A Balancing Act: Ball and Beam Control

For ENGR 454. By Richard Arkusinski and David Egolf.

# Project Goal

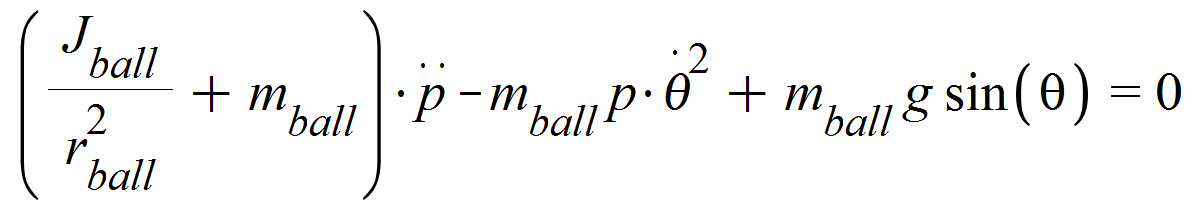
The idea behind our project was to balance a ball on a beam by controlling the tilt of the beam, as shown in Figure 1. Our goal was to balance the ball so that it stayed in a given spot on the beam using digital control.

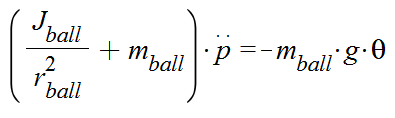
# Pre-Construction Analysis

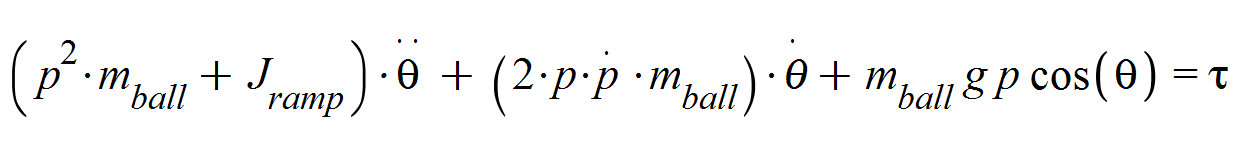
Figure 1: Project Concept

Before purchasing sensors and constructing the system, we needed to know if the system would be controllable and observable. Therefore, we needed to model the system.

We chose to model our system using Lagrangian equations. The input to our system was the torque applied to the center of the beam, and the outputs were the angle of the beam and the position of the ball. The resulting equations were non-linear, and so we linearized them by assuming the ball was close to the center of the beam (*p* is small) and that the beam was not tilted at a very large angle (θ is small). Further discussion can be found in “Ball on Ramp Derivation.”







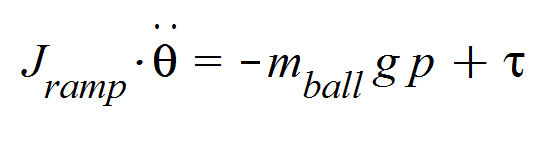


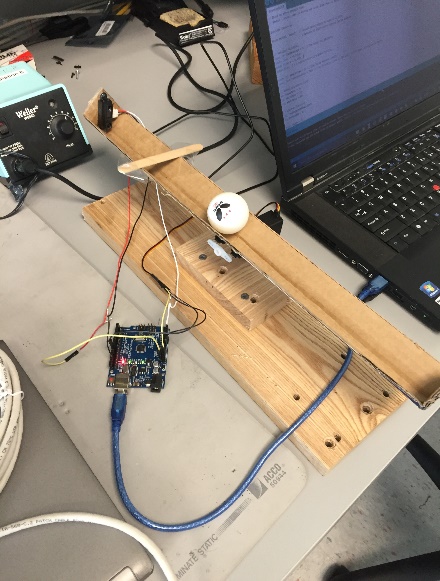
Figure 2: The Lagrangian equations (left) and their linearized form (right).

We found the system was both controllable and observable when both the position of the ball and the angle of the ramp were measured. Somewhat surprisingly, we found the system was still observable and controllable if we measured only the ball position or only the ramp angle. As a measure of controllability, we calculated the conditioning number for the system – the smaller the conditioning number, the better (see the MATLAB script “calcConditionNumber”).

Measuring the position of the ball was key to a small conditioning number. So, we purchased a sensor to measure the position of the ball, and decided to forgo measuring the beam angle until shown to be necessary. The servo command angle also serves as a first approximation of beam angle.

# System Components and Construction

## System Components

Before beginning construction, we had to decide on the basic physical configuration of the system. We considered two ideas. The first option was to drive the beam from the middle, and the second option was to pivot the beam on one end and drive the other end using a gearing system. We chose to drive the beam from the middle for ease of construction.

To drive the beam, we chose to use servos since they were readily available. For simplicity, we used an already available Arduino UNO to control the servos. A ping-pong ball worked well for the ball component.

To measure the ball position we had several sensor options to choose from: ultrasonic sensing, infrared (IR) sensing, a camera, or some sort of resistive sensor. Again for simplicity, we chose to order a cheap infrared sensor. We hoped the infrared sensor would have better resolution than an ultrasonic sensor we had available.

We had some trouble finding a beam, but we eventually created one by cutting out the edge of a cardboard box. The resulting beam was light, but not very rigid or uniform. Our base was originally a flimsy paper bowl, but Professor Stirling upgraded us to a solid wooden base.

Figure 3: Our constructed system

## System Construction

At the start of construction we still had not selected a beam. In order to be sure we had enough driving torque for whatever beam we chose, we coupled two servos together. We hot-glued the servo horns together so they would move in unison. It turned out that coupling the two servos in this way caused some problems, and that one servo was sufficient for our light beam. Therefore, we removed the second servo.

We hot-glued the beam to the top of the servo horn and hot-glued the IR sensor to the end of the beam. Since the IR sensor does not measure distance properly at very short range, we taped on a popsicle stick as a block to ensure the ball stayed far away enough from the sensor.

Finally, we wired all the servos and motors to the Arduino, and connected the Arduino to the computer. Using the Arduino servo sample code, we successfully wrote command angles to the servos, which then moved the beam to the desired angle. At this point, we had a functioning system and were ready to start writing control algorithms.

# PID Control

We began by using PID control because it is simple to implement, does not require a system model, and it only requires a measurement of the ball position since that is the variable we are trying to control. See the folder “BeamControlV4 - with PID” for details.

## Sensor Noise: Approximating Position and Derivative

We faced a significant challenge in getting accurate data from our sensor. The sensor would periodically report spikes way above the general trend line. In order to combat this noise, we took the lowest couple of data points and averaged them to get a smooth signal. This technique worked pretty well. For the derivative, we used a backwards difference approximation on the proportional data.

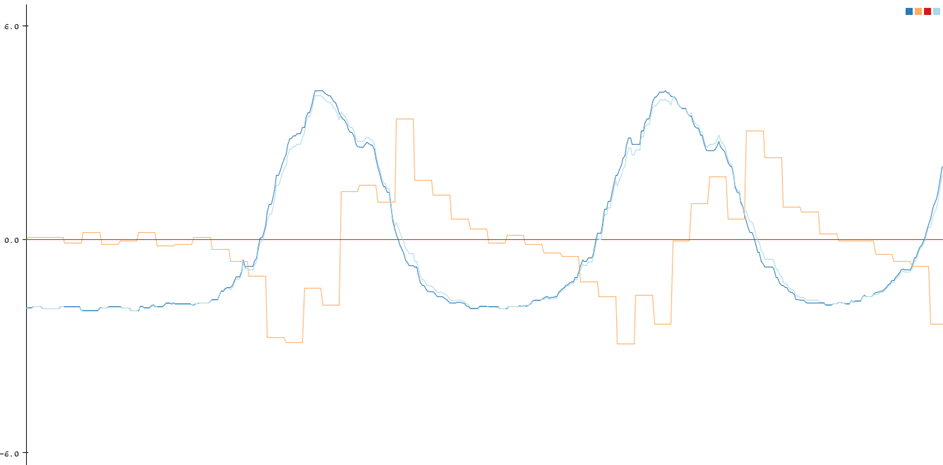
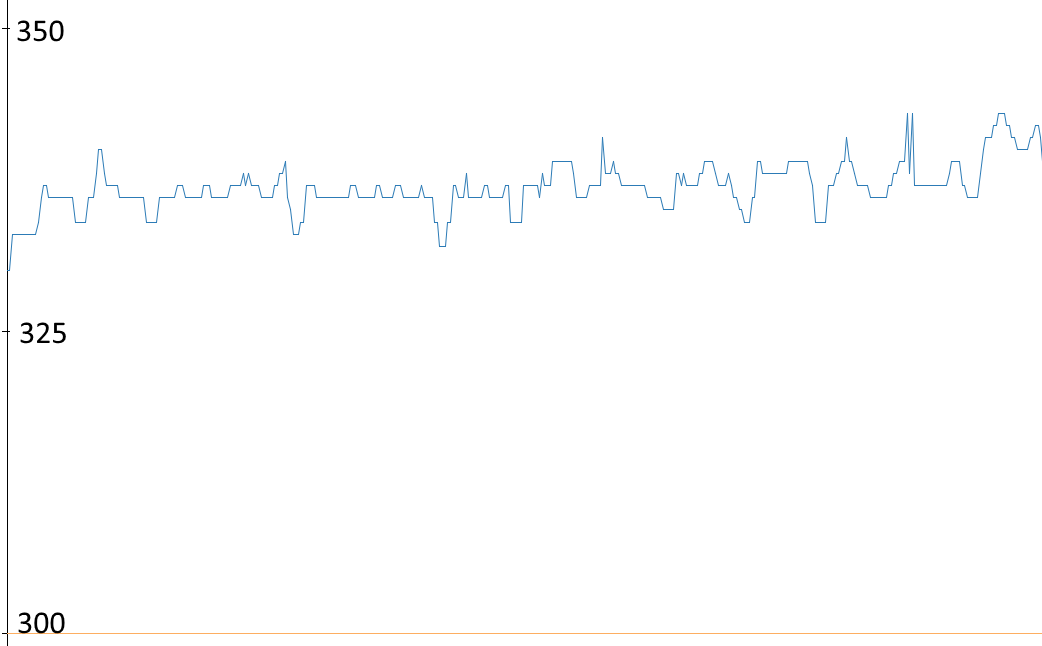


Figure 4: Position and derivative approximation.

Dark blue = position approximation, orange = (negative of) derivative approximation

## Steady State Error

Without an integral term, our system suffered from significant steady state error. By introducing a term proportional to a running total of beam angle error, we were able to correct this problem. In order to prevent this integral term from dominating, we put a limit on its magnitude. Figure 5 shows the effect of the integral term.



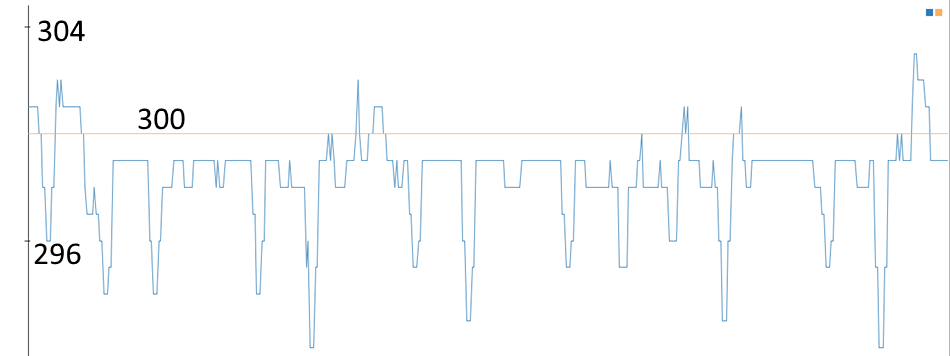


Figure 5: Integral term reduces steady state error.

These plots show ball position without (above) and with (below) the integral term. The desired position is 300.

## Tuning Strategy

We found it was an effective tuning strategy to start with the proportional term and set it to a reasonable value around where oscillation just starts. Then we moved on to the derivative term and finally to the integral term. This incremental approach was essential to our eventual success.

## Steady State Oscillation

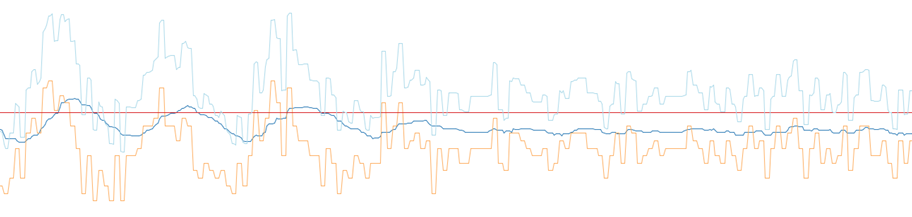
Using the PID controller, the system moderately quickly moves the ball to the correct location. However, there are occasionally oscillations in steady state which appear and disappear unpredictably. Figure 6 shows these steady state oscillations dying out temporarily.

Figure 6: Steady state oscillation.

These oscillations appear and disappear unpredictably.

Dark blue = position approximation, orange = (negative of) derivative approximation

# LQR Control

The next step was to design an LQR control to hopefully better control the system. An LQR controller allows us to pick which states we want to optimize for and control multiple variables like the position of the ball and acceleration of the ramp.

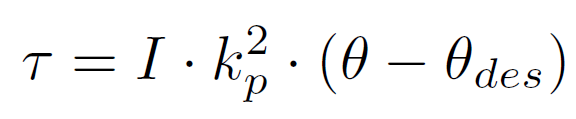
## Second Order System

To begin with, we started with a simplified model, treating our system as only second order. To do this, we pretended the beam was stationary and teleported to various angles to change the acceleration of the ball. This let us bypass some issues with our inputs and get a basic system up and running. Since the two states in the system were the position and velocity of the ball, this turned out to just be a PD controller. We chose to heavily weigh the position of the ball over the derivative since that is the main variable we care about. However this approach tuned the system very poorly. When compared to the manually tuned PID controller, the proportional term was much too large. While there is almost certainly some coefficients that would tune the system well, we already had a working PID controller and decided it would be more beneficial to just move to a more complicated system.

## Servos

In order to do a more detailed model of the system, we had to be able to model our input. While we originally chose servos for their ease of use, they gave us several problems during the modeling process. We assumed the servos were internally controlled by a proportional controller. The speed of the motor is proportional to the difference between the desired angle it is given and the angle it is currently at. We were able to come up with an equation for the torque in our derived equations (see Figure 8), but there are problems with the derivation when the desired angle is nonconstant – which is certainly the case.

Figure 7: Servo



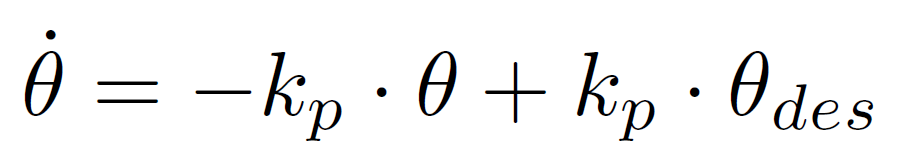


Figure 8: Servo Equations

## Fourth Order System

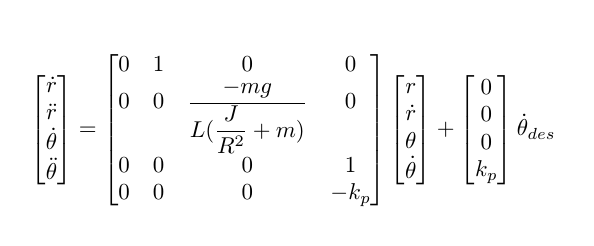
We were able to create a state space model using the proportional model for the servo, but it has some problems (see figure 9). As it turns out, we needed the derivative of both the current angle and the desired angle. So far we had been approximating the current angle as the desired angle, which would make our angular acceleration go to zero. In order to find the actual angle, we would need to add more hardware. It turns out this model is not observable with just the position of the ball. Unfortunately, we had run out of time to expand on our hardware. We would also have needed the proportional coefficient internal to the servo, but that could have been determined experimentally.

Figure 9: Fourth Order State Space Equations

# Conclusion

We successfully built and controlled a ball on a beam system using a PID controller. Our physical system was non-linear and non-ideal and our sensors were noisy. These factors, together with the limitations of PID control result in some steady state oscillation.

In addition, we modelled the system using Lagrangian equations and then created an approximate linear system. Using MATLAB, we were able to create an LQR control law for this system, but were unable to implement for two reasons: the input to the system is desired angle (not torque), and we did not have a method for measuring or estimating the angle of the beam with sufficient precision. To build on this work, we would need to measure or estimate the angle of the beam. We would also need to validate our method for approximating the servo motor torque as a function of desired and current beam angle.