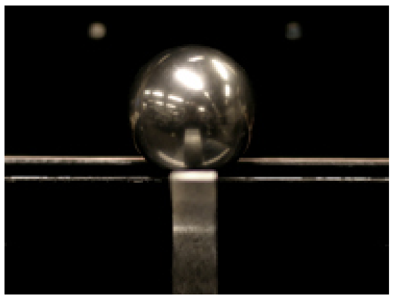
A Robotic Ball Balancing Beam

Jeff Lieberman

February 10, 2004

Abstract



I designed a beam to balance and move a chrome steel 1.25" diameter ball, to a desired position/waveform through time, stably and with high performance behavior. I tested the ball to make sure it could retain its central position, as well as following different frequency sine and square waves of desired position, with or without human disturbances, and it succeeds in these tasks, up to a sine wave frequency of roughly 0.5 Hertz [at an amplitude of roughly 80% of the overall system travel], at which point the linearization of the system model breaks down and the mechanical system becomes unstable.

1 Introduction

The Ball Balancing Beam is a common feedback control system project, due mostly to its ease in construction, interesting asthetics, and its use in learning about applying control to stabilize an otherwise unstable system. The setup is reasonably simple - some sort of beam is mounted on a motor shaft [or commonly through some transmission]. The beam is set to stably hold a ball in one dimension, [usually by being composed of two parallel beams], so that it can only roll back and forth in the one unconstrained dimension. This system is extremely unstable [as will be explained in more detail below], and needs compensation to stablize. Once stabilized, the system is usually commanded to a zero [or centralized] position, and outside disturbances such as people pushing the ball are rejected by the system. After that was accomplished, I decided to test alternate control signals to explore dynamic behavior.

2 Dynamic Model

The ball balancing beam setup is shown in Figure [1] with relevant labels.

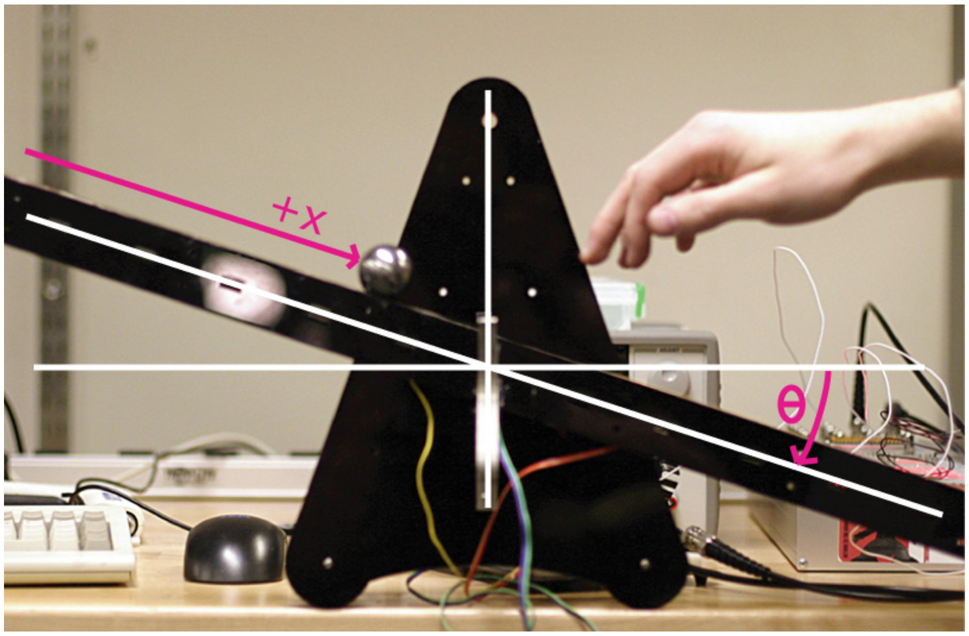


Figure 1: Dynamics of the ball balancing beam system.

As can be seen, the beam angle controls acceleration, not position.

Specifically, from gravity we have

$$F_{gravity} = m_{ball} g \sin(\theta),$$

and since we assume the ball is rolling without slipping, we also have $r\omega = \dot{x}$. The moment of inertia of the solid ball is $J = 2mr^2/5$. Given centripetal acceleration with rotation of the rod as $a_{centripetal} = x\dot{\theta}^2$, altogether we have

$$\ddot{x} - x\dot{\theta}^2 = -g \sin\theta - \frac{2}{5}\ddot{x},$$

which, when we approximate θ small, gives us a transfer function of

$$G(s) \equiv \frac{x}{\theta} = \frac{5g}{7s^2} = \frac{c}{s^2},$$

for $c = 5g/7$. The important thing here is that this transfer function introduces two poles at the origin.

The dc motor inertia follows the typical behavior, introducing yet another pole at the origin from the standard steady state $\omega \propto V$ relation.

3 Sensing Technologies

Two sensors are required for the stability of this system. First, since the angle of the beam is needed to determine the ball's acceleration, I attached a potentiometer along the front of the beam [at the rotational axis]. After calibration in dSpace [described below] this measured the angle of the beam.

The more difficult of the two sensors was by far the linear sensor, which in the end proved the most difficult part of this project. I attempted two different techniques in solving this problem, with usable results in the end, but still with great room for improvement.

The first technique I attempted was a magnetic induction sensor [the full model is shown in figure [2] and a closeup of the sensor loop wiring is in figure [3]]. The two pipes that made the beam were both made of aluminum, and were kept insulated from each other. A long insulated wire was wrapped in a loop inside these wires roughly twenty times, and a large AC signal was run through it. Two wire probes were attached to the two pipes. When the ball was laid on the beam, it formed a flux area concentric to the AC signal, which was

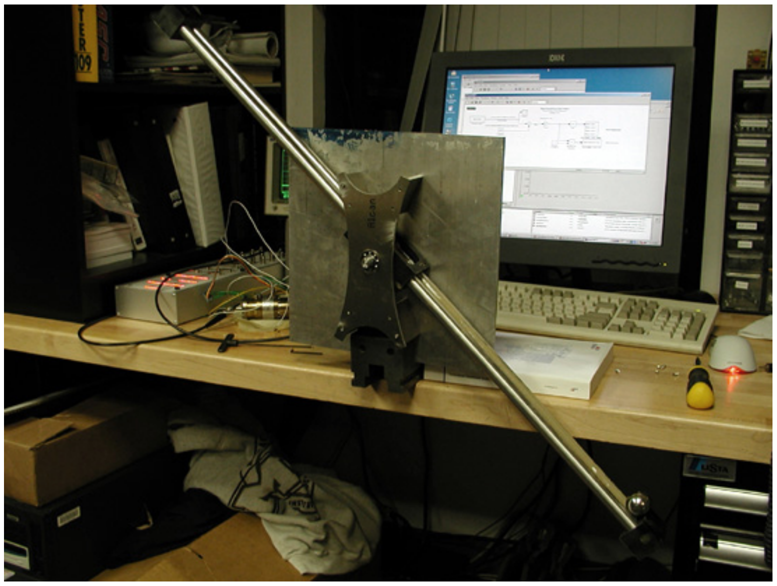


Figure 2: The final first revision induction sensor model.

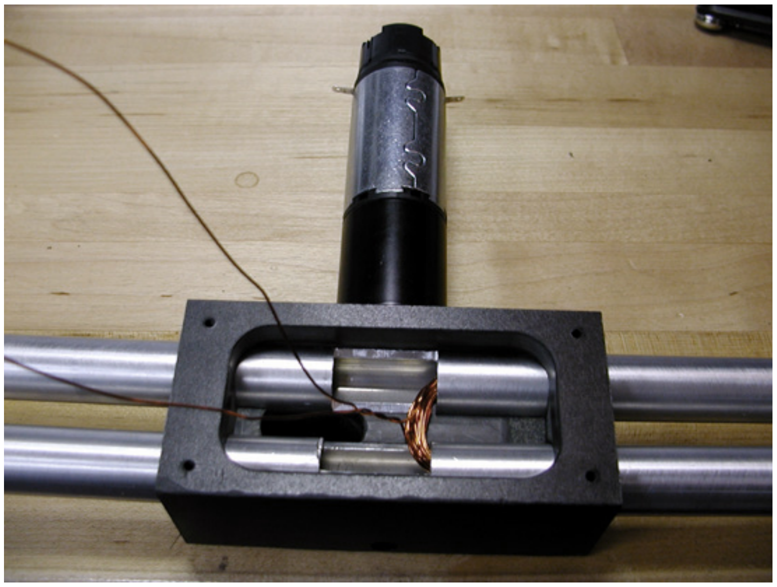


Figure 3: Closeup of the induction sensor for the old model.

[conveniently] proportional to the position that the ball lay along the beam. This signal could be measured to give a measure of position. Running the AC control signal at resonance frequency with the coil [with an added capacitor] greatly magnified the picked-up signal.

Although that sensing technology held promise, it proved extremely noisy as well as untrustworthy, as the signal often disappeared. One cause of this was the rolling of the ball - the roughness of the surfaces and the ball [microscopic but non-negligable] caused many breaks in conductivity of the signal current path. Even after the application of conductive grease and some filtering attempts, the signal was extremely shaky. It would be an interesting sensing type to try in the future.

The next idea for the linear sensor was to use a resistive wire, running along the top of one half of the beam, and a copper wire on the other half, as a linear potentiometer. One end of the resistive wire was held at 5V, the other at ground. Thus, when the ball moved linearly across the resistive wire, it would be held at a voltage proportional to its distance from ground to the 5V rail. The copper wire, also making contact with the ball, would carry that voltage off the ball/beam, where it could be sampled.

For the wire, I used Nickel Chromium, available in a variety of sizes, with the resistance of course scaling down with larger diameter wire. However, to get a reasonable signal at reasonable voltage/current, I used size 28 wire, which has a resistance of roughly 5Ω/ft. This provided adequate signal. The wires were laid on top of two plates of 1/8" acrylic (one plate with the wire being mounted is shown in Figure [3], and a setup of an original test is shown in Figure [4]), the edges of which were beveled to the proper angle so as to be perpendicular to the ball. They were then adhered with super glue, and then sanded down to resurface the conductive wiring [as they needed to be slightly covered with glue to get enough of a hold on the acrylic].

This sensing approach worked, and was the final choice in the design. However, the rolling of the ball [which causes breaks in conductivity] causes extreme noise in the signal. Also, imperfections of alignment reduce signal quality. The signal was usable, however, since derivatives of this signal were needed for compensation, the signal needed to be filtered so as to not cause infinite spikes. This reduced the ability to design a better compensation technique, and the introduction of low pass filters added yet another pole near the origin to deal with in compensation design.

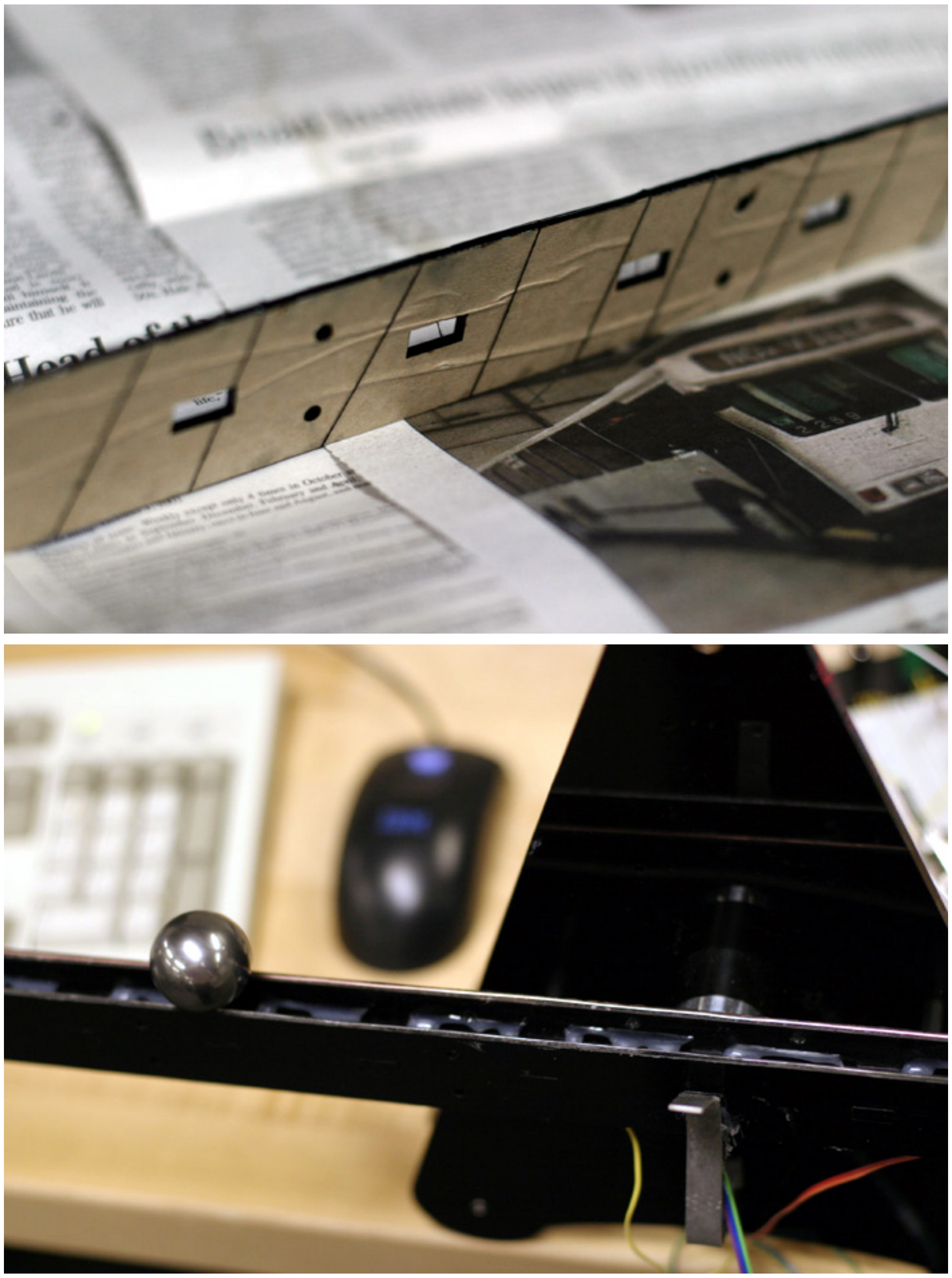


Table 1: Nichrome wire being mounted on one of the acrylic beam panels, and a look at the final sensor setup on the beam.

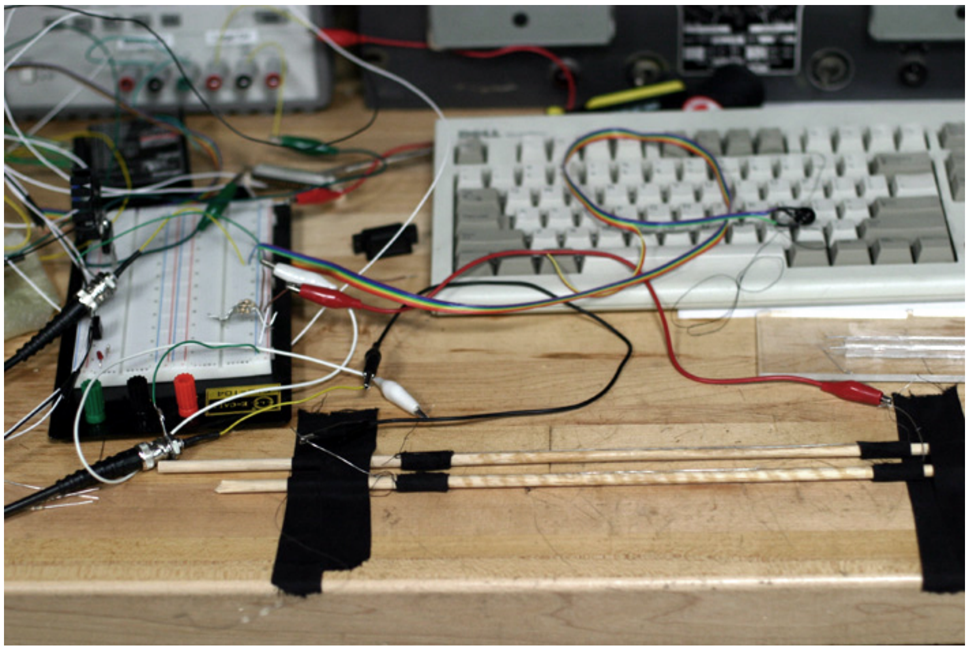


Figure 4: A view of an early test of the nichrome wire as linear sensor.

4 dSpace/Simulink

For feedback design I used Matlab and the Simulink toolbox. The Simulink toolbox allows quick and easy setting up of feedback control systems and testing of their response to different types of input functions, such as a step response.

dSpace is a hardware platform that interfaces with Simulink, to allow real sensory input, and real motor signal outputs. In this way, a theoretically tested feedback controller can be wired up to a real mechanical construction, and then run with real feedback on the real system [in contrast to being built in an analog system with op amps, etc]. This is extremely useful for testing a variety of feedback controllers in a short amount of time, which was done in this case.

Using dSpace and Simulink did cause several problems, however. First, with already noisy data, taking derivatives of the linear sensor measurements caues impulses in the data stream. The initial design to include two derivatives of the input data [explained below] was impossible with this data as the derivatives in this case blew up quickly out of the range of Simulink's variables. Thus, a one-derivative model controller needed to be designed. Possibly with a different averaging method of computing derivatives, better responses could be created.

Also, placing poles in certain locations seemed to present digital problems that it shouldn't have - when attempting to place poles extremely close to the origin, for some reason Simulink would make the signal completely unstable, even though as the pole reached its desired location, there seemed no indication that the designed controller should be approaching any instability. This seemed to be some sort of bug in Simulink, as far as I could find.

5 Compensator Design

The first part of the controller that was designed was an minor loop compenator for the motor to achieve any desired angle. First I found the important motor parameters, namely $R_a = 4.1\Omega$, $k_t = 35.1e - 1$ Nm/A, and $k_w = 24/600$, and including the gear ratio of the motor, roughly 100, the total motor and beam inertia, $J = 5.4e - 6kgm^2$, as well as $k_p = \frac{0.1}{\pi/4}$. The overall transfer function for the minor loop is

$$\frac{\theta}{V} = \frac{1/k_w}{\frac{JR_a s}{k_t k_\omega} + 1} \frac{1}{s}.$$

Instantiating a proportional feedback controller with a gain of 25 caued a nearly first order response with a closed-loop bandwidth of roughly 100 Hz. This bandwidth is far higher than needed in this ball balancing application, which runs at closer to 1Hz given the inertia of the ball.

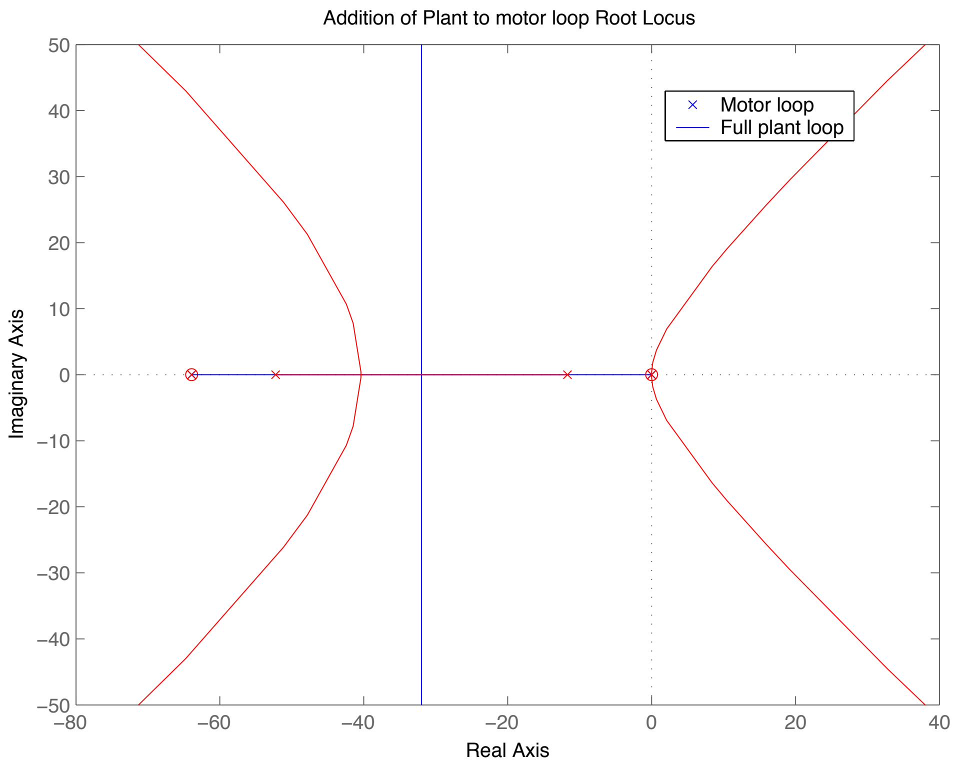


Figure 5: A comparison of the motor closed loop root locus to that of the full plant. Note that there becomes a triple pole at the origin.

Next was the design of the outer loop, in order to control position of the ball by control of the beam angle. Given the limitations found in using dSpace/Simulink to design a controller, finding a stable solution proved difficult. The plant introduces two new poles at the origin, which need to be dealt with to stabilize the system. Figure [5] shows the comparison of root locus methods on the motor closed loop to the full system plant loop. In order to get any reasonable sort of data out of the linear potentiometer [which involved a derivative] the data needed to be filtered. The $\frac{1}{0.1s+1}$ filter achieves this without too much degredation in the bandwidth, but of course adds yet another pole to the system.

By adding a lead compensator of the form

$$\frac{0.1s + 0.01}{0.01s + 1},$$

the phase margin at crossover becomes stable. Although I attempted to improve response with a higher order controller, this would have required another derivative in the position sensor feedback. This was impossible given dSpace and the quality of the linear sensor. Also, trying to place some of the feedback poles toward negative infinity resulted in some unexpected and buggy behavior from Simulink. This would've resulted in less overshoot and lighter ringing response.

6 Results

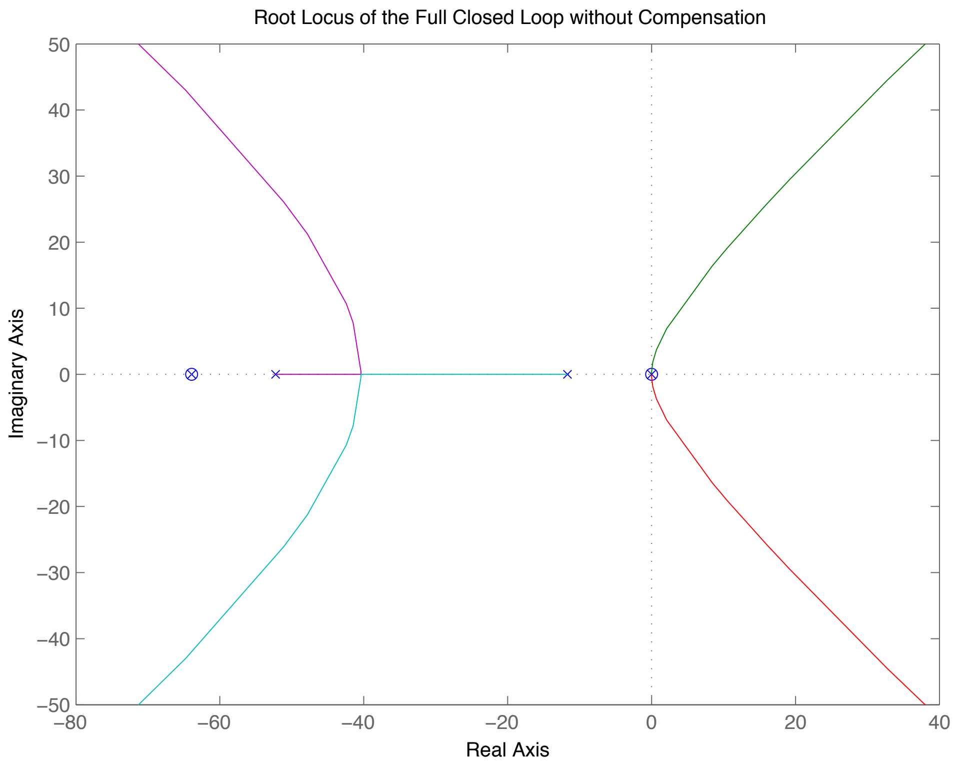
Figure [6] shows the comparison of root locus plots before and after compensation. Notably, the poles from the origin move to the left half plane briefly before crossing back over to instability.

Figure [7] shows the theoretical system response of the beam system, before compensation [giving us roughly 0° phase margin], and after compensation [where it becomes about 60°].

Figure [8] shows the theoretical response to a unit step applied to the beam system, once again before and after compensation. Clearly, before compensation the system is unstable, and after compensation, not only is it stable, but it is reasonably fast, and appears almost critically damped. In reality, differences in motor parameters, nonlinearities in the plan, and sensor variations/offsets change pole/zero locations as well as loop gains. This causes slightly more second-order effects than expected otherwise, which could be fixed by finer tuning of the compensator. However, without fine tuning, the system performs satisfactorily.

The entire feedback system design in dSpace/Simulink is shown in Table [6].

In reality, the system performed quite well. Table [6] shows a couple views of the final system in action. There was modest overshoot, for reasons previously explained. But it's ability for disturbance rejection and for path tracking [sawtooth, sine, and square waves were



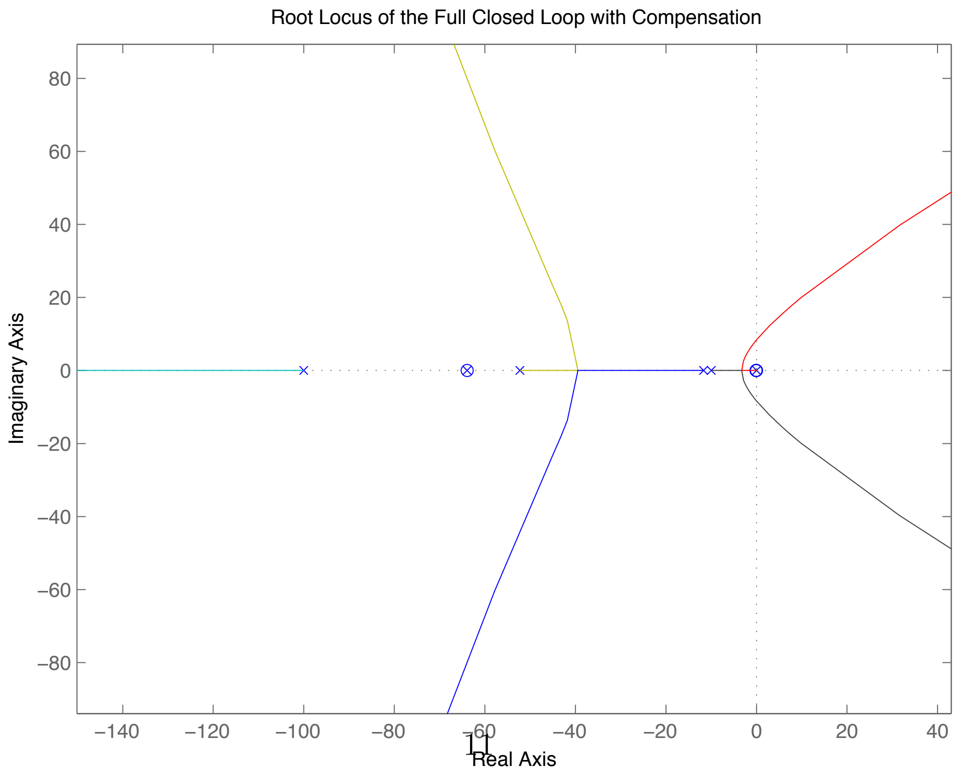


Figure 6: A comparison of root locus plots before and after applied compensation.

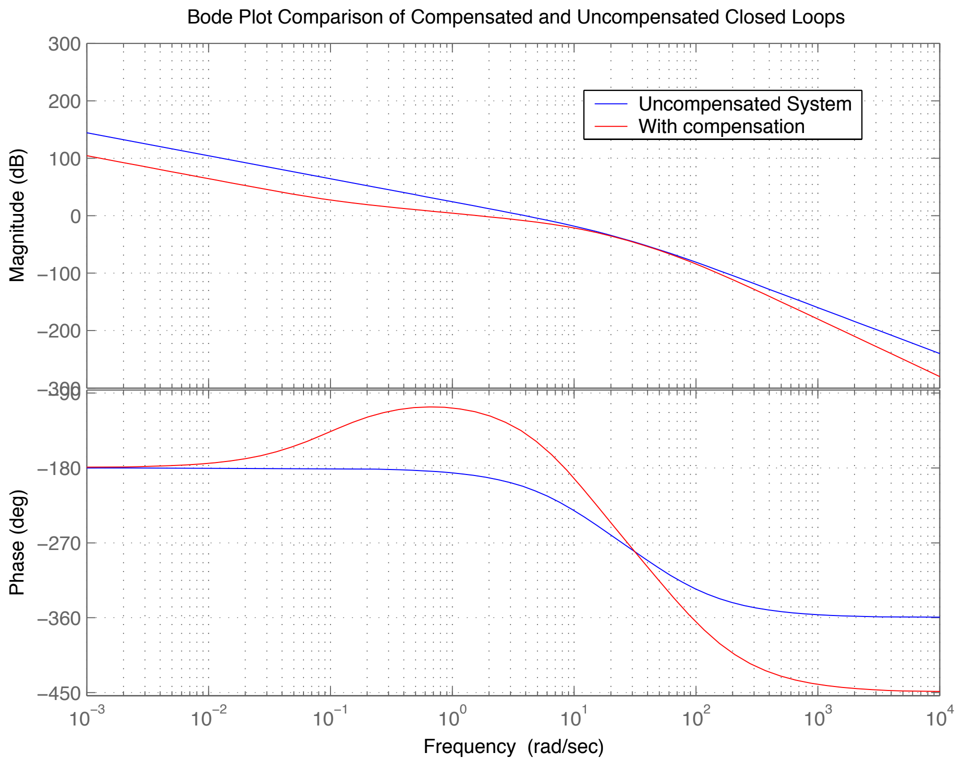


Figure 7: A comparison of system response before and after applied compensation.

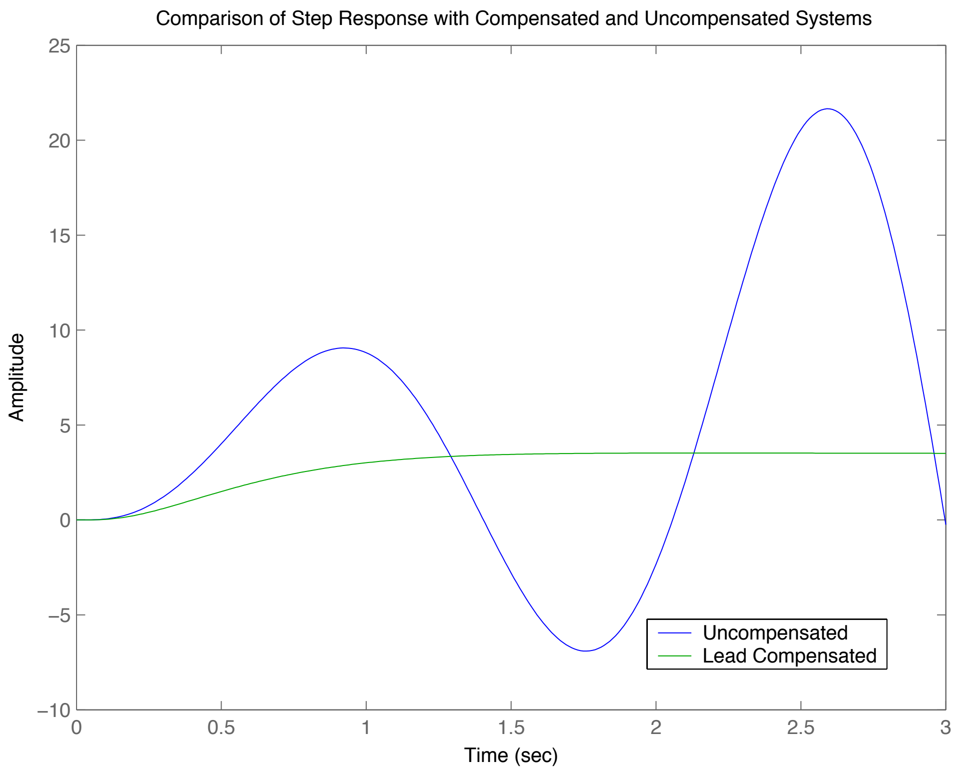


Figure 8: A comparison of system response to a step function before and after applied compensation.

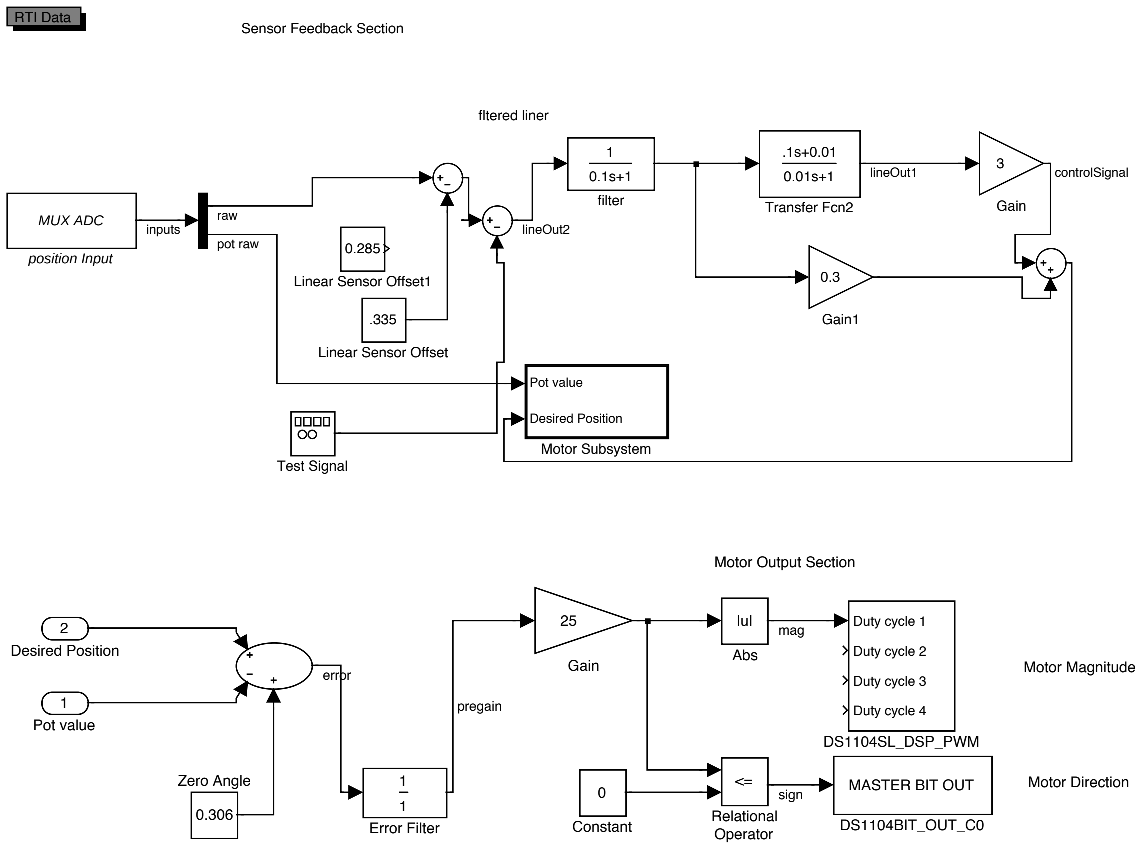


Table 2: The full simulink beam controller model.

tried] was high. For videos of actual system response over time, please view the movie at `http://bea.st/sight/rbbb/rbbb.mov` .

7 Acknowledgements

Many thanks to Jack Holloway, for compensation advice, especially when dSpace/Simulink started freaking out and ideal feedback controllers wouldn't respond.

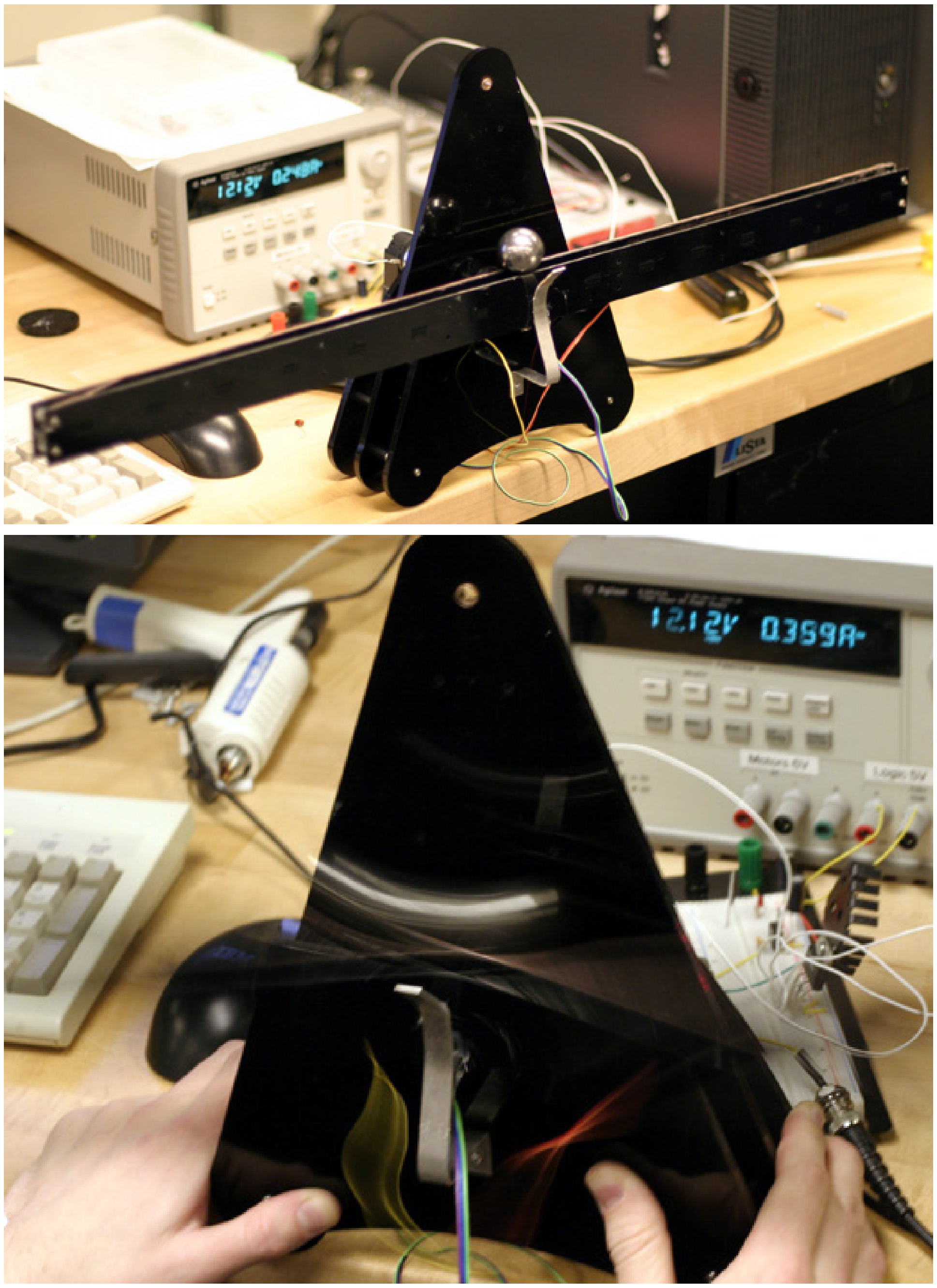


Table 3: Some views of the final model, static and in motion.