Controls Design Project:

Rotary Inverted Pendulum

Forrest Bourke & Evan Dorsky

Due May 4 2015

Abstract

We've made a "twist" on the ordinary inverted pendulum and created a rotary inverted pendulum. It consists of a DC gearbox motor driving the angle of a shaft which has a pendulum with one degree of freedom, perpendicular to the shaft—an example is shown in Figure 1. Our goal for this project was to make a highly polished final product, and put significant effort into tuning our control system based on an accurate mathematical model.

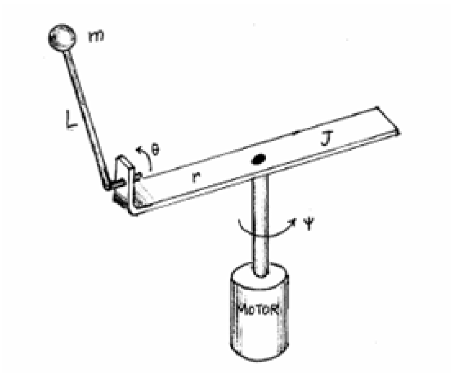


Figure 1: Rotary Inverted Pendulum

1 Plant Data

1.1 Analysis

In our model of the system, we decided to lump the motor driver, DC permanent magnet motor, power supply, gearbox, belt system, and horizontal arm into a single DC permanent magnet "motor" with one inertial moment J, no damping, and negligible inductance. We have also ignored the effects of backlash in our model of the system.

We've determined the parameters necessary for this first-order model: K_e and τ_m.

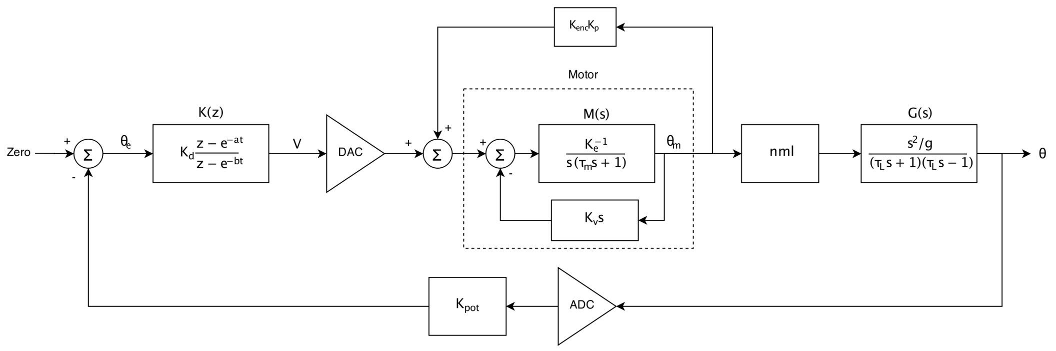
$$\tau_m = 0.0596\text{sec}$$

$$K_e = 0.0111 \frac{\text{V sec}}{\text{rad}}$$

Our first-order approximation for the motor is thus:

$$\frac{K_e^{-1}}{1 + \tau_m s}.$$

The block diagram for our system is shown in Figure 2. The first block, $K(z)$, is our compensator in the digital domain. The constants a and b are the locations of the zero and pole, and K_d is the DC gain of the discrete time compensator, corresponding to K_c, the DC gain of the continuous time compensator.



Zero
θe
K(z)
$K_d \frac{z - e^{-at}}{z - e^{-bt}}$
V
DAC
$K_{enc}K_p$
Motor
M(s)
$\frac{K_e^{-1}}{s(\tau_m s + 1)}$
θm
$K_v s$
nml
G(s)
$\frac{s^2/g}{(\tau_L s + 1)(\tau_L s - 1)}$
θ
ADC
K_{pot}

Figure 2: System Block Diagram

1.2 Modeling

To convert from a continuous compensator (lag or lead) to a discrete time implementation, we used the following z-transform (matched pole-zero).

$$z = e^{sT} \tag{1}$$

$$K_c \frac{s + a}{s + b} \rightarrow K_d \frac{z - e^{-aT}}{z - e^{-bt}} \tag{2}$$

$$K_d = \frac{K_c a}{b} \left(\frac{1 - e^{-aT}}{1 - e^{-bt}} \right) \tag{3}$$

$$K_d \frac{z - e^{-aT}}{z - e^{-bt}} = \frac{V}{\theta} \tag{4}$$

$$\theta K_d \left(z - e^{-aT} \right) = V \left(z - e^{-bT} \right) \tag{5}$$

$$K_d \left(\theta [k+1] - \theta [k] e^{-aT} \right) = V [k+1] - V [k] e^{-bT} \tag{6}$$

$$V [k+1] = K_d \left(\theta [k+1] - \theta [k] e^{-aT} \right) + V [k] e^{-bT} \tag{7}$$

$$V [k] = K_d \left(\theta [k] - \theta [k-1] e^{-aT} \right) + V [k-1] e^{-bT} \tag{8}$$

The last line, Equation (8), shows how we determine the output to the motor, based on the previous output voltage as well as the current and previous positions of the pendulum. This method of continuous to discrete transformation can be extended to other methods of compensation as well.

1.3 Measurement

We took the majority of our measurements using the 64 click-per-revolution encoder built into our motor and the potentiometer in our system. In addition to these measurements, several other system measurements were taken. We manually balanced our pendulums to find their centers of mass and then measured them. The data from these measurements is shown in Table 1.

Table 1: Pendulum Data

Pendulum	Center of Mass
5.0”	0.0936m
6.5”	0.1195m
10” Delrin	0.1651m
10”	0.1944m
15”	0.2659m

In addition to these manual measurements, we had two sensors in the system: a continuous rotation potentiometer attached to the pendulum, and a 64 count-per-revolution quadrature encoder attached to the ungeared output of the motor. With the motor gear reduction of 30:1 and the belt drive reduction of 2:1, the encoder can measure the angle of the output arm with a precision of 960 counts per revolution, or 152.8 counts per radian.

To determine the K_e of our motor, we ran it at different voltages while measuring both the speed and the current drawn from the motor. These data are shown in Figure 3.

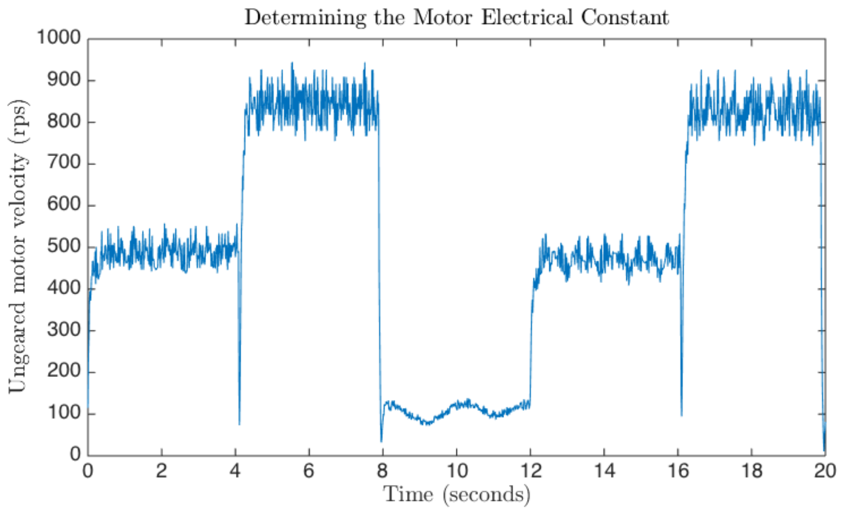


Figure 3: System Step Response

Additionally, we studied the motor reversing direction as a step response to determine the τ_m of our motor, shown in Figure 4 on the next page.

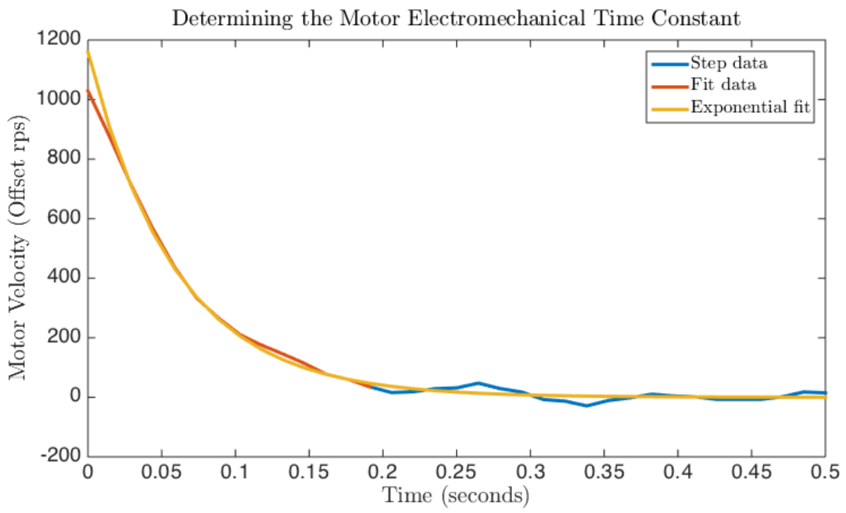


Figure 4: System Step Response

2 Hardware

2.1 Design

Our first iteration of hardware design used 3D printed components, since 3D printing was the fastest and most accessible fabrication method. We had never envisioned the 3D-printed base being the final product, but the rigidity issues with the PLA parts proved to be even more significant than we had envisioned. We quickly moved to redesign the system to be laser cut from 0.25" MDF, which is much more rigid. Additionally, we added gussets to structural areas to further increase the structural integrity of the system. Our MDF cutsheet (shown on the size of the laser bed) is shown in Figure 6 on the following page.

The motor is held into the system by the 6 screws into the tapped holes on the faceplate of the gearbox. These screws (when loose) are free to slide in a 0.5" long slot, allowing the belt to be tensioned by simply moving the motor along the slot and re-tightening the screws. The motor shaft is attached to a 1.00" OD pulley with 12 teeth. We chose XL-series timing belts and pulleys due to the reduced cost and size compared to other timing belt systems. We used an 11" belt to connect to the second pulley, which has a 1.75" OD and 24 teeth. This pulley is screwed into a hollow 5/16" shaft with a wall thickness of 0.049". This wall thickness is clearly overkill for our application, but it allows us to grind large flats into the sides of the shaft—reducing the rotation of all of the set-screw collars used to hold the shaft in place. The shaft is supported horizontally by flanged bearings and vertically by set-screwed shaft collars. The arm is sandwiched between two MDF couplers and two shaft collars, as well as glued onto the shaft. The arm supports the potentiometer, which doubles as a bearing for the pendulum. The pendulum is a press fit onto the flatted shaft of the potentiometer. On the end of the pendulum, there is room for 1/2-20 hardware to add mass to the end of the pendulum. A cutaway of the side of the system is shown in Figure 5 on the next page. A wireframe model of our system can be seen in Figure 7 on page 6.

We investigated direct driving the shaft from the output of the motor. While this would've reduced the backlash we see from the timing belt system, it would have been very

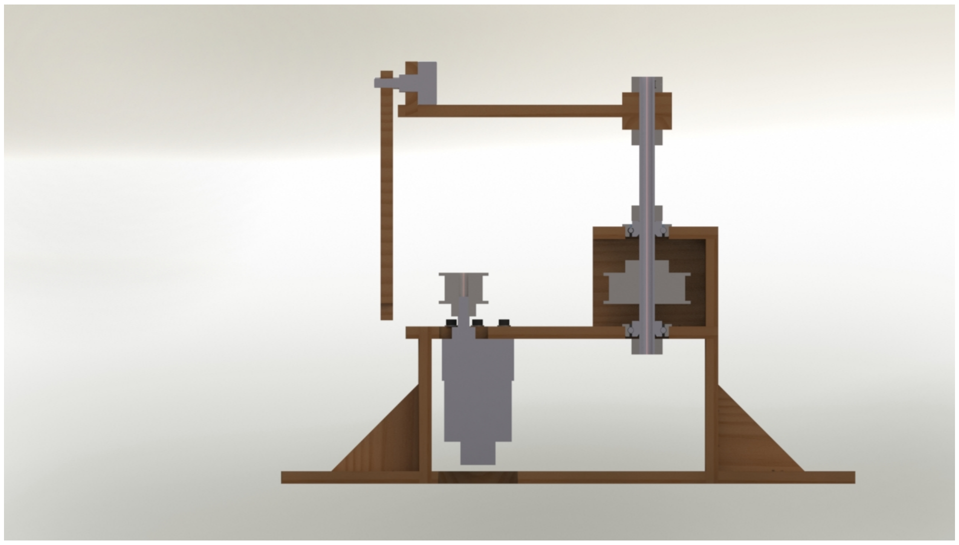


Figure 5: Our system, shown with a section view through the middle

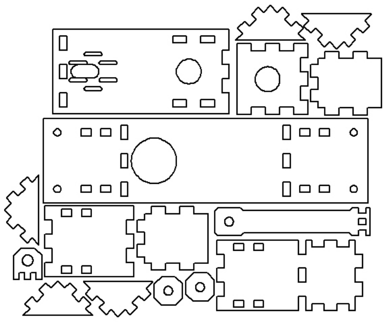


Figure 6: Our MDF Cutsheet

costly, since it required a slip ring with a hollow shaft. We considered using a normal slip ring with our hollow shaft system, but we decided to just run long enough wires that the occasional twisting would not be a factor. This method has fortunately proved reliable.

2.2 Construction

Our final rendered design can be seen in Figure 8 on the next page, and our fully completed mechanical system can be seen in Figure 9 on page 7 (though we have used several pendulums, not included in the picture). We found that while Delrin pendulums were more

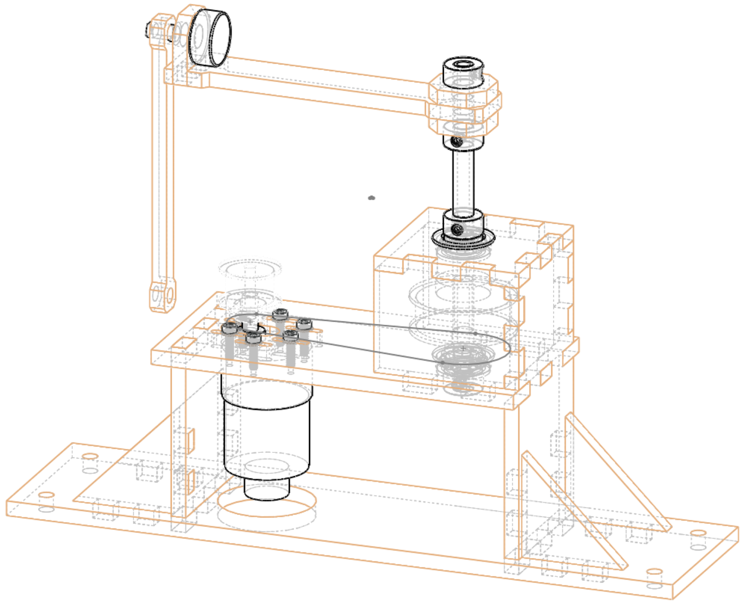


Figure 7: Our system, wireframe rendering

resistant to breakage, they were less rigid and increased oscillations in the unsupported axis of the pendulum. After testing with delrin pendulums, we implemented safety measures in software and moved back to MDF pendulums.

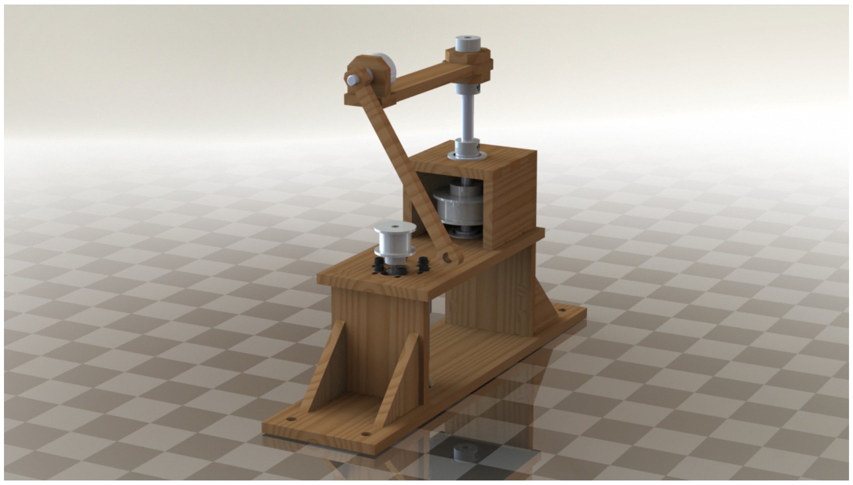


Figure 8: Our system, rendered from CAD files that we used to laser cut the parts

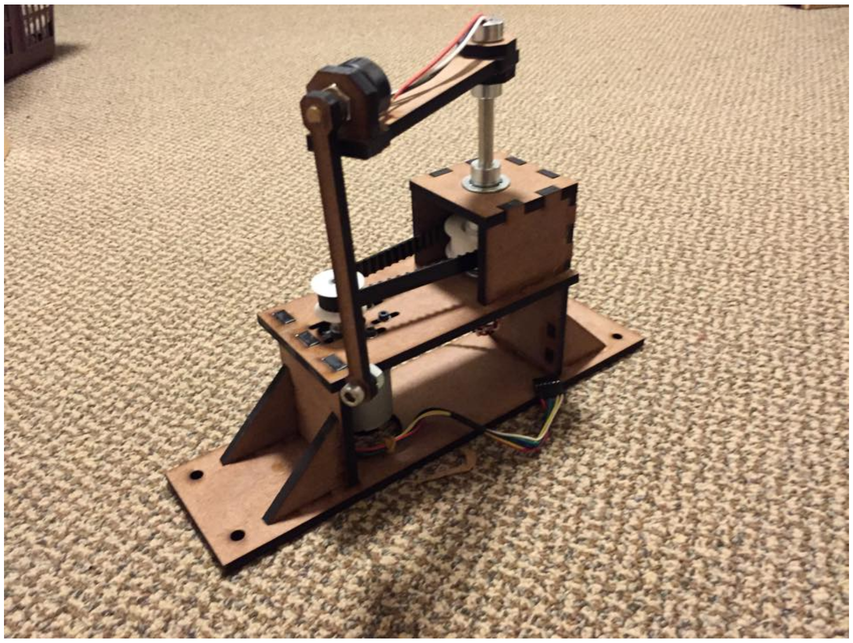


Figure 9: Our system, in real life!

3 Final System

3.1 Compensation

Our tuning process for the final system involved inputting physical constants to a matlab simulation, which allowed us to develop a continuous compensator. While Matlab conveniently provides `c2d` function, we wrote our own matched pole-zero code to ensure that the discrete DC gain corresponded to the continuous DC gain. Our microcontroller code is designed so we can directly plug in the constants produced by our Matlab script.

3.1.1 Matlab Overview

This section will use the same constant names as the block diagram in Figure 2 on page 2. In Matlab, we use the data we collected on the motor to find the motor transfer function

```
%% motor tf
Ke = .0111; % V*sec/rad
n = 1/30; % gearbox reduction
m = 1/2; % pulley reduction
Larm = .1332; % m length of the horizontal (5.245in)
Ke = Ke*Larm/n/m; % corrected (and linear) motor constant
tauM = .0596; % sec
M = 1/Ke/(s*(1 + tauM*s)); % X/Vin
```

We then used the motor transfer function, `M`, to implement minor loop feedback:

```
% velocity minor loop
kV = .04;
enc = kV*s;
Mc = 1/(1/M + enc); % X/Vin
```

We added position positive feedback like so:

```
% position positive feedback
% 960 counts per motor rot -> 480 per arm -> 0-2pi
kPosConv = (2*pi)/960;
kP = 0.2;
Mpc = 1/(1/Mc - kP)
```

Followed by a manually tuned lag compensator (and a continuous to discrete transformation using the Matched Pole-Zero method):

```
%% series compensator tf (lag)
T = 1e-3;
tauA = 1/8;
tauB = 1/.3;
Kc = sqrt(10)*db2mag(2.4+4+4.5);
Ktest = Kc*(tauA*s + 1)/(tauB*s + 1)
K = Ktest;
eaT = exp(-1/tauA*T)
ebT = exp(-1/tauB*T)
Kd = Kc*tauB/tauA*((1 - eaT)/(1 - ebT))
KMp = K*Mpc;
```

And the pendulum transfer function, based on the measured length, which combines with the compensated motor transfer function into the overall system transfer function:

```
% pendulum tf
g = 9.8; % m/sec^2
% l = 0.0936; % m (5", broken MDF)
% l = 0.1195; % m (6.5")
% l = 0.2659; % m (15")
% l = 0.1944; % m (10")
% l = 0.1651; % new delrin 10"
l = 0.18923; % new mdf 10"
tauL = sqrt(l/g); % sec
tauL = sqrt(l/g); % sec
G = -s^2/g/((tauL*s + 1)*(tauL*s - 1)) % Theta/X

% Overall system transfer function
Sys = KMp*-G; % X/Theta (- because G is -)
```

3.1.2 Compensated System Properties

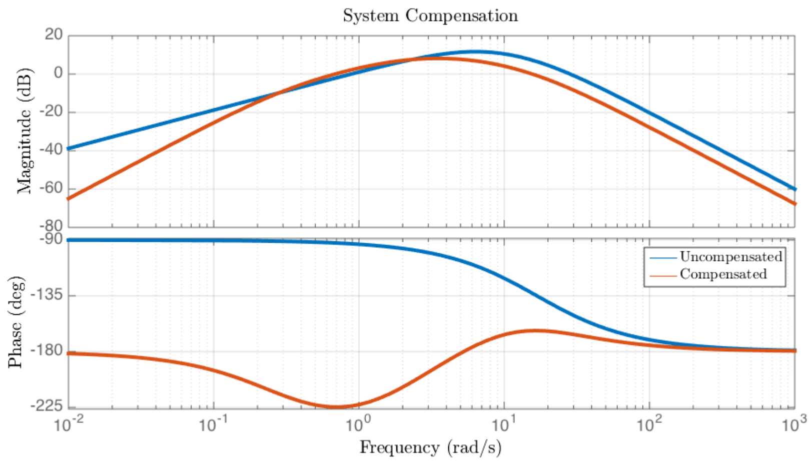


Figure 10: The compensated system with improved low frequency crossover phase margin (red) and the uncompensated system (blue)

Figure 10 shows the effect of the compensator on the system frequency response. The positive position feedback moves the motor pole away from the origin, reducing the low-frequency phase to -180°. We positioned the pole and zero of the lag compensator to place the negative phase bump at the lower frequency crossover point, maximizing the gain while still maintaining an acceptable phase margin. The low-frequency crossover phase margin is 43° (negated because the crossover is up-going instead of down-going) and the high-frequency crossover phase margin is 28°.

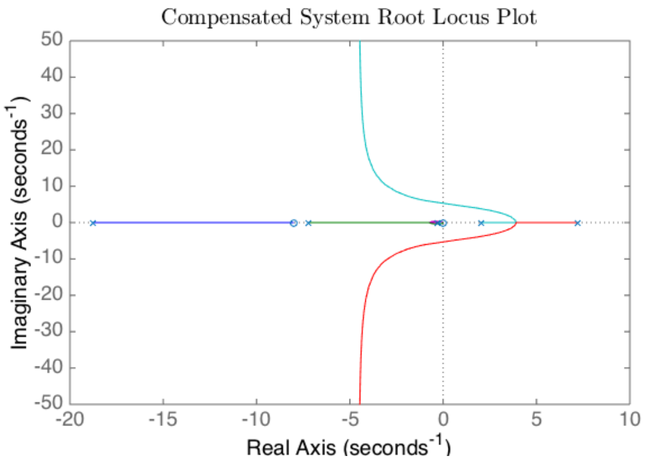


Figure 11: Root locus showing all poles moving into the LHP with increasing gain

The root locus plot (figure 11 on the previous page) shows that with this compensator, the system will be stable if the gain is high enough. Additionally, the nyquist plot (figure 12) of the compensated system shows two counter-clockwise (negative) encirclements of the -1 point. As the open loop transfer function has two poles in the RHP, the nyquist plot shows that the system is stable.ve

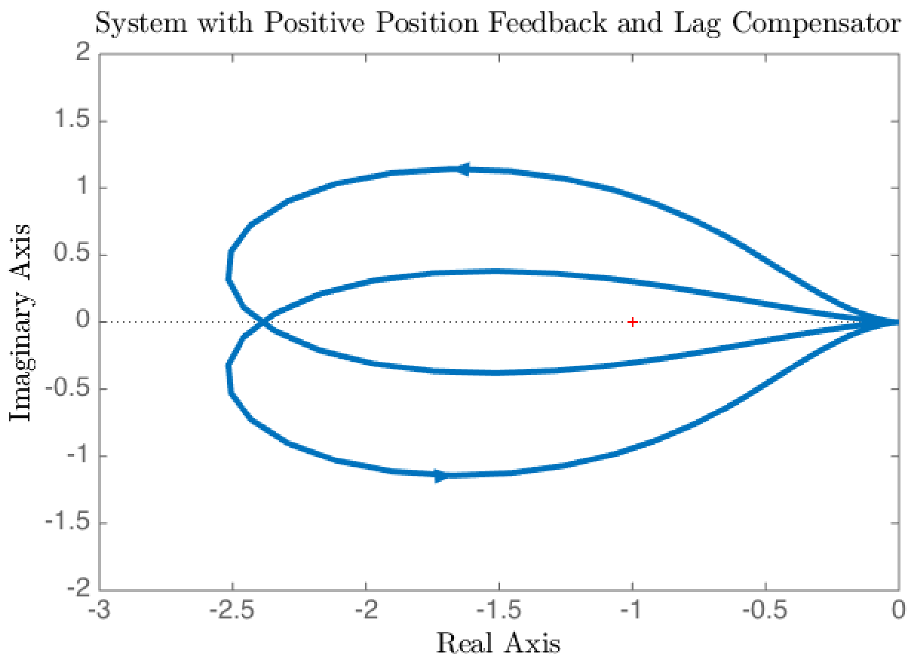


Figure 12: Nyquist plot showing two counter-clockwise (negative) encirclements of the -1 point

3.1.3 Microcontroller Overview

Our microcontroller code must perform the following functions:

- *Update motor voltage output from transfer function*
- *Check interrupt B to determine the motor speed*
- *Check interrupt A to determine the motor direction*
- Set the motor speed and direction
- Communicate with the attached computer over serial

The italicized functions require precise timing and are attached either to hardware interrupts (for the encoders) or a software timer interrupt in the case of the transfer function update loop. The serial runs at 115200 baud to reduce the time wasted sending data to the computer—for further increased performance, serial-based output can be disabled altogether. Setting the motor speed and direction is simply performed as frequently as possible.

```
void control() //runs every 1000uS
{
  posFb = mpos*kPosF; // kPosF determined by matlab
  // if (pulsetime)
  //   mvel = mdir*1.0/((float)(pulsetime*32e-6))*.0022;
  thetak = analogRead(pot)*kP - theta0;

  Voutk = Voutk1*.99970004 + 7857.1779*(thetak - .99203191*thetak1) + posFb/* - .01*mvel*/;

  drivedir = Voutk < 0? BACKWARD : FORWARD;
  // Motor output must be clipped later
  thetak1 = thetak;
  Voutk1 = Voutk;
}
```

The constants in the difference equation with `Voutk` are from the output of the continuous to discrete conversion in matlab.

3.2 Performance

We found we experienced diminishing returns with pendulums of excessive length (i.e. 15 inches and longer). We believe the reason for this is the fact that we do not take gyroscopic effects into account in our compensator. Longer pendulums also wobble more in axes that we can't control.

We are unfortunately still unsure of the effect of doing floating point math in our real time calculations. Floating point math is significantly slower than integer math on our ATMega328P, but we believe the calculations finish in the time of the control loop. Additionally, reading the onboard ADC is very slow—moving to integer math will not speed up the ADC reading.

Additionally, the motor draws between 3 A and 4 A when switching directions, which is taxing for a power supply that may be sufficient for running the motor continuously in one direction. While our model does not take this clipping of the supply rails into account (which limits transient response rate), the system performs adequately despite the current limiting issues.

Currently, our system is fully stable when subject to very small disturbances (of a few degrees). With more tuning of the position feedback loop, we believe we could make the system robust to larger disturbances. The system initially recovers from a larger disturbance, but the position loop subsequently diverges, causing the pendulum to fall.