measured 2D Ball Balancer closed-loop position step response when using the PD gains found in Section 4.3.2.2 is shown in Figure 37. To generate this response, execute the *meas_2dbb_specs.m* script with the saved MAT files *sample_rsp_pd_step_x.mat*, *sample_rsp_pd_step_theta_l_x.mat*, and *sample_rsp_pd_vm.mat*. Alternatively, to generate a Matlab figure from a new experimental run do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, and *Tp* = Inf.
2. Run the *q_2dbb_pos_soln* Simulink model with *bsd* = 0 in both PID X and Y control systems.
3. Stop QUARC.
4. Run the *meas_2dbb_specs.m* script.

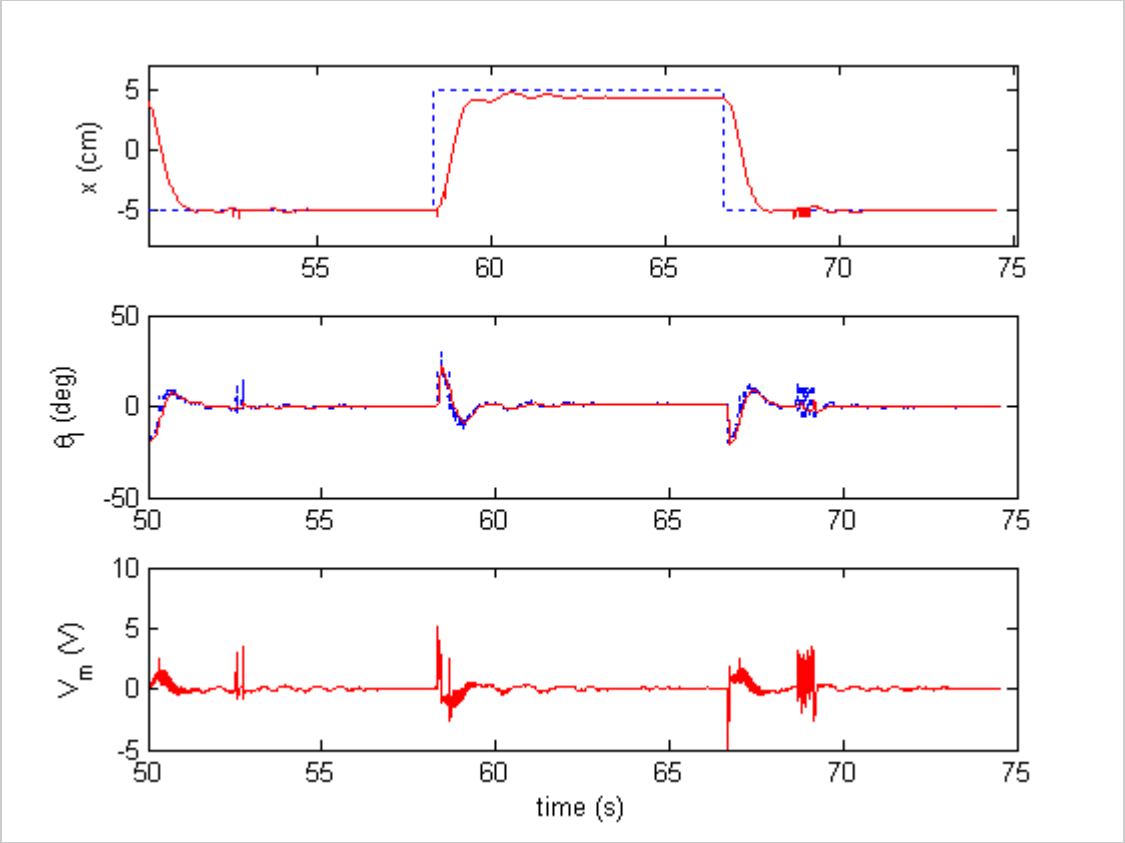

Figure 37: Measured 2DBB PD step response along x-axis .

0	1	2

12. Measure the steady-state error, the settling time, and the percentage overshoot. Does the response satisfy the specifications given in Section 4.2.1?

Solution:
The steady-state error measured in Figure 37 at (before the step goes down for another cycle) is

$$e_{ss} = 0.629\,[cm] \quad \text{[s84]}$$

The settling time and percentage overshoot of the response shown in Figure 37 is

$$t_s = 1.11\,[s] \quad \text{[s85]}$$

and

$$PO = 5.1\,[\text{"\%"}] \quad \text{[s86]}$$

The steady-state error specification is not satisfied because it exceeds the 0.1 cm requirement. The settling time and percentage overshoot meet the requirements.

To find the steady-state error, settling time, and percentage overshoot of a response saved in *data_x* automatically, run the *meas_2dbb_specs.m* script after running *q_2dbb_pos_soln* or using the responses saved in the MAT files: *sample_rsp_pd_step_x.mat*, *sample_rsp_pd_step_theta_l_x.mat*, and *sample_rsp_pd_step_vm.mat*.

0	1	2

13. Generate a Matlab figure showing the y-axis ball position, servo angle response, and input voltage. Attach it to your report. Although the setpoint is only in the *x* direction, notice that the y-axis servo is still performing control. Explain why this occurs. Does the model developed in Section 4.1.3 take this into account?

Solution:
The position response along the y-axis when using the PD controller is shown in Figure 37. To generate this response, execute the *meas_2dbb_specs.m* script with the saved MAT files *sample_rsp_pd_step_y.mat*, *sample_rsp_pd_step_theta_l_y.mat*, and *sample_rsp_pd_vm(:,3).mat*.

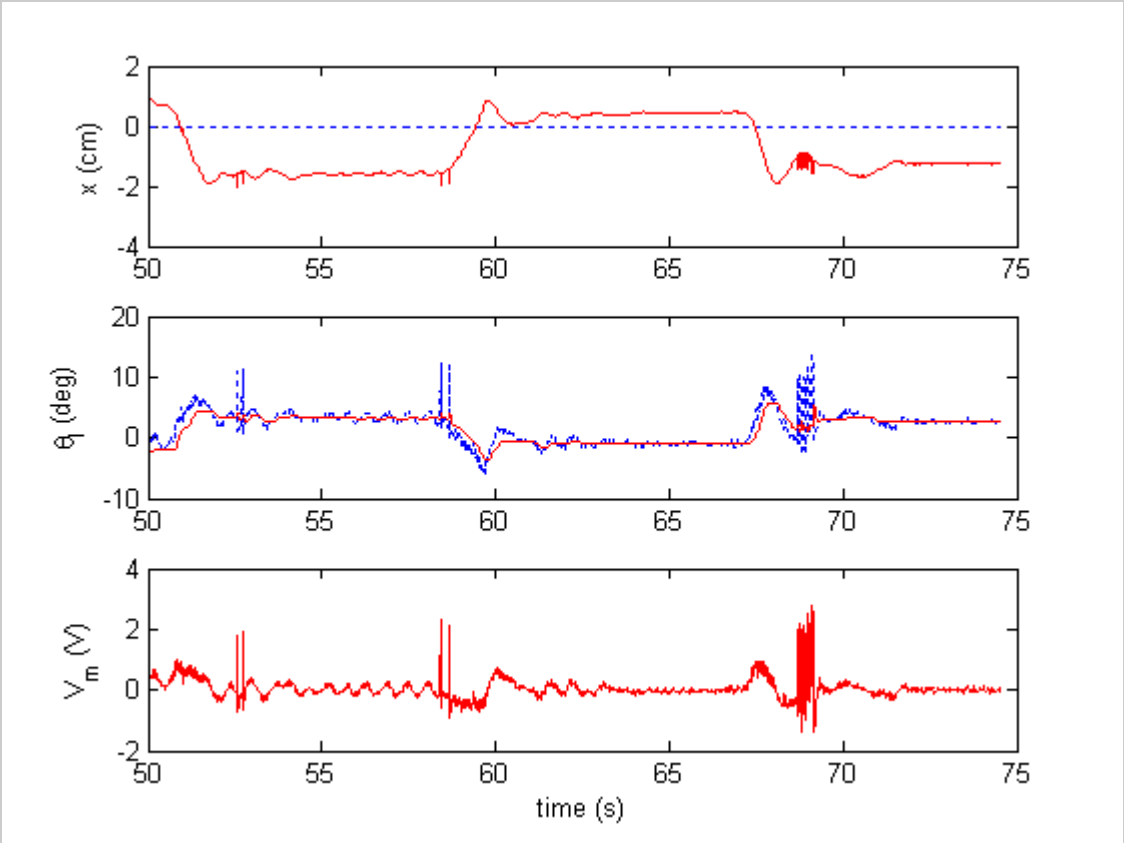

Figure 38: Measured 2DBB PD step response along y-axis.

Contrary to what is assumed in the model, the plate on the actual device is not perfectly balanced. The gimbals on each servo have 2 DOF. As a result, when the x-axis servo rotates to control the position of the ball in the *x* direction, the angle of the plate in the *y* direction will vary as well. In the simulation, as the ball is being controlled in the x-axis it does not move in the *y* direction. Therefore there is some coupling between the two axes in the actual device.

0	1	2

14. Give another unmodeled effect that may contribute to the response not behaving as expected.

Solution:
There friction present in the gimbals, the servo gears, and between the ball and the plate. This is not a characteristic that is modeled. In particular due to stiction, it takes a certain minimum amount of voltage to get the load gear in motion as well as the ball in motion. In effect, friction introduces a nonlinear deadband effect into the system.

0	1	2

15. Set the *Amplitude Y (cm)* slider gain block to 5 to generate a step with an amplitude of 5.0 centimeters in the y-axis. This is now a planar position control problem as the ball has to track the steps in both the x and y directions simultaneously. Attach a Matlab figure depicting the position response in the x-axis and y-axis along with the accompanying input voltages. Compare the *x* position response when running in dual-axis mode with the previous response, that is, when the ball was only controlled along the x-axis.

Solution:
The response obtained when commanding a step reference of 5.0 cm on both the x and y axes with the PD control is depicted in Figure 39. To generate this response, execute the *plot_2dbb_rsp.m* script with the saved MAT files *sample_rsp_pd_step_xy_x.mat*, *sample_rsp_pd_step_xy_theta_l_x.mat*, *sample_rsp_pd_step_xy_y.mat*, *sample_rsp_pd_step_xy_theta_l_y.mat*, and *sample_rsp_pd_xy_vm.mat*.

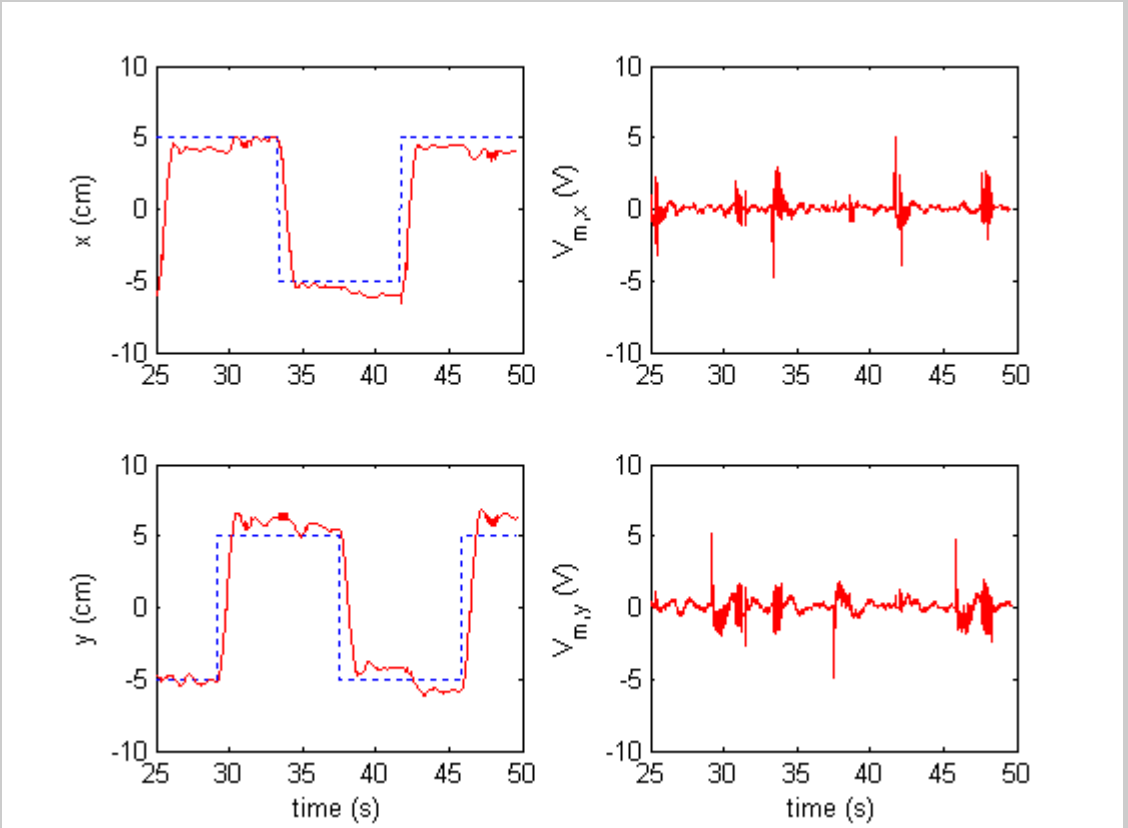

Figure 39: Measured 2DBB PD dual-axis step response.

As previously discussed, there is coupling between the two axes. Thus when the y-axis setpoint goes from -5.0 cm to 5.0 cm at the 29.1 second mark the response in x-axis plot also goes up, by at least 0.5 cm. Essentially each axis creates a disturbance on the other axis when tracking a setpoint.

0	1	2

16. As explained in Section 5.1.1, the setpoints are setup such that the ball tracks a square on the plate. Attach a sample response in the *X-Y Figure* when using the PD control.

Solution:
A sample square PD response is shown in Figure 40. Notice the steady-state error that was previously noticed as well as the coupling effect that occurs between the axes. Due to the large steady-state error, we cannot obtain a perfect square response with the PD control.

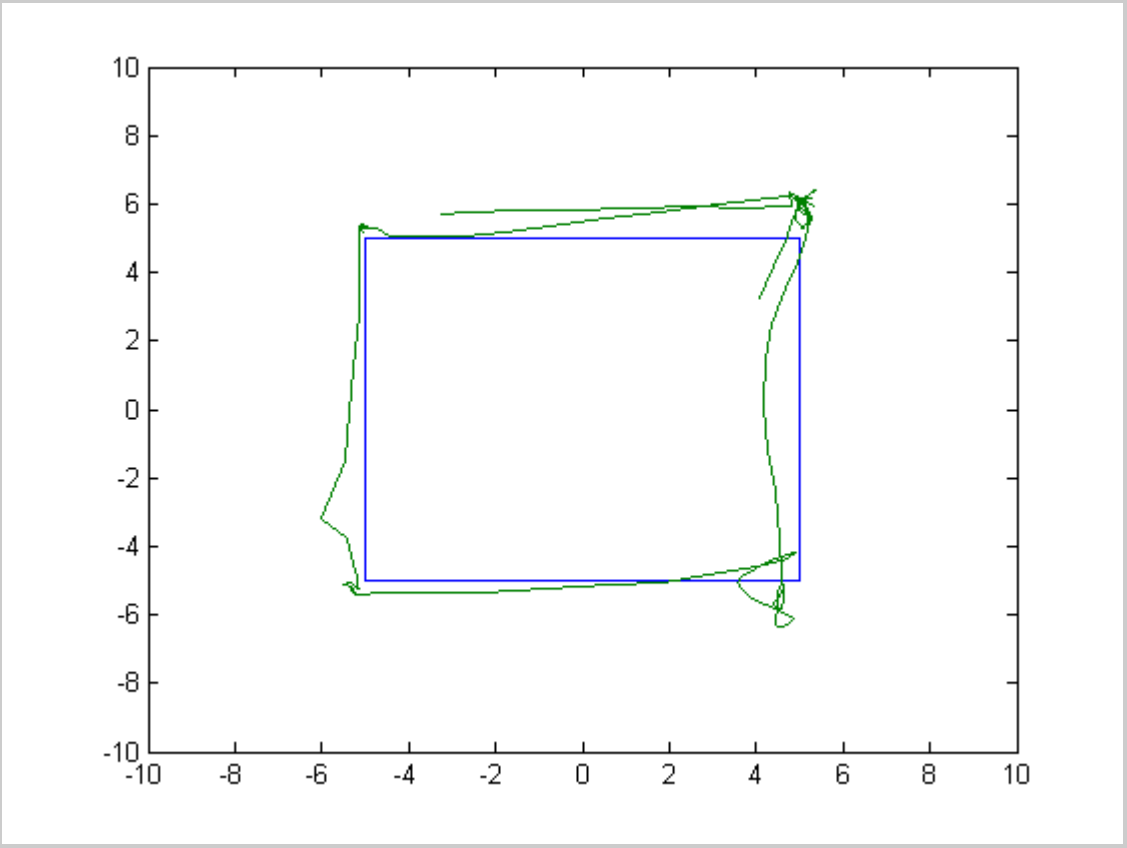

Figure 40: PD square response.

0 1 2

17. Make sure QUARC is stopped.
18. Shut off the amplifier power if no more experiments will be performed on the SRV02 in this session.

5.2.4. Step Response using the PID Control

The ball is now to track a step reference on the 2DBB device when using the developed proportional-integral-derivative control. Measurements will then be taken to see if the specifications are satisfied.

Follow the steps below:

1. Go through the steps in Section 5.1.2.1 for *q_2dbb_pos* to setup the Matlab workspace for the PID controller.
2. Enter the integrator windup parameter, INT_MAX, that was tuned in Section 5.1.2.2 to satisfy

the specifications.

3. Follow steps 2-9 in Section 5.2.3 to begin running the controller.
4. Set the *Amplitude X (cm)* slider gain block to 5 to generate a step with an amplitude of 5.0 centimeters in the x-axis. The ball should begin going back-and-forth along the x direction of the plate.
5. When a suitable response is obtained, click on the *Stop* button in the Simulink diagram tool bar (or select QUARC | Stop from the menu) to stop running the code. Generate a Matlab figure showing the x-axis ball position, servo angle response, and input voltage. Attach it to your report.

Solution:
The measured 2D Ball Balancer closed-loop x-axis position step response when using the PID gains found in Section 4.3.2.3 is shown in Figure 41. To generate this response, execute the *meas_2dbb_specs.m* script with the saved MAT files *sample_rsp_pid_step_x.mat*, *sample_rsp_pid_step_theta_l_x.mat*, and *sample_rsp_pid_vm.mat*. Alternatively, to generate a Matlab figure from a new experimental run do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, *Tp* = 1.0, *INT_MAX* = 1.6*pi/180, and *Tr* = 0.001.
2. Run the *q_2dbb_pos_soln* Simulink model with *bsd* = 0 in both PID X and Y control systems.
3. Stop Quarc.
4. Run the *meas_2dbb_specs.m* script.

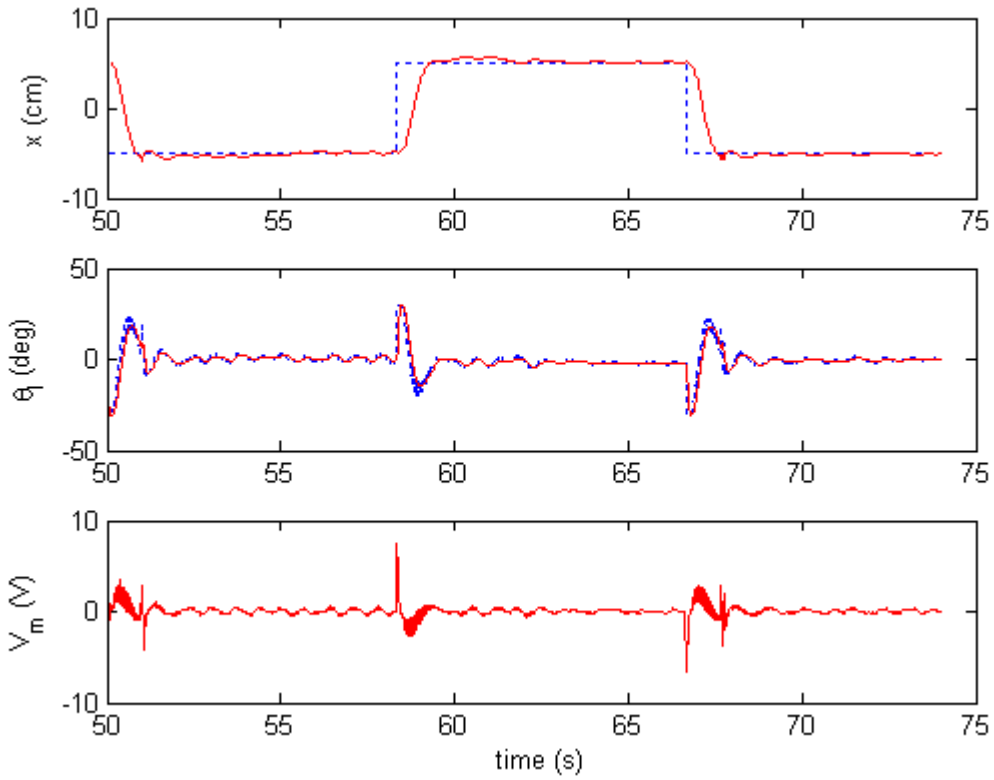

Figure 41: Measured 2DBB PID step response along x-axis .

0	1	2

6. Measure the steady-state error, the settling time, and the percentage overshoot. Does the response satisfy the specifications given in Section 4.2.1?

Solution:
The steady-state error measured in Figure 41 at (before the step goes down for another cycle) is

$$e_{ss} = -0.0962\,[cm] \quad \text{[s87]}$$

The settling time and percentage overshoot of the response shown in Figure 41 is

$$t_s = 3.31\,[s] \quad \text{[s88]}$$

and

$$PO = 6.57\,[\text{"\%"}] \quad \text{[s89]}$$

The steady-state error and percentage overshoot specifications are satisfied but the settling time does not.

To find the steady-state error, settling time, and percentage overshoot of a response saved in *data_x* automatically, run the *meas_2dbb_specs.m* script after running *q_2dbb_pos* or using the responses saved in the MAT files: *sample_rsp_pid_step_x.mat*, *sample_rsp_pid_step_theta_l_x.mat*, and *sample_rsp_pid_step_vm.mat*.

0	1	2

7. Compare the step PID response with the PD. Does it compensate for the unmodeled effects successfully?

Solution:
The steady-state error is greatly improved when using the PID. Thus it does compensate for some unmodeled effects in that respect.

However, perhaps because the plate is not perfectly balanced, the x-axis controller is still perturbed by the y-axis controller enough for the response to take longer to settle.

0	1	2

8. Tune the integrator saturation parameter INT_MAX and see if the PID step response can be improved. Describe its effect on the response when it is increased. To do this, set INT_MAX to a value in radians, such as INT_MAX = 5.0 * pi / 180 (i.e. 5 degrees) in the Matlab command prompt and click on *Edit* | *Update Diagram* in the Simulink model to apply the changes (or select the Simulink model and press CTRL-D).

Solution:
When increasing the INT_MAX parameter from 1.6 degrees to 10 degrees, the overshoot increases substantially and the settling time becomes longer. Thus when dealing with a step reference, it is best to keep the integrator saturation parameter conservative.

0	1	2

9. Reset the INT_MAX parameter to the original value (that was found in Section 5.1.2.2). Set the *Amplitude Y (cm)* slider gain block to 5 to generate a step with an amplitude of 5.0 centimeters in the y-axis. The ball should now be tracking a 5.0 x 5.0 cm^2 square. Attach a sample response in the *X-Y Figure* when using the PID control. Compare the square PID response with the PD one. Has the 2D tracking performance improved?

Solution:
A sample square PID response is shown in Figure 42. There is not much improvement over the PD control because the INT_MAX cannot be set high enough for the integrators to compensate for the coupling between the axes. If INT_MAX is set too high then there is too much overshoot.

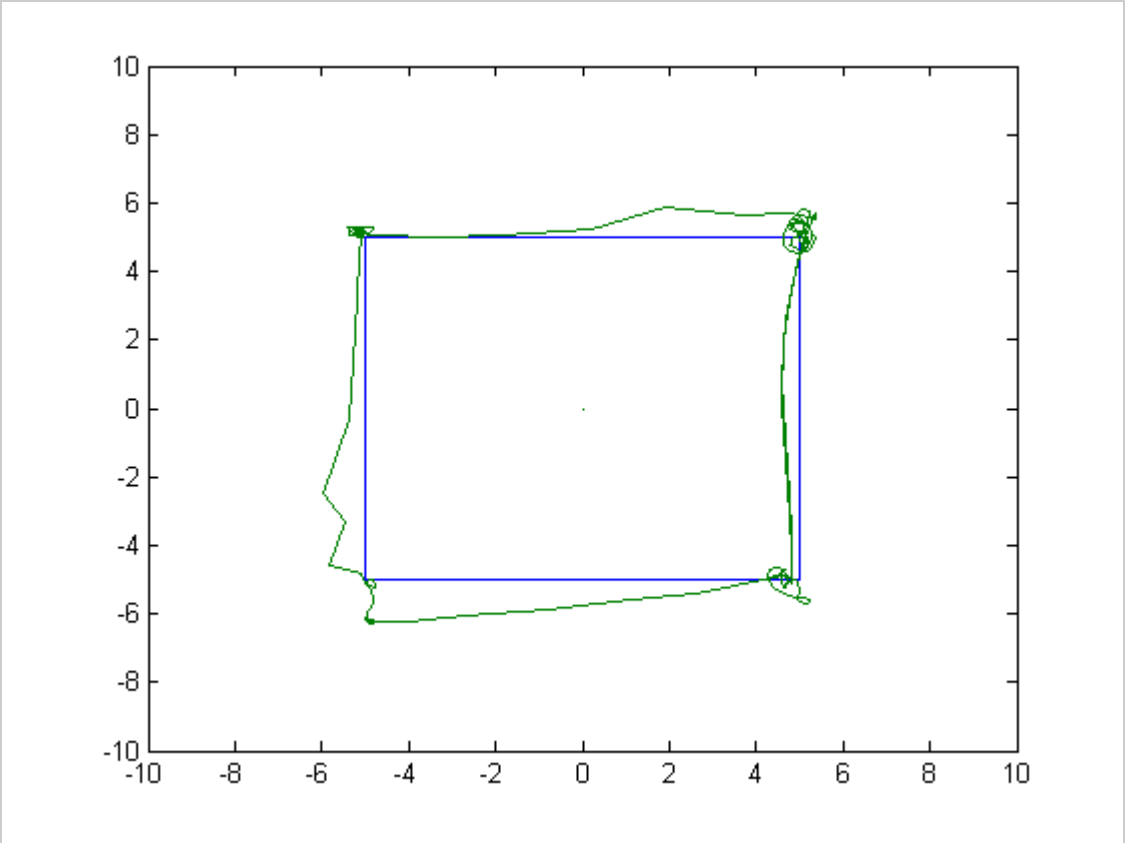

Figure 42: PID square response.

0	1	2

10. Make sure QUARC is stopped.
11. Shut off the amplifier power if no more experiments will be performed on the SRV02 in this session.

5.2.5. Ramp Response using PD and PID

In this section, the ramp response is evaluated when using the PD control with varying set-point weights and the PID control. The goal is for the ball to track a diamond shaped trajectory in the XY plane.

Follow the steps below:

1. Go through the steps 2-9 in Section 5.2.3 to being running the controller.

2. In the *Setpoints* subsystem, select triangle in the *Signal Type* field of both the *SRV02 Signal Generator X* and *SRV02 Signal Generator Y* in order to generate a ramp reference.
3. Make sure the set-point velocity weight is set to 0. Thus set the *bsd* Slider Gain block in both the *Cascade Control\2DBB PID Position Control: X* and *Cascade Control\2DBB PID Position Control: Y* subsystems to 0.
4. Set the *Amplitude X (cm)* and *Amplitude Y (cm)* slider gain blocks to 5 to generate a ramp with an amplitude of 5.0 centimeters in the x and y axes. The scopes should be displaying responses similar to figures 43, 44, 45, 46, and 47.

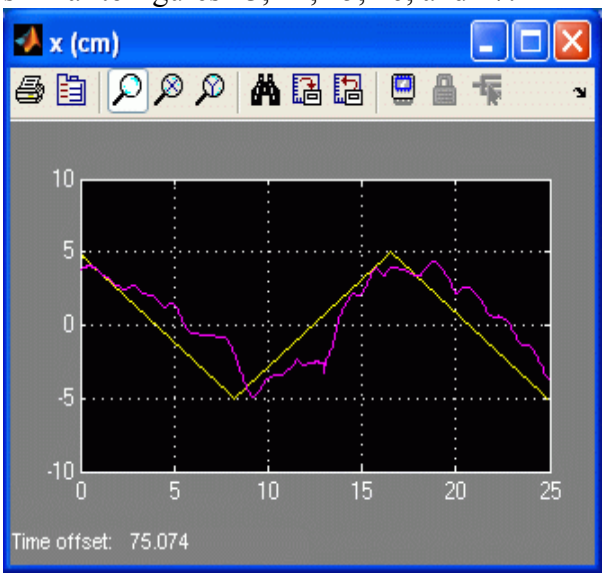

Figure 43: PD ramp x position response.

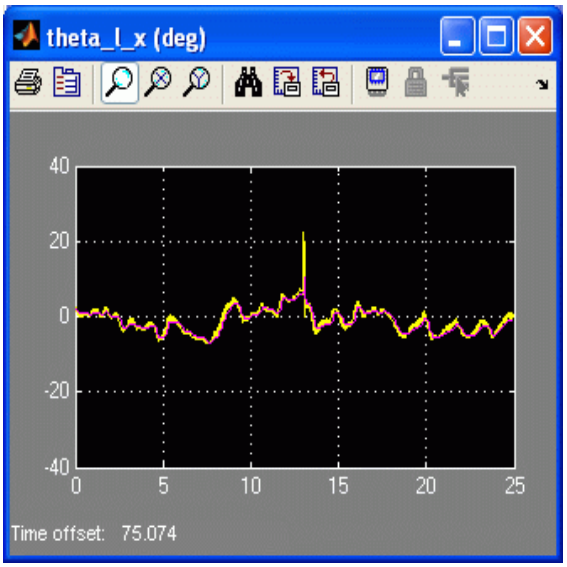

Figure 44: PD ramp load gear x angle response.

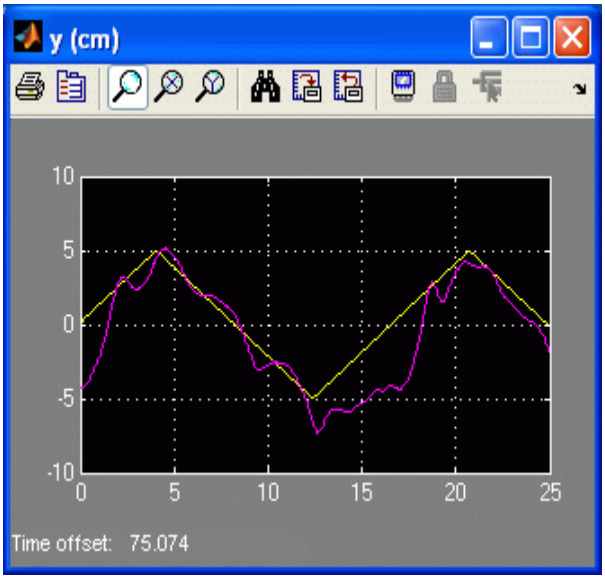

Figure 45: PD ramp y position response.

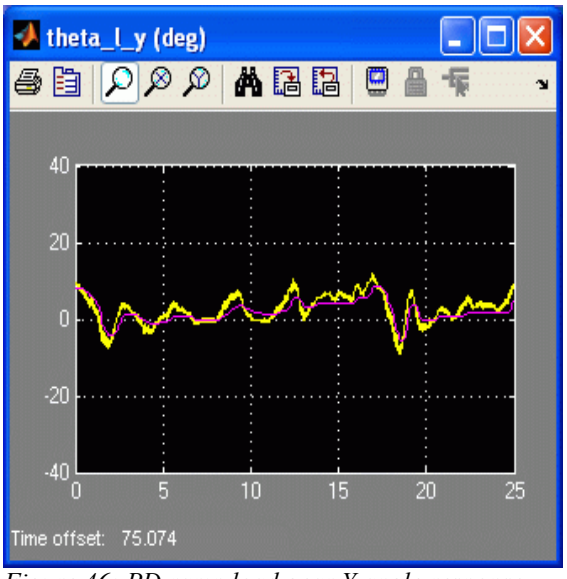

Figure 46: PD ramp load gear Y angle response.

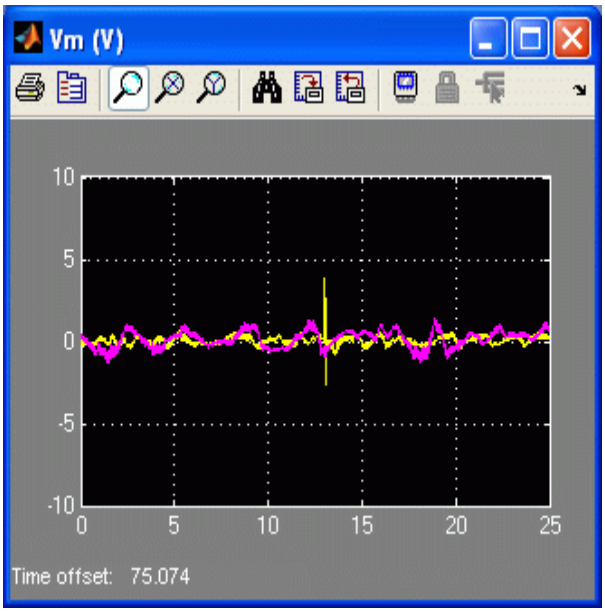

Figure 47: PD ramp input motor voltage.

5. When a full response is obtained, click on the *Stop* button in the Simulink diagram tool bar (or select QUARC | Stop from the menu) to stop running the code. Generate a Matlab figure showing the x and y ball positions and input voltage responses. Attach it to your report..

Solution:
The measured PD ramp response when the set-point velocity weight parameter is set to zero, i.e. $b_{sd} = 0$, is shown in Figure 48. To generate this response, execute the *plot_2dbb_rsp.m* script with the saved MAT files *sample_rsp_pd_ramp_bsp0_x.mat*, *sample_rsp_pd_ramp_bsp0_y.mat*, *sample_rsp_pd_ramp_bsp0_theta_l_x.mat*, *sample_rsp_pd_ramp_bsp0_theta_l_y.mat*, and *sample_rsp_pd_ramp_bsp0_vm.mat*. Alternatively, to generate a Matlab figure from a new experimental run do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, and *Tp* = 1.0.
2. Run the *q_2dbb_pos_soln* Simulink model with the bsd = 0 in both PID X and Y control systems.
3. Stop Quarc.
4. Run the *meas_2dbb_specs.m* script.

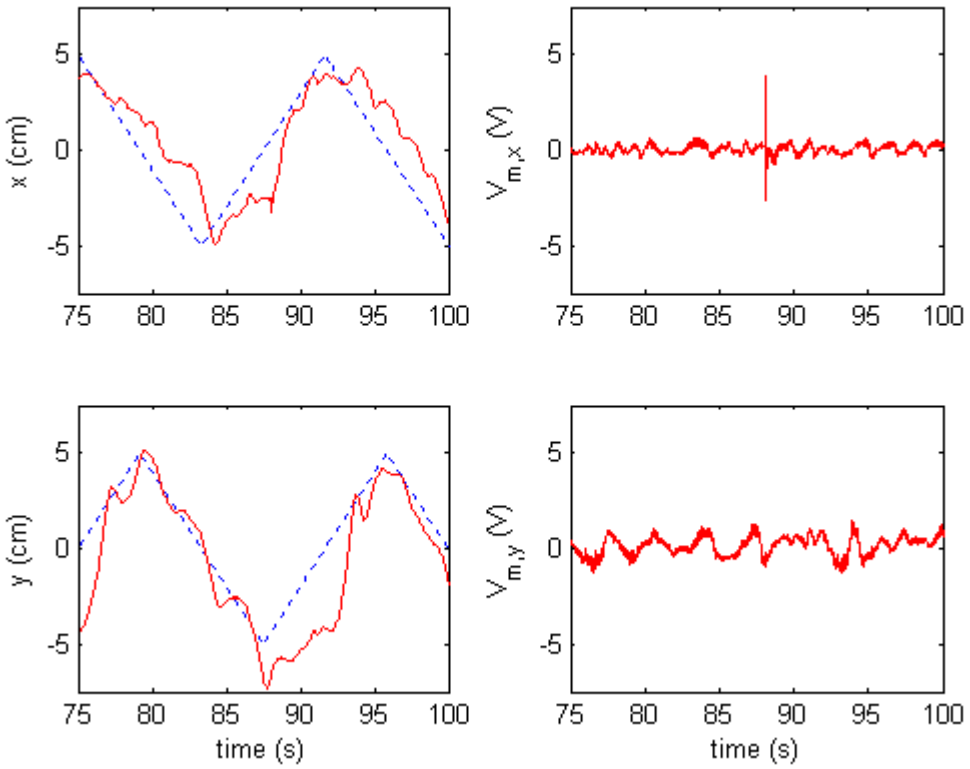

Figure 48: Measured 2DBB PD ramp response with $b_{sd} = 0$.

0	1	2

6. Now run the PD controller when the set-point velocity weight is set to 1. Similarly to the previous plot, attach a Matlab figure showing the x and y positions and input voltages.

Solution:
The measured PD ramp response when the set-point velocity weight parameter is set to one, i.e. $b_{sd} = 1$, is shown in Figure 48. To generate this response, execute the *plot_2dbb_rsp.m* script with the saved MAT files *sample_rsp_pd_ramp_bsp1_x.mat*, *sample_rsp_pd_ramp_bsp1_y.mat*, *sample_rsp_pd_ramp_bsp1_theta_l_x.mat*, *sample_rsp_pd_ramp_bsp1_theta_l_y.mat*, and *sample_rsp_pd_ramp_bsp1_vm.mat*. Alternatively, to generate a Matlab figure from a new experimental run do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, and *Tp* = 1.0.
2. Run the *q_2dbb_pos_soln* Simulink model with the bsd = 1 in both PID X and Y control systems.
3. Stop Quarc.
4. Run the *meas_2dbb_specs.m* script.

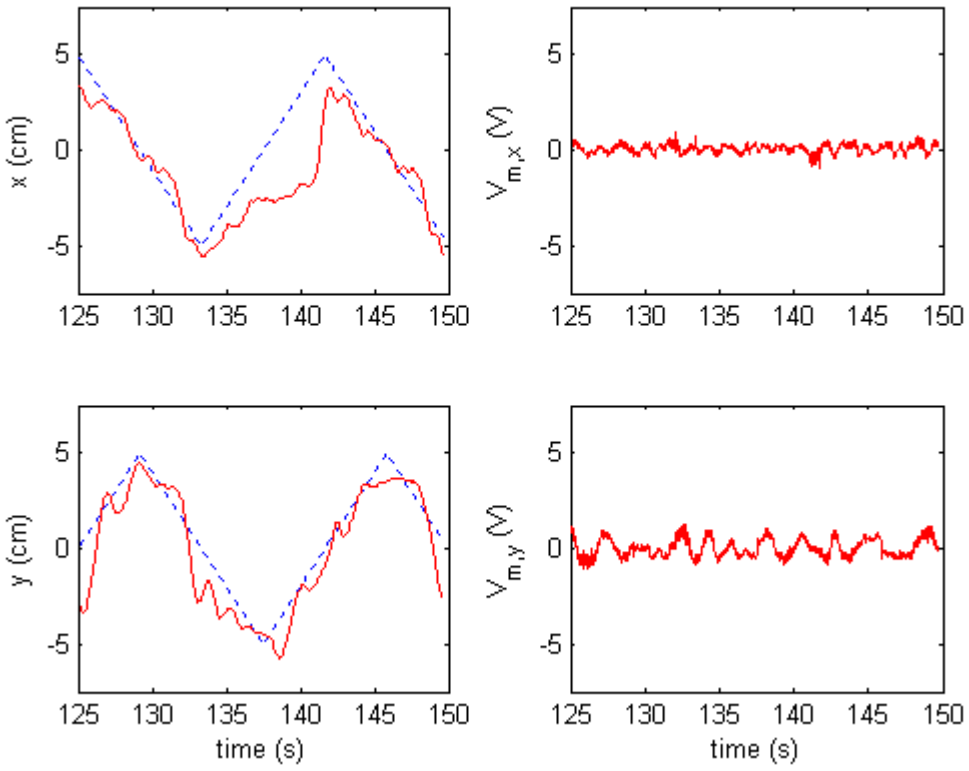

Figure 49: Measured 2DBB PD ramp response with $b_{sd} = 1$.

0	1	2

7. In general, is the tracking performance of the PD as expected in the simulation? Also, compare the response when bsd = 0 and when bsd = 1.

Solution:
The ramp is a much slower response then the step and it requires a lot less control effort, i.e. motor voltage, to control the ball position. Because of this, unmodeled effects such as friction and the cross-axis coupling has a greater effect on the response. As a result, the response obtained on the actual device is quite different then previously simulated.

The set-point weight does not have much effect. The PD response with either b_{sd}=0, Figure 48 or b_{sd} = 1, Figure 49, are quite similar.

0	1	2

8. Run the controller with the PID controller designed. Go through the steps in Section 5.1.2.1 to setup the Matlab workspace for the PID controller. Attach a Matlab figure of the response.

Solution:
The measured PID ramp response is shown in Figure 48. To generate this response, execute the *plot_2dbb_rsp.m* script with the saved MAT files *sample_rsp_pid_ramp_x.mat*, *sample_rsp_pid_ramp_y.mat*, *sample_rsp_pid_ramp_theta_l_x.mat*, *sample_rsp_pid_ramp_theta_l_y.mat*, and *sample_rsp_pid_ramp_vm.mat*. Alternatively, to generate a Matlab figure from a new experimental run do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, and *Tp* = 1.0.
2. Run the *q_2dbb_pos_soln* Simulink model with the bsd = 1 in both PID X and Y control systems.
3. Stop Quarc.
4. Run the *meas_2dbb_specs.m* script.

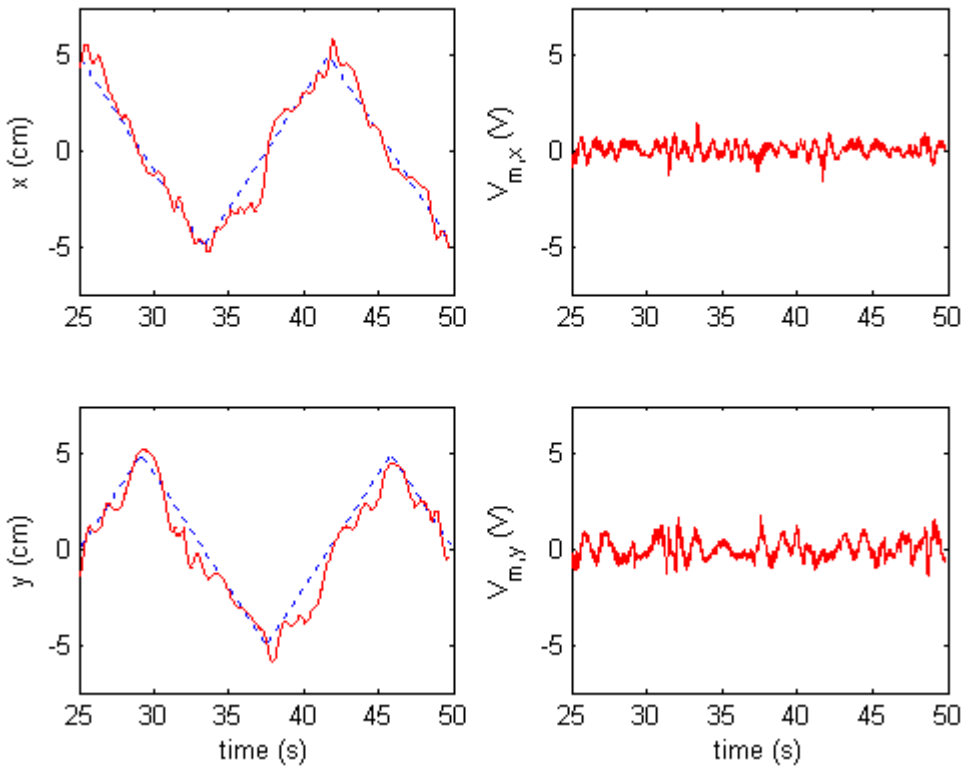

Figure 50: Measured 2DBB PD ramp response with b_{sd} = 1.

0	1	2

9. The control effort required to track the ramp reference is less than when tracking a step setpoint. The pole decay time constant specification will therefore be tuned as the controller is running to obtain a better response. Write a Matlab script that computes the PID gains *kp_bb*, *ki_bb*, and *kd_bb* from the percentage overshoot, settling time, and pole decay time specifications. Attach a copy of the script to your report.

Solution:
The function in the *d_2dbb_pid_gain.m* script is given in Text 3. The equations are taken from the Section 4.3.2.1. See the actual script for a description of the function.

```
function [ kp, ki, kv ] = d_2dbb_pid_design( K_bb, PO, ts, c_ts, Tp )
    % Damping ratio from overshoot specification.
    zeta = -log(PO/100) * sqrt( 1 / ( ( log(PO/100) )^2 + pi^2 ) );
    % Natural frequency from specifications (rad/s)
    wn = -log( c_ts * (1-zeta^2)^(1/2) ) / (zeta * ts);
    % Pole location (rad/s)
    p0 = 1 / Tp;
    % Proportional gain (rad/m)
    kp = wn*(wn +2*zeta*p0) / K_bb;
    % Integral gain (rad/m/s)
    ki = wn^2*p0 / K_bb;
    % Velocity gain (rad.s/m)
    kv = (2*zeta*wn + p0) / K_bb;
end
```

Text 3: Sample function that computes the PID gains based on the 2DBB specifications.

0	1	2

10. To make the controller tighter, decrease the time constant of the pole decay time by increments of 0.05 seconds. This bring the desired pole further into the left-hand plane and increases the gains – especially the integral gain. Thus run the script to generate a new set of gains and update the Simulink diagram to apply these updated gains to the running controller. Once satisfied, attach a plot of the response.

Solution:
The tuned PID ramp response is shown in Figure 48. To generate this response, execute the *plot_2dbb_rsp.m* script with the saved MAT files *sample_rsp_pid_tuned_ramp_x.mat*, *sample_rsp_pid_tuned_ramp_y.mat*, *sample_rsp_pid_tuned_ramp_theta_l_x.mat*, *sample_rsp_pid_tuned_ramp_theta_l_y.mat*, and *sample_rsp_pid_tuned_ramp_vm.mat*. Alternatively, to generate a Matlab figure from a new experimental run do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, and *Tp* = 0.3.
2. Run the *q_2dbb_pos_soln* Simulink model with the bsd = 1 in both PID X and Y control systems.
3. Stop Quarc.
4. Run the *meas_2dbb_specs.m* script.

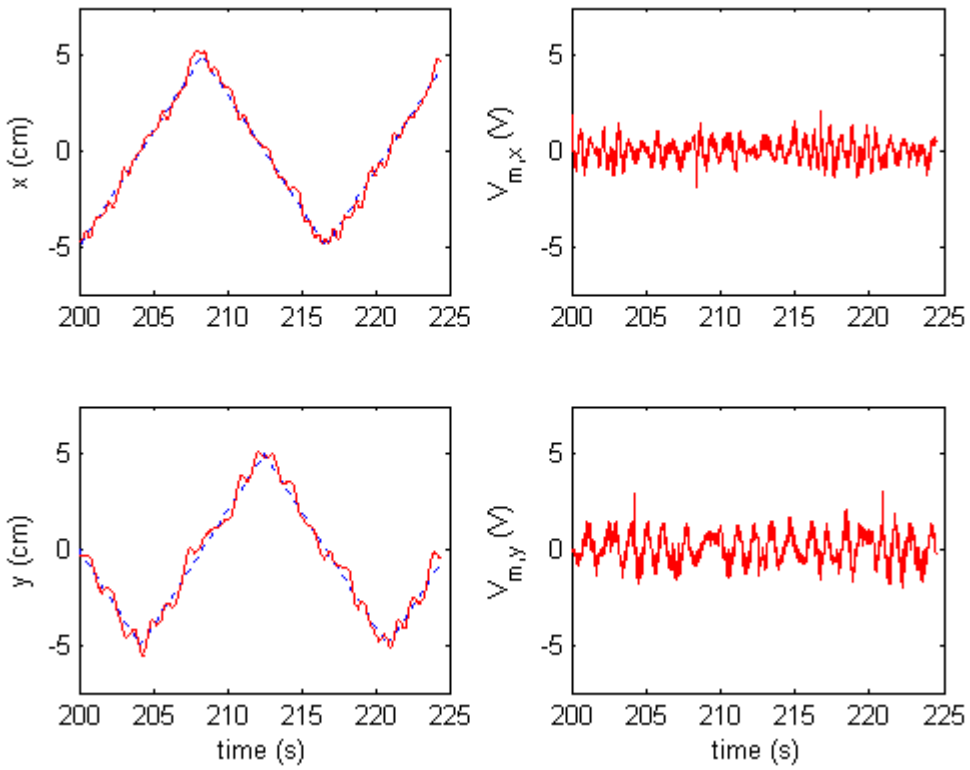

Figure 51: Measured 2DBB PID ramp response with T_r = 0.3 seconds.

0	1	2

11. List the decay time used as well as the PID gains generated.

Solution:

The PID controller was tuned with the pole decay time

$$T_p = 0.3\,[s] \quad . \qquad \text{[s90]}$$

With this new specification, the PID gains are

$$k_{p,\,bb} = 14.4\left[\frac{rad}{m}\right], \qquad \text{[s91]}$$

$$k_{i,\,bb} = 18.5\left[\frac{rad}{m\,s}\right], \qquad \text{[s92]}$$

and

$$k_{d,\,bb} = 6.01\left[\frac{rad\,s}{m}\right] \quad . \qquad \text{[s93]}$$

0	1	2

12. Show the resulting X-Y Figure with the newly tuned PID controller.

Solution:
The diamond trajectory in the XY plane when using the tuned PID controller is illustrated in Figure 52.

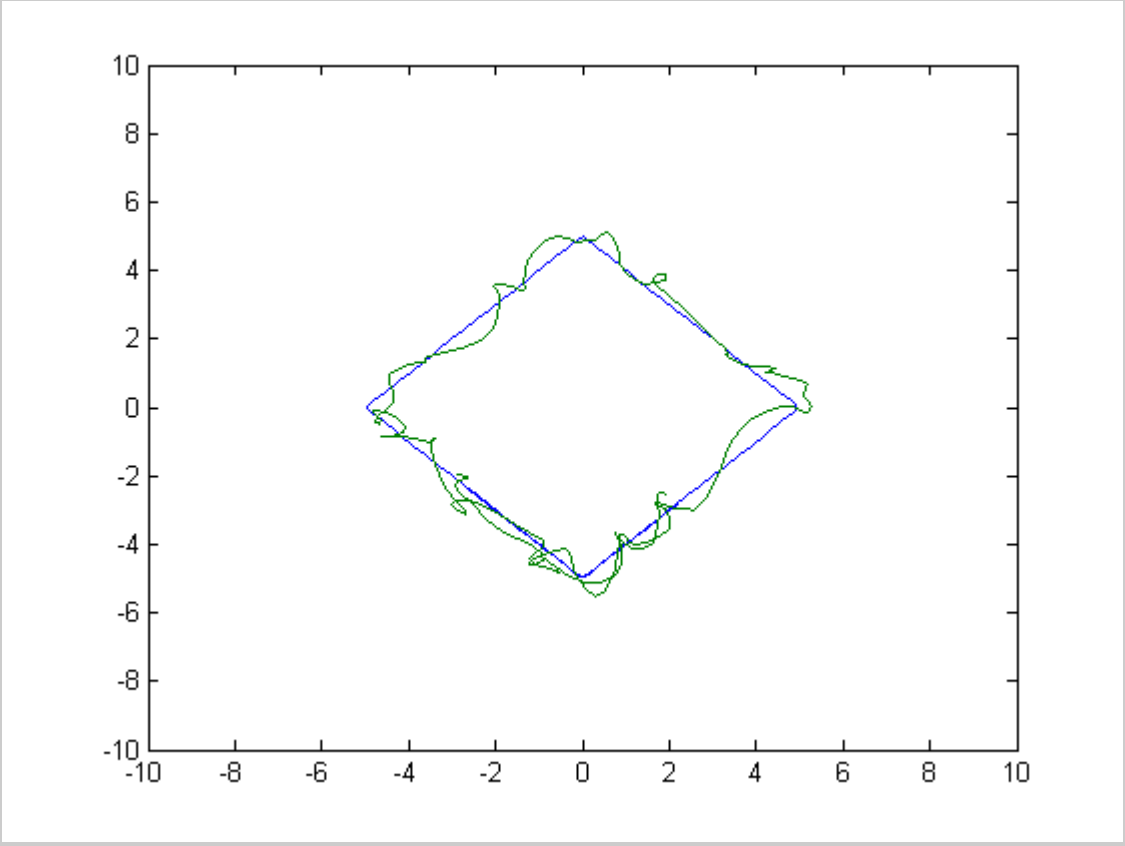

Figure 52: Tuned PID diamond response.

0	1	2

13. Make sure QUARC is stopped.
14. Shut off the amplifier power if no more experiments will be performed on the SRV02 in this session.

5.2.6. Sine Response using PID Control

Using the PID controller, the ball is to track a sinusoidal reference in both axes to form a circle in the 2D plane.

Follow the steps below:

1. Go through the steps 2-9 in Section 5.2.3 to start the controller.
2. In the *Setpoints* subsystem, select *sine* in the *Signal Type* field of both the *SRV02 Signal Generator X* and *SRV02 Signal Generator Y* in order to generate a sinusoidal reference.

3. Place the ball approximately in the center of the plate and adjust the servo load gears so the ball remains in place.
4. Make sure the set-point velocity weight is set to 1. Thus set the *bsd* Slider Gain block in both the *Cascade Control\2DBB PID Position Control: X* and *Cascade Control\2DBB PID Position Control: Y* subsystems to 0.
5. Set the *Amplitude X (cm)* and *Amplitude Y (cm)* slider gain blocks to 5 to generate a step with an amplitude of 5.0 centimeters in the x and y axes. The scopes should be displaying responses similar to figures 53, 54, 55, 56, and 57 (depending on the control gains used).

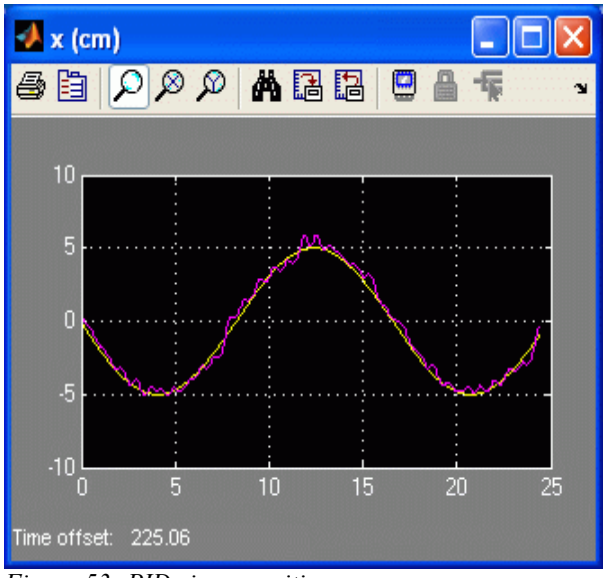

Figure 53: PID sine x position response.

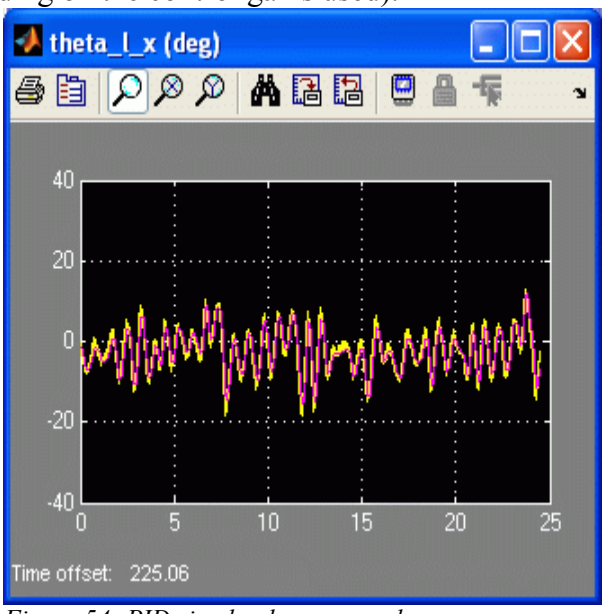

Figure 54: PID sine load gear x angle response.

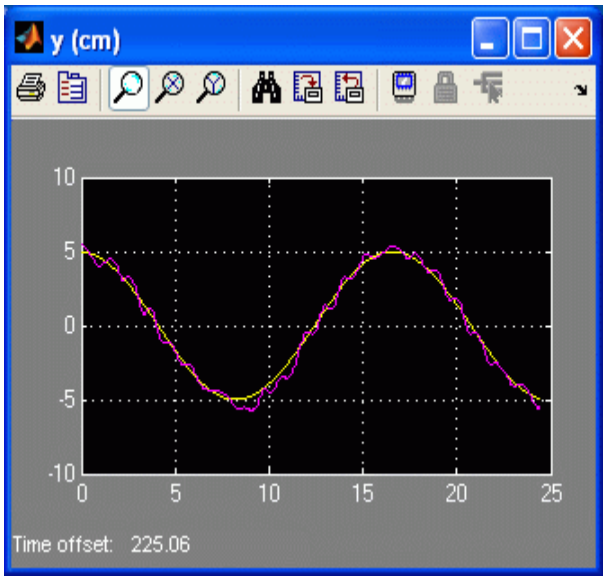

Figure 55: PID sine y position response.

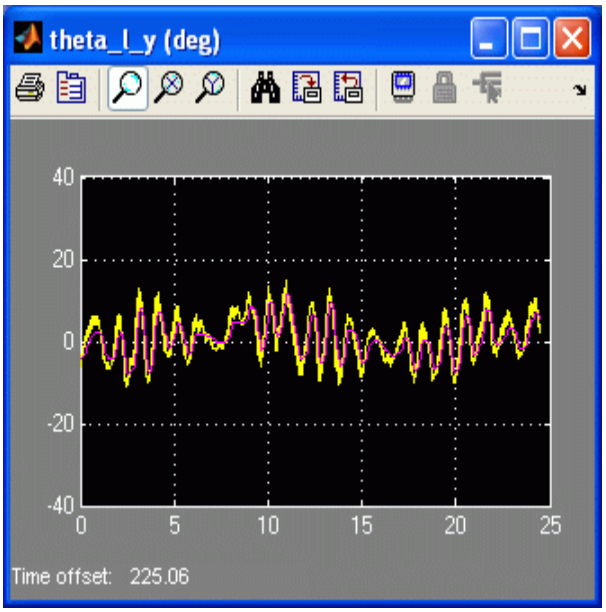

Figure 56: PID sine load gear Y angle response.

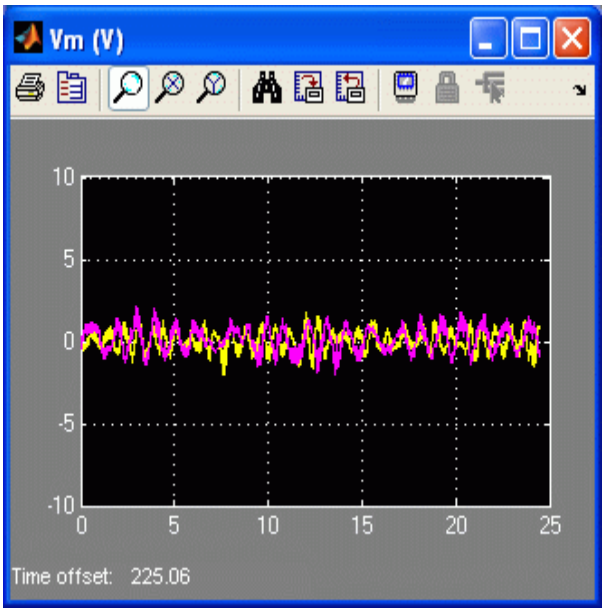

Figure 57: PID sine input motor voltage.

6. As with in the previous experiment with the ramp setpoint, use the your script to tune the pole decay time specification, Tp, and generate new gains accordingly. Decrease the parameter in increments of 0.05 seconds, run the script, update the diagram, and examine its effect on the response. Make sure the SRV02 load gears and the input voltage do not become saturated. When a suitable response is obtained, stop running the controller and generate a Matlab figure showing the x and y ball positions and input voltage responses. Attach it to your report..

Solution:
The measured PID sinusoidal response is shown in Figure 58. To generate this response, execute the *plot_2dbb_rsp.m* script with the saved MAT files *sample_rsp_pid_tuned_sine_x.mat*, *sample_rsp_tuned_sine_y.mat*, *sample_rsp_pid_tuned_sine_theta_l_x.mat*, *sample_rsp_pid_tuned_sine_theta_l_y.mat*, and *sample_rsp_pid_tuned_sine_vm.mat*. Alternatively, to generate a Matlab figure from a new experimental run do the following:

1. Execute the *setup_srv02_exp17_2dbb.m* script with CONTROL_TYPE = 'AUTO', *c_ts* = 0.04, *ts_bb* = 2.5, *PO_bb* = 7.5, and *Tp* = 0.3.
2. Run the *q_2dbb_pos_soln* Simulink model with the bsd = 1 in both PID X and Y control systems.
3. Stop Quarc.
4. Run the *meas_2dbb_specs.m* script.

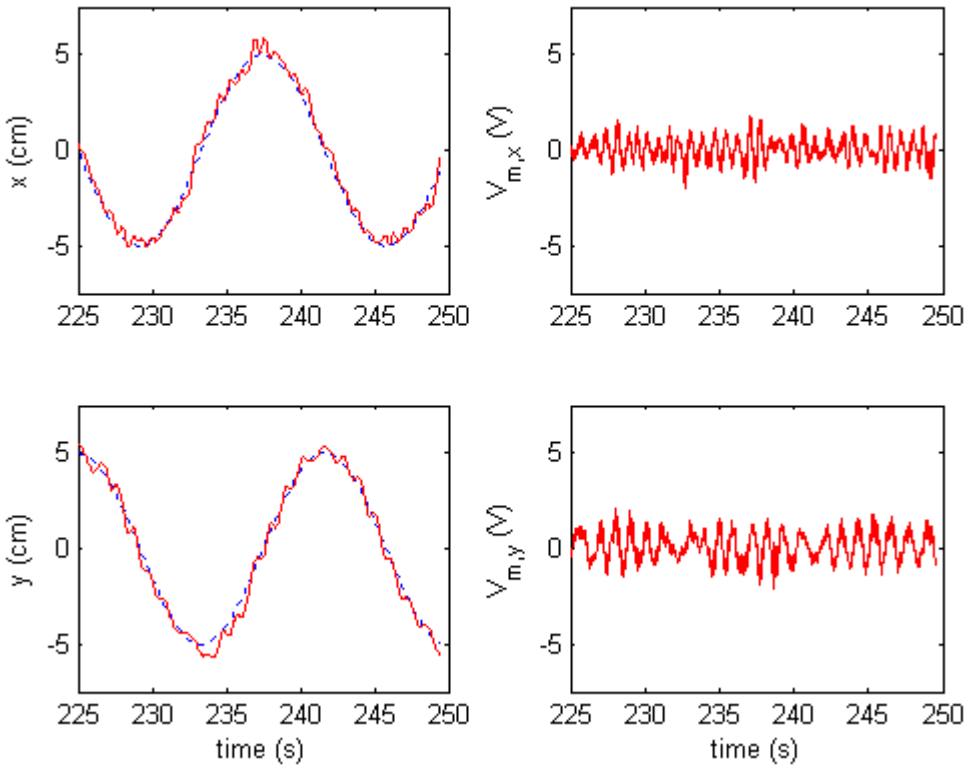

Figure 58: Measured 2DBB tuned PID sine response.

0	1	2

7. List the decay time used as well as the PID gains generated.

Solution:
The PID controller was tuned with the pole decay time

$$T_p = 0.3\,[s] \quad . \qquad \text{[s94]}$$

With this new specification, the PID gains are

$$k_{p,\,bb} = 14.4\left[\frac{rad}{m}\right], \qquad \text{[s95]}$$

$$k_{i,\,bb} = 18.5\left[\frac{rad}{m\,s}\right], \qquad \text{[s96]}$$

and

$$k_{d,\,bb} = 6.01\left[\frac{rad\,s}{m}\right] \quad . \qquad \text{[s97]}$$

0	1	2

8. Show the resulting circle in the X-Y Figure with the tuned PID controller.

Solution:
The circle trajectory in the XY plane when using the tuned PID controller is shown in Figure 59.

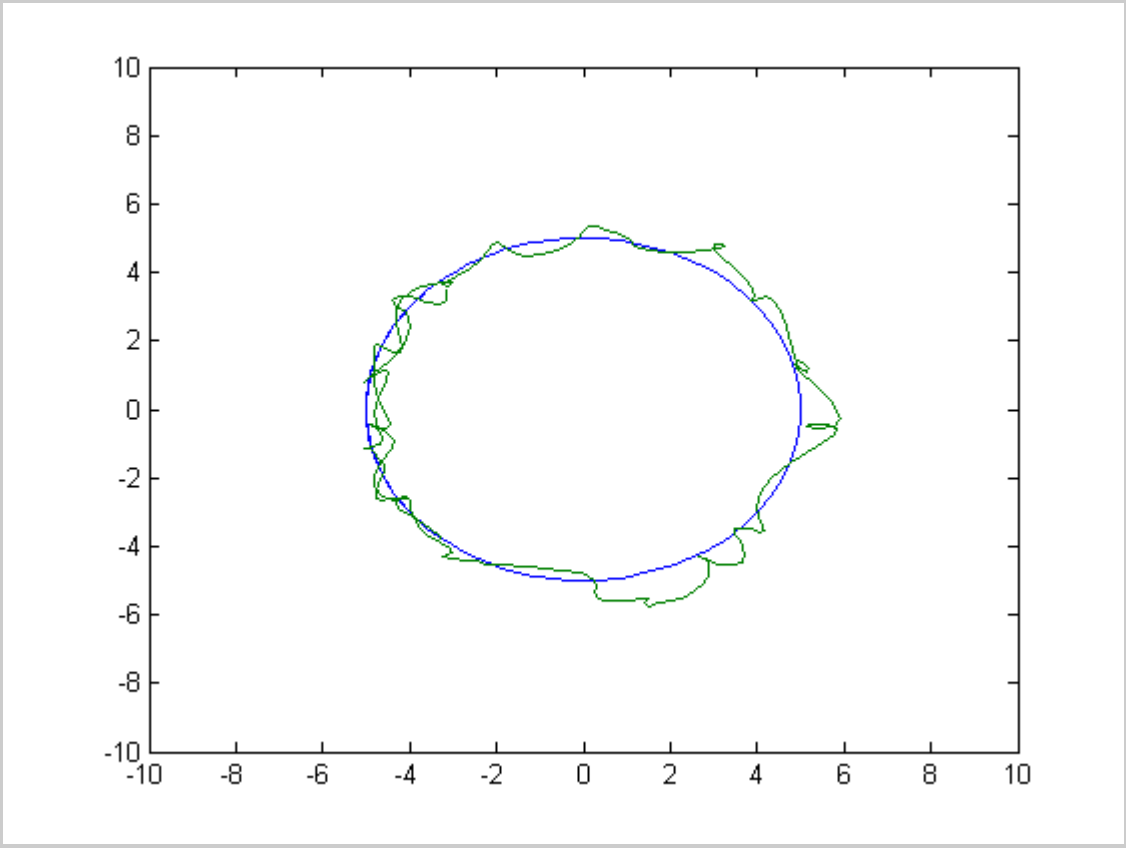

Figure 59: Tuned PID circle response.

0	1	2

9. Make sure QUARC is stopped.
10. Shut off the amplifier power if no more experiments will be performed on the SRV02 in this session.

6. Results Summary

Fill out Table 2, below, with the pre-laboratory and in-laboratory results such as the PD and PID gains computed and the measured settling time, percentage overshoot, and steady-state error obtained from the simulated and implemented responses.

Section	*Description*	*Symbol*	*Value*	*Unit*
4.3.1	***Pre-Lab: Model Parameters***			
1.	Open-Loop Steady-State Gain	K	1.76	rad/(V.s)
1.	Open-Loop Time Constant	τ	0.0285	s
4.3.1	***Pre-Lab: SRV02 PV Gain Design***			
3.	Proportional gain	k_p	13.5	V/rad
3.	Velocity gain	k_v	0.078	V.s/rad
4.3.2.2	***Pre-Lab: PD Control Design***			
1.	Proportional gain	$k_{p,bb}$	3.69	rad/m
1.	Derivative gain	$k_{d,bb}$	2.14	rad.s/m
4.3.2.3	***Pre-Lab: Practical PD Control Design***			
1.	Proportional gain	$k_{p,bb}$	5.83	rad/m
1.	Derivative gain	$k_{d,bb}$	2.91	rad.s/m
1.	Integral gain	$k_{i,bb}$	3.69	rad/(m.s)
4.3.3	***Pre-Lab: Steady-State Error***			
2.	PD step steady-state error	e_{ss}	0	cm
4.	PD ramp steady-state error	e_{ss}	0.696	cm
5.	Set-point velocity weight needed to meet steady-state error specification.	b_{sd}	0.86	
5.1.1.2	***In-Lab Simulation: Step PD Response***			
7.	Steady-state error	e_{ss}	0.00	cm
7.	Settling time	t_s	2.31	s
7.	Percentage overshoot	PO	5.48	%
5.1.1.3	***In-Lab Simulation: Ramp PD Response***			
7.	Steady-state error	e_{ss}	0.698	cm

5.1.2.2	***In-Lab Simulation: PID Step Response***			
	Default PID			
	Anti-windup tracking time constant	T_r	5.0	s
	Anti-windup saturation limit	INT_MAX	2.5*π/180	rad
9.	Steady-state error	e_{ss}	-0.0152	cm
9.	Settling time	t_s	6.02	s
9.	Percentage overshoot	PO	8.28	%
	Tuned PID			
	Anti-windup tracking time constant	T_r	0.001	s
12.	Tuned anti-windup saturation limit	INT_MAX	1.6*π/180	rad
13.	Steady-state error	e_{ss}	0.00	cm
13.	Settling time	t_s	2.47	s
13.	Percentage overshoot	PO	5.66	%
5.2.2	***In-Lab Implementation: Camera Calibration***			
11.	X-Axis ball displacement measured by ruler.	Δx_{ruler}	20.3	cm
11.	X-Axis ball displacement measured by camera.	Δx	20.5	cm
12.	Y-Axis ball displacement measured by ruler.	Δy_{ruler}	17.8	cm
12.	Y-Axis ball displacement measured by camera.	Δy	16.8	cm
5.2.3	***In-Lab Implementation: PD Step Response***			
12.	Steady-state error	e_{ss}	0.629	cm
12.	Settling time	t_s	1.11	s
12.	Percentage overshoot	PO	5.1	%
5.2.4	***In-Lab Implementation: PID Step Response***			
12.	Steady-state error	e_{ss}	-0.0962	cm
12.	Settling time	t_s	3.31	s
12.	Percentage overshoot	PO	6.57	%
5.2.5	***In-Lab Implementation: Ramp Response***			
19.	Pole decay time.	T_p	0.30	s
19.	Proportional gain	$k_{p,bb}$	14.4	rad/m
19.	Derivative gain	$k_{d,bb}$	18.5	rad.s/m
19.	Integral gain	$k_{i,bb}$	6.01	rad/(m.s)

5.2.5	*In-Lab Implementation: PID Sine Response*			
13.	Pole decay time.	T_p	0.30	s
13.	Proportional gain	$k_{p,bb}$	14.4	rad/m
13.	Derivative gain	$k_{d,bb}$	18.5	rad.s/m
13.	Integral gain	$k_{i,bb}$	6.01	rad/(m.s)

Table 2: SRV02 Experiment #17: 2D Ball Balancer control results summary.

7. References

[1] DAQ User Manual.
[2] QUARC User Manual (type `doc quarc` in Matlab to access).
[3] QUARC Installation Manual.
[4] Power Amplifier User Manual.
[5] SRV02 User Manual.
[6] SRV02 QUARC Integration.
[7] Rotary Experiment #1: SRV02 Modeling.
[8] Rotary Experiment #2: SRV02 Position Control.
[9] Rotary Experiment #4: Ball and Beam Position Control.
[10 2D Ball Balancer User Manual.