University of Colorado Boulder

ASEN 3200 - Orbital Mechanics/Attitude Dynamics & Control

Lab A-3: Spacecraft Pitch Axis Control

March 24, 2023

Author:
Ben Chapel[1]

Author:
Daniel Mascarenas[2]

Author:
Rishab Pally[3]

Professor:
Casey Heidrich

[0]SID: 109099510
[1]SID: 109137988
[2]SID: 109519936

Smead Aerospace
UNIVERSITY OF COLORADO **BOULDER**

Contents

This paper provides an analysis on how PID controls can be utilized to model a spacecraft's reference trajectory while concurrently using spacecraft rigid body mass properties and feedback controller gains to better understand the closed loop tacking behavior. Values for moment of inertia, K_d, and K_p were found to meet the requirements for percent settling time and maximum overshoot. These values were 7.49 $kg \cdot m^2$ for I, 85.49 for K_p, and 29.92 for K_d. It was also found that the integral control gain was important in minimizing the sign error between the inputted and position values. From analysis of the measured vs. reference angular position, we were able to theorize about possible sources of error, such as the spin module potentially being unbalanced or unwanted gyroscopic effects.

1 Preliminary Questions

1.1 Moment of Inertia Derivation

$$L = I\alpha \rightarrow I = \frac{L}{\alpha} \rightarrow I = L\left(\frac{t}{\omega}\right) \tag{1}$$

Initially we provide a command torque to the base of the spacecraft. Knowing this command torque allows us to then determine the moment of inertia of the spacecraft base by using the equation: $I = L\left(\frac{t}{\omega}\right)$. Another way is to graph the angular velocity vs. time and fit a linear graph to it. This allows the team to determine the angular acceleration

1.2 Block Diagram

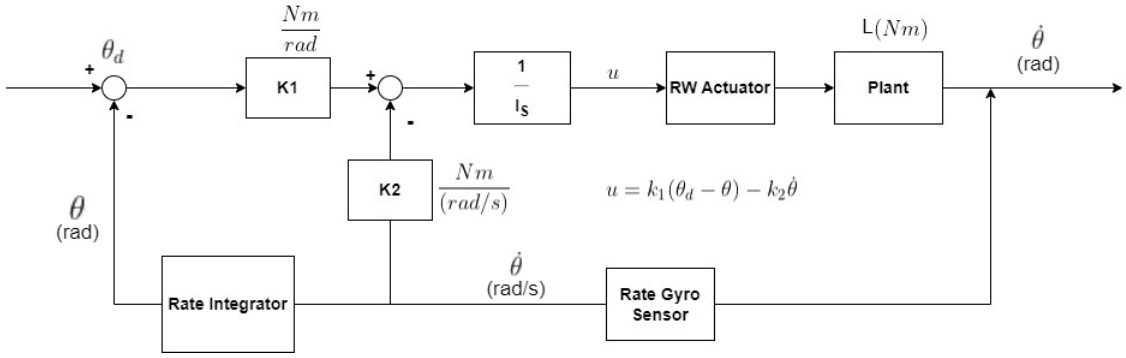


Figure 1: Control Block Diagram

1.3 System Poles, ζ, ω_n, K_p, K_d

The equation for maximum overshoot was utilized to solve for ζ:

$$M_p = e^{\frac{-\zeta}{\sqrt{1-\zeta^2}}} \tag{2}$$

Since the value for M_p is known, ζ was calculated as 0.59115. This value was then plugged into Equation (5) to get $\omega_n = 3.37839$. With ζ and ω_n known, the following equations were used to find the required K_p and K_d values.

$$\frac{K_2}{I} = 2\zeta\omega_n \tag{3}$$

$$\omega_n = \sqrt{\frac{K_1}{I}} \tag{4}$$

From these, expressions relating the moment of inertia of the spin module to the required K_p and K_d values were found:

$$K_1 = K_p = 11.4135I \tag{5}$$

$$K_2 = K_d = 3.9943I \tag{6}$$

The system pole locations in the s-plane were calculated from the following:

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \tag{7}$$

By plugging in the calculated values for ζ and ω, the values for s were found to be -1.9972 $\pm$ 2.72487i.

2 Experiment

2.1 Inertia

In order to find the required K_p and K_d values, the moment of inertia of the spin module was determined experimentally. Data was collected with a set applied external torque of [3,5,7,8,10] mNm. We graphed the response of angular Velocity vs time. A line of best fit was fit to the linear section of the time vs. angular velocity data and its slope ($\frac{\omega_3}{t}$) was noted determining the angular acceleration of the Reaction wheel. These values were then plugged into Equation (3) to calculate the moment of inertia. This value was found to be

$$MOI = [7 \pm 2] \quad kg \cdot m^2$$

2.2 Control Design

By following the procedure laid out in Preliminary Question 3 and using the moment of inertia calculated experimentally, values for K_p and K_d to meet the requirements were found. These values were calculated as

$$K_p = [85.4931] \quad Nm/rad$$

$$K_d = [29.9194] \quad Nm/rad/sec$$

2.3 Control Results

The K_p and K_d values that we calculated using the moment of inertia were then entered into the LabVIEW program and the angular position vs. time data was recorded. Similar to the method to calculate the moment of inertia, the most ideal oscillation in the data file was found and plotted. The figures for the reference vs. measured angular positions are found below.

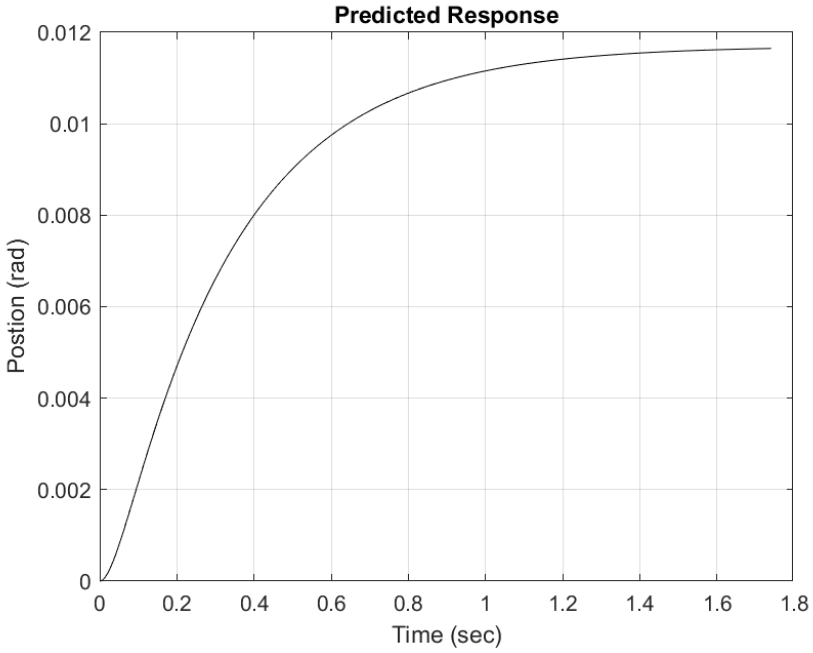


Figure 2

3 Analysis

3.1 Moment of Inertia Calculation

As discussed in Sections 1.1 & 2.1, to calculate the moment of inertia of the spacecraft multiple different known torques L were applied to the spacecraft and the angular velocity ω was measured over time during this process. For each torque value, the resulting angular velocity linearly increased over time, so a linear fit was applied and the slope of each was recorded, corresponding to the angular acceleration α. Each torque L was then divided by the corresponding α, giving a moment of inertia estimate for each trial. These values were then averaged to find the final value of $I = 7 \pm 2kg \cdot m^2$. This is a very large uncertainty, and there are many potential sources. One potential source of error is friction between the spacecraft and the base, as this would resist the applied torque and change the $\frac{L}{\alpha}$ ratio. The relative size of the effect may also vary between different speeds, thus affecting each trial differently and increasing the uncertainty. Additionally, there is likely some error in exactly how much torque the motor produced, something that would also have different effects at different torque settings. This likely also contributed to the uncertainty in the moment of inertia.

3.2 Expected Step Response

The closed loop poles were calculated by first using the percent settling time of 5% with Equation (4) to get a relationship between ω_n and ζ. Next, Equation (6) was used with the desired maximum overshoot of [10 percent] to calculate the ζ value, which was then used with Equation (5) to obtain the ω_n value. Finally, these ω_n and ζ values were plugged into Equation (11) to solve for s, the closed loop poles. The plot for the expected step response for this closed-loop system based on the design requirements is below. The poles were found to be S = -5.4655 ± 7.4580i. Based on the Below graph and the poles it can be determined that the system is under-damped and the spacecraft will take longer to settle within the desired position.

3.3 Experimental Results

As seen in the Figure 4 below, both tests produced very similar results. During the test in which a reference step was provided, the actual position settled within 5% well before the desired 1.5s and did not overshoot. It did have some steady-state error that remained well after the position had settled.

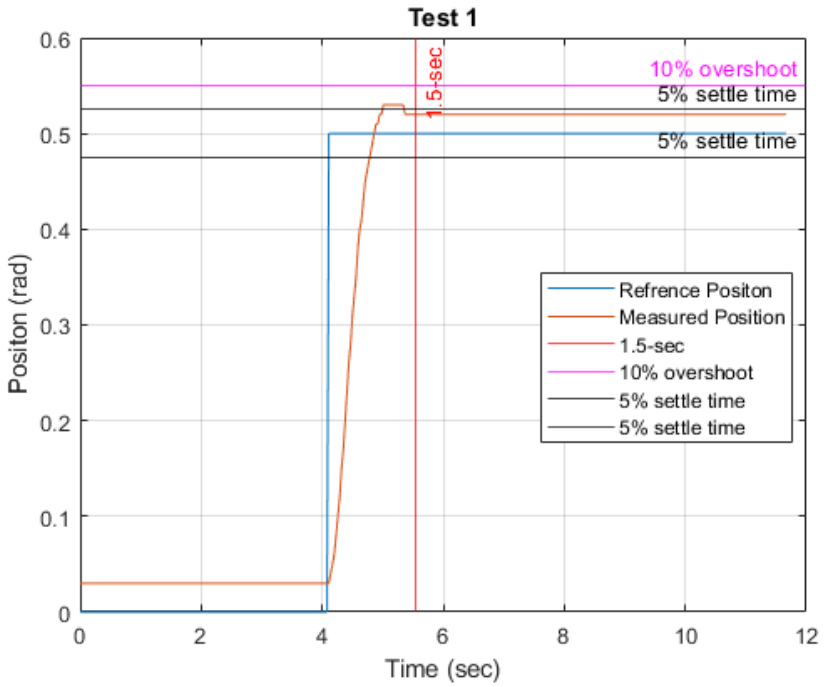


Figure 3

During the test in which a disturbance was applied with a zero reference command, the response is incredibly similar as seen in Figure 6. The only noticeable difference is a slight (1%) overshoot, followed by the position seemingly perfectly aligning with the commanded position.

3.4 Behavior of Reaction Wheel

The reaction wheel spins in the opposite direction of the spacecraft module which then results in the spacecraft module to spin in the desired direction due to conservation of angular momentum. The current data also correlates with the reaction wheel as the angular acceleration of the spacecraft module is opposite of the reaction wheel whenever the step input changes and the current approaches zero when the spacecraft module has reached its desired angle/state.

3.5 Response to Disturbance Input

When a disturbance input by hand gets added, the reaction wheel directs the spacecraft module back to its original orientation. For example, when an initial displacement of 1 rad in the counterclockwise direction is input into the spacecraft module, the reaction wheel starts to accelerate in the positive counterclockwise direction so that the torque is able to align in the opposite clockwise direction which results in a rotation towards the initial point. As it is returning to the original orientation of $\theta = 0$, there will be an overshoot since we have our derivative gain set to 0. To restore the module completely back to its original neutral orientation, the reaction wheel continues to oscillate and loses energy due to friction. The reaction wheel repeats this process and after every iteration, the overshoot continues to get smaller until it reaches its original orientation. This correlates well with real life space applications although errors such as friction and measurements in rotation need to be accounted for as it can affect the reaction wheel's ability to direct the spacecraft back to

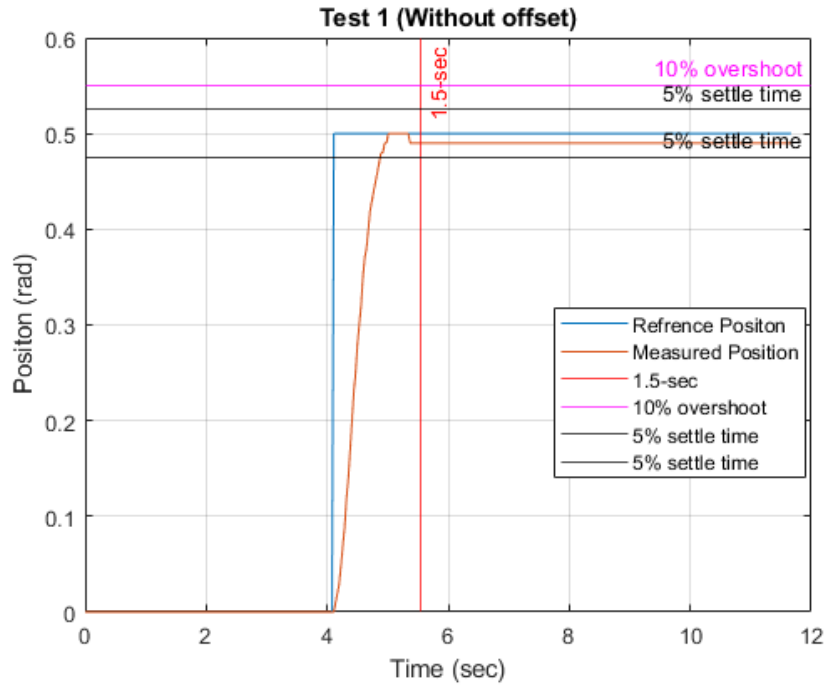


Figure 4

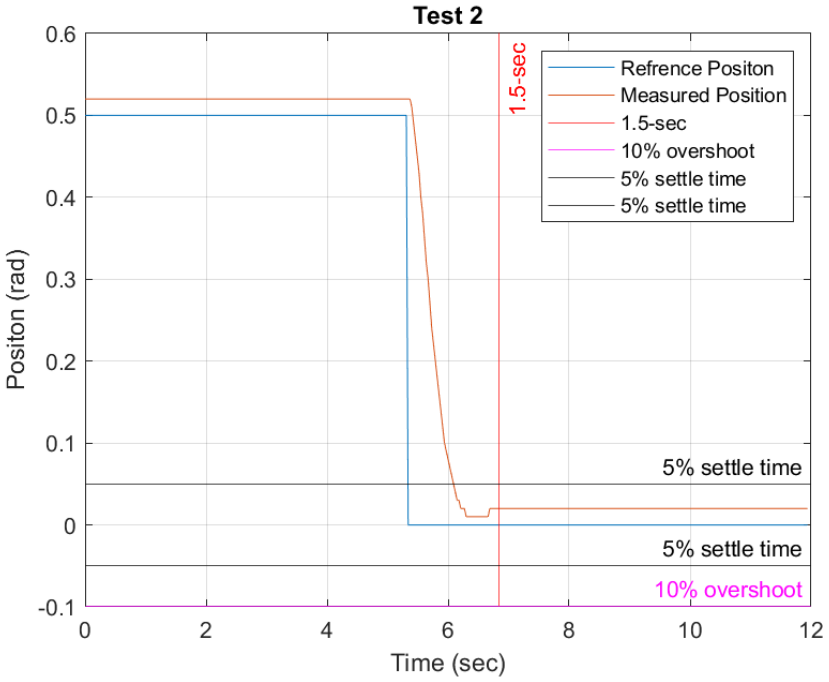


Figure 5

its exact original position. This is how derivative gains were implemented to stop the spacecraft from moving. The below figures show how the reaction wheels torque is applied during the duration of the experiment.

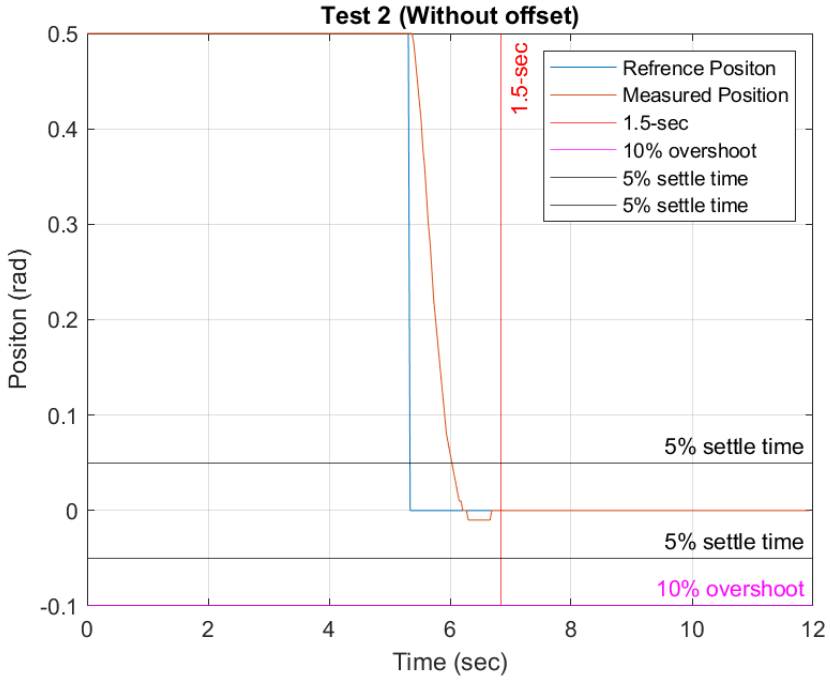


Figure 6

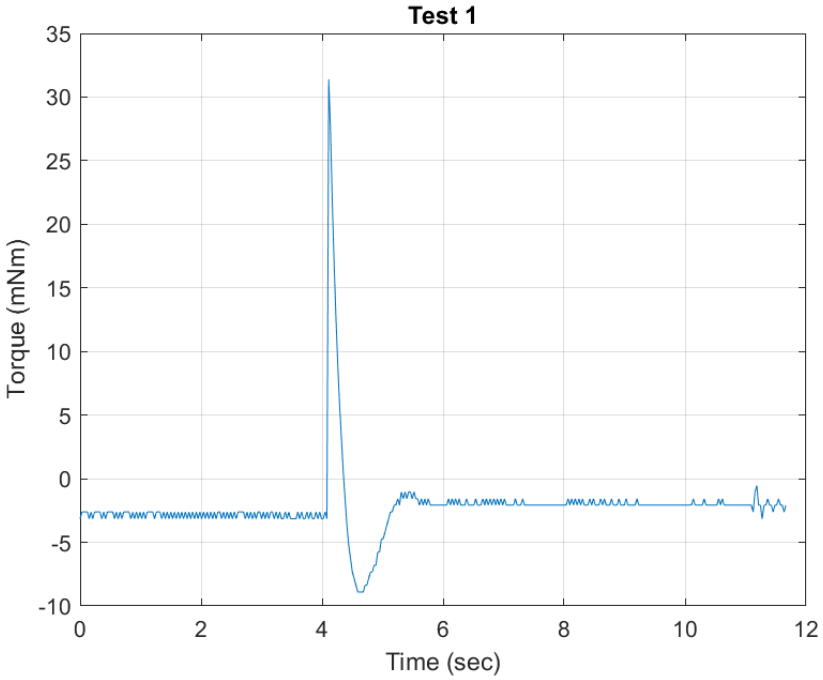


Figure 7

3.6 Response of Modified Control System

When modifying the control system, the system receives an input to hold at $\theta = 0$, with the K_p set as a non-zero value and K_d set to 0. Note that the presence of damping is necessary to settle the angular position

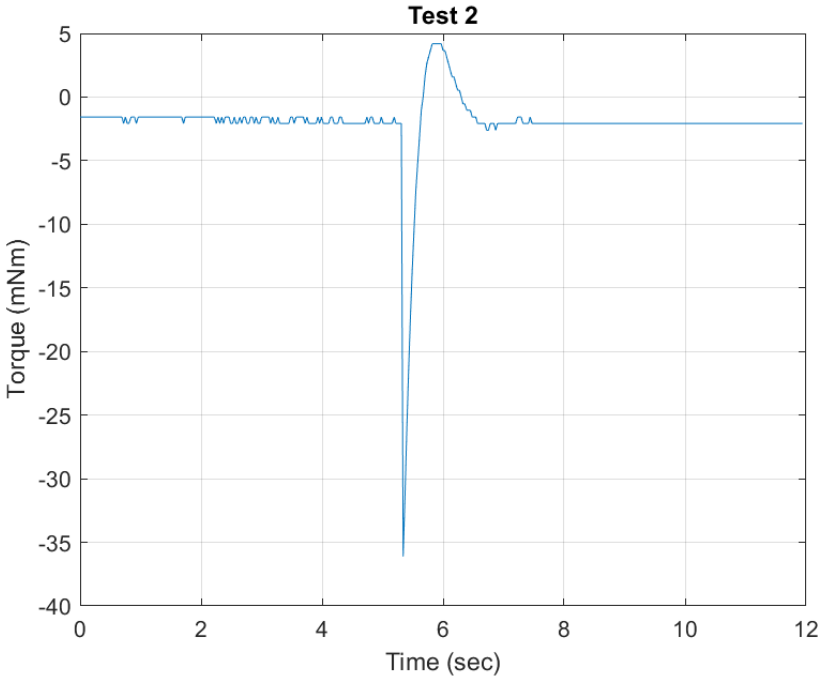


Figure 8

to a desired value. The natural friction in the system provides the necessary damping. Having a larger K_p means more oscillations and a longer settling time. If the K_p is too large, the spacecraft spins out of control. It seemed that a relatively small proportional gain was best, though it would not respond to the disturbance as quickly.

3.7 Effect of Integral Control Gain

Integral control gain accounts for the sign error between the input and position values. The controller will require a step input reference command that will result in an increase in the position value (despite being close to its desired value) if the integrated difference is positive. This can potentially result in an overshoot but can be controlled using a derivative control subsystem. To make the system even more accurate and responsive, the integral control gain is used alongside the proportional and derivative controls. This together forms a Proportional Integral Derivative controller also known simply as a PID controller. Therefore it is important to understand the effect the integral control gain aspect has on the overall system and the data being input or output. The addition of the integral control subsystem does not help the system to settle more quickly or overshoot less, but it does help combat the settling error. A larger K_i helps the system settle closer to the desired value, which is important in this experiment as the inherent imbalances causes an error in the position that gets worse over time.

4 Conclusion and Recommendations

In this lab, the users were able to visually see the effects of changing the proportional and derivative gains in order to implement them in the respective control feedback. Using MATLAB, we were able to calculate the necessary parameters given in this lab to ensure that the requirements were met in overshoot and settling time. The differences between the proportional gain and derivative gain as the proportional gain measures the error and the derivative gain measures the rate of change of the response. We believe that more experiments and results using the integral gain could have been more useful in our understanding of and experience with implementing integral gains into a controller.

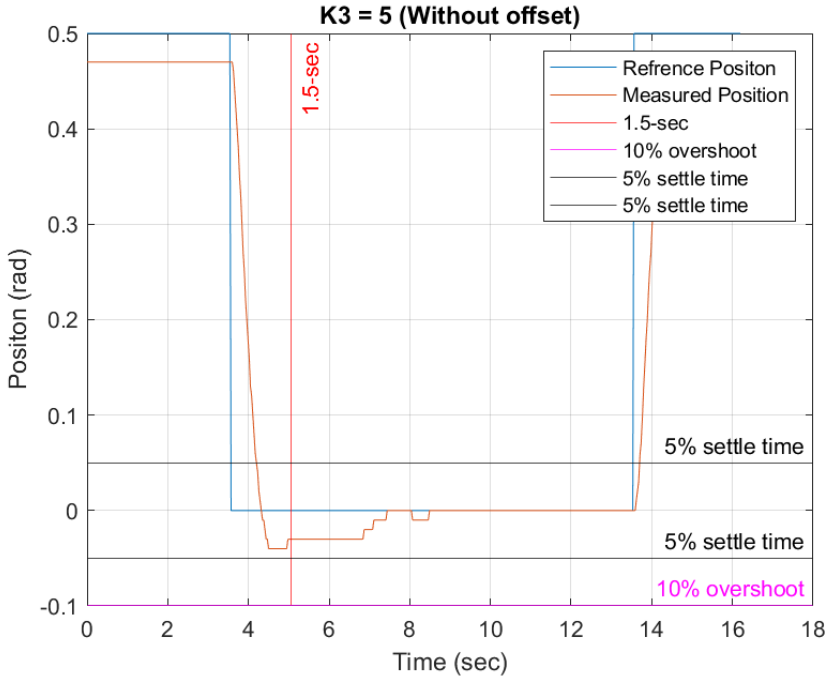


Figure 9

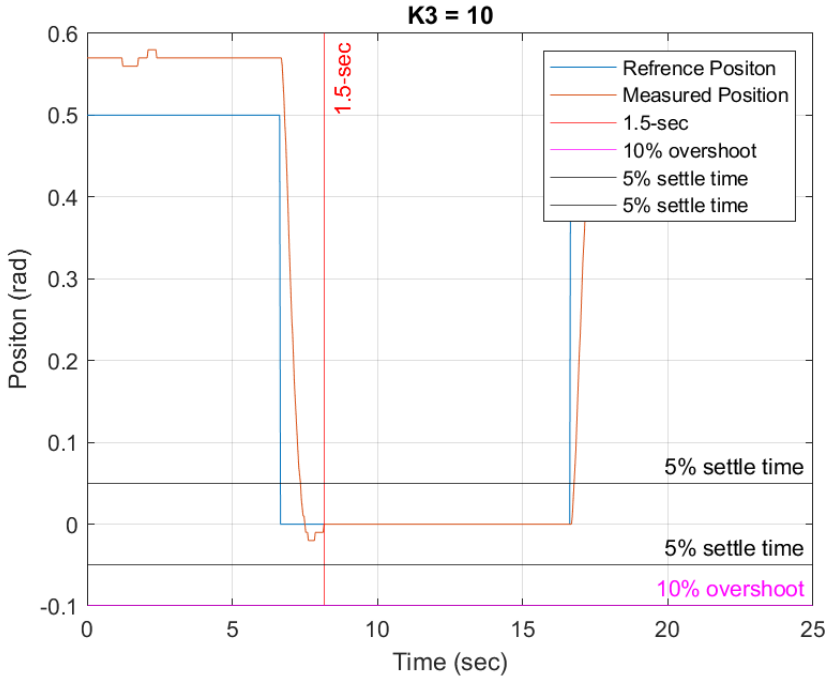


Figure 10

5 Acknowledgments and Contribution

Our group would like to acknowledge the help of the entire teaching team including our professor Dr. Casey Heidrich.

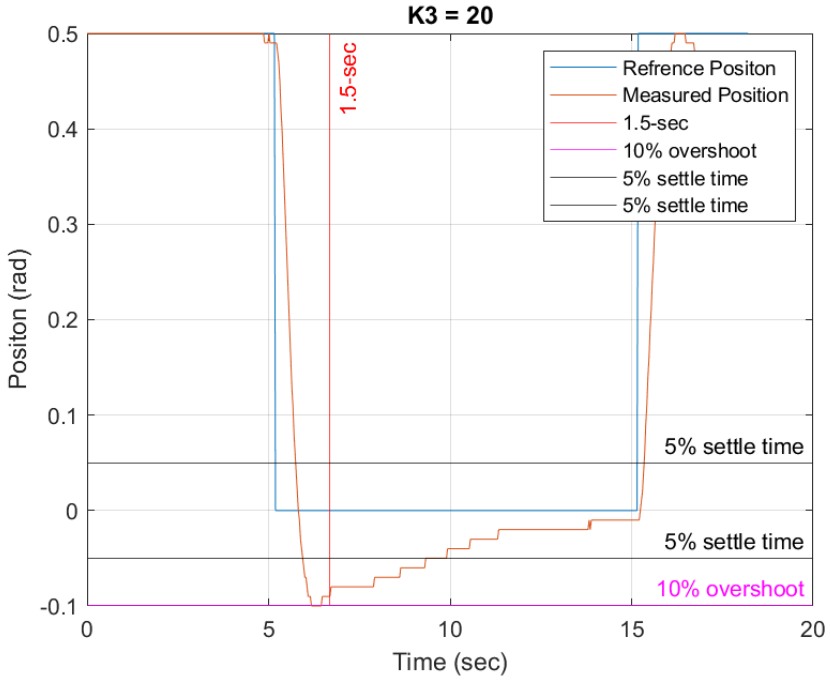


Figure 11

5.1 Member Contributions

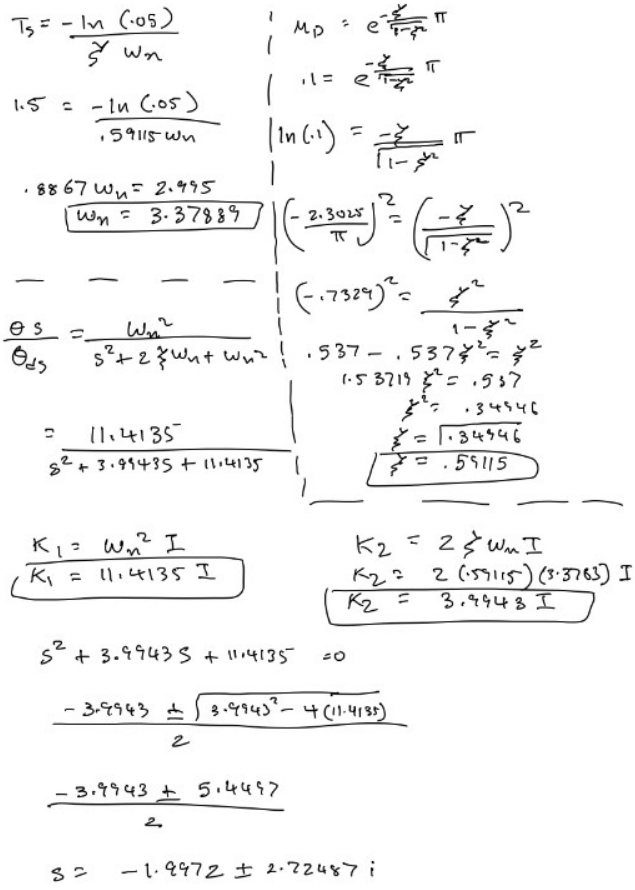
Deliverable	**Ben Chapel**	**Daniel Mascarenas**	**Rishab Pally**
Abstract	0%	0%	100%
Introduction	0%	100%	0%
Preliminary Questions	33%	33%	33%
Experiment Analysis	33%	33%	33%
Response Analysis	33%	33%	33%
Conclusions/Recommendations	0%	0%	100%
Matlab Component	33%	33%	33%
Style and Clarity	33%	33%	33%
TOTAL	**33%**	**33%**	**33%**

6 References

- ASEN 3200 Lab A-3 Spacecraft Pitch Axis Control Lab Document

7 Appendix

7.1 System Poles Derivation



7.2 MATLAB Script

```
% Analysis 2
clc;clear;close all;

kp = 85.4931 ;

t = linspace(0,6);
w_n = sqrt(kp);
z =  0.5911;


num = [1];
den = [1 29.9194 85.4931];
systf = tf(num,den);


[beta1, t1] = step(systf);

s1 = -z.*w_n+((w_n).*sqrt(z.^2-1));
s2 = -z.*w_n-((w_n).*sqrt(z.^2-1));
```

```
S = [s1;s2]

figure()
plot(t1,beta1,"-k");
title('Predicted Response')
grid on
xlabel('Time (sec)')
ylabel('Postion (rad)')
%saveas(gcf,"Predicted_Response.png")
```

```
%% Analysis 3


%% Experimental Design

T1 = readmatrix("tested_gains1");
T2 = readmatrix("tested_gains2");

t1 = T1(:,1)/1000 - T1(1,1)/1000; %% Converts ms to sec
ref_pos1 = T1(:,2); % Position in radians
mes_pos1 = T1(:,3);
mes_p1 = mes_pos1 - T1(1,3); %% If perfect response
Torque_1 = T1(:,4)*33.5;

figure()
% subplot(2,1,1)
plot(t1, ref_pos1);
grid on
hold on
plot(t1, mes_pos1);
xline(5.53,'-r', {'1.5-sec'})
yline(0.55,'-m', {'10% overshoot'})
yline(0.475,'-k', {'5% settle time'})
yline(0.525,'-k', {'5% settle time'})
legend("Refrence Positon", "Measured Position", "1.5-sec","10% overshoot","5% settle time",
title("Test 1")
xlabel('Time (sec)')
ylabel("Positon (rad)")
% saveas(gcf,"Fig_1_Test1.png")

figure()
plot(t1, ref_pos1);
grid on
hold on
plot(t1, mes_p1);
xline(5.53,'-r', {'1.5-sec'})
yline(0.55,'-m', {'10% overshoot'})
```

```
yline(0.475,'-k', {'5% settle time'})
yline(0.525,'-k', {'5% settle time'})
legend("Refrence Positon", "Measured Position", "1.5-sec","10% overshoot","5% settle time",
title("Test 1 (Without offset)")
xlabel('Time (sec)')
ylabel("Positon (rad)")
% saveas(gcf,"Fig_2_Test1_processed.png")

figure()
plot(t1,Torque_1)
title('Test 1')
xlabel('Time (sec)')
ylabel('Torque (mNm)')
grid on
% saveas(gcf,"Test1_Torque.png")

%% Test 2


t2 = T2(:,1)/1000 - T2(1,1)/1000;%% Converts ms to sec
ref_pos2 = T2(:,2);% Position in radians
mes_pos2 = T2(:,3);
mes_p2 = mes_pos2 - 0.02; %% If perfect response
Torque_2 = T2(:,4)*33.5;

figure()
plot(t2, ref_pos2);
grid on
hold on
plot(t2, mes_pos2);
xline(6.84,'-r', {'1.5-sec'})
yline(-0.1,'-m', {'10% overshoot'})
yline(-0.05,'-k', {'5% settle time'})
yline(0.05,'-k', {'5% settle time'})
legend("Refrence Positon", "Measured Position", "1.5-sec","10% overshoot","5% settle time",
title("Test 2")
xlabel('Time (sec)')
ylabel("Positon (rad)")
 saveas(gcf,"Fig_3_Test2.png")

figure()
plot(t2, ref_pos2);
grid on
hold on
plot(t2, mes_p2);
xline(6.84,'-r', {'1.5-sec'})
yline(-0.1,'-m', {'10% overshoot'})
yline(-0.05,'-k', {'5% settle time'})
yline(0.05,'-k', {'5% settle time'})
legend("Refrence Positon", "Measured Position", "1.5-sec","10% overshoot","5% settle time",
title("Test 2 (Without offset)")
xlabel('Time (sec)')
ylabel("Positon (rad)")
 saveas(gcf,"Fig_4_Test2_processed.png")
```

```
figure()
plot(t2,Torque_2)
title('Test 2')
xlabel('Time (sec)')
ylabel('Torque (mNm)')
grid on
% saveas(gcf,"Test2_Torque.png")



% Analysis 7


T_i_1 = readmatrix("k3_5");
T_i_2 = readmatrix("k3_10");
T_i_3 = readmatrix("k3_20");

t1_i = T_i_1(:,1)/1000 - T_i_1(1,1)/1000; %% Converts ms to sec
ref_pos1_i = T_i_1(:,2); % Position in radians
mes_pos1_i = T_i_1(:,3);
mes_p1_i = mes_pos1_i + .03;%- T_i_1(1,3); %% If perfect response
Torque_1_i = T_i_1(:,4)*33.5;


t2_i = T_i_2(:,1)/1000 - T_i_2(1,1)/1000; %% Converts ms to sec
ref_pos2_i = T_i_2(:,2); % Position in radians
mes_pos2_i = T_i_2(:,3);
mes_p2_i = mes_pos2_i -.07;%- T_i_2(1,3); %% If perfect response
Torque_2_i = T_i_2(:,4)*33.5;


t3_i = T_i_3(:,1)/1000 - T_i_3(1,1)/1000; %% Converts ms to sec
ref_pos3_i = T_i_3(:,2); % Position in radians
mes_pos3_i = T_i_3(:,3);
mes_p3_i = mes_pos3_i ;%- T_i_3(1,3); %% If perfect response
Torque_3_i = T_i_3(:,4)*33.5;


figure()
plot(t1_i, ref_pos1_i);
grid on
hold on
plot(t1_i, mes_pos1_i);
xline(5.07,'-r', {'1.5-sec'})
yline(-0.1,'-m', {'10% overshoot'})
yline(-0.05,'-k', {'5% settle time'})
yline(0.05,'-k', {'5% settle time'})
legend("Refrence Positon", "Measured Position", "1.5-sec","10% overshoot","5% settle time",
title("K3 = 5 (Without offset)")
xlabel('Time (sec)')
ylabel("Positon (rad)")
```

```
 saveas(gcf,"k3_5P.png")

figure()
plot(t2_i, ref_pos2_i);
grid on
hold on
plot(t2_i, mes_pos2_i);
xline(8.16,'-r', {'1.5-sec'})
yline(-0.1,'-m', {'10% overshoot'})
yline(-0.05,'-k', {'5% settle time'})
yline(0.05,'-k', {'5% settle time'})
legend("Refrence Positon", "Measured Position", "1.5-sec","10% overshoot","5% settle time",
title("K3 = 10")
xlabel('Time (sec)')
ylabel("Positon (rad)")
 saveas(gcf,"k3_10P.png")

figure()
plot(t3_i, ref_pos3_i);
grid on
hold on
plot(t3_i, mes_pos3_i);
xline(6.69,'-r', {'1.5-sec'})
yline(-0.1,'-m', {'10% overshoot'})
yline(-0.05,'-k', {'5% settle time'})
yline(0.05,'-k', {'5% settle time'})
legend("Refrence Positon", "Measured Position", "1.5-sec","10% overshoot","5% settle time",
title("K3 = 20")
xlabel('Time (sec)')
ylabel("Positon (rad)")
 saveas(gcf,"k3_20P.png")
```