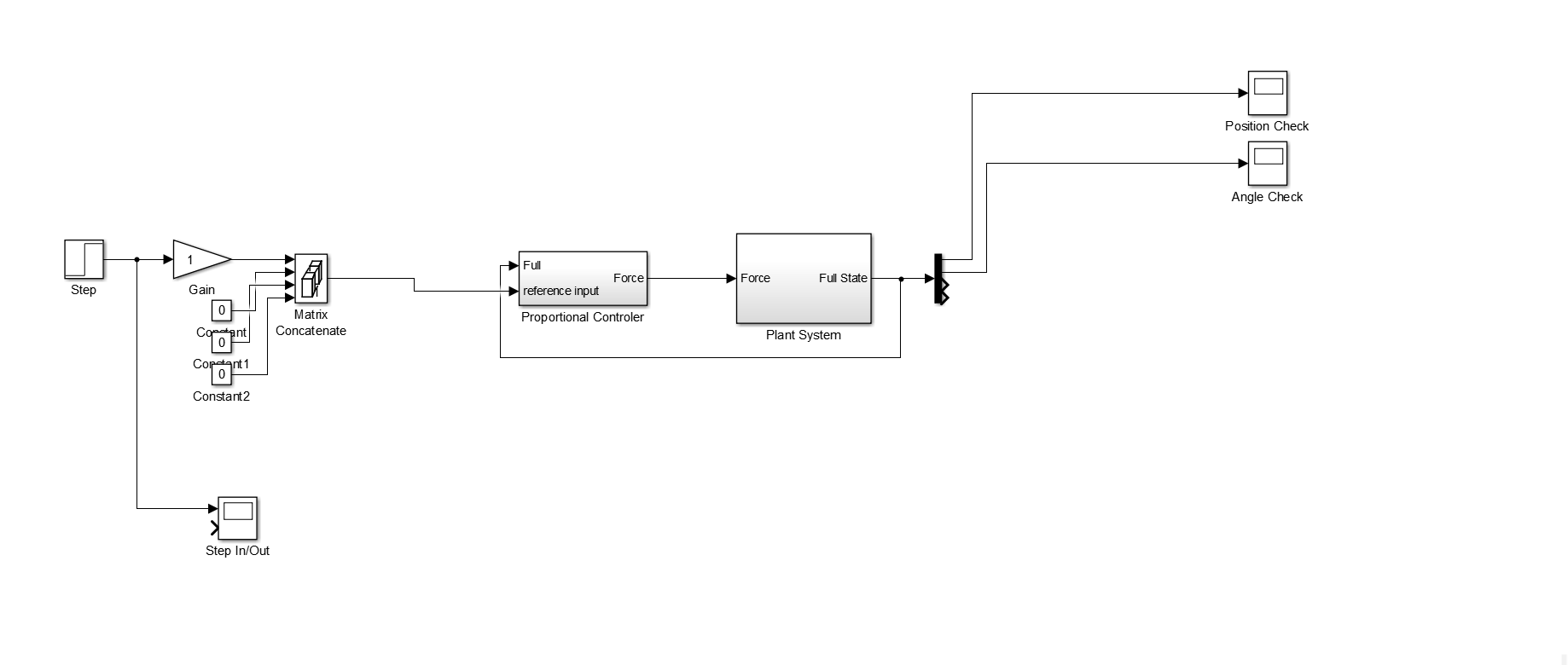
# Lukas Gemar, ES158, Lab 4 II

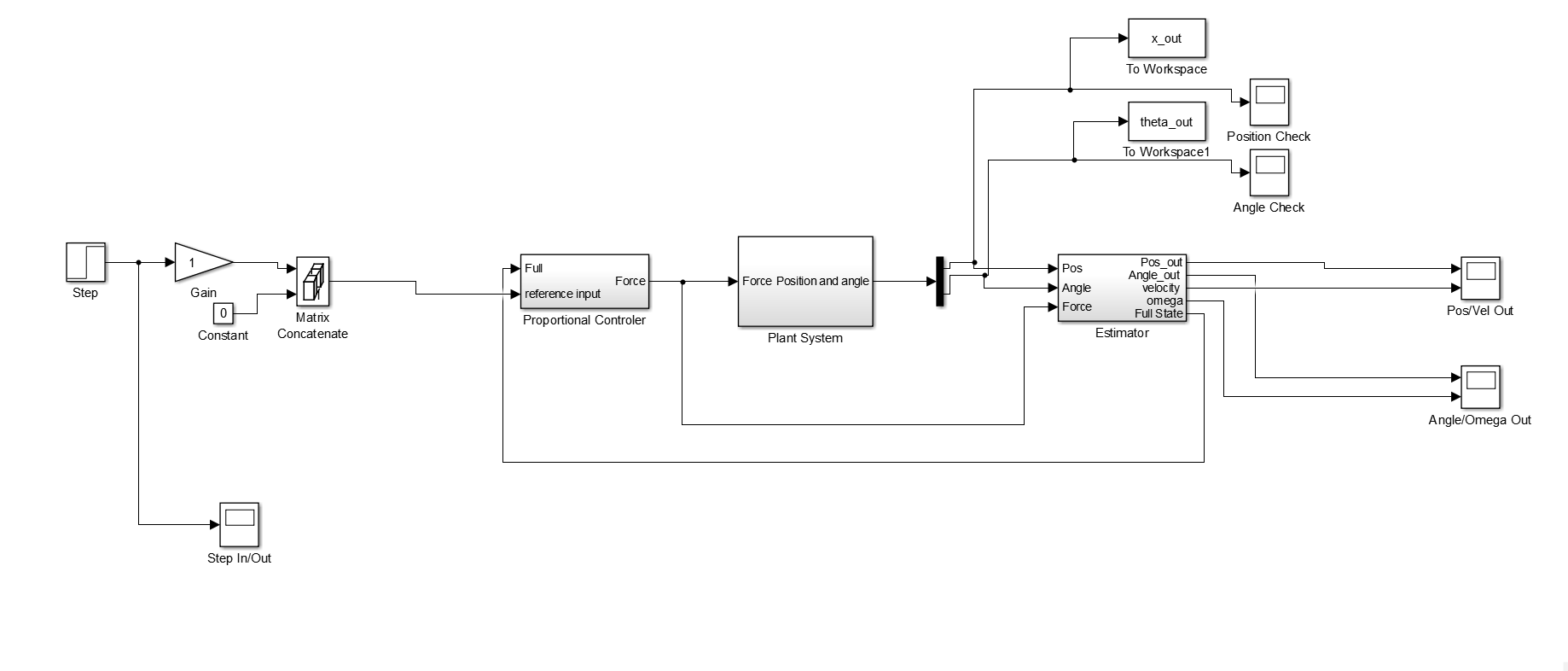
# State Feedback Controller

## System Diagrams

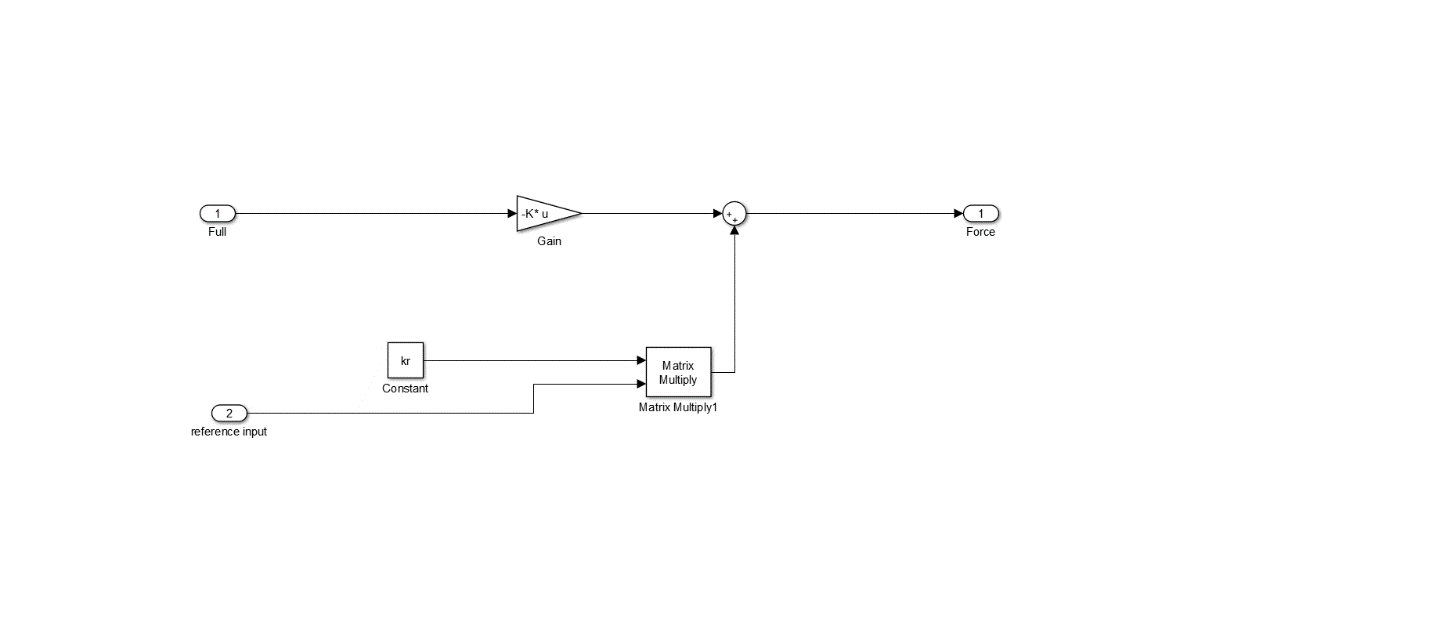
### State Feedback Controller (without estimator)



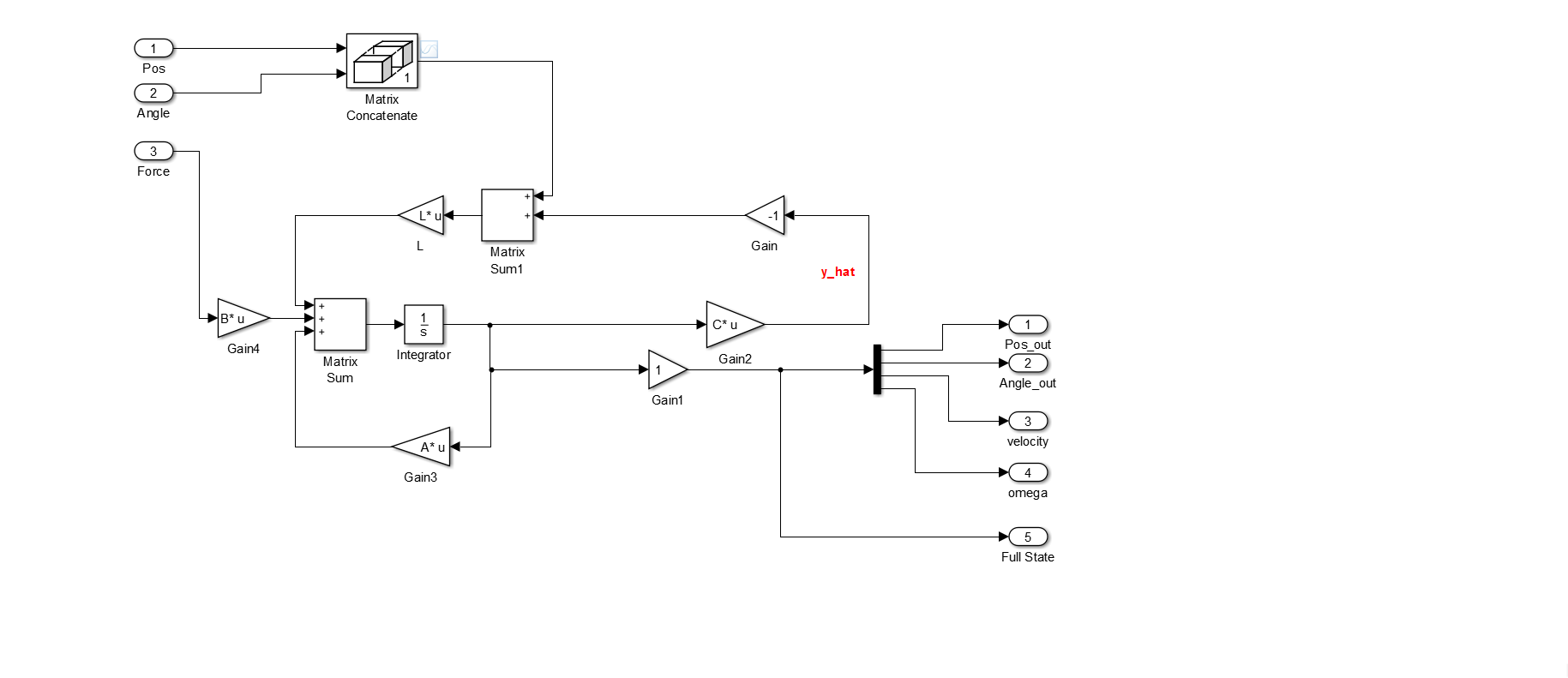
### State Feedback Controller (with estimator)



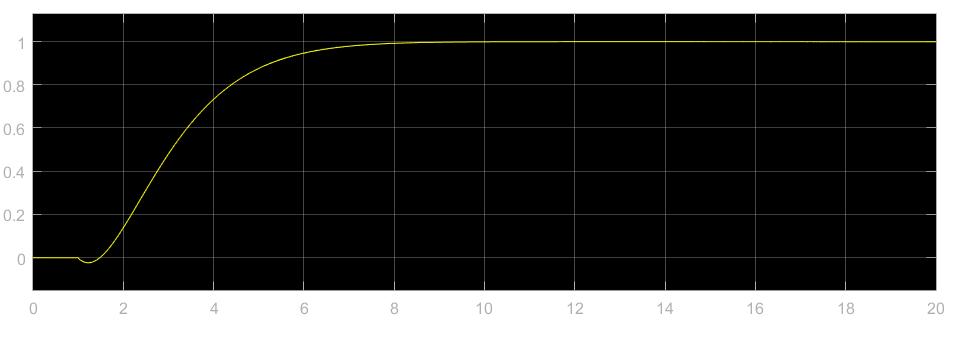
### Proportional Controller



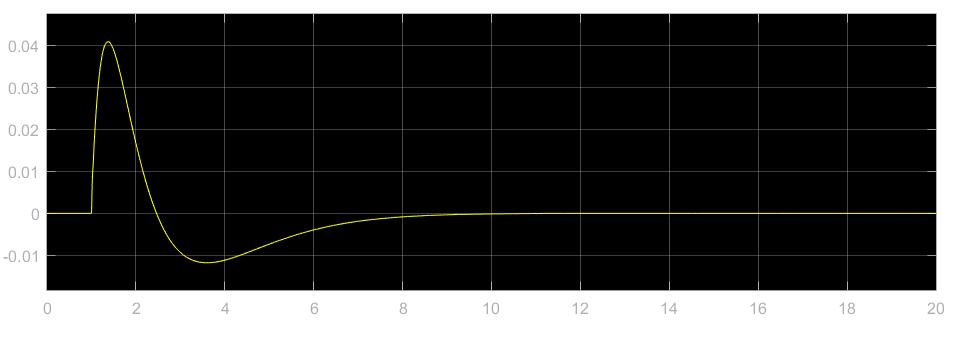
### Estimator



## Step Response: Position



## Step Response: Angle



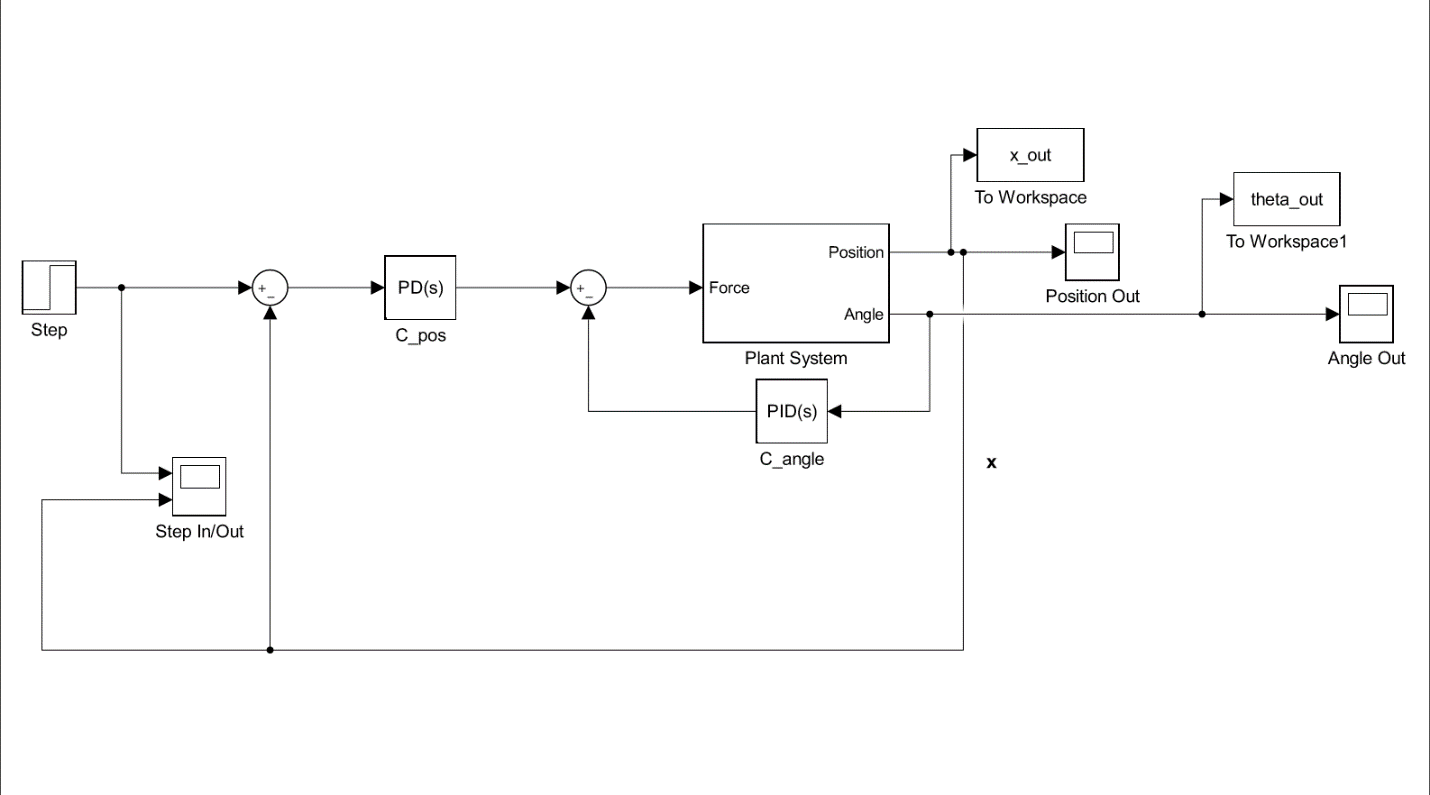
## Design and Optimization

The primary parameter that I designed was the K in the controller u = -K \* x + k\_r \* r. First, I designed K by placing the eigenvalues of the A-B\*K matrix in the left hand plane (LHP) at values similar to the eigenvalues of the original system defined by A.

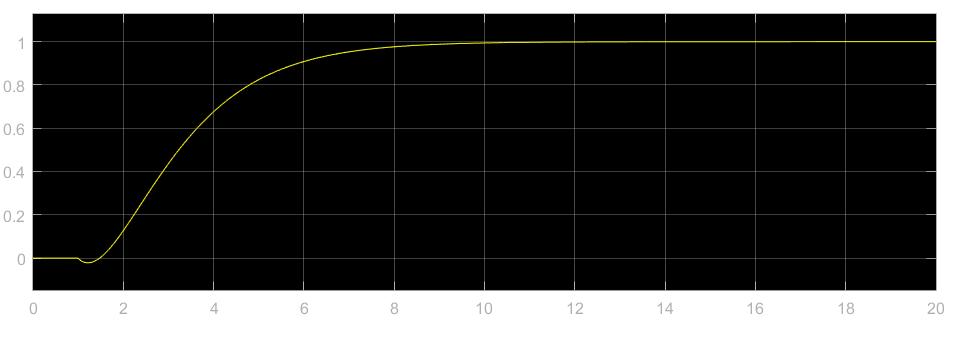
To optimize the system I programmatically looped through various placements of the poles of the A-B\*K matrix by defining a range of pole locations. I set the lower and upper limits of the search for poles by trying various pole locations by hand and testing the scores of those locations. Once I determined the bounds for the search, I looped through the values of the poles and saved the scores of all of those locations. I set the poles of A-B\*k to be the set of poles that maximized the score.

# PID Controller

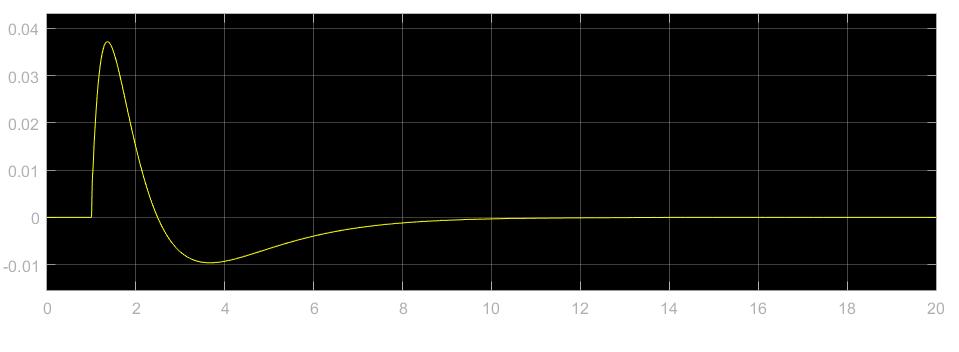
## System Diagram



## Step Response: Position

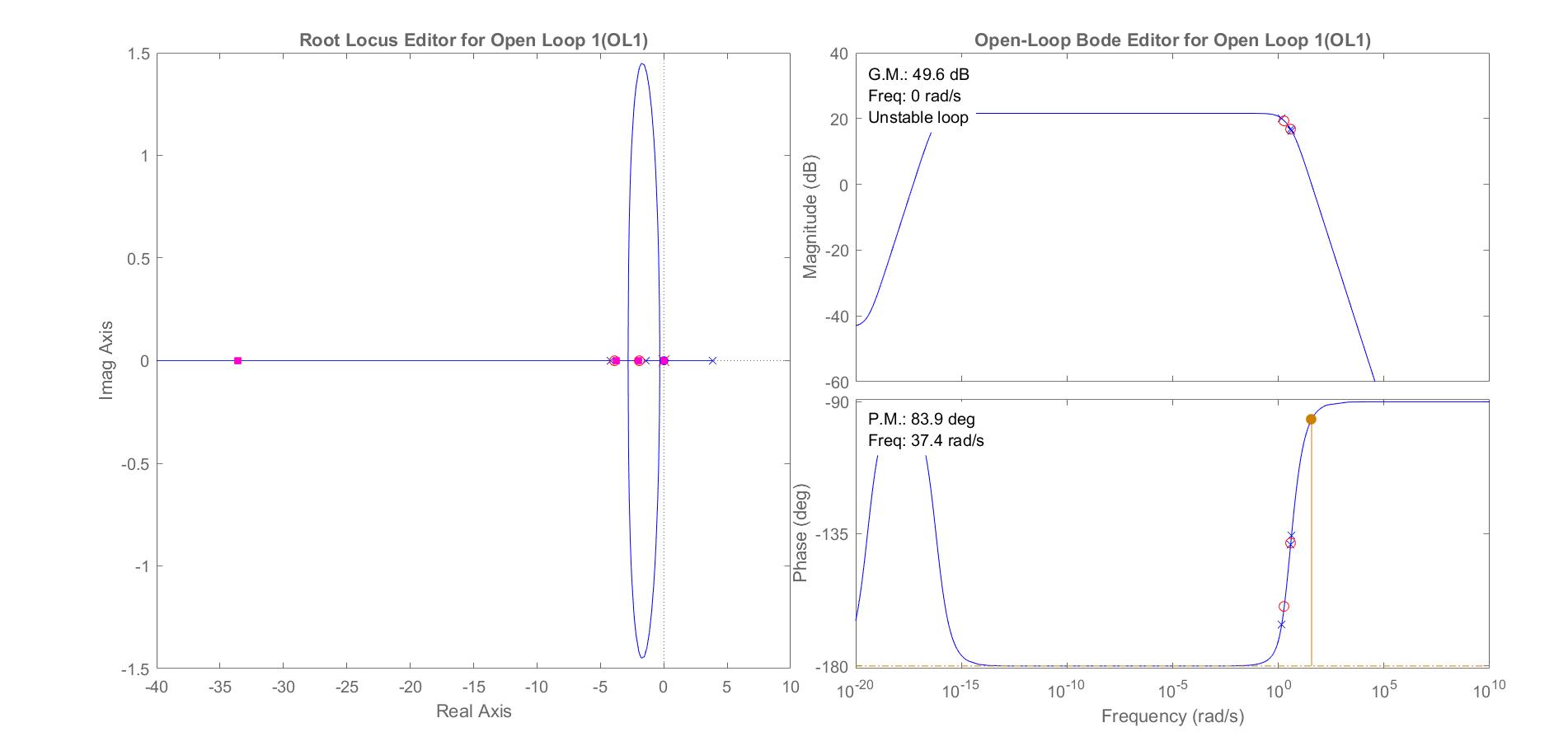


## Step Response: Angle

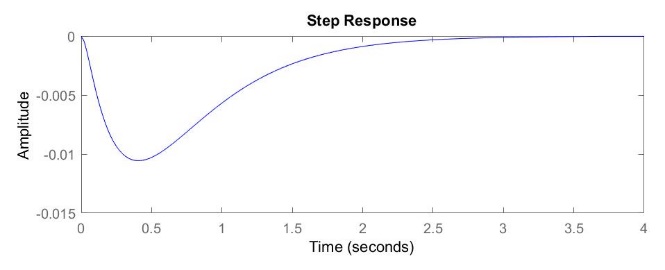


## Design and Optimization

To design the PID controller for the inverted pendulum system, I first designed a PID controller for the angle. To design this PID, I used the root-locus plot to place the poles and zeros of the angle controller, C\_angle for the plant P\_angle. The frequency design is shown below:



Given this frequency design of C\_angle, the step response of the plant for the angle (P\_angle) is shown in the graph below:

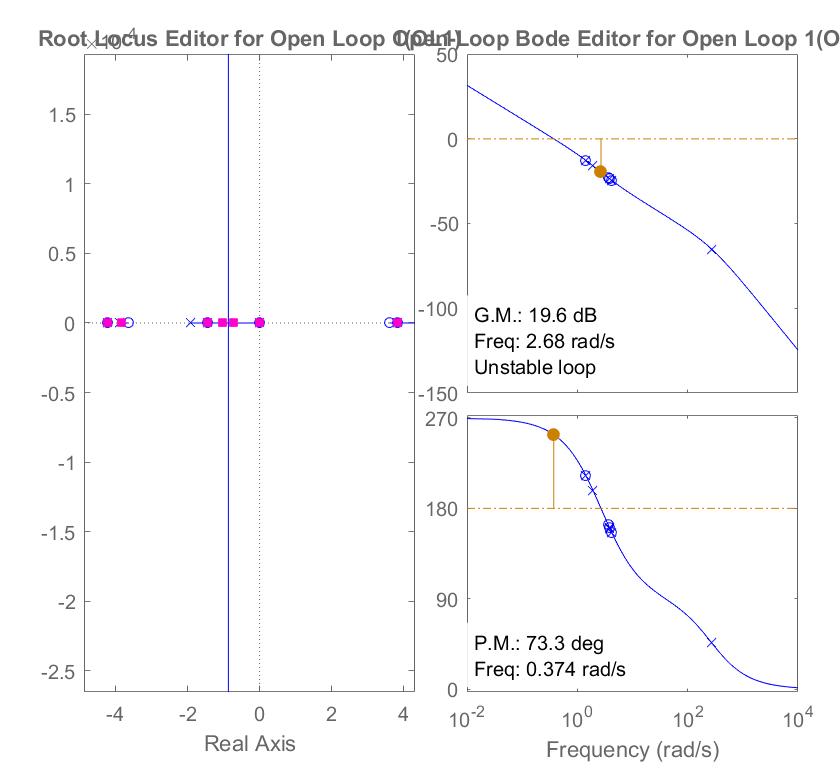


After finding a controller C\_angle that could control the angle of the inverted pendulum, I used the PID approach to design a position controller, C\_pos. I found that the transfer function from the input force to position – after the PID controller was in place for angle – was given by the equation,

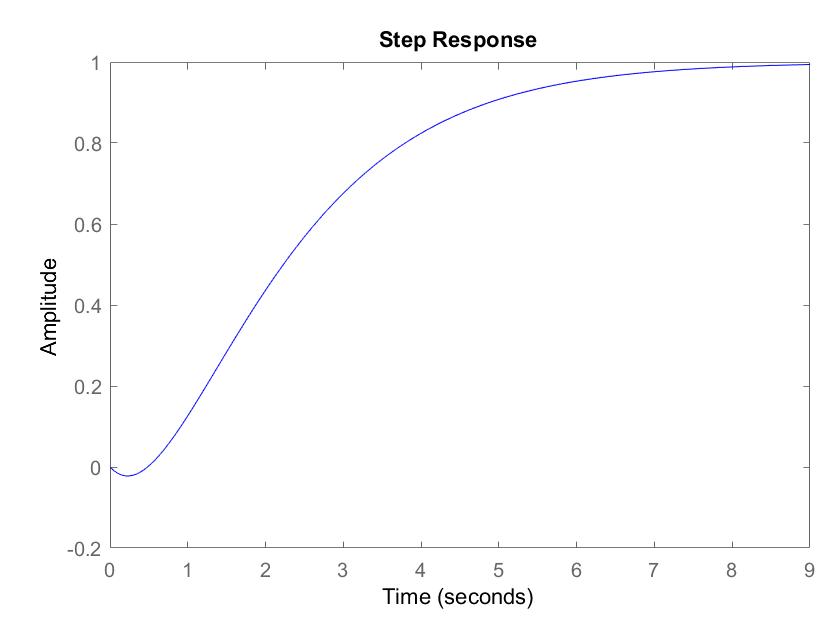
x\_over\_u = P\_pos / (1 + P\_angle \* C\_angle);

where x\_over\_u is the transfer function from the input force to the position, P\_pos is the plant for the position, P\_angle is the plant for the angle, and C\_angle is the controller for the angle.

Similar to my design approach for the C\_angle controller, I designed the set of parameters for the PID controller by using the root-locus plot. The frequency space design is shown in the graph below:



I found that simply using a proportional controller was sufficient for controlling the position. The step response is shown below:



I optimized the pair of controllers for position and angle by programmatically searching through a set of options for the zeros of the C\_angle controller, gain for the C\_angle controller, and gain for the C\_pos controller. The final set of parameters that I used was the set that produced the maximum score for the controller.

# Final Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Controller | Overshoot, OSx | Settling Time STx | Maximum Angle, Mtheta | Score |
| State Feedback | 0 | 15.43 | 0.0095 | 119.40 |
| PID | 0 | 9.21 | 0.0372 | 117.96 |