NC STATE UNIVERSITY

College of Engineering

Department of Mechanical and Aerospace Engineering

MAE 435

Principles of Automatic Control

Ball and Beam Controller

Written By: **Christina Lasdin**
Jake Janosko
Maia Judd
Mateen Sattari

Date: **04/25/2023**

Abstract

This study and the corresponding report were conducted with the goal of optimizing a controller for an Acrome Ball and Beam system by myControl. This system is designed to balance a metal ball on a teetering beam with a specific controller implemented. By modeling and designing the plant and implementing the sensor dynamics, a controller can be designed using root locus analysis. The study found many possible controllers in Simulink, but in real life practice, the controllers struggled to track and react quick enough to the ball's movement, resulting in positional error. This is likely due to the discrepancies in the Ball and Beam system itself. A poor quality sensor can result in large amounts of phase lag, as well as create readout noise that can confuse and delay the controller's actions, and some of the given variables (like the ball weight and radius) may be slightly off. Despite this, a solution was found for the control system that had reasonable settling time and steady state error.

1. Introduction

Control systems work to manipulate one or more system inputs so that one or more system outputs can track a reference. Systems that are stable have bounded responses to bounded inputs[1]. Controllers can be designed to stabilize unstable systems to obtain bounded responses.

The controller for this system should be able to balance the ball about an input reference position along the beam with both reasonable settling time and steady state error.

The plant can be modeled as a combination of the servo dynamics, the appropriate angle conversion of the motor arm to the beam, and the ball dynamics. A simple open loop block diagram of the plant can be seen below:

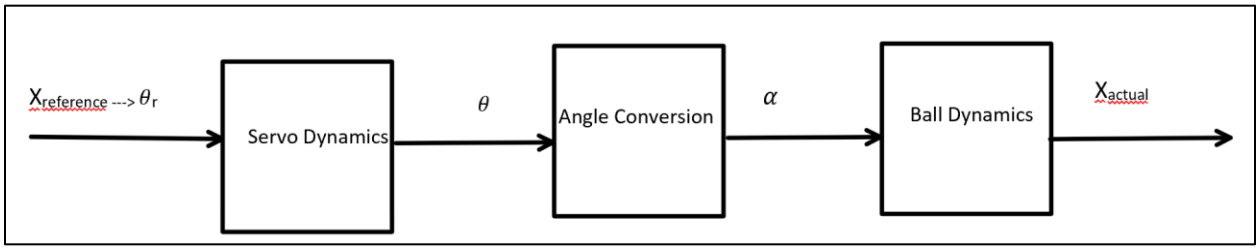


Figure 1: Open-Loop Block Diagram of Plant

The Simulink controller sends an actuation signal to an Arduino microcontroller, which then powers the servo actuator via pulse-width modulation (PWM) where the PWM input signal is decoded and is converted into a voltage that results to a corresponding angle[2]. A block diagram of the RC servo theory of operation can be seen in Appendix C.

The servo actuator dynamics can be approximated as a first order transfer function in standard form. As provided in the courseware, the transfer function is then the following:

$$P_{servo} = \frac{1}{0.01s + 1}$$

The angle conversion, as will be more thoroughly derived later will be described with the following relationship:

$$\sin(\alpha) = \left(\frac{r_m \sin\theta}{L_r}\right)$$

The position of the ball with respect to the beam x_b, is measured with a resistive sensing tape which is passed through the Arduino and the Simulink controller. The corresponding ball dynamics, as will be more thoroughly derived later, can be described with the following equation of motion and subsequent transfer function.

$$\ddot{x}_b = \frac{5}{7} g \sin\alpha$$

$$P_{ball\ dynamics} = \frac{1.0322}{s^2}$$

2. Methods

To model the plant, a solid sphere rolling down an incline was considered. The equations of motion were obtained for the ball, with the positive x-axis along the incline pointing in the left direction and the positive y-axis being perpendicular up from the x-axis.

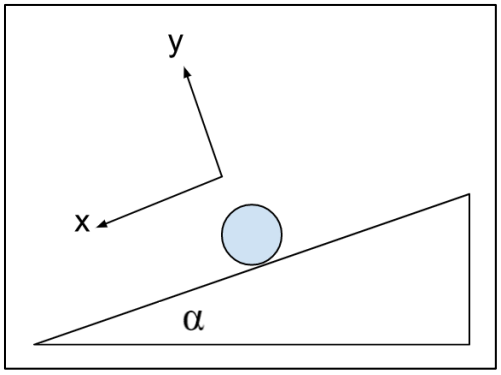


Figure 2: Model of a Rolling Ball on a Ramp

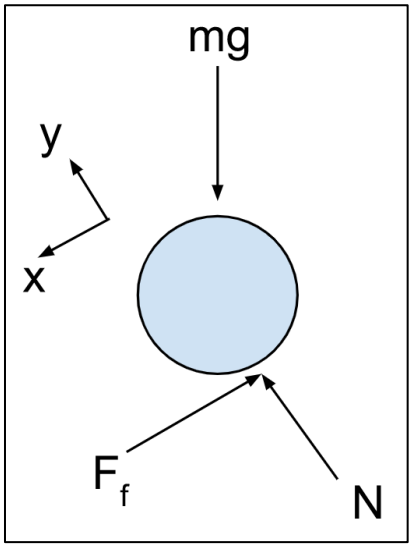


Figure 3: Free Body Diagram of a Rolling Ball on a Ramp

Analyzing the forces applied to the ball, the acceleration in the x axis was solved as follows, replacing the incline angle α with ϕ to avoid confusion with rotational acceleration:

$$\sum F_x = mg \sin\phi - F_f = ma_{G,x} \tag{a}$$

$$\sum M_G = I_G \alpha = -F_f r$$

$$F_f = -\frac{I_G \alpha}{r} \tag{b}$$

Assuming the condition of no slipping, a third equation is presented:

$$a_{G_x} = -\alpha r$$

$$\alpha = -\frac{a_{G_x}}{r} \tag{c}$$

Plugging in all known variables and the moment of inertia for a solid sphere: $I_G = \frac{2}{5}mr^2$:

$$mg \sin\phi - \frac{\left(\frac{2}{5}mr^2\right)\alpha_{G_x}}{r^2} = ma_{G_x}$$

$$a_{G_x} = \ddot{x}_b = \frac{5}{7}g \sin\phi = \frac{5}{7}g \sin\alpha \tag{d}$$

The angle θ of the servo motor arm is translated to a change in the beam angle that moves the ball, corresponding to angle α on figure 1. The conversion of these angles was computed to determine how the angle of the beam follows the angle of the servo motor arm.

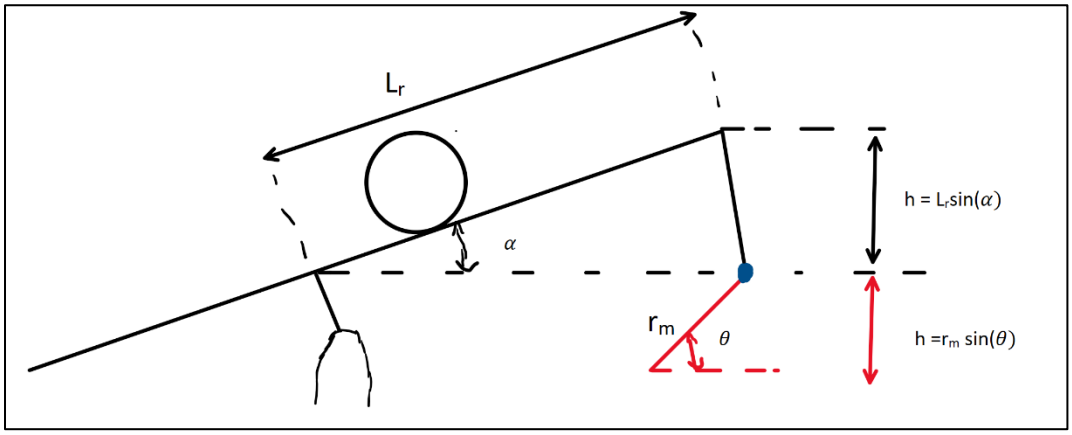


Figure 4: Angle Relations

Assuming the change in angles θ and α contribute to the same change in y (Δy), the change in y can be equated and the angles can be solved:

$$\mathrm{h} = \Delta y = r_m \sin\theta = L_r \sin\alpha \tag{e}$$

Solving for α,

$$\alpha = \sin^{-1}\left(\frac{r_m \sin\theta}{L_r}\right) \tag{f}$$

Plugging in α from equation (f) into equation (d) and substituting known values, we are left with:

$$\ddot{x}_b = \frac{5}{7} g \sin\left(\sin^{-1}\left(\frac{r_m \sin\theta}{L_r}\right)\right)$$

$$\ddot{x}_b = \frac{5}{7}(9.81)\left(\frac{0.0245}{0.29}\right)\sin\theta \tag{g}$$

Linearizing this function using the MATLAB command *linmod* we are left with a transfer function:

$$P(s) = \frac{1.0322}{s^2(0.01s+1)} \tag{h}$$

The transfer function of the sensor was given and was then put into standard form:

$$H(s) = \frac{15}{s+15} = \frac{1}{0.067s+1} \tag{i}$$

The plant and the feedback were then combined into an open loop system by multiplying P and H together. Using the MATLAB command *rltool* a root locus was then modeled of the open loop system.

$$PH(s) = \frac{1.0322}{s^2(0.01s+1)(\frac{1}{15}s+1)} \tag{j}$$

A general form of the controller that satisfied the design requirements in a sufficient manner was found using root analysis and MATLAB's *rltool* functionality and control system designer.

$$C(s) = (k_p)\frac{(\tau_{z1}s + 1)(\tau_{z2}s + 1)}{s(\tau_p s + 1)}$$

To stabilize the system, a controller was designed using root locus analysis. Many controllers were designed that stabilized the system successfully in Simulink modeling, but upon experimental testing were insufficient on the physical system. These included, but were not limited to, single pole, pole and zero, addition of a damping term, and imaginary pole use.

A two-zero two-pole system was decided as being the most successful way to stabilize the system. An integrator was added in the controller to limit steady-state error. Using lead compensation, zeros were placed close to the origin with a pole farther away in the left-hand plane to promote a faster response and reduce overshoot. While it was expected that a zero closer to the origin is preferrable, certain zeros too close to the pole would result in excessive input angles that were not realistic. The second pole needed to be placed substantially in the left-hand plane to pull the asymptote away from instability, but too far left could lead to it becoming insignificant and cause the system to become unstable anyway.

By adding equivalent zeros and poles, the number of the system remains at four. If more zeros were added compared to poles, this would be considered an improper controller. The addition of more poles to zeros would increase the overall number of the system, meaning that the asymptote would shift toward the right-hand plane, toward instability, and the number of angles would increase, decreasing high-gain stability.

Recognizing that settling time should be limited, a settling time of less than 1.5 seconds was considered adequate for the system. Considering these design requirements, the controller was determined to be:

$$C(s) = (0.2)\frac{(s + 1)(10s + 1)}{s(0.03s + 1)}$$

3. Results

The root locus of the open loop system, with a controller gain of 1 showed extreme instability, as expected.

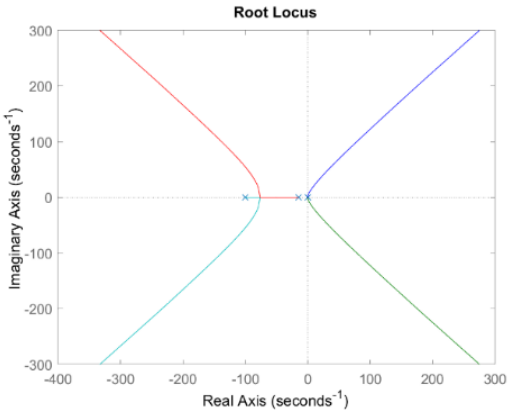


Figure 5: Root Locus of Open Loop System with a Proportional Gain of 1

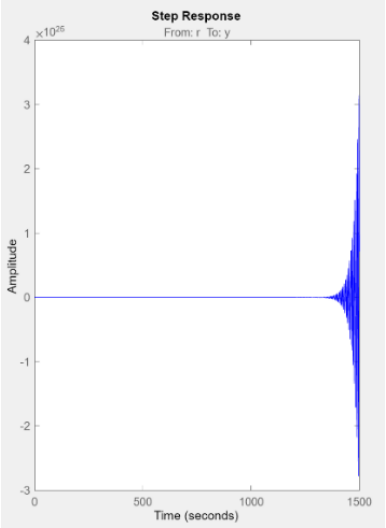


Figure 6: Response of the Open Loop System with a Proportional Gain of 1

After the careful iteration of several designed controllers, the final controller consisted of two zeroes at -1 and -1/10, as well as two poles at 0 and -33.3. Poles too far in the left-hand plane could lead to the pole becoming insignificant and possibly leading to the system going unstable, so the location of s=-33.3 was found to be ideal.

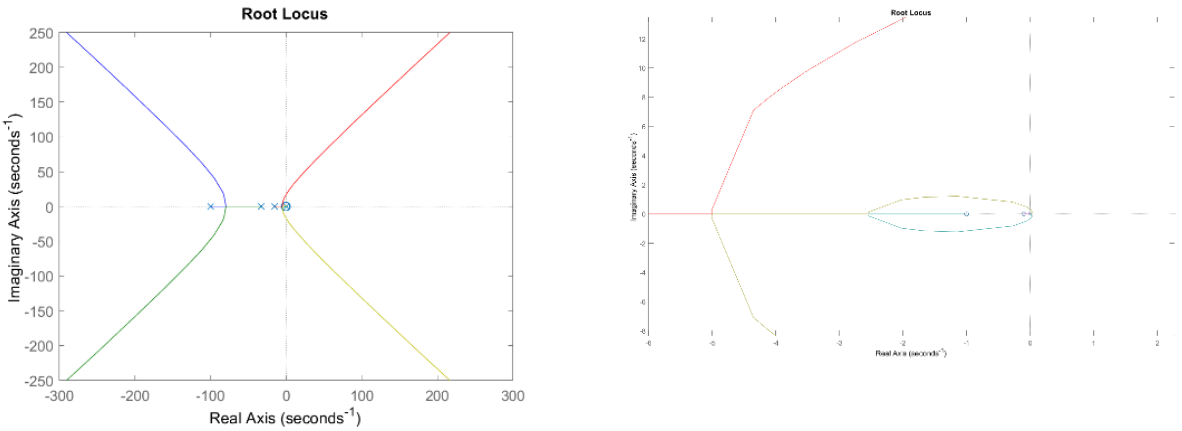


Figure 7: Root Locus of System with Designed Controller

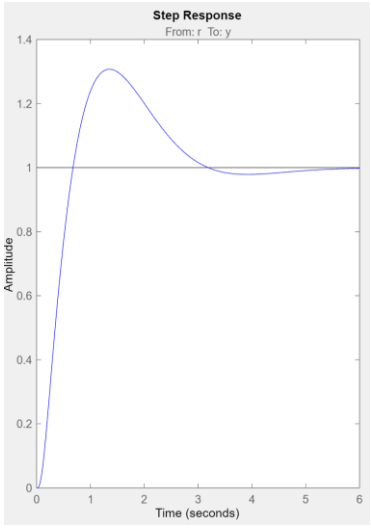


Figure 8: Response of the System with the Designed Controller

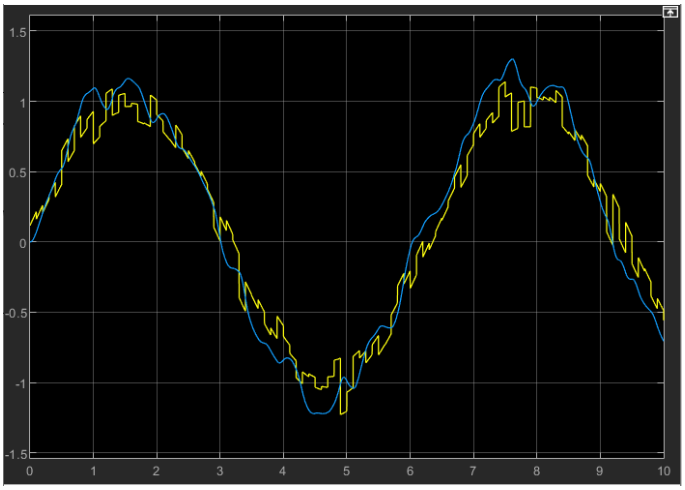


Figure 9: Simulink Response to a Chaotic System Input

Upon implementing the designed controller into the experimental system, it was found that gains lower than 0.7 led to better controllers. Specifically, a gain of 0.2 resulted in the most preferrable experimental result.

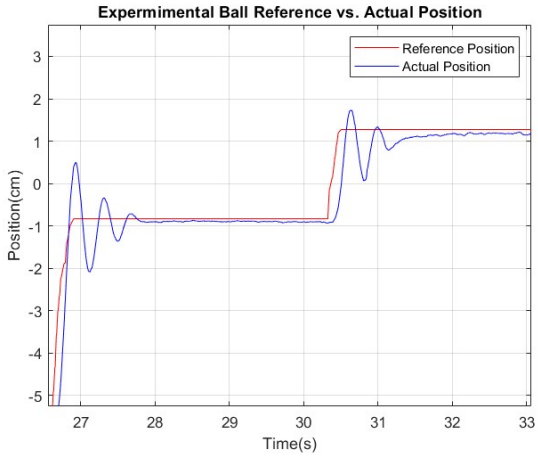


Figure 10: Experimental Ball Reference vs Actual Ball Position Step

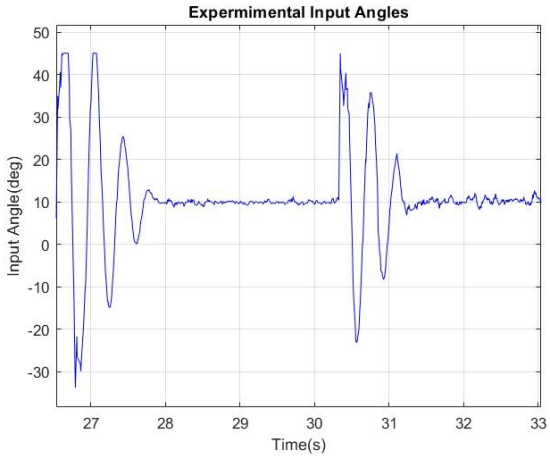


Figure 11: Experimental Input Angles Step

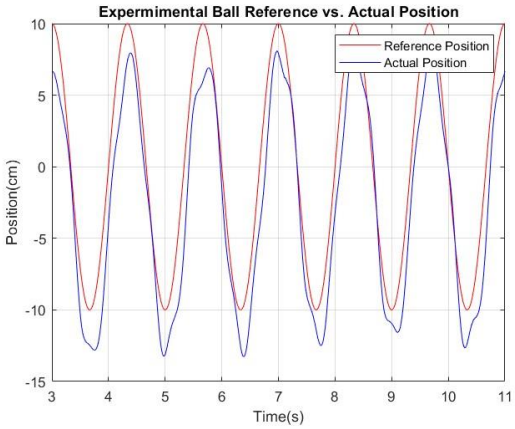


Figure 12: Experimental Ball Reference vs Actual Ball Position Sine

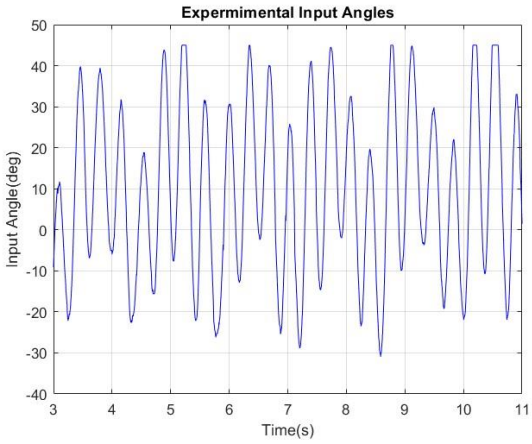


Figure 13: Experimental Input Angles Sine

4. Conclusions

The study showed that an adequate controller can be made using root locus design with compensating zeros and poles. An integrator was included to limit steady-state tracking error. The range of acceptable values for adequate controllers was extremely low, and low gain was a must for any level of stability. As the project went on, more and more errors were found, such as the ball mass, diameter, and sensor distortion. However, when a specific controller with extra poles and zeros is used, even noticeable overshoot can result in a decent control of the ball, which responded well to position step functions and sine functions. The biggest issue the controller has is that its control of the servo motor is a little bit jittery instead of being smooth. Overall, this specific controller was adequate because it had low gain, close zeros, a far pole, and an integrator term, which are all topics of discussion for controller design in the latter half of MAE 435.

5. References

[1] Buckner, G. D. (Spring 2023). *MAE 435 Course Notes* [Course Notes]. NC State University. MAE435.

[2] *ACROME Ball and Beam Courseware. myControl*.
https://moodlecourses2223.wolfware.ncsu.edu/mod/resource/view.php?id=754589

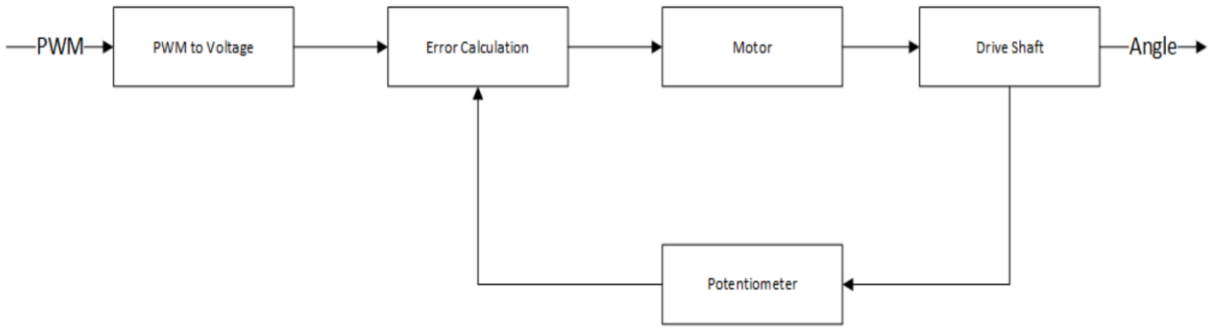
6. Appendix

Appendix A: Ball and Beam Balance by Acrome

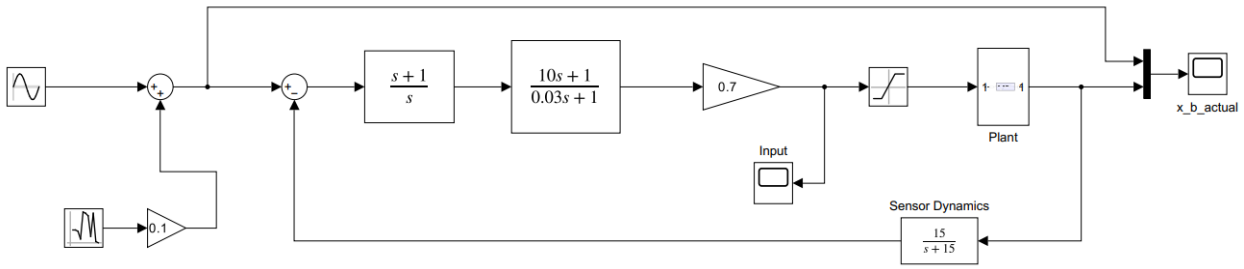
Servo	**Ball**	**Beam**	**Sensor**
• Actuator arm radius (rm): 0.0245m • Connecting rod length (Lc): 0.048m • Actuator arm angle: θ - when θ=0∘, α=0∘ • Actuator range: θ min=−45∘, θ max=+45∘ • Corresponding beam angle range: α min=−3∘, α max=+3∘	• Radius (rb): 0.020 m • Mass(mb): .2631 kg • MMOI (Jb): 0.00002143 kgm2	• Length (L): 0.5m • Actuator Moment Arm (Lr): 0.29m • MMOI: 0.01kg·m2	• Transfer function: H(s)=1/0.067s+1

• Transfer function: G1(s)=θ(s)/θr(s)=1/0.01s+1			

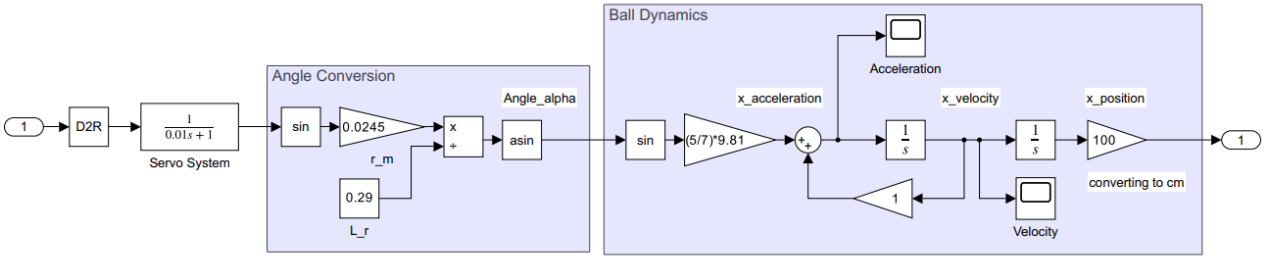
Appendix B: Table of Given System Values



Appendix C: RC Theory Servo Operation



Appendix D1: Simulink System Overview



Appendix D2: Simulink Plant Model