Mini Project 3: Detumbling a Spacecraft

ME 743: Satellite Systems, Dynamics and Control  
Spring 2020

COVID-19 Era



Nathan Gunter

Charlie Nitschelm

# Abstract

In order to stop the tumbling of a spacecraft after deployment from a rocket, a PID controller was developed. The control system uses inputs of the current body rates, recorded from the spacecraft’s on-board instrumentation, to control electronic momentum wheels. The momentum wheels are used in a specific sequence in order to bring the spacecraft into a state referred to as “Safe Mode”, or when all body rates are zero. The controller is designed to bring the spacecraft to Safe Mode before it has completed its first orbit around the earth, which is around 90 minutes.

# Problem Description

A spacecraft has been deployed into a low earth orbit using a rocket. During deployment, the spacecraft began to tumble and its body rates were measured using the on-board equipment. In order to stabilize the spacecraft, the ground station needs to send a signal to enter safe mode that would cause the spacecraft to have zero body rotation. A control system is to be designed to be used to detumble the spacecraft within one orbit, or around 90 minutes. On-board the spacecraft are momentum wheels which saturate at a maximum absolute momentum of 5 N-m. The control system is to be designed to use as little battery reserve power as possible while still achieving the detumbling requirement.

# The Strategy and Solution

The spacecraft that needed to be de-stumbled was a unique body for it having the exact moment of inertias in the x and y directions while the z axis was 10 times bigger than the other two. What this means is that the xx and y body rates will naturally decay over time, meaning if that decay is slower then how long it will take to stabilize the z body rate, controllers are not needed for the x and y axis.

The system was designed in MATLAB Simulink. The system uses three functions for each axis which continuously calculate the for each axis, where the system pushes those values into a body rate, ω, which also contain the initial conditions of the satellite seen below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | X-Axis | Y-Axis | Z-Axis |
| Initial Body Rate (rad/s) | 4 | -3 | 5 |

The system body rates enter a mux that compile a vector of the moment applied to the spacecraft and its respective current body rates. Those values continue back into the system recalculating a new moment force from the inertial wheels to apply to the satellite. A PID controller was selected as it is the most widely used controller throughout the space industry and also offers an optimal controller for this problem. The x and y PID controllers were all set to zero, rendering them non-existent, to see if they could be eliminated from the control system given the innate qualities of the spacecraft inertia. The z PID controller was given random controller values for an initial guess. The first run proved that the x and y axis naturally decay very quickly compared to the z axis with a controller, also given that the best time the z axis can achieve stabilizing is 1000 seconds given the max output of the inertia wheels and the starting body rate and second moment of inertia.

A picture containing screenshot

Description automatically generated

Figure - The MATLAB Simulink model for the control system needed to stabilize the spacecraft

The z axis PID controller was run multiple times until the controller landed at the best time it could, which landed to be 1000.00 seconds. The parameters that we landed on for our z axis PID controller is below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | X-Axis | Y-Axis | Z-Axis |
| Proportional (P) | 0 | 0 | 1,000,000,000 |
| Integral (I) | 0 | 0 | 0 |
| Derivative (D) | 0 | 0 | 1 |
| Filter Coefficient | 100 | 100 | 100 |

We did not have to experiment with the x and y axis PID controllers as we knew if those were activated, it might reduce the time in the x and y axis, but it wouldn’t matter to the total time the satellite would need to completely stabilize. Adding controllers in those axis for the hell of it would only add to the total control effort that the spacecraft would need to exert.

The controller we laned on is very effective, meaning it achieves the fastest time possible to stabilize and also minimizes the amount of energy used to do so. It does this by removing the need for controllers in the x and y axis and just focus on creating a perfect z controller. This not only improves our time and energy used for the process, but also allows the engineers of the spacecraft to not put on controllers for the x and y axis making it more lightweight in general, if wanted. Of course, keep the ability for the controllers to be on board if you would like to be able to control the orientation of the satellite in real time, not just stabilize it and allow it to go into any orientation with respect to ground station.

The three graphs below show the x, y, and z body rates simulated as the controller works the intertia wheels on the spacecraft to deorbit the craft. It is obvious that the x and y axis show a naturally decaying body rate behavior, meaning a controller is not needed to achieve zero body rate. It also decays much faster naturally compared to the z axis, which requires a constant wheel on at -5 N m of total energy being exerted until it rapidly shuts off as it reaches zero body rate.

A close up of a map

Description automatically generated

Figure - The x-axis body rate with respect to time during the stabilize phase of the orbit. The spacecraft stabilizes itself naturally in 48.93 seconds.

A close up of a map

Description automatically generated

Figure - The y-axis body rate with respect to time during the stabilize phase of the orbit. The spacecraft stabilizes itself naturally in 48.82 seconds.

A close up of a map

Description automatically generated

Figure - The z-axis body rate with respect to time during the stabilize phase of the orbit. The spacecraft stabilizes in 1000.00 seconds, the theoretical best time, with tuned PIUD control to instantly identify zero body rate and shut off the wheel respectively.

The figure for the applied moment that was imparted on the spacecraft is below. It is obvious that the applied moments in the x and y axis are zero because there is no need to apply anything in that axis as it naturally decays over time. The applied moment in the z axis is consistently -5 N m until it instantly shuts off at 1000.00 second as it achieves zero body rate in that axis.

A screenshot of a cell phone

Description automatically generated

Figure - The moment imparted on the spacecraft by the inertia wheels during its stabilization phase in each axis.

Another important parameter to show is the total control effort of the spacecraft and its inertia wheels during this maneuver. To do so, the Simulink model in Figure 1 calculates the total control effort during the stabilization phase by summing the moments and integrating them with respect to time. This outputs a curve on each axis that continuously adds to the final value, which represents the total control effort that was needed to do the maneuver.

A screenshot of a cell phone

Description automatically generated

Figure - The control effort the spacecraft needs to impart on itself by the inertia wheels in each axis during the stabilization phase.

It is easily shown that the x and y axis do not contribute to the total control effort as the controllers are non-existent with 0’s for all its parameters. The z axis PID is the only contributor as it keeps its wheel on for the full 1000.00 second at -5.0 N m until it reaches a body rate of 0. It achieves the theoretical best control effort of 5,000 N s shown above and turns off right after to no longer contribute to any control effort.

# Conclusion and Final Recommendation

# Appendix

clear all

close all

ix =100;

iy = 100;

iz = 1000;

sim('model')

tol=.01;

for i = 2:length(u)

if mean(abs(u(i,4))+abs(u(i+1,4))+abs(u(i+2,4))+abs(u(i+3,4))+abs(u(i+4,4))) < tol

u1\_end = i;

break

end

end

for i = 2:length(u)

if mean(abs(u(i,5))+abs(u(i+1,5))+abs(u(i+2,5))+abs(u(i+3,5))+abs(u(i+4,5))) < tol

u2\_end = i;

break

end

end

tol=.0001;

for i = 2:length(u)

if mean(abs(u(i,6))+abs(u(i+1,6))+abs(u(i+2,6))+abs(u(i+3,6))+abs(u(i+4,6))) < tol

u3\_end = i;

break

end

end

figure(1)

plot(tout,u(:,4))

hold on

plot(u1\_end/100,0,'o')

title('X-Axis Body Rates')

xlabel('Time (s)')

ylabel('Body Rates (rad/s)')

xlim([0,120])

text(50,1,'Stabilizes at t=48.93 seconds')

figure(2)

plot(tout,u(:,5))

hold on

plot(u2\_end/100,0,'o')

title('Y-Axis Body Rates')

xlabel('Time (s)')

ylabel('Body Rates (rad/s)')

xlim([0,120])

text(50,1,'Stabilizes at t=48.82 seconds')

figure(3)

plot(tout,u(:,6))

hold on

plot(u3\_end/100,0,'o')

title('Z-Axis Body Rates')

xlabel('Time (s)')

ylabel('Body Rates (rad/s)')

xlim([0,1200])

ylim([-.5,5])

text(400,0,'Stabilizes at t=1000.00 seconds')

figure(4)

plot(tout,controleffort(:,1))

hold on

plot(tout,controleffort(:,2))

plot(tout,controleffort(:,3))

title('Control Effort of PID Controllers')

xlabel('Time (s)')

ylabel('Control Effort (J s)')

xlim([0,1200])

ylim([0,5500])

legend('X-Axis PID','Y-Axis PID','Z-Axis PID','location','northwest')

figure(5)

plot(tout,u(:,1))

hold on

plot(tout,u(:,2))

plot(tout,u(:,3))

title('Moment Added to Satellite Axis')

xlabel('Time (s)')

ylabel('Moment (N m)')

xlim([0,995])

ylim([-5.5,.5])

legend('X-Axis','Y-Axis','Z-Axis')

## Code and Simulink Model

 