Angle (Attitude) Control Assignment

Table of Contents

Name: Tanner Lex Jones

Preliminaries

```
% This cleans all variables and sets the format to display more
 digits.
clearvars
close all
clc
format long
warning('off')
```

ctrlpref

```
% Addpath to Attitude Representations Folder
addpath('../01 Attitude Representations')

% Addpath to Attitude Kinematics Folder
addpath('../02 Attitude Kinematics')

% Addpath to Attitude Dynamics Folder
```

```
addpath('../03 Attitude Dynamics')

% Load qBus
load qBus.mat

% Load Mass Properties
mass_properties
```

Parameters

```
dt_delay = 0.01; % seconds

% amount of time to run simulation
t_sim = 6; % seconds

% Initial attitude
e = [1; 1; 1]; e = e/norm(e);
q0_BI = e2q(e, 0*pi/180);
% Desired Attitude
qstar_BI = e2q(e, 180*pi/180);
ramp_slope = pi; % ramp slope (rad/s)

A0_BI = q2A(q0_BI);
A0_IB = A0_BI';

% Initial Satellite Angular Velocity
wbi0_B = [0; 0; 0];%*pi/180; % rad/s
wbi0_P = A_PB*wbi0_B;

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

% Disturbance parameters
d_type = -1;
dist_level = 0.01;
```

Calculate the principal open loop plant models

```
G1 = 1/(J_C_P(1,1)*s);
display(G1);
G2 = 1/(J_C_P(2,2)*s);
display(G2);
G3 = 1/(J_C_P(3,3)*s);
display(G3);
```

```
G1 =

      1
  ----------
  0.007089 s

Continuous-time transfer function.
```

```
G2 =

      1
  ---------
  0.03852 s

Continuous-time transfer function.


G3 =

      1
  ---------
  0.03994 s

Continuous-time transfer function.
```

Inner Loop Proportional Control Design

```
% Open loop gain crossover frequency in the controller
PM = 65*pi/180; % Phase Margin (rad)
DC_phase = -90*pi/180; % Phase at DC (rad)
% -pi = DC_phase - Phase Margin - Xover_angle
% Xover_angle = DC_phase -PM + pi
Xover_angle = DC_phase - PM + pi;

% crossover frequency
w_crossover = Xover_angle/dt_delay; % rad/s
display(w_crossover/(2*pi), 'Open Loop Gain Crossover Frequency
 (Hz)');

% Design proportional control gains such that all three axes have
 identical
% responses.
Kd1 = 1/bode(G1, w_crossover);
display(Kd1);
Kd2 = 1/bode(G2, w_crossover);
display(Kd2);
Kd3 = 1/bode(G3, w_crossover);
display(Kd3);

% Control gains as diagonal matrix for input into simulink.
Kd = diag([Kd1; Kd2; Kd3]);
```

```
Open Loop Gain Crossover Frequency (Hz) =

   6.944444444444444
```

```
Kd1 =

   0.309328870616099


Kd2 =

   1.680748066474290


Kd3 =

   1.742853620369665
```

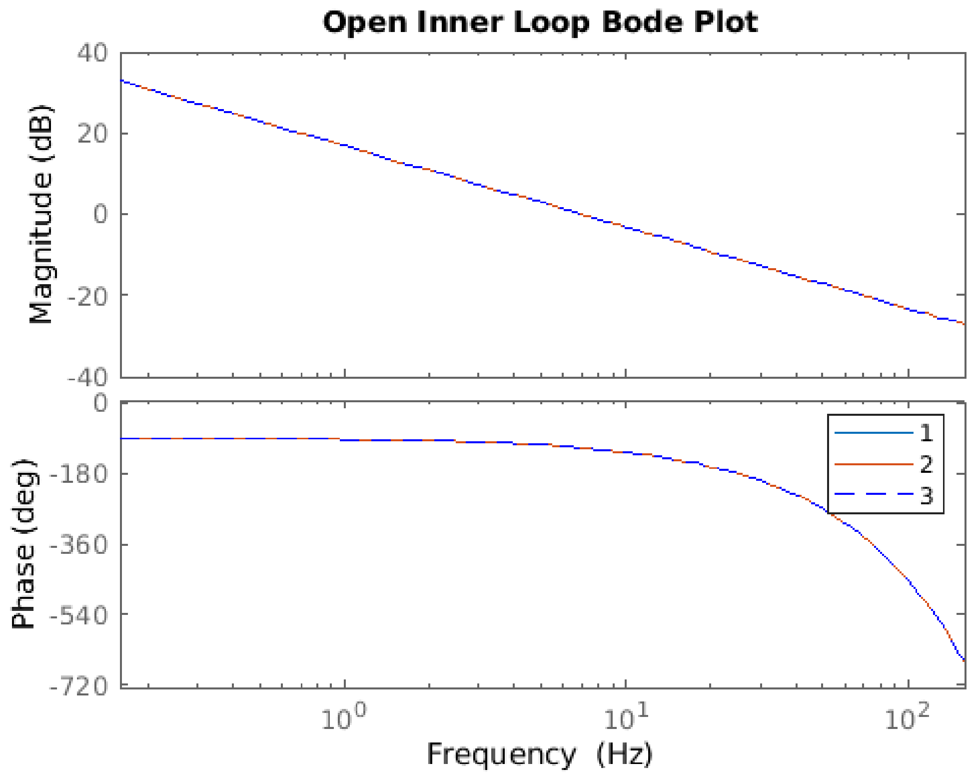
Open Inner Loop Bode Plot of each axis with the controller in the loop

This bode plot shows the response of the open loop system. The magnitude shows a response of logarithmic decrease and a phase that starts at -90deg and rolling off which are both typical of a proportional controller with an integrator plant. The three axes are equal because the gains were all calculated to achieve the same crossover frequency.

```
% Define each controller
C1 = tf(Kd1,'OutputDelay', dt_delay);
display(C1);
C2 = tf(Kd2,'OutputDelay', dt_delay);
display(C2);
C3 = tf(Kd3,'OutputDelay', dt_delay);
display(C3);

figure
bode(C1*G1, C2*G2, C3*G3, {1,1000}, '--');
title('Open Inner Loop Bode Plot');
legend('1', '2', '3');
```

```
C1 =

  exp(-0.01*s) * (0.3093)

Continuous-time transfer function.


C2 =

  exp(-0.01*s) * (1.681)

Continuous-time transfer function.


C3 =
```

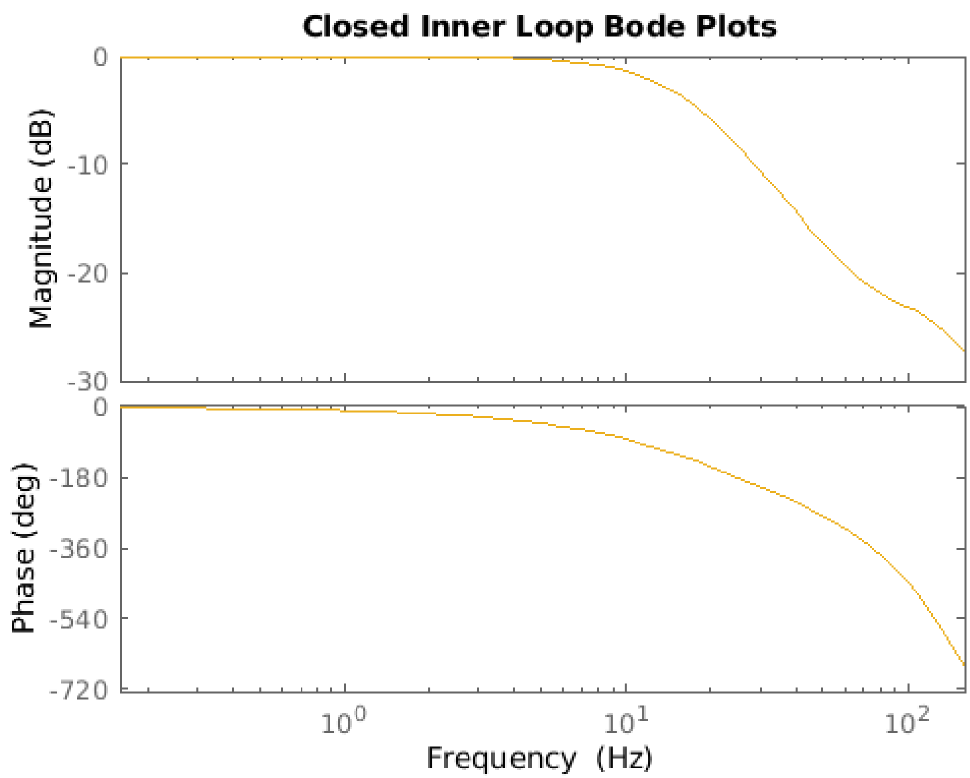
```
  exp(-0.01*s) * (1.743)

Continuous-time transfer function.
```



Closed Inner Loop Bode Plot for each of the three axes

This closed loop bode of the inner loop show that this part of the system acts as a low pass filter which is seen by the magnitude plot. It is mostly flat in the lower frequencies and after the 3db bandwidth frequency (~14Hz) it rolls off more steeply. The phase also decreases with increase in frequency, starting at 0deg and already at ~-115deg at the bandwidth frequency.

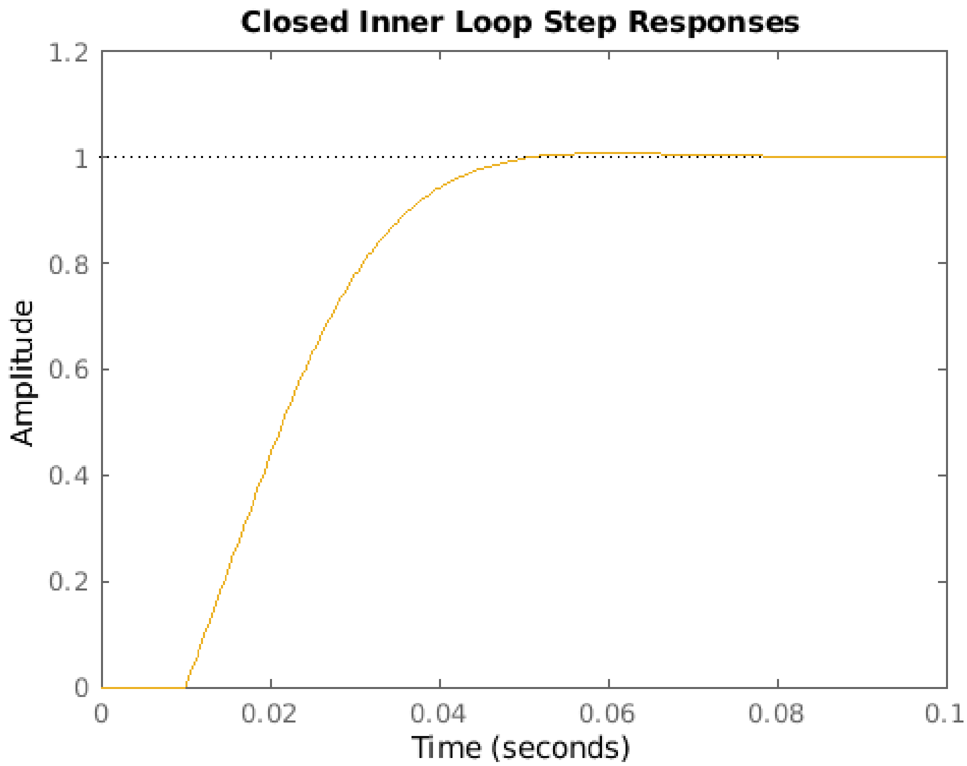
```
CLTF1 = feedback(C1*G1, 1);
CLTF2 = feedback(C2*G2, 1);
CLTF3 = feedback(C3*G3, 1);
figure
bode(CLTF1, CLTF2, CLTF3, {1,1000})
title('Closed Inner Loop Bode Plots');
```



Closed Inner Loop Step responses

The step responses also are exactly the same. You can see the rise and settling time are both good in this plot. Also notice that the overshoot appears to be small.

```
figure
step(CLTF1, CLTF2, CLTF3, 0.1)
title('Closed Inner Loop Step Responses');
```



Inner Loop Phase and Gain Margins

Each axis has the same phase and gain margins: 65deg, and 11.126dB. The PM is above the design requirement by 5 degrees and the GM is well above the requirement of 6dB. These margins will ensure good stability even if the true values may be off due to measurements or other errors.

```
[GM1, PM1] = margin(C1*G1);
display(mag2db(GM1), 'Gain Margin 1 (dB)');
display(PM1, 'Phase Margin 1 (degrees)');
```

```
Gain Margin 1 (dB) =

  11.126050015345745


Phase Margin 1 (degrees) =

  64.999999999999972
```

Inner Loop Step Response Information

Because the design has an equal phase margin for each axis the step response is also exactly the same. Choosing the 65deg phase margin gives a quick 24ms rise time, with a small overshoot of 0.8%. The most important design criteria was first stability and then settling time, which was 45ms. With a faster system, for example 60deg PM, the settling time increased to 60ms, and a slower system (70deg PM) also had increased settling time (72ms).

```
stepinfo1 = stepinfo(CLTF1);
display(stepinfo1);
```

```
stepinfo1 = 

  struct with fields:

        RiseTime: 0.024144674359168
    SettlingTime: 0.045243684099428
     SettlingMin: 0.901331578912983
     SettlingMax: 1.007954289762815
       Overshoot: 0.795428976281487
      Undershoot: 0
            Peak: 1.007954289762815
        PeakTime: 0.060167040273008
```

Outer Loop Control Design

The first design criteria is to stabilize the system. The next most important criteria was again chosen to be the settling time of the step response. For this the PM was chosen to be 65deg.

```
[num, den] = pade(dt_delay, 8);
C_pade8 = tf(num, den);

config = sisoinit(6);
config.G1.value = G1;
config.C1.value = 1; % initially just 1, to be tuned below
config.C2.value = Kd1*C_pade8;
config.G2.value = 1/s;
config.OL1.View = {'bode'};
config.OL2.View = {};
% controlSystemDesigner(config);

Ko = 2948.2; % outer loop control gain
Zo = -0.01; % zero location (rad/s)
Po = -188.5; % pole location (rad/s)
Co = Ko*(s-Zo)/(s*(s-Po));
OLTF = Co*CLTF1/s;
% config.C1.value = Co;
% controlSystemDesigner(config);
```

Outer Loop Design Questions

What is the gain crossover frequency of your outer loop? 2.46 Hz

Why was that gain crossover frequency chosen? In order to get a 65 deg PM because it gives a great settling time.

How does this gain crossover frequency compare to the gain crossover frequency of the inner loop? It is at less than half the frequency of the inner loop. That is due to the inner loop acting as a lowpass filter. It will necessarily decrease the available bandwidth of the outer loop.

What is the frequency of the real zero? 1.59E-3 rad/s **What effect does the zero have on disturbance rejection?** The lower the frequency of the zero, the longer it takes to reject a constant disturbance. Increasing the frequency makes the disturbance rejection faster, but may also decrease stability because the zero is used to buy back phase from the integrator.

What effect does the zero have on the closed loop step response? The effect of the zero on the system is first to create stability where it otherwise could not exist because the phase response of the system is -180 deg at DC. The tradeoff of the location of the zero is that when it is at higher frequency it cannot quickly settle to a commanded input, but it does reject constant disturbance very quickly. On the other hand, when it is placed at a very low frequency it will not noticeably affect the step response, but it will take much longer to reject a constant disturbance.

What effect does the zero have on the ability to track a ramp? The zero makes the system stable, but it makes it harder to track a ramp because it undoes the effect of the integrator.

Why did you place the zero where you did? This is a question of the disturbance rejection vs stability. I placed the zero such that with ~12 min of control, any constant disturbance is rejected. This assumes that the controller will be used for long periods of time and that the initial ~12 min is relatively a low cost. Because it is a slow zero the system has a lot more range of stability.

What is the frequency of the real pole? 30 rad/s **What effect does the pole have on system stability?** The pole may cause instability if it is placed at a low enough frequency. If the pole is too slow there may not exist a gain such that a 65 deg phase margin can be implemented.

Does the pole have any other effect on the system? Yes, the reason for the pole is that the controller can be used without infinite power, because the pole rolls off the magnitude response for higher frequencies. Any practical controller would otherwise be impossible because of high frequency noise that will enter any system.

Open Outer Loop Bode Plot

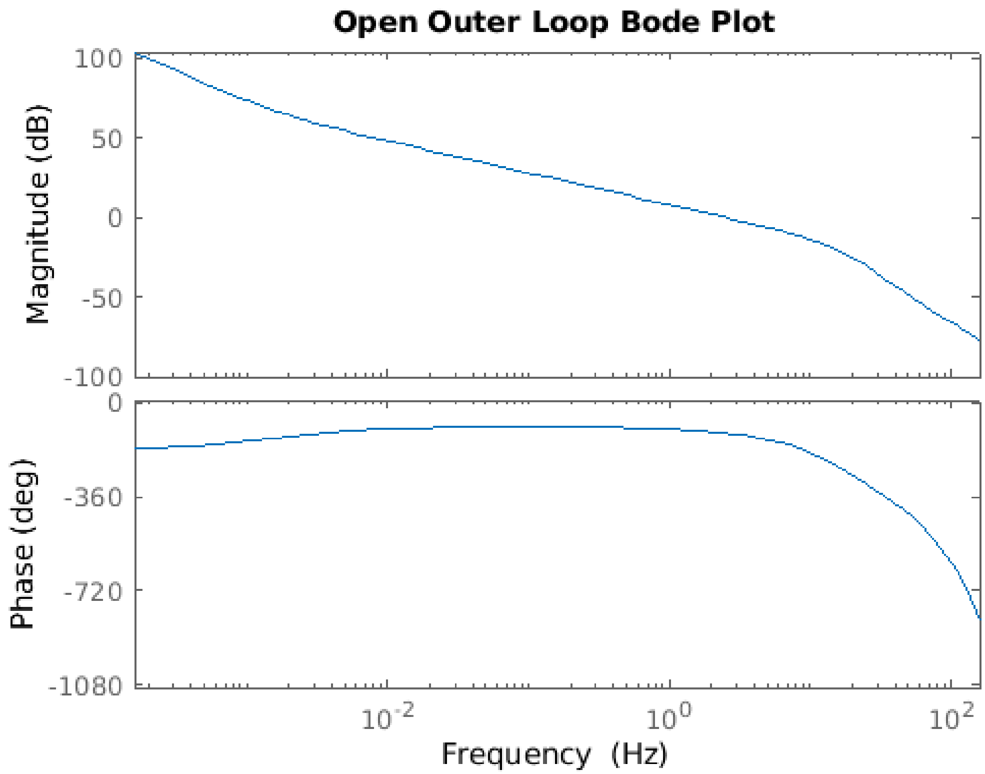
The Open Loop Bode Plot for the Outer loop shows both the gain margin and the phase margin. This is useful for determining the stability of the system. You can see how the phase bump between the zero and the pole creates a region of stable phase margin from which to choose the gain.

What is the phase of your controller at low frequency? -180 degrees

What happens to the phase when the zero is added? At the zero frequency the phase is increased by 45 degrees and it adds a total of 90 phase to the system, most of which is added within the next decade.

What happens to the phase when the pole is added? The pole causes a decrease in phase just as the zero causes an increase in phase.

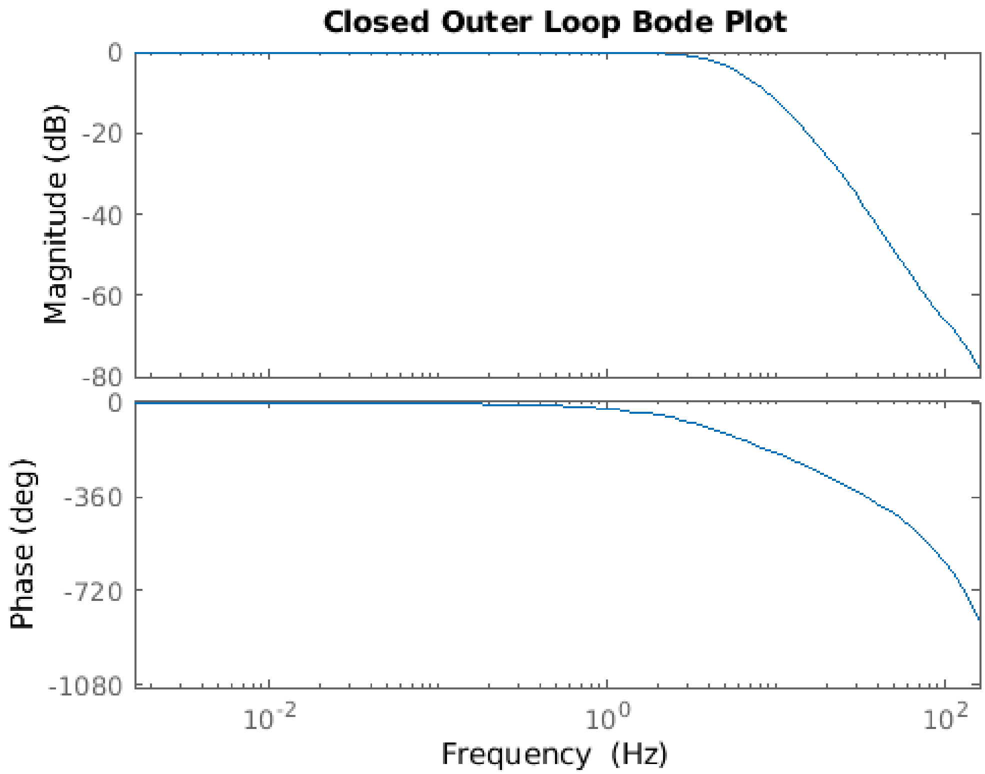
```
figure
bode(OLTF, {0.001, 1000});
title('Open Outer Loop Bode Plot');
```



Closed Outer Loop Bode Plot

As seen from the closed loop bode plot the system acts as a low pass filter. This is the ultimate tool if you need to know, given a sinusoidal input exactly what the output would be. The magnitude plot tells you what amplitude to expect relative to the input and the phase plot tells you how much the output will lag behind the input.

```
COLTF = feedback(OLTF,1);
figure
bode(COLTF, {0.01, 1000});
title('Closed Outer Loop Bode Plot');
```



Relative Stability Margins

The chosen phase margin is very close to the actual phase margin which will give a good settling time. The gain margin is much higher than the requirement of 6dB, so this should be a relatively stable system.

```
% The phase margin and gain margin of the outer loop
[GMO, PMO] = margin(OLTF);
display(mag2db(GMO), 'Gain Margin 1 (dB)');
display(PMO, 'Phase Margin 1 (degrees)');
```

```
Gain Margin 1 (dB) =

  12.487430932888126


Phase Margin 1 (degrees) =

  64.896938620074664
```

Step Response Performance Characteristics

The rise time of the system is 72ms and settling time is 126ms which is good considering the stability and also accounting for disturbance rejection that has also constrained the system. The overshoot is 1.66% which is noticable, but tolerable considering the priority is to get the best settling time.

```
% Outer loop step response performance characteristics
COLTF = feedback(OLTF, 1);
stepinfoO = stepinfo(COLTF);
display(stepinfoO);
```

```
stepinfoO =

  struct with fields:

        RiseTime: 0.072009746015994
    SettlingTime: 0.125677456328153
     SettlingMin: 0.900644466840607
     SettlingMax: 1.016573062740016
       Overshoot: 1.657306274001558
      Undershoot: 0
            Peak: 1.016573062740016
        PeakTime: 0.168231409776350
```

Derivations

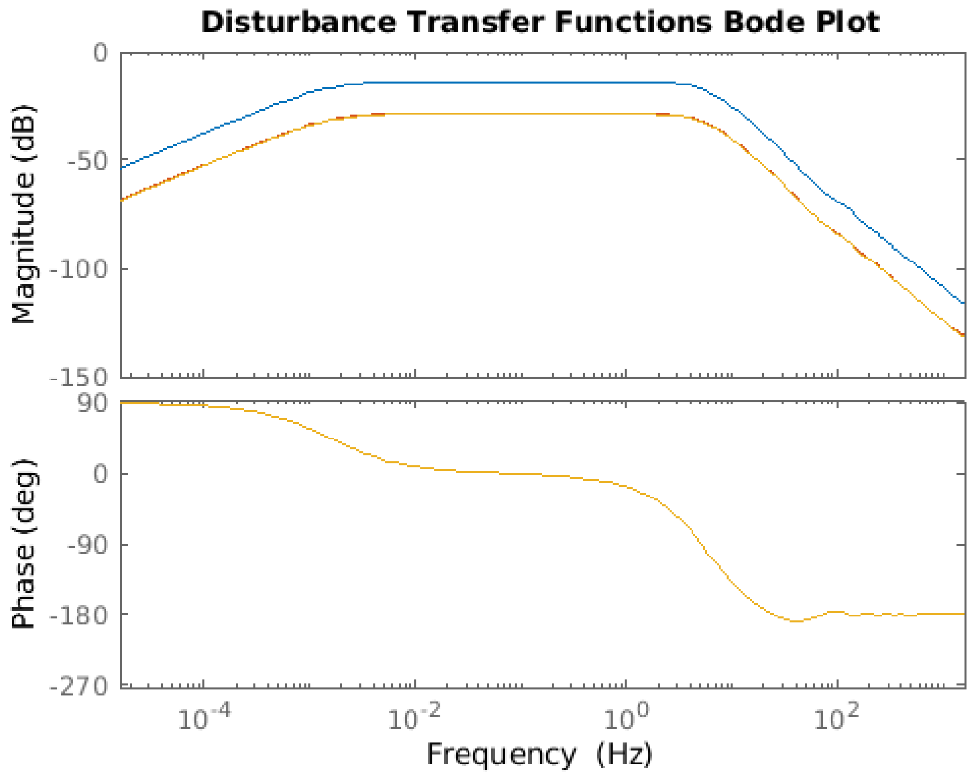
The derivation of the steady state error of the system to a ramp input is included elsewhere in this report.

The derivation of the transfer function between a disturbance torque input and the angle output is also included elsewhere in this report.

Disturbance Transfer Function Bode Plot

The bode plots of the disturbance transfer functions show how the system would respond to sinusoidal disturbances. These plots show that they behave as band-pass filters, rejecting the low and high frequencies. Note that the separate axes have different magnitudes, but that the phase is the same. This is because of the different moments of inertia for each axis.

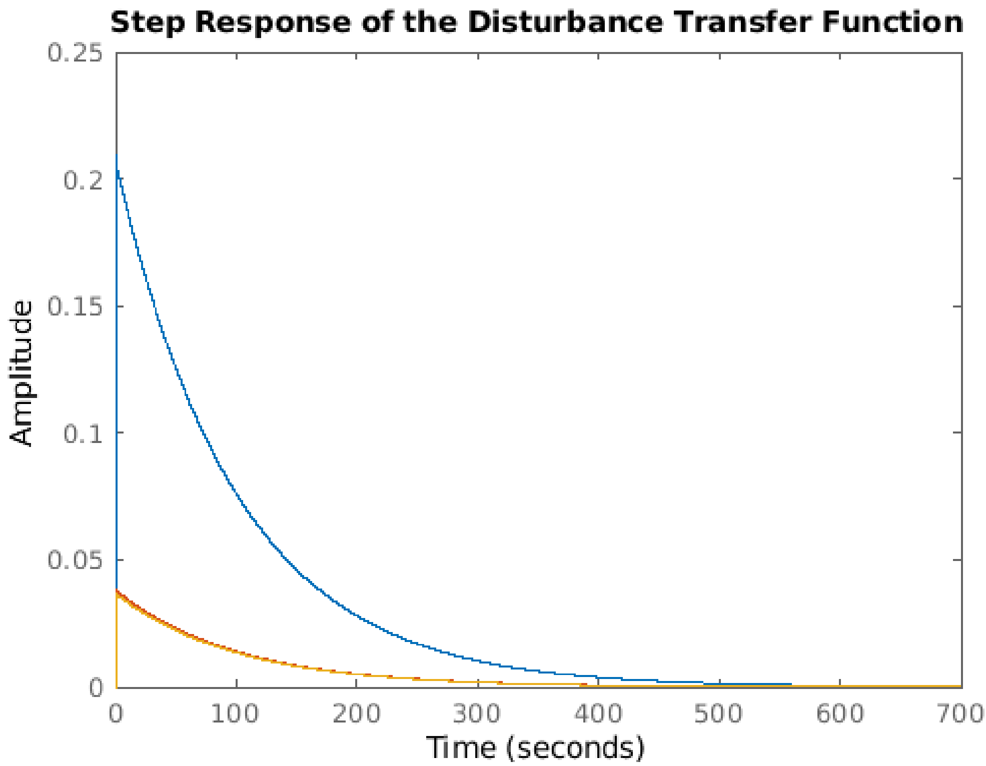
```
Go = 1/s;
DTF1 = G1*Go/(1 + C1*G1 + C1*Co*G1*Go);
DTF2 = G2*Go/(1 + C2*G2 + C2*Co*G2*Go);
DTF3 = G3*Go/(1 + C3*G3 + C3*Co*G3*Go);
figure
bode(DTF1, DTF2, DTF3, {.0001, 10000});
title('Disturbance Transfer Functions Bode Plot');
```



Step Response of the Disturbance Transfer Function

As discussed above on the question of where to place the zero in the outer loop control, there is great advantage to having a slow zero, the drawback is the time needed for the controller to reject constant disturbance. This step response of the disturbance transfer function shows that ~12min is needed. After that time there is no perceivable effect of a constant disturbance. Also note that because each axis has a different transfer function for the disturbance, the step response plots are different in magnitude.

```
figure
step(DTF1, DTF2, DTF3, 700);
title('Step Response of the Disturbance Transfer Function');
```



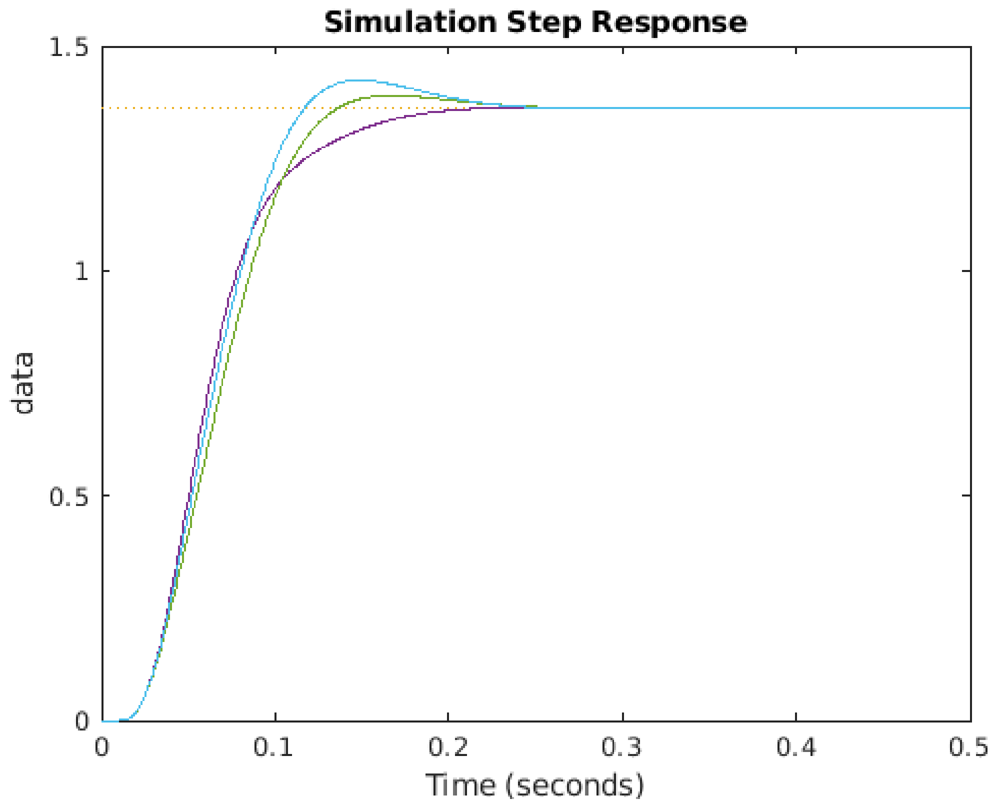
Simulink Simulation - Step Response

The response of the system to a commanded step input is what would be expected based on the analytical results. The simulation shows that there could only be a negligible difference with the analytical step response. The curves from each approach appear identical.

Did the system behave like you expected? Yes, the system behaves just as expected, based on the rise time, settling time, and overshot values that were calculated analytically.

Does it behave like a linear system? Yes, this behavior is indistinguishable from the analytically calculated linear step response.

```
q0_BI = e2q(e, -45*pi/180);
qstar_BI = e2q(e, 135*pi/180);
input_type = 1; % constant input
input_param = 0; % unused parameter for constant input type
sim('AngleControl', 0.5)
figure
plot(theta_in, ':')
hold on
plot(theta_out)
title('Simulation Step Response')
```



Simulink Simulation - Ramp Response

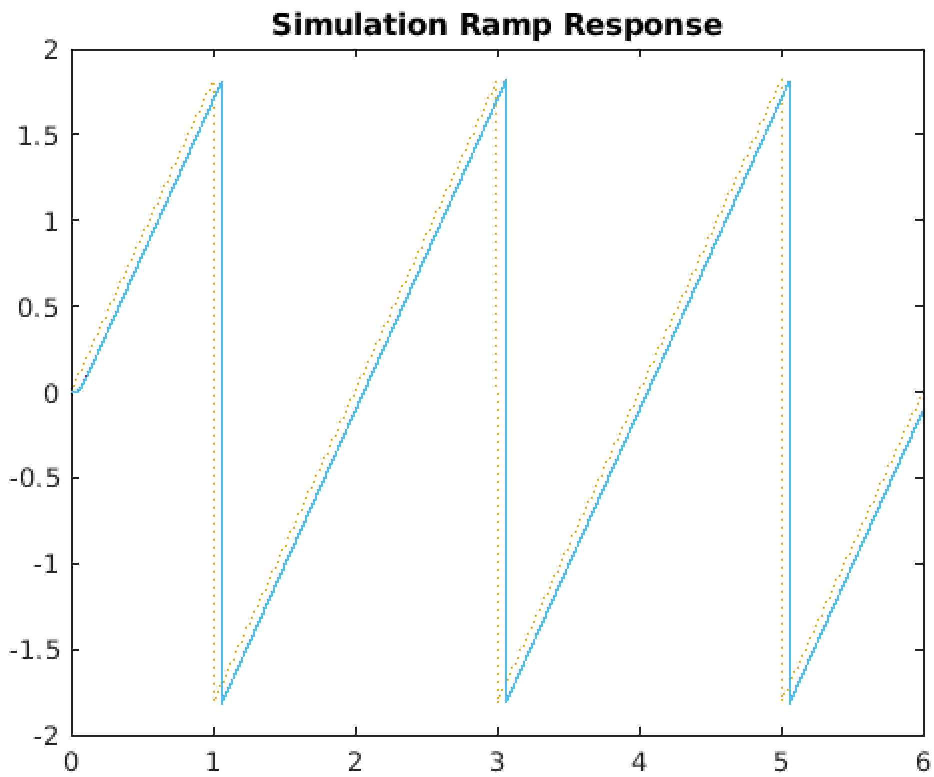
The system can follow a ramp pretty well to begin with. In the beginning it will of course lag behind the commanded input. Eventually it converges to the commanded input. This is about the same time required for disturbance rejection.

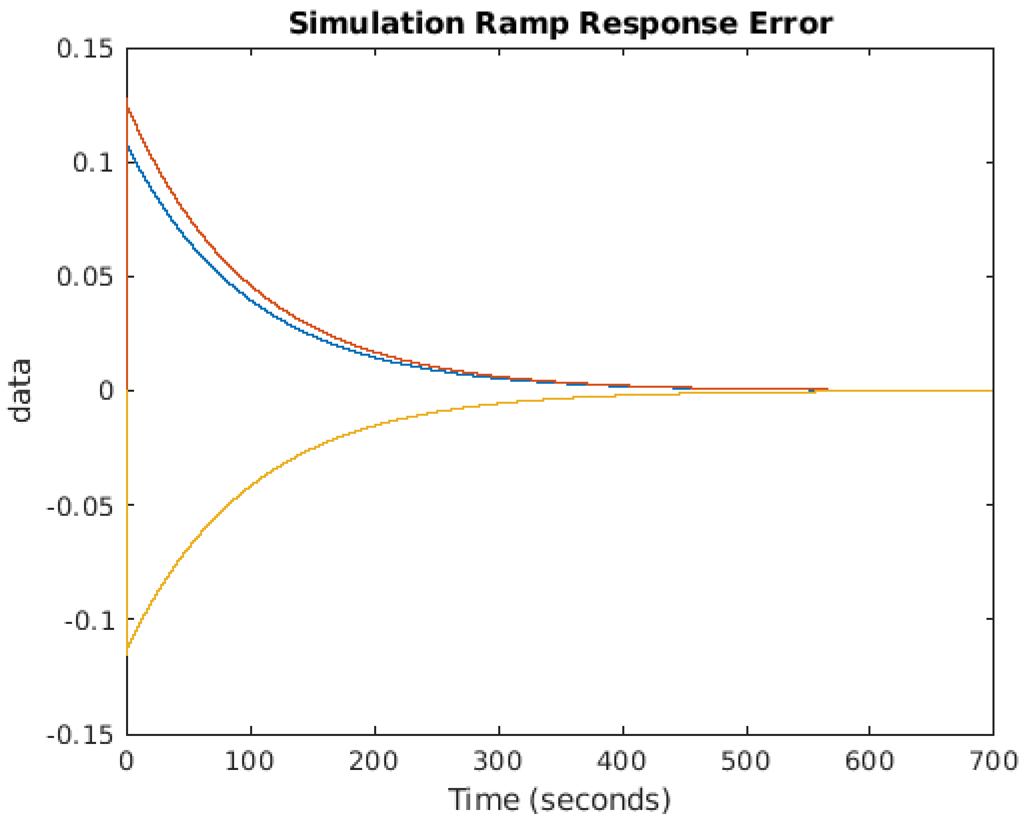
Did the system behave like you expected? I expected a lag in the tracking of a ramp, but I didn't really expect that the system would catch up. Then I derived the steady state error and believed that it would catch up with no error, but didn't know how that would look until this simulation was ran.

How long does it take to track a ramp? It takes as long as is needed for disturbance rejection. For this system that means ~12 minutes.

```
input_type = 2; % ramp input
q0_BI = e2q(e, 0*pi/180);
qstar_BI = q0_BI;
ramp_slope = pi; % ramp slope (rad/s)
input_param = ramp_slope;
sim('AngleControl', 700)
figure
plot(theta_in.Time(1:12000), theta_in.Data(1:12000,:), ':')
hold on
plot(theta_out.Time(1:12000), theta_out.Data(1:12000,:))
title('Simulation Ramp Response')
figure
plot(theta_error)
```

```
title('Simulation Ramp Response Error')
```





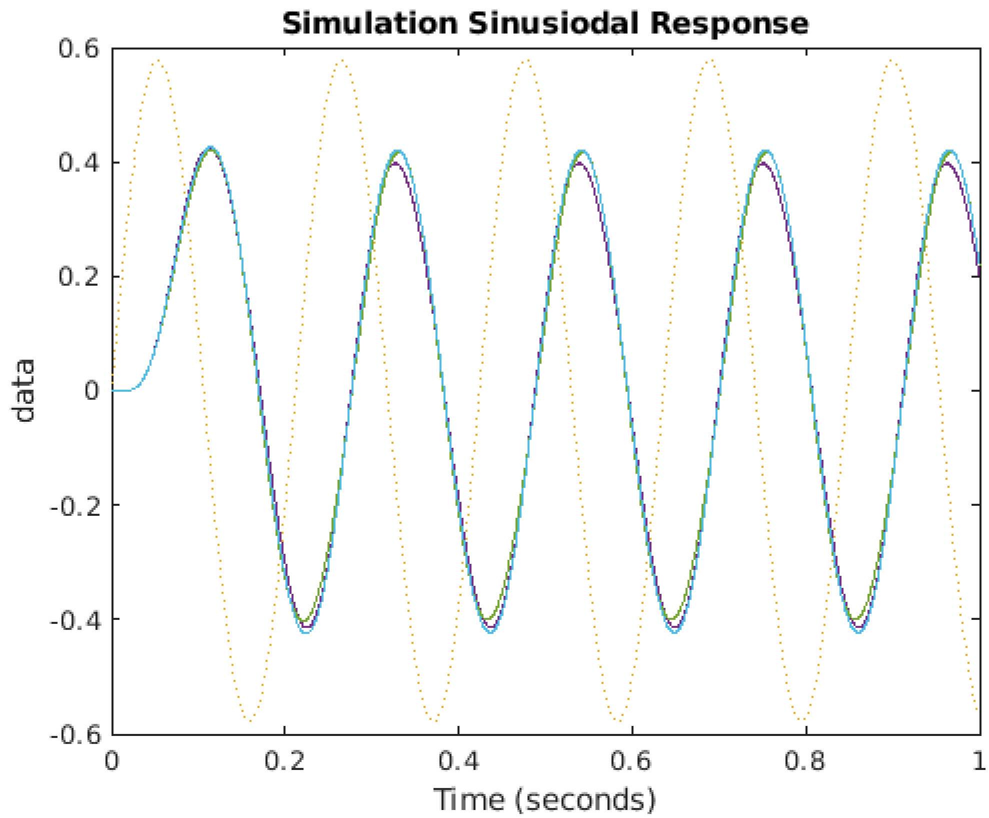
Simulink Simulation - Sinusoidal Input

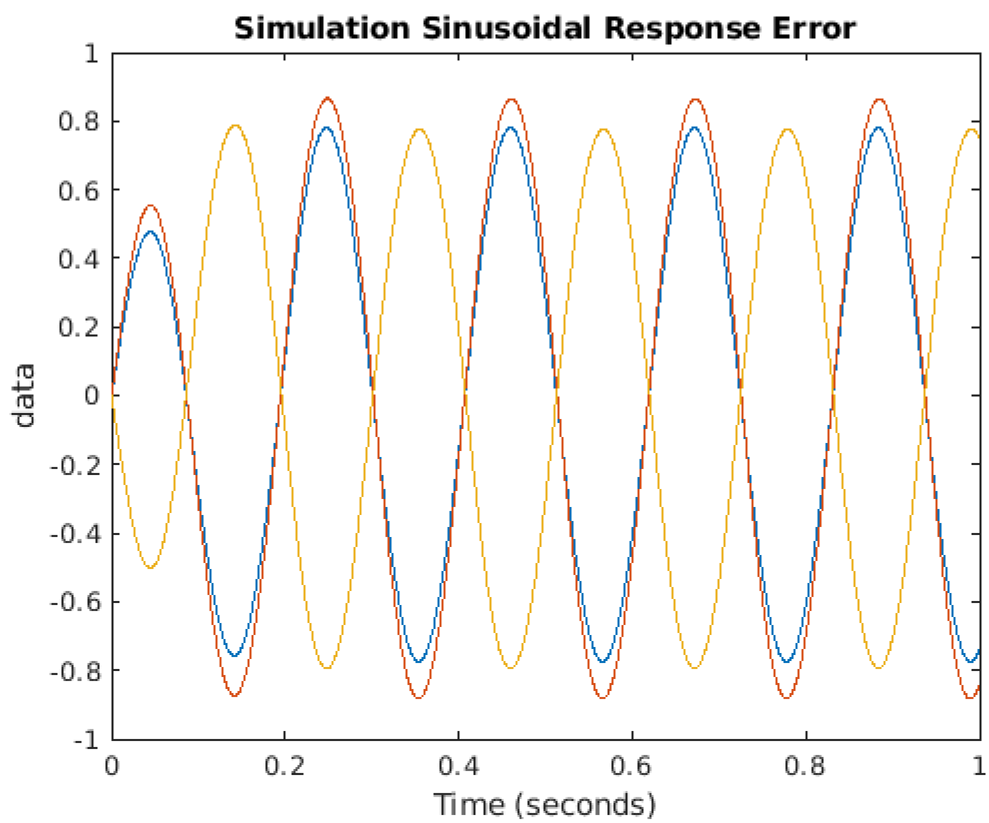
The output signal is attenuated and off phase as expected when referencing the bode plot. Even though this is the bandwidth of the system, I wouldn't say it is a frequency that is very useable. But then what good is rotating a satellite back and forth like this? The important thing here is to have another metric with which to classify the system's performance. So, I think it is a good thing that the controller can shake the satellite so, though it should not be done.

Did the system behave like you expected? Yes

How does the output sine wave compare to the commanded? It is attenuated by the 3dB (~70%), and the phase appears to lag about 90 deg.

```
input_type = 3; % sinusoidal input
freq = bandwidth(COLTF); % frequency of 3dB bandwidth (rad/s)
input_param = freq;
sim('AngleControl', 1)
figure
plot(theta_in, ':')
hold on
plot(theta_out)
title('Simulation Sinusiodal Response')
figure
plot(theta_error)
title('Simulation Sinusoidal Response Error')
```


Simulation Sinusiodal Response
data
Time (seconds)
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
0.2
0.4
0.6
0.8
1

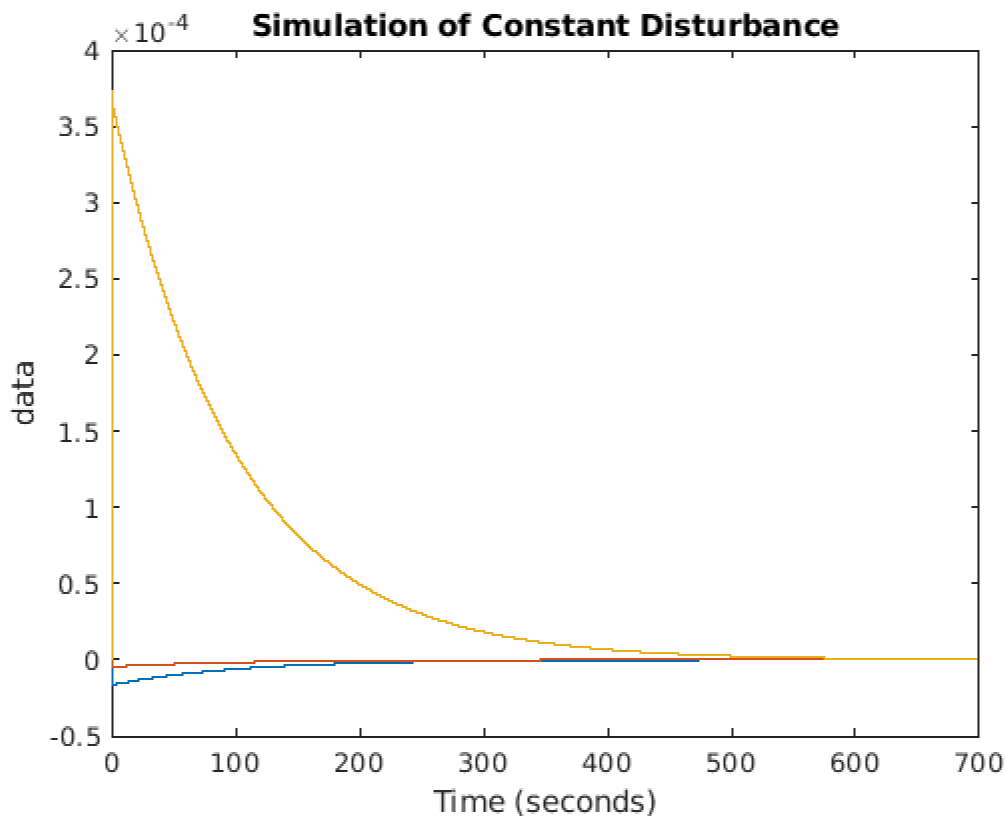

Simulation Sinusoidal Response Error
data
Time (seconds)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.2
0.4
0.6
0.8
1

Simulink Simulation - Constant Disturbance

Due to the integrator used in the outer loop controller there is a low frequency rolloff that you see in the disturbance transfer function bode plot. The results of this simulation prove that this is true.

Does the controller reject the constant disturbance? Yes **How long does it take to reject it?** Looking at the simulation plot shown it takes about 12 minutes as the analysis above would support.

```
q0_BI = e2q(e, 0*pi/180);
qstar_BI = q0_BI;
input_type = 1; % constant input
d_type = 1; % constant disturbance
sim('AngleControl', 700)
figure
plot(theta_error)
title('Simulation of Constant Disturbance')
```



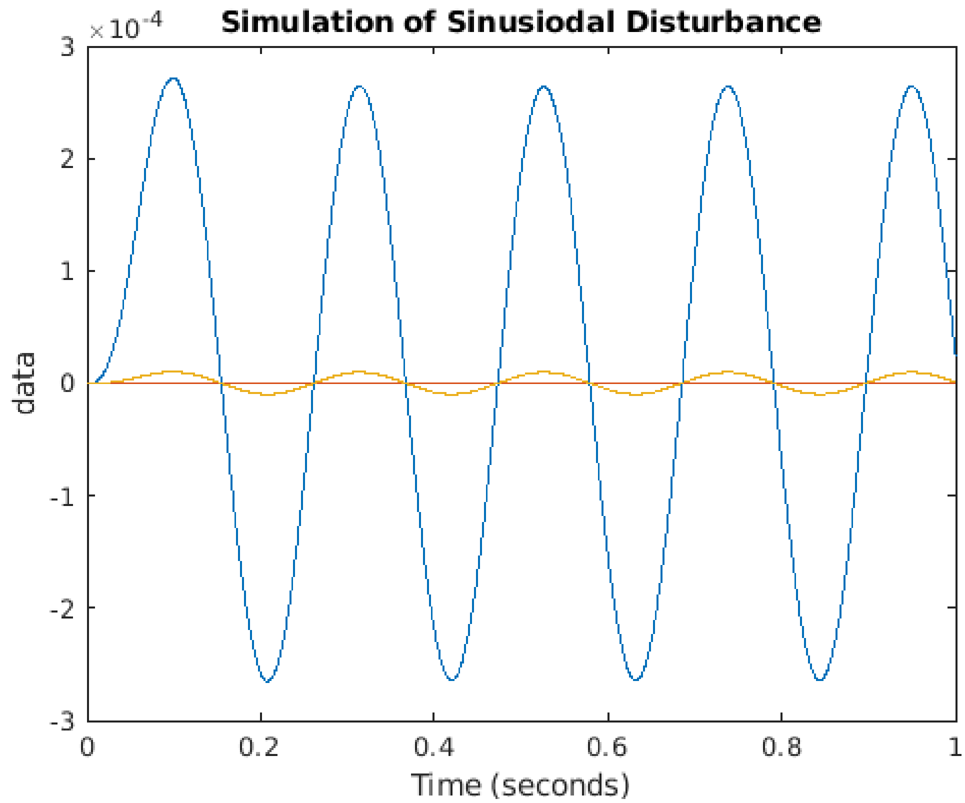
Simulink Simulation - Sinusoidal Disturbance

The bode plot for the disturbance transfer function show a band-pass filter for disturbances. Any sinusoidal disturbance in the middle frequencies will be greatly attenuated, but any extreme frequency will be rejected.

Does the system attenuate the disturbance like you expected? Yes, the output signal is greatly attenuate as the bode plot shows it should be.

Is the phase difference between the input and output like you expected? Yes, it appears to lag the input by ~90deg which is near where the bode plot has it.

```
q0_BI = e2q(e, 0*pi/180);
qstar_BI = q0_BI;
input_type = 1; % constant input
d_type = 3; % sinusoidal disturbance
freq = bandwidth(COLTF); % frequency of 3dB bandwidth (rad/s)
input_param = freq;
sim('AngleControl', 1)
figure
plot(theta_out)
title('Simulation of Sinusiodal Disturbance')
```



Attitude Control Summary

The design process began with proportional velocity control of the satellite. The important parameters to choose here are the stability margins. The result of this design becomes the inner loop of the attitude controller.

In order to reject disturbances it is necessary to employ an integrator in the attitude controller. This, however would necessarily create a system that is not stabilizable because each of the 2 integrators in the system decrease the phase by 90 degrees. As a result the phase starts at -180 degrees. To address this a zero is used to buy back phase. The zero will account for a total increase of 90 degrees. One consequence of the zero is that the system ends up with a flat response in the high frequencies to infinity. Even if it is attenuated this would require the system to use infinite power to control where any high frequency noise exists. Therefore a pole is added which rolls off the high frequency response. The zero is placed where it can buy back a useable amount of phase, but still reject constant disturbance in a timely way. The pole is placed far enough away from the zero that it doesn't cancel it out too soon and there is still a useable phase bump, but not so far that noise becomes a power drain on the system. Once these are done the gain is adjusted to achieve the desired stability margins.

Conclusions

The tradespace is very large when it comes to satellite attitude control. Before beginning any design, it is very important to fully define the critical requirements and parameters. In the case of this satellite, the moments of inertia were in place before an inner loop controller were even began.

Simulation is a critical step for determining stability. Stability is of highest importance. It is clear that an unstable system is an unuseable system. Theory and analysis may show stability where it does not exist. Only after a system has been proven by both theory and simulation may you begin to trust that it can be implemented.

Published with MATLAB® R2020a