Jose Santiago

Dejan Mulitinović

ECE 141

Project 5

(a) For the control loop presented in Fig. 2, take $K_i = 0$ and use the root locus to find the parameters K_p and K_d, so that the damping factor of the system ≥ 0.7 and all zeroes and poles of the closed loop system are in the complex plane within the circle of $_n < \frac{1}{5T_s}$, $T_s = 0.1s$, $_n < 2\frac{rad}{s}$.

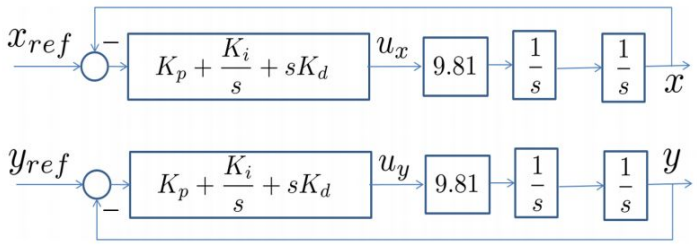


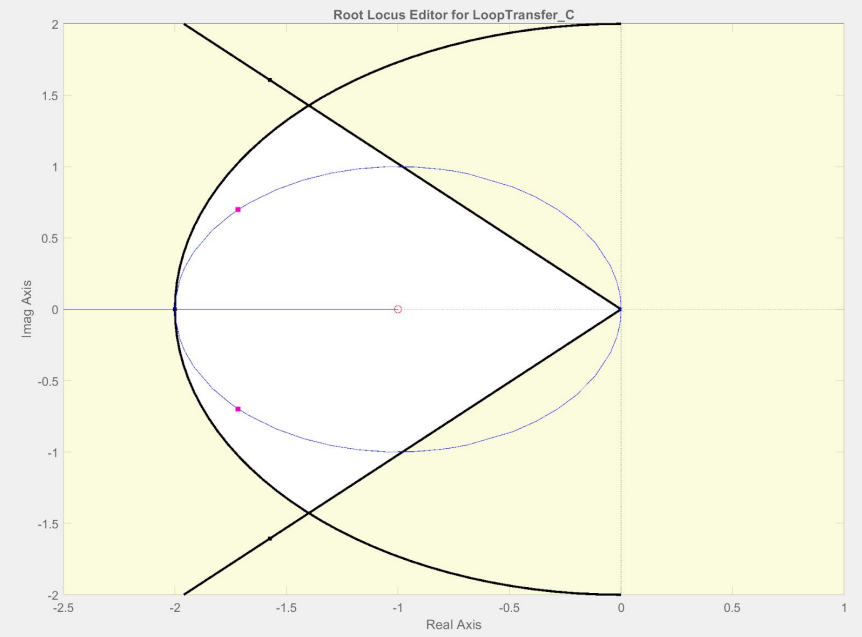
Figure 2:

Solution:

To solve this part of the lab I used MATLAB's rltool(_) function to analyze the root locus of the system. I defined the transfer function of the system, $G(s) = \frac{9.81}{s^2}$, in matlab and called rltool(G) as follows:

```
%ECE 141 Project #5
%Jose Santiago

%Plot Root Locus for Part (a)

%define transfer function variable s
s = tf('s');

%use s to write transfer function G
G = (9.81)/(s^2);

%print transfer function
G

%Plot transfer function with rltool
rltool(G);
```

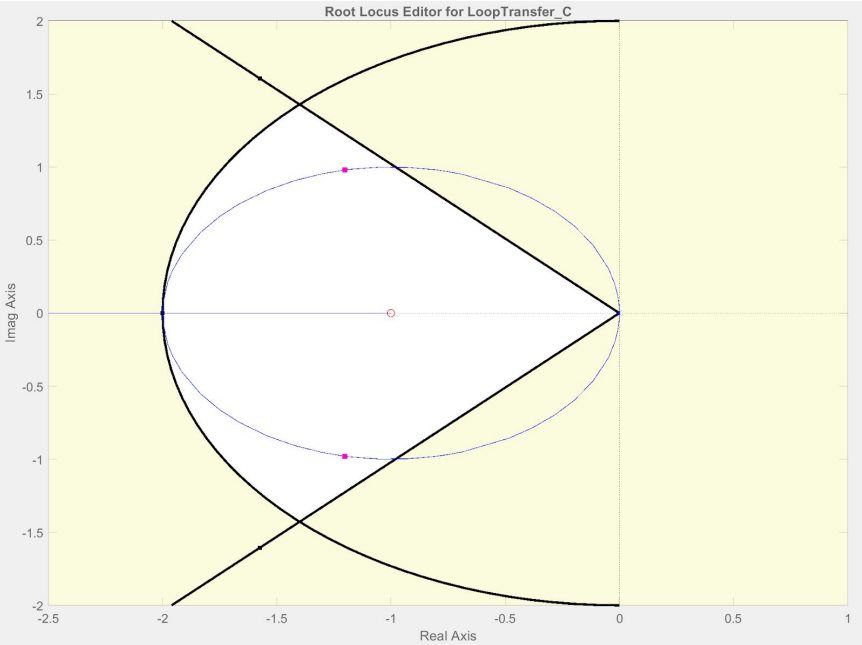
After this I added a real zero to represent the K_{PD} controller we want to design. The initial value of the zero was not important as I had yet to include the design requirements. Following the addition of the zero, I added the design requirements that are requested in part (a). I set the damping ratio, > 0.7, and the natural frequency, $_n < 2\frac{rad}{s}$. Setting these gave me the constraints for where the zeroes and poles of the system should be. I then settled on a zero value of -1 since it is a nice whole number. Then by editing the compensator in the Root Locus Editor, I was able to pick values of K_p and K_d that satisfy the design requirements. With my zero at -1 I could use $K_p = 0.35$ and $K_d = 0.35$ to create poles that lie within the design requirements. The following is the Root Locus Editor showing the poles and zero lying within the desired area with the given design requirements:



(b) If you know that the motor thrust can drop quickly and reduce the gain 9.81 to 70% of its value, can you find K_p and K_d to achieve ≥ 0.7 and $_n < \frac{1}{5T_s}$, $T_s = 0.1s$, $K_i = 0$.

Solution:

To solve for this part I used the same zero from part (a) of -1. Then I modified the K_p value to reduce the gain to 70% of its value. Since the Kp value is multiplied to the 9.81 gain, it is like reducing the gain of 9.81 to 70% of its value via the associative property of multiplication. I multiplied the K_p value from part (a) with 0.7 and then checked to see if the poles of the system remained within the design requirements of the damping ratio, ≥ 0.7, and natural frequency, $_n < 2\frac{rad}{s}$. This resulted in $K_p = 0.35 * 0.7 = 0.245$. With this modification, my poles remained within the desired design requirements, as seen in the following image of the Root Locus Editor:



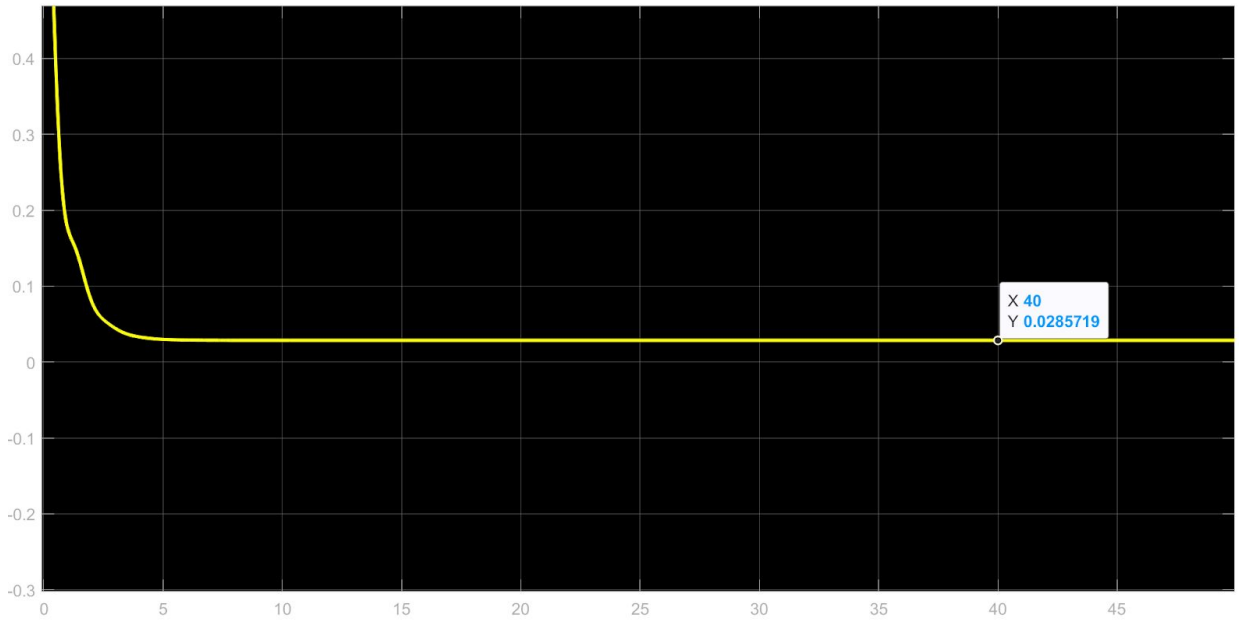
(c) You are provided with the Simulink model (quadExperiment.mdl) which models the dynamics (1) - (2) and the matrix R^{-1}. The model also has the discrete time implementation of the PID controller. Set the K_p, K_d and set $K_i = 0$, and provide the

response along the x and y coordinates. What are the steady state errors $e_x = x_{ref} - x$ and $e_y = y_{ref} - y$?

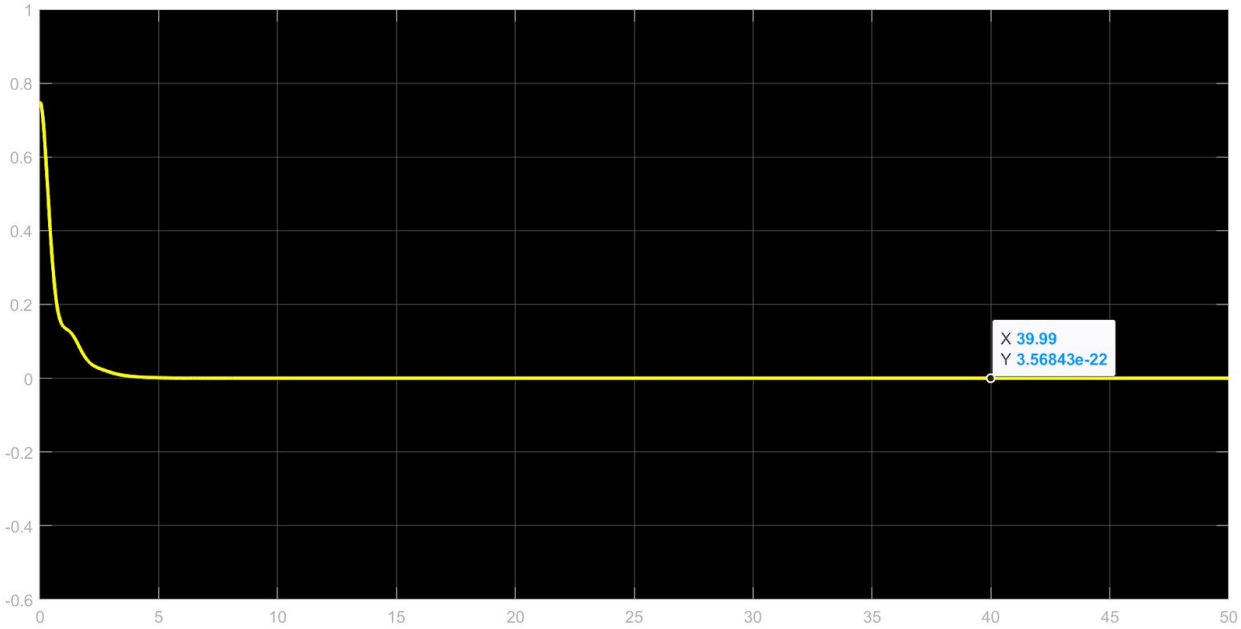
Solution:

For this part I opened the quadExperiment.mdl Simulink model and set my K_p and K_d values that I found in parts (a) and (b), $K_p = 0.35$, $K_d = 0.35$. I kept the x and y reference inputs as zero. Afterwards I ran the Simulink model and observed the x and y output waveforms. The waveforms showed that the x output converged to a non-zero value, $x(40) = 0.0285719$ from the scope, while the y output converged to a very small value close to zero, $y(39.99) = 3.56843 * 10^{-22}$. From the visuals we see on the respective scopes we can say that as time goes to infinity, $x \approx 0.0285719$ and $y \approx 0$. Therefore the steady state error $e_x = 0 - 0.0285719 = -0.0285719$, and $e_y = 0 - 0 = 0$. Below are the waveforms I have based my answers for this part off of.

x waveform:



y waveform:



(d) Use the diagrams below in Fig. 3 to find the transfer function from $d_x(d_y)$. The disturbances d_x and d_y model a displacement of the quadrotor's center of mass and you can consider them to have constant values. Can $K_i \neq 0$ eliminate their impact and result in $e_x(\infty) = e_y(\infty) = 0$?

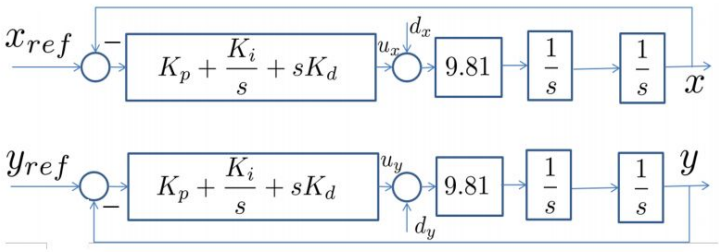


Figure 3:

Solution:

For this part of the project I started by solving for the transfer function of each system for x and y. My derivation for transfer function $\frac{x(s)}{d_x(s)}$ is as follows:

$$x(s) = (d_x(s) + u_x(s))(\tfrac{9.81}{s^2})$$

$$u_x(s) = (0 - x(s))(K_p + \frac{K_i}{s} + K_d s)$$

$$u_x(s) = (-x(s))(K_P + \frac{K_i}{s} + K_d s)$$

$$x(s) = (d_x(s) + (-x(s))(K_p + \frac{K_i}{s} + K_d s))(\frac{9.81}{s^2})$$

$$x(s) = (d_x(s)(\frac{9.81}{s^2})) - (x(s))(\frac{(K_p + \frac{K_i}{s} + K_d s)(9.81)}{s^2})$$

$$x(s) + x(s)(\frac{(K_p + \frac{K_i}{s} + K_d s)(9.81)}{s^2}) = d_x(s)(\frac{9.81}{s^2})$$

$$x(s)(1 + (\frac{(K_p + \frac{K_i}{s} + K_d s)(9.81)}{s^2})) = d_x(s)(\frac{9.81}{s^2})$$

$$\frac{x(s)}{d_x(s)} = \frac{\left(\frac{9.81}{s^2}\right)}{\left(1 + (\frac{(K_p + \frac{K_i}{s} + K_d s)(9.81)}{s^2})\right)}$$

$$\frac{x(s)}{d_x(s)} = \frac{\left(\frac{9.81}{s^2}\right)}{\left(\frac{s^2}{s^2} + (\frac{(K_p + \frac{K_i}{s} + K_d s)(9.81)}{s^2})\right)}$$

$$\frac{x(s)}{d_x(s)} = \frac{\left(\frac{9.81}{s^2}\right)}{\left(\frac{s^2 + (K_p + \frac{K_i}{s} + K_d s)(9.81)}{s^2}\right)}$$

$$\frac{x(s)}{d_x(s)} = (\frac{9.81}{s^2})\left(\frac{s^2}{s^2 + (K_p + \frac{K_i}{s} + K_d s)(9.81)}\right)$$

$$\frac{x(s)}{d_x(s)} = \frac{9.81}{s^2 + (K_p + \frac{K_i}{s} + K_d s)(9.81)}$$

$$x(s) = \left(\frac{9.81}{s^2 + (K_p + \frac{K_i}{s} + K_d s)(9.81)}\right)(d_x(s))$$

The derivation for the transfer function $\frac{y(s)}{d_y(s)}$ is very similar to this derivation for $\frac{x(s)}{d_x(s)}$, and follows the same pattern, so I won't display the work here. The result from this derivation gives us the following:

$$\frac{y(s)}{d_y(s)} = \frac{9.81}{s^2+(K_p+\frac{K_i}{s}+K_d s)(9.81)}$$

$$y(s) = \left(\frac{9.81}{s^2+(K_p+\frac{K_i}{s}+K_d s)(9.81)}\right)(d_y(s))$$

After deriving these transfer functions, we can account for $K_i = 0$, which gives us the transfer functions:

$$\frac{x(s)}{d_x(s)} = \frac{9.81}{s^2+(K_p+K_d s)(9.81)}$$

$$x(s) = \left(\frac{9.81}{s^2+(K_p+K_d s)(9.81)}\right)(d_x(s))$$

$$\frac{y(s)}{d_y(s)} = \frac{9.81}{s^2+(K_p+K_d s)(9.81)}$$

$$y(s) = \left(\frac{9.81}{s^2+(K_p+K_d s)(9.81)}\right)(d_y(s))$$

Then we can start analyzing the error signal of our systems, given by the following equations:

$$e_x(s) = x_{ref}(s) - x(s)$$

$$e_y(s) = y_{ref}(s) - y(s)$$

$$\rightarrow x_{ref}(s) = 0$$

$$\rightarrow y_{ref}(s) = 0$$

$$e_x(s) = -x(s)$$

$$e_x(s) = -\left(\frac{9.81}{s^2+(K_p+K_d s)(9.81)}\right)(d_x(s))$$

$$e_y(s) = -y(s)$$

$$e_y(s) = -\left(\frac{9.81}{s^2+(K_p+K_d s)(9.81)}\right)(d_y(s))$$

Then we can use the Final Value Theorem to analyze the error as time tends to infinity as follows:

$$\lim_{t\to\infty} e(t) = \lim_{s\to 0}(s)(e(s))$$

$$\lim_{s\to 0}(s)(e_x(s)) = (s)\left(-\left(\frac{9.81}{s^2+(K_p+K_d s)(9.81)}\right)(d_x(s))\right)$$

$$\to d_x(s) = \frac{d_x}{s}$$

$$\lim_{s\to 0}(s)(e_x(s)) = (s)\left(-\left(\frac{9.81}{s^2+(K_p+K_d s)(9.81)}\right)\left(\frac{d_x}{s}\right)\right)$$

$$\lim_{s\to 0}(s)(e_x(s)) = \left(-\left(\frac{9.81(d_x)}{s^2+(K_p+K_d s)(9.81)}\right)\right)$$

$$\lim_{s\to 0}(s)(e_x(s)) = \left(-\left(\frac{9.81(d_x)}{(0)+(K_p+(0))(9.81)}\right)\right)$$

$$\lim_{s\to 0}(s)(e_x(s)) = \left(-\left(\frac{9.81(d_x)}{(K_p)(9.81)}\right)\right)$$

$$\lim_{s\to 0}(s)(e_x(s)) = \left(-\frac{d_x}{K_p}\right)$$

From this derivation we can see that if $K_i = 0$, our error signal, $e_x(s)$, converges to a finite value. The derivation of the Final Value Theorem for the error signal $e_y(s)$ is the same as the derivation of the for the error signal $e_y(s)$, so I won't display the full derivation, but I will provide the solution here:

$$\lim_{s\to 0}(s)(e_y(s)) = \left(-\frac{d_y}{K_p}\right)$$

With this derivation we can see that if $K_i = 0$, our error signal, $e_y(s)$, converges to a finite value as well. To check to see if our system will work with $K_i \neq 0$, we can use the transfer function we found originally before we set $K_i = 0$ to evaluate

with the Finite Value Theorem. The following is the derivation of the FVT with $K_i \neq 0$:

$$x(s) = \left(\frac{9.81}{s^2+(K_p+\frac{K_i}{s}+K_d s)(9.81)} \right) (d_x(s))$$

$$\rightarrow d_x(s) = \frac{d_x}{s}$$

$$x(s) = \left(\frac{9.81}{s^2+(K_p+\frac{K_i}{s}+K_d s)(9.81)} \right) \left(\frac{d_x}{s}\right)$$

$$\lim_{s\rightarrow 0}(s)(e_x(s)) = \left(\frac{9.81}{s^2+(K_p+\frac{K_i}{s}+K_d s)(9.81)} \right) \left(\frac{d_x}{s}\right) (s)$$

$$\lim_{s\rightarrow 0}(s)(e_x(s)) = \left(\frac{(9.81)(d_x)}{s^2+(K_p+\frac{K_i}{s}+K_d s)(9.81)} \right)$$

$$\lim_{s\rightarrow 0}(s)(e_x(s)) = \left(\frac{(9.81)(d_x)}{s^2+(K_p+\frac{K_i}{s}+K_d s)(9.81)} \right) \left(\frac{s}{s}\right)$$

$$\lim_{s\rightarrow 0}(s)(e_x(s)) = \left(\frac{(9.81)(d_x)(s)}{s^3+(s)(K_p+\frac{K_i}{s}+K_d s)(9.81)} \right)$$

$$\lim_{s\rightarrow 0}(s)(e_x(s)) = \left(\frac{(9.81)(d_x)(s)}{s^3+(K_p s+K_i+K_d s^2)(9.81)} \right)$$

$$\lim_{s\rightarrow 0}(s)(e_x(s)) = \left(\frac{(9.81)(d_x)(0)}{(0)^3+(K_p(0)+K_i+K_d(0)^2)(9.81)} \right)$$

$$\lim_{s\rightarrow 0}(s)(e_x(s)) = 0$$

From this derivation we can see that if $K_i \neq 0$, our error signal, $e_x(s)$, converges to zero. The derivation for the error signal $e_y(s)$ as time goes to infinity is very similar to the one for $e_x(s)$ so I won't show the whole derivation, but will note that it also converges to zero:

$$\lim_{s\rightarrow 0}(s)(e_y(s)) = 0$$

Therefore from these derivations we can conclude that if $K_i \neq 0$, the impact of the disturbances d_x & d_y are eliminated, resulting in $e_x(\infty) = 0$, and $e_y(\infty) = 0$.

(e) If you know that the motor thrust can drop quickly and reduce the gain 9.81 to 70% of its value, can you find K_p, K_d, K_i to achieve ≥ 0.7 and $_n < 2\frac{rad}{s}$. Provide the response along the x and y coordinates. To simulate your control system realistically with the measurement noise, set all the constants NX, NY, NYaw in the Simulink model to 1. Record the data, and compute the mean value and the standard deviation of x and y.

Solution:

For this part of the lab I defined the transfer function $G(s)$ slightly different than I did in part (a) since $K_i \neq 0$. To do this I manipulated our K_{PID} equation as follows:

$$K_{PID} = \frac{K_i}{s} + K_p + sK_d$$

$$K_{PID} = K_i\left(\frac{1}{s} + \frac{K_p}{K_i} + s\,\frac{K_d}{K_i}\right)$$

$$K_{PID} = K_i\left(\frac{1}{s} + \frac{K_p}{K_i}\left(\frac{s}{s}\right) + s\,\frac{K_d}{K_i}\left(\frac{s}{s}\right)\right)$$

$$K_{PID} = K_i\left(\frac{1+(s)\frac{K_p}{K_i}+(s^2)\frac{K_d}{K_i}}{s}\right)$$

$$K_{PID} = K_i\left(\frac{1+(s)\frac{K_p}{K_i}+\left(s\sqrt{\frac{K_d}{K_i}}\right)^2}{s}\right)$$

After achieving this form of our controller, we can factor out the $\frac{1}{s}$ term from the K_{PID} controller and instead multiply it to our system $G(s) = \frac{9.81}{s^2}$ as follows:

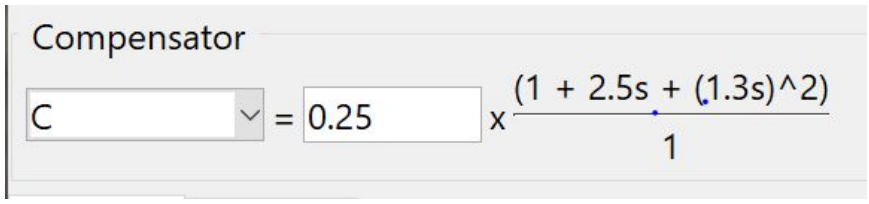
$$K_{PID} = K_i \left[1 + (s)\frac{K_p}{K_i} + \left(s\sqrt{\frac{K_d}{K_i}} \right)^2 \right] \left(\frac{1}{s}\right)$$

$$G(s) = \frac{9.81}{s^2}\left(\frac{1}{s}\right)$$

$$G(s) = \frac{9.81}{s^3}$$

$$K_{PID} = K_i \left[1 + (s)\frac{K_p}{K_i} + \left(s\sqrt{\frac{K_d}{K_i}} \right)^2 \right]$$

This is nice because it is simpler to modify the $G(s)$ equation than it is to add a pole. After modifying the $G(s)$ I ran the rltool(_) function on it. The K_{PID} modification was made because the Compensator in the Root Locus Editor displays the K_p, K_i, and K_d in this form as shown below:



To get this form in the Compensator I had to add two complex zeros to the Root Locus design. This is because the introduction of K_i introduces two zeroes and a pole. The pole is taken into consideration within the new derivation of our $G(s)$, but we must add the complex zeros ourselves within the Root Locus Editor. After doing this I added the design requirements that were requested in part (e), ≥ 0.7 and $_n < 2\frac{rad}{s}$. These design requirements also had to be met with a gain of 70%. So after testing various different values, I landed at the values displayed in my Compensator window above. After solving for the controller values algebraically by using substitution with my K_{PID} I found above, I found values:

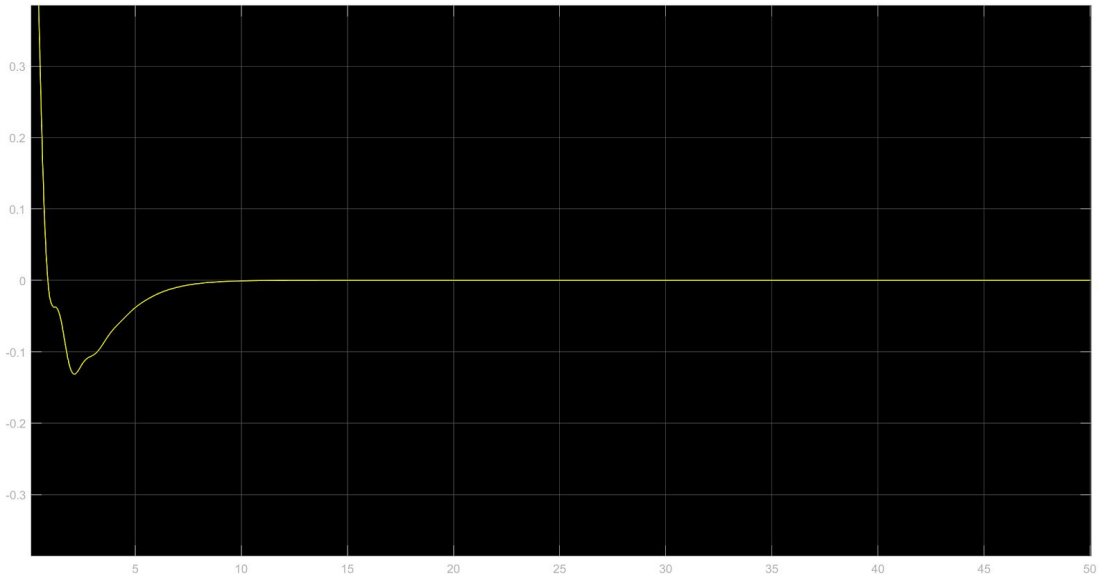
$K_i = 0.25$

$K_p = 0.625$

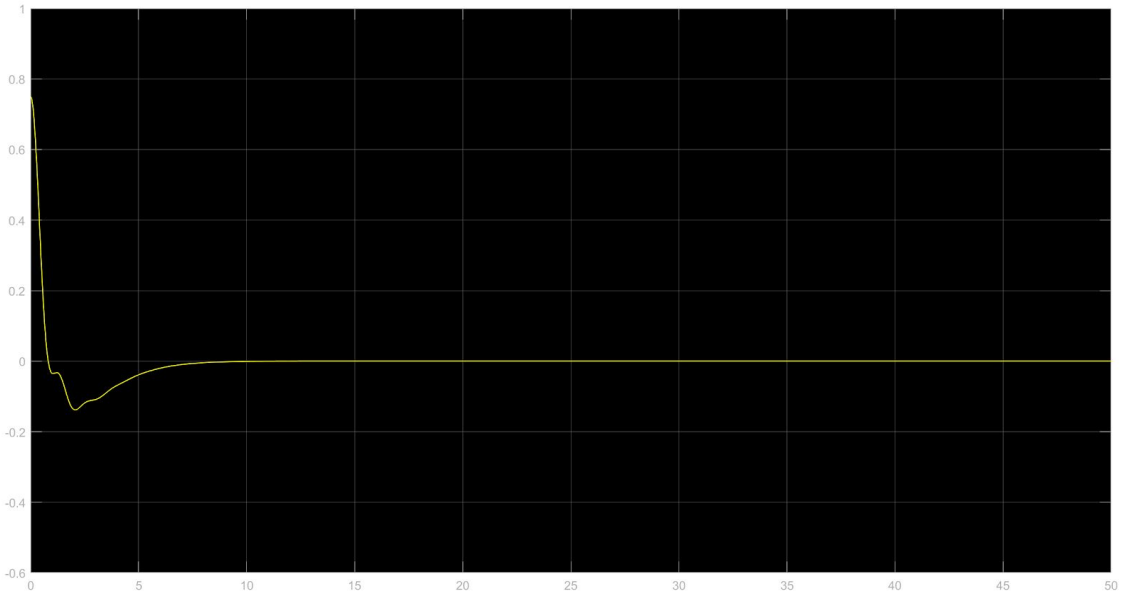
$K_d = 0.4225$

After inputting these controller values into the Simulink model I found the x and y waveforms depicted below:

x waveform:

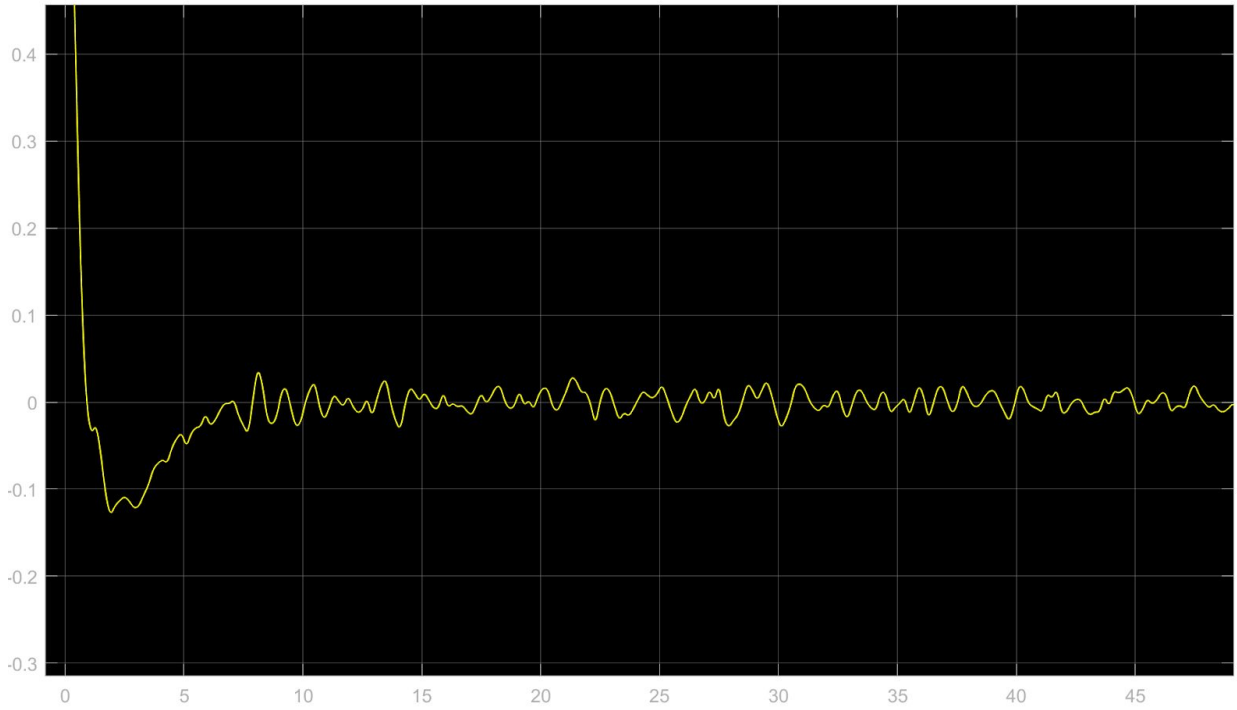


y waveform:

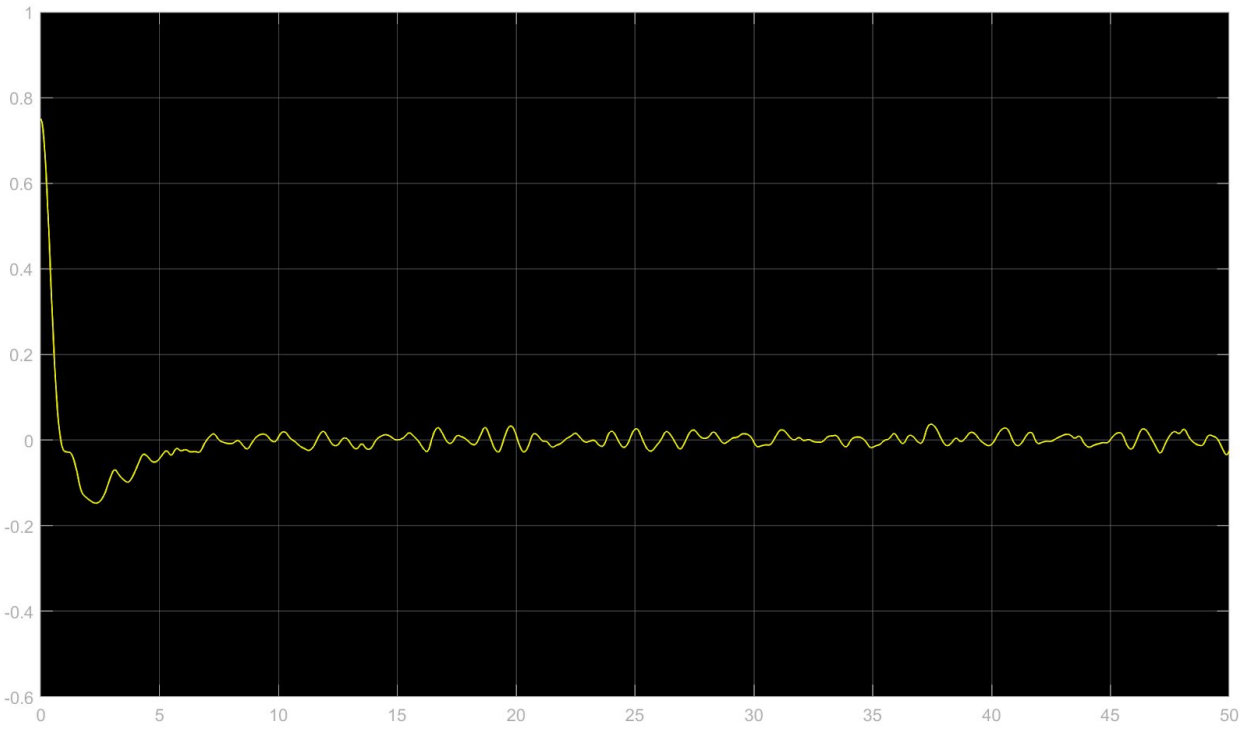


Then after setting NX, NY, NYaw in the Simulink model to 1, I observed the waveforms depicted below:

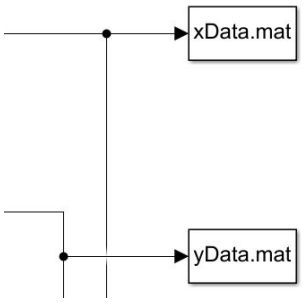
x waveform:



y waveform:



To calculate the mean and standard deviation of my data from outputs x and y, I had to first add a To File output block to my Simulink model. I connected it to the x and y outputs so that I would be able to access the data points from a file within MATLAB. After storing the data in two separate .mat files I accessed them in MATLAB by using the load command. I named the files xData.mat and yData.mat.



After loading the xData and yData as variables, I was able to use the mean(_) and std(_) functions to find the mean and standard deviation of the respective outputs, x and y. The following is the script I used, as well as the mean and standard deviation of the output data with the noise added to the system:

```
%ECE 141 Project #5
%Jose Santiago

%load xData and yData
load xData
load yData

%calculate xData and yData means
disp('mean(xData.Data):')
m_x = mean(xData.Data);
disp(m_x)
disp('mean(yData.Data):')
m_y = mean(yData.Data);
disp(m_y)

%calculate xData dn yData SD's
disp('std(xData.Data):')
s_x = std(xData.Data);
disp(s_x)
disp('std(yData.Data):')
s_y = std(yData.Data);
disp(s_y)
```

```
>> Project5_part_e_2
mean(xData.Data):
   7.3979e-04

mean(yData.Data):
  -3.4798e-04

std(xData.Data):
    0.0740

std(yData.Data):
    0.0719
```

To conclude, the mean and standard deviations of the x and y outputs are as follows:

$mean(xData) = 0.00073979$

$mean(yData) = -0.00034798$

$std(xData) = 0.0740$

$std(yData) = 0.0719$