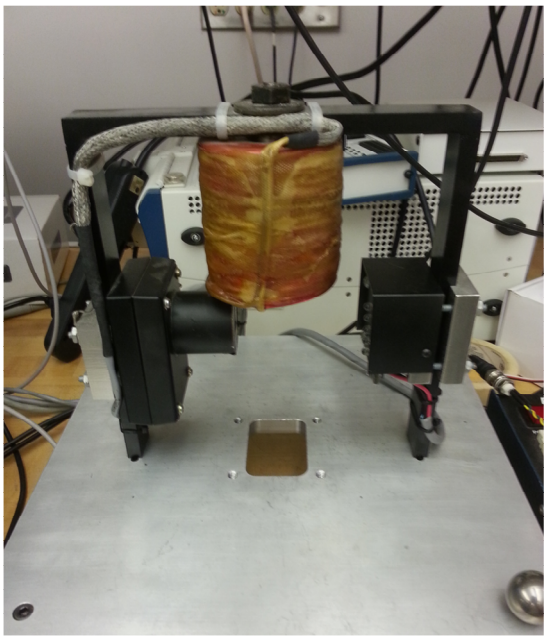
# 2.171 – Final Project Report
# Magnetic Suspension of a steel ball
Michael Everett and Wyatt Ubellacker

**Introduction**

Magnetic suspension is useful in a many applications, most notably in high-performance and high-speed system, where the non-contact and very low friction is desired. However, magnetic systems are inherently unstable and nonlinear, which makes the control of such systems crucial. For this project, we are studying and stabilizing a simple 1-DOF magnetic suspension.

**Hardware**

The plant we are working with consists of a large coil-wound electromagnet with a linear current control amplifier, which translates a 0 to -10V reference to electromagnet current, and a light-based position sensor for determining the height of the steel ball. The output of the light sensor is in the range of 0 to -6V. The figure below shows the physical system.



To create the digital control, we decided to use an Arduino nano microcontroller. We wanted to implement a controller with limited processing power and as a difference equation. This microcontroller contains a 16MHz clock, 0-5V analog inputs, and no onboard DAC. Though the clock is 12MHz, our loop time was ultimately limited to 4KHz. To overcome the lack of DAC, we used a PWM output through a scaling and inverting op amp circuit to drive the linear regulator. A similar strategy was used to scale the sensor values to a readable range.

**Linearization**

The system in question is governed by the following equations of motion:

$$F_{mag} - mg = m\ddot{x}$$

$$F_{mag} = C\left(\frac{i}{x}\right)^2$$

As see, the force of the magnet is non-linear in both x and y. To use the vast array of tools we have for linear control, we must linearize the system. By choosing an operating point and operating current we can arrive at the following linearization:

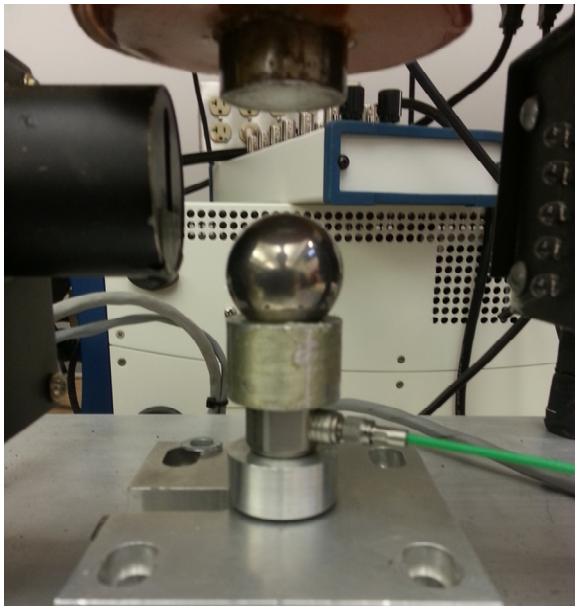
$$F_{mag} = C\left(\frac{i_0}{x_0}\right)^2 + \frac{\partial F_{mag}}{\partial i} * \tilde{\imath} + \frac{\partial F_{mag}}{\partial x} * \tilde{x}$$

Which we can plug in the original system dynamics, and select out operating point to balance the gravitational force to achieve the final linearized system:

$$m\ddot{x} = -mg + C\left(\frac{i_0}{x_0}\right)^2 + \frac{\partial F_{mag}}{\partial i} * \tilde{\imath} + \frac{\partial F_{mag}}{\partial x} * \tilde{x}$$

$$m\ddot{x} = \frac{\partial F_{mag}}{\partial i} * \tilde{\imath} + \frac{\partial F_{mag}}{\partial x} * \tilde{x}$$

To determine the partial derivatives the steel ball was glued to a micrometer and force sensor as in the following figure:



The force was then measure for a range of heights and input currents. For these measurements, the output duty cycle of the PWM was used as a proxy to current, and the analog read bits as a proxy to height.

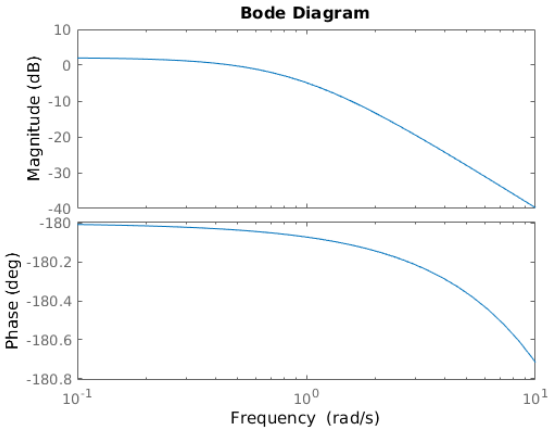
This results in:

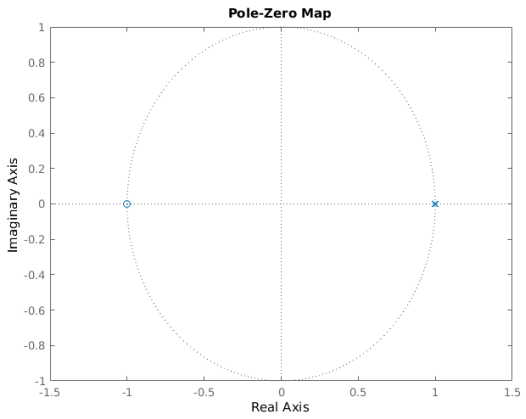
$$\frac{\partial F_{mag}}{\partial x} = -0.81 \qquad \frac{\partial F_{mag}}{\partial i} = 1.034$$

And a final continuous time transfer function of:

$$\frac{PWM}{AnalogRead} = \frac{1.034}{s^2 - .81}$$

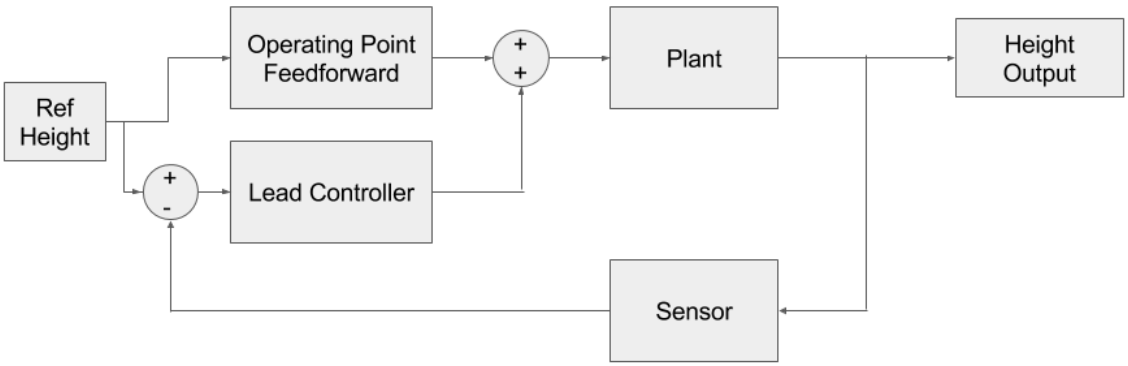
As expected, this transfer function has a naturally unstable pole at .9 in continuous time. For the digital control, this continuous time transfer function's zero order hold equivalent was calculated with a sampling time of 250ms. The resulting bode plot and pole zero map are show below. The plant is still unstable, as expected and will need to be stabilized.




**Controller Design**

After the plant was linearized, we were able to use the linear control methods we have learned throughout the semester.



A lead compensator uses the height error from the two most recent samples, scales by Kp, and outputs a number of bits which compensate for the height error. This error compensation is added to a feedforward term. The feedforward term is the number of bits corresponding to magnet voltage required for levitation at the reference height. This term is the reason why the constant Fmag term cancelled with mg in the linearization.

We designed the lead compensator to have a great deal of phase margin (73 degrees), because this corresponds to high damping and stability. It sacrifices speed of response, but we were more interested in stability than speed for this stage of the project.

We were limited by the resolution of the control effort. To increase crossover frequency, Kp must increase because the plant has a -2 slope in our operating area. But the output of the lead term is multiplied by Kp, so a small error multiplied by a huge Kp means the control effort will only take a few values within the operating range of an 8- or 10-bit DAC. Instead of having a nice linear relationship between error and control effort, the control effort is almost always saturated either high or low. So, we had to decrease Kp, which lowered the crossover and lowered the speed of the response. This was not something we anticipated because theoretical homework problems usually assume continuous, unlimited control effort. Instead, the limited Arduino hardware brought up an interesting, real-life trade-off.

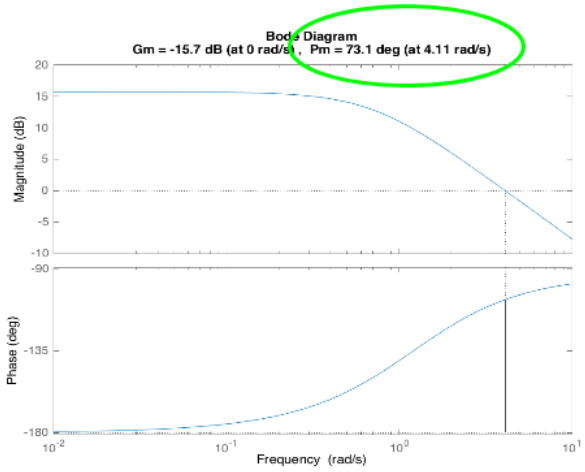
The compensator values are shown in the figure below.

$$C(z) = K_p \frac{z-a}{z-b}$$

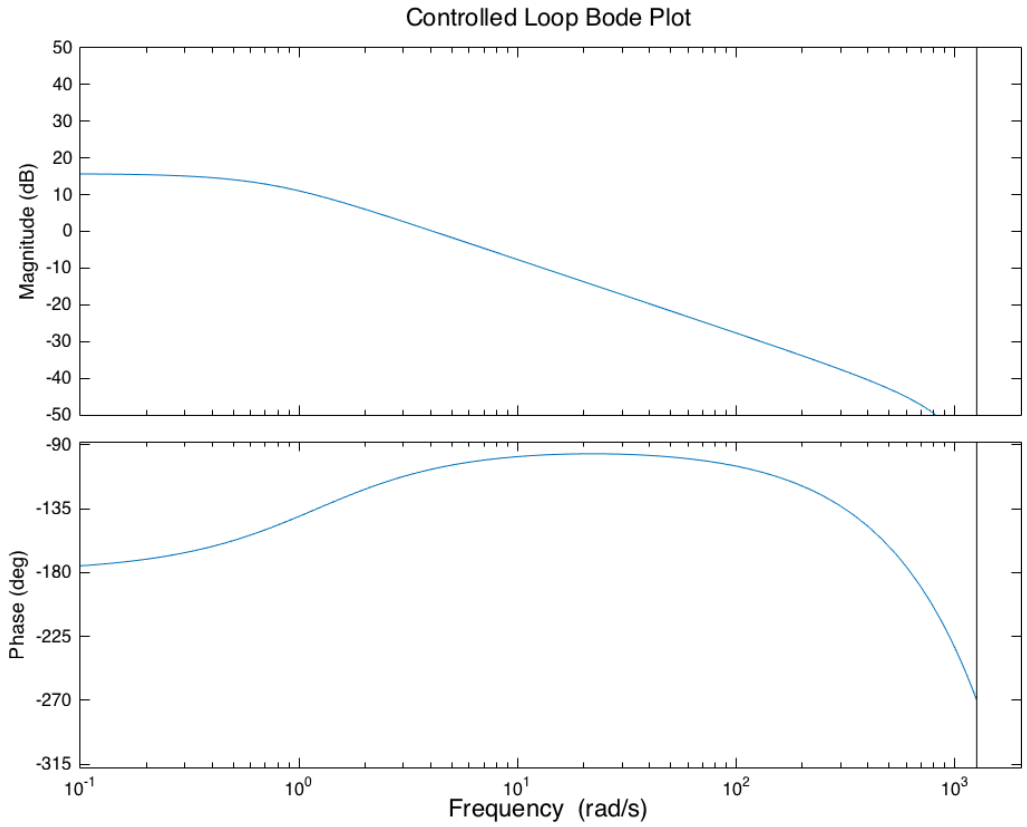
$$a = .997$$

$$b = 0$$

$$K_p = 1600$$

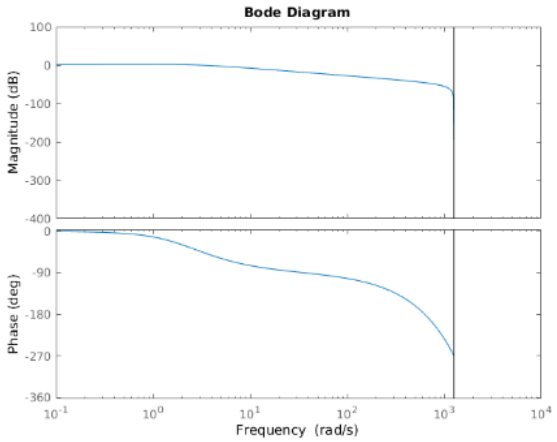


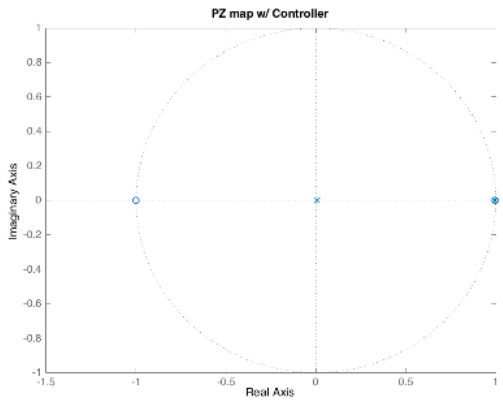

The plot below shows a zoomed out version of the plot above. It shows how the choice of large alpha causes very wide lead compensator phase bump.




The closed loop transfer function shows that the unstable plant pole has moved inside the unit circle, indicating stability.

Closed Loop transfer function, Bode plot, and pole zero map

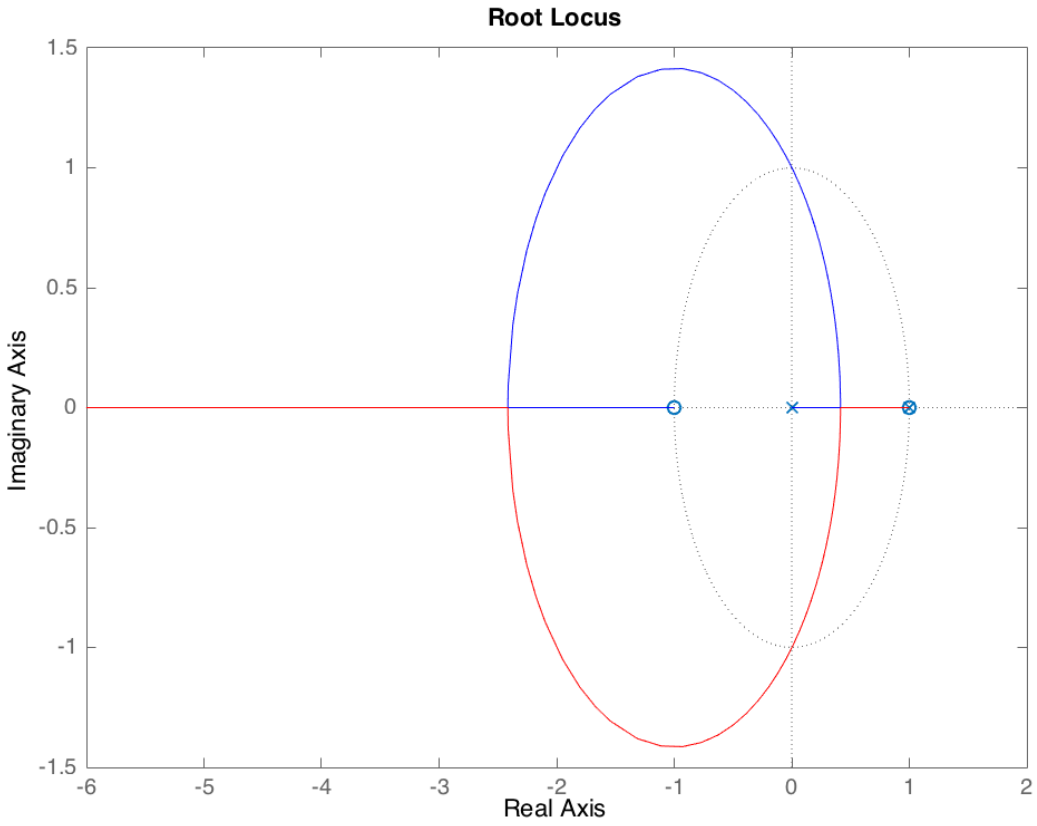



$$\frac{0.00517\,(z+1)\,(z-0.997)}{(z-0.9957)\,(z-0.9939)\,(z-0.005208)}$$

Sample time: 0.0025 seconds

The root locus of the controlled loop also indicates high stability, because it would require a lot more gain to move any of the poles outside of the unit circle.




**Challenges**

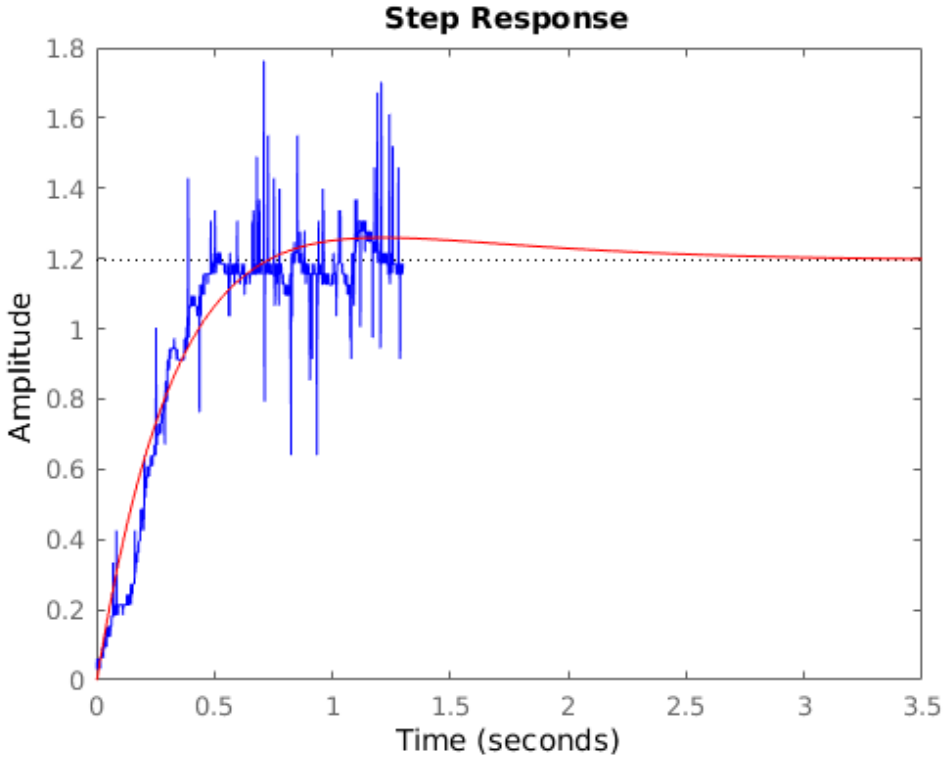
Almost all of the challenges we faced were because of the Arduino hardware.

- No DAC hardware, only PWM
    - Standard PWM is 500 Hz on Arduino
    - Too slow for good approximation of an analog voltage, especially with loop at 4kHz
- Low output resolution via PWM
    - Switched from 8- to 16-bit timer to increase output resolution
    - High Kp causes serious quantization of control effort
- Couldn't RC filter PWM output
- Hard to measure Bode Plot
    - Arduino running 4kHz loop is already pushing its limits
    - Sine calculation too slow
    - Lookup table also too slow

    - Next step is to try either Op Amp sum of sensor + disturbance signal, or voltage divider method
- Unknown ~0.1Hz sensor noise
    - Should add anti-aliasing filter before A/D input
- Unknown ~100kHz sensor noise
    - Maybe from driving linear current controller with PWM

**Results**

| | Simulated | Experimental |
|---|---|---|
| Time Constant | 260ms | 250ms |
| Overshoot | 5% | 3% |
| S.S. error | 20% | 16% |


Step Response
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
0
0.5
1
1.5
2
2.5
3
3.5