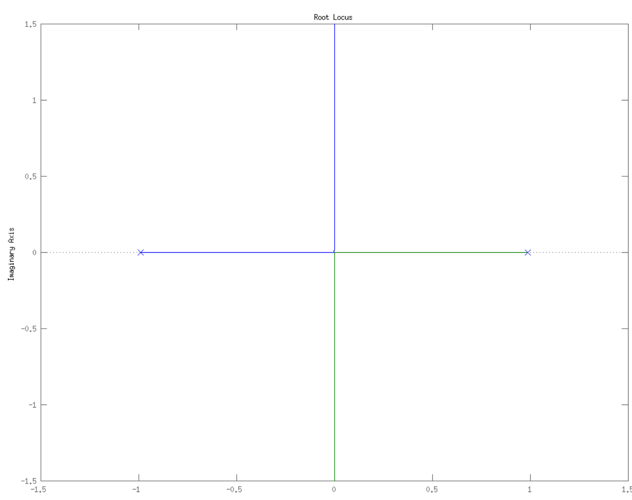
Problem 1

Consider an inverted pendulum with all mass centered at the end of the pendulum. Take the input to be an applied torque at the base of the pendulum, and the output to be the angle of the pendulum. Use Evans' method to analyze the performance of a proportional controller in unity feedback.

$$\begin{aligned}
I\ddot{\theta} &= mgl\sin\theta + T \\
ml^2\ddot{\theta} - mgl\theta &= T \\
l\ddot{\theta} - g\theta &= T \\
\Theta(s) &= \frac{1/l}{s^2 - g/l}T(s)
\end{aligned}$$



part b

Relate what you seen in the root locus to what happens in the physical system as k changes from 0 to ∞ (there are three separate "regions" that you should analyze.)

If the controller is "weak," then it won't overpower gravity, and the pendulum will still fall (but more slowly, depending on how strong the controller is. If the controller is "strong," then it will beat gravity, and push the pendulum back to the upright position. Of course, because the controller torque is proportional to the deviation, it continues to push until the angle crosses zero. This causes oscillations, so the poles are imaginary. In between these regimes, if the controller exactly *cancels out gravity, then the system is a double integrator. The physical behavior is an unstable "drift"; the impulse response is a ramp.*

part c

You decide to implement your controller. Unfortunately, the amplifier you use to map the error signal to the control input is not perfect: in particular, it isn't very accurate at high frequencies (we'll see how to interpret this using Bode plots next week). Instead of being

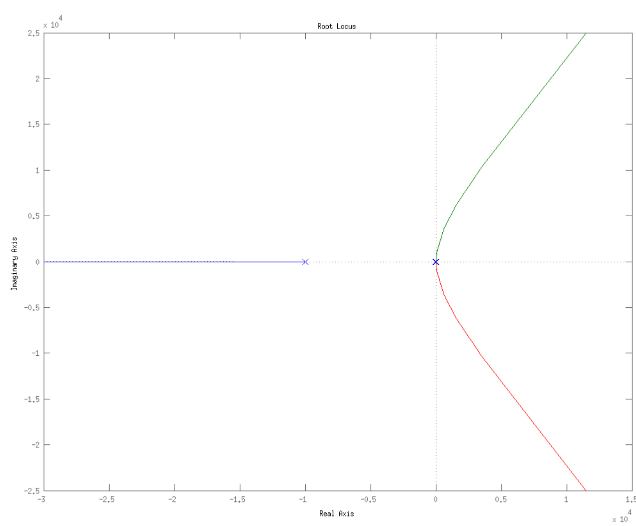
$$K(s) = k$$

the controller is

$$K(s) = \frac{k}{s/\tau + 1}$$

where $\tau \gg 0$ is the time constant of the controller. First, sketch the step response of this controller. Then plot the root locus of the closed-loop system. How does this relate to parts a/b?

No matter how "fast" the pole is, the closed loop poles are always in the open RHP (except for the double integrator case). Because the controller now has some "lag," the system goes unstable. Note the scale. For really fast poles, the RL is locally similar to part a.

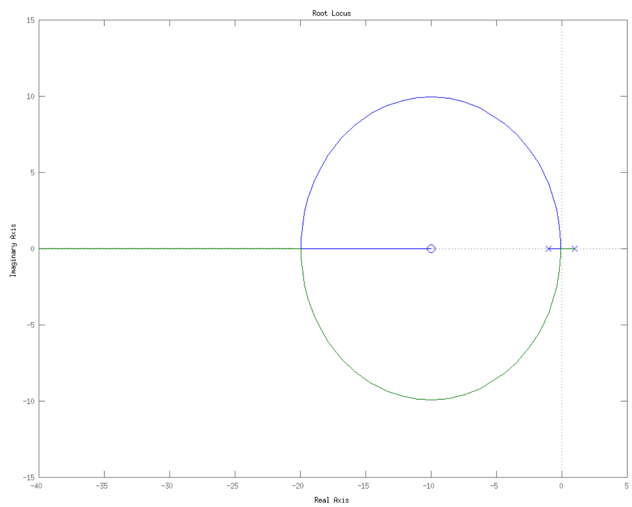


part d

A proportional controller is probably not the best controller for this plant. Use intuition from Evans' root locus to design a better controller.

Try making some zeros to "attract" the poles over to the LHP. A PD controller works:

$$K = k(s + 10)$$



Note: it doesn't really matter if you keep the fast pole here or not, the result is pretty much the same.

part e

If time permits...

The lift and tail forces on an aircraft generate torque around the center of mass. A (very) simplified model is that the torque is proportional to the *angle of attack* (pitch angle) of the aircraft, so

$$T = C\alpha$$

where T is the torque, α is the angle of attack, and C is some constant. Compare the mathematical model to that of a pendulum, assuming that the pilot can use control surfaces to apply an external torque to the aircraft.

The equation of motion is

$$I\ddot{\alpha} = C\alpha + T_{app}$$

so the transfer function is

$$A(s) = \frac{1}{Is^2 - C} T_{app}(s)$$

So this is the same as the pendulum system: for $C > 0$*, the poles are at* $\pm\sqrt{C/I}$ *(inverted pendulum). For* $C < 0$*, the poles are at* $\pm j\sqrt{C/I}$ *(hanging pendulum).*

part f

The NASA X-29 test aircraft was designed so that $\sqrt{C/I} \approx 38$ Hz. How long does a pendulum have to be so that the poles of the inverted pendulum system are at the same location as the X-29's poles? Do you think it was possible to fly this aircraft manually?

$$\begin{aligned} p &= \sqrt{g/\ell} \\ \ell &= g/p^2 \\ \ell &= 9.8/6^2 \\ &\approx 27cm \end{aligned}$$

If you try this out, you find that it's possible to balance such a pendulum manually for a dozen seconds or so, but it requires your full attention, and would be quite dangerous on an aircraft. The MATLAB *code for this problem is listed below*

```
clc
clear
cd ~/Dropbox/E105-Win2013/Assignments/assignment6-nick/section/

% PART A
g=9.8;
l=10;
s=tf('s');
P=(1/l)/(s^2-g/l);
rlocus(P)
print -dpng p1a.pdf

% PART C
rlocus(P/(1/10000*s+1));
print -dpng p1c.pdf

% PART D
rlocus(P*(s+10));
print -dpng p1d.pdf
```