Benjamin Okoronkwo (bco26), Ethan Kim (ejk252), Olivia Lee (ol67), Aloyse Maille (aam299)

FINAL GROUPEWORK PROPOSAL

Title: North Star (not the dining hall)

Topics of Interest: Satellite Pointing

- Developing a state space model of the satellite
- Deriving performance requirements for pointing
- Learning about and modeling disturbances
- Developing a MATLAB script or live script to simulate the system

Abstract: The system under study is a satellite located in a geostationary orbit pointing at a static location on the Earth surface. It was selected as it is a common interest to all of our group members to understand better how pointing or attitude control in a satellite operates. Those systems can have a lot of applications, from commercial purposes to research and defense, and even if we don't actively work with such systems in the future, we still use them directly or indirectly in our daily lives. They are used for radio and television communications but can also now be used for messaging on our phones and other fun things.

Students/Roles

Student	Tasks/ Roles	Portfolio link
Aloyse	I worked on the state space model and block diagram, as well as the disturbance model of the satellite. Additionally, I participated in working on the design performance parameters and the MATLAB modelization of the system.	https://cornell-mae-ug.github.io/spring-2025-portfolio-aloysemaille/
Ethan	I worked on the state space model and also contributed to the Matlab script. From this project, I gained a deeper understanding and appreciation for space systems.	https://cornell-mae-ug.github.io/fa25-portfolio-ejk252/
Olivia	For this project, I helped create the state space model, block diagram and MATLAB code. I learned how a satellite points at a specific location on Earth and how disturbances affect the model.	https://cornell-mae-ug.github.io/fa25-portfolio-ol67/
Benjamin	I helped in the creation and calculation of the state space modeling, I worked on defining performance requirements for the satellite, and I aided the creation of the MATLAB simulation and analysis.	https://cornell-mae-ug.github.io/fa25-portfolio-benjioko/

Skills to explore and demonstrate:

1. Equations $J\theta(doubledot) = T$
2. ODEs
3. State space model
4. Block diagram
5. Active control

Benjamin Okoronkwo (bco26), Ethan Kim (ejk252), Olivia Lee (ol67), Aloyse Maille (aam299)

1) System Definition

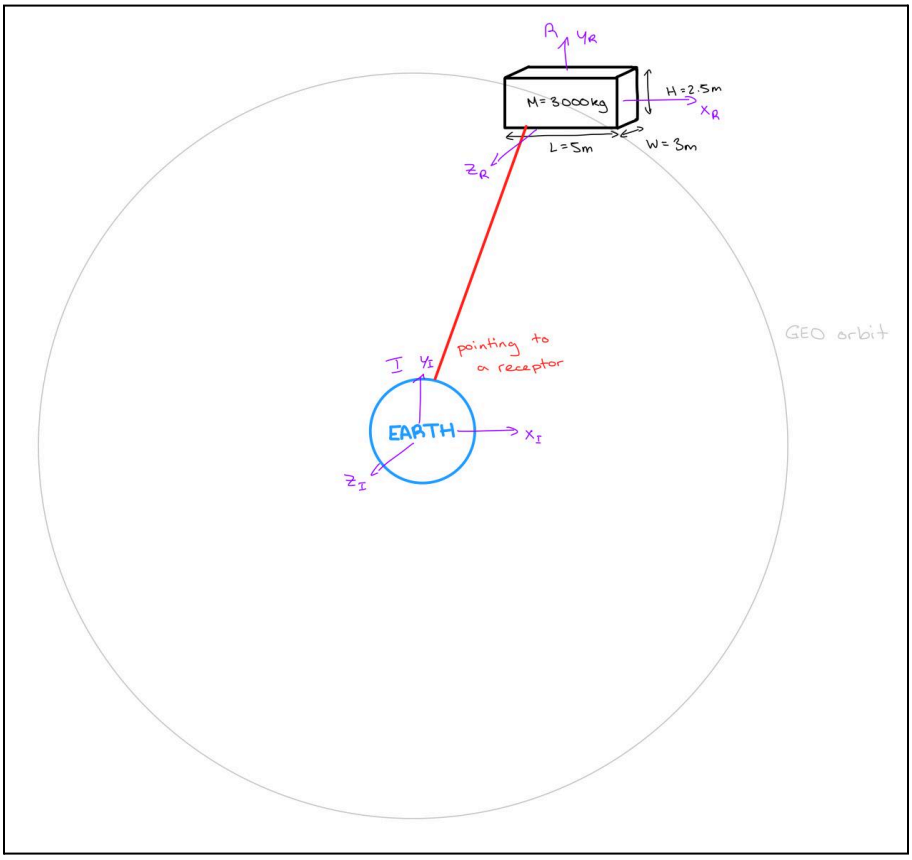

Fig.1 : Schematic of the system and its orbit

The satellite under study can be approximated by a rectangular volume with the following properties:

Mass = M = 3000 kg
Length = L = 5 m
Width = W = 3 m
Height = H = 2.5 m

It operates in a geostationary circular (GEO) orbit 35,786km above the Earth surface, or 42,164 km from the Earth's center. And the angular velocity of a fixed body-centered frame (R) relative to an inertial frame (I) is:

ω^{IR} = [0 $-\omega_0$ 0]

Its inertia components can be computed as follow:

$I_1 = (1/12)*M*(W^2+H^2)$
$I_2 = (1/12)*M*(L^2+H^2)$

$I_3 = (1/12)*M*(L^2+W^2)$

The system is also studied under the assumption it is made of reflective material.

2) State-Space Model and Block Diagram without disturbances

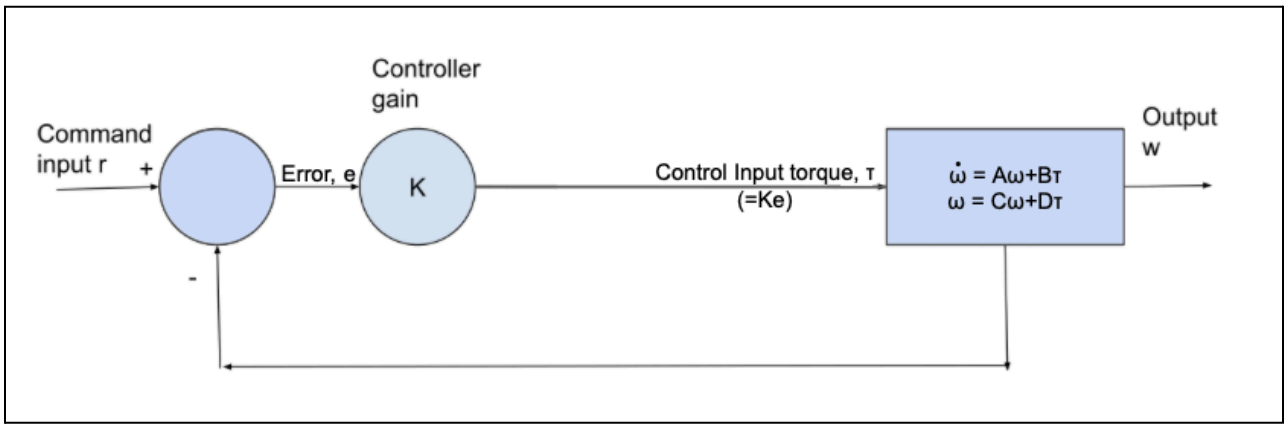

Fig.2 : Block Diagram for Satellite angular velocity control without disturbances

The Equations of Motion of the system come from Newton's second law of motion analogy for rotational motion: torque τ = dh/dt equation with the angular momentum h =I*ω with I being the moment of inertia matrix of the system and ω the angular velocity vector of the system.

$$\tau_1 = I_1 \dot{\omega}_1 + (I_3 - I_2)\,\omega_3\,\omega_2$$
$$\tau_2 = I_2 \dot{\omega}_2 + (I_1 - I_3)\,\omega_1\,\omega_3$$
$$\tau_3 = I_3 \dot{\omega}_3 + (I_2 - I_1)\,\omega_2\,\omega_1$$

Which gives us:

$$\dot{\omega}_1 = \frac{\tau_1}{I_1} - \frac{(I_3 - I_2)\,\omega_3\,\omega_2}{I_1}$$
$$\dot{\omega}_2 = \frac{\tau_2}{I_2} - \frac{(I_1 - I_3)\,\omega_1\,\omega_3}{I_2}$$
$$\dot{\omega}_3 = \frac{\tau_3}{I_3} - \frac{(I_2 - I_1)\,\omega_2\,\omega_1}{I_3}$$

with $\boldsymbol{\omega} = [\omega_1\ \omega_2\ \omega_3] = \omega^{IR} + \partial\omega$ *(under the assumption that the system deals with small angles so that only small variations of angular velocity are needed.)*[1]

Plugging in the small variation and constant angular velocity components and with $\omega^{IR} = [\,0\ -\omega_0\ 0]$, the following state-space model can be derived:

$$\begin{bmatrix}\dot{\omega_1}\\ \dot{\omega_2}\\ \dot{\omega_3}\end{bmatrix} = \begin{bmatrix}0 & 0 & \frac{(I_3-I_2)\omega_0}{I_1}\\ 0 & 0 & 0\\ \frac{(I_2-I_1)\omega_0}{I_3} & 0 & 0\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} + \begin{bmatrix}\frac{1}{I_1} & 0 & 0\\ 0 & \frac{1}{I_2} & 0\\ 0 & 0 & \frac{1}{I_3}\end{bmatrix}\begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\end{bmatrix}$$

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} + \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\end{bmatrix}$$

Furthermore, the angular velocity outputs can be integrated through a MATLAB function to obtain the angular position/orientation of the spacecraft used for pointing. Giving us the block diagram below. This could be pushed even further and the input itself would be an angular position.

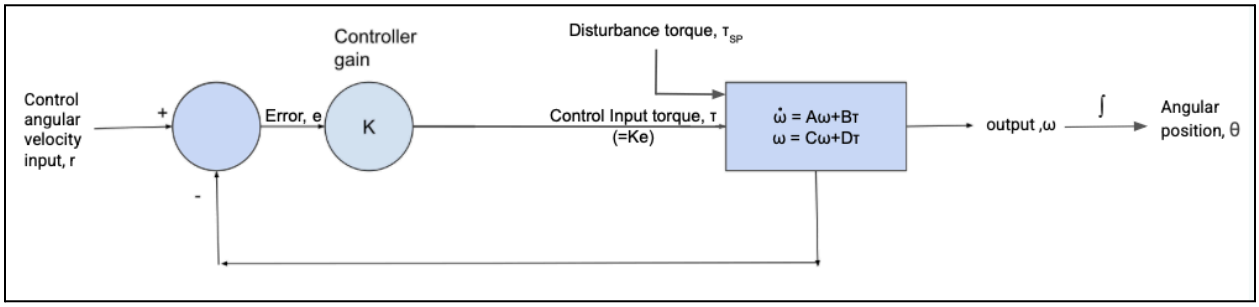

Fig.3 : Block Diagram for Satellite angular velocity control without disturbances and with angular position output

3) State-Space Model and Block Diagram with disturbances

Three of the main disturbance sources that can apply to a spacecraft are gravity, atmospheric drag and solar pressure. However, for altitudes greater than about 800 km, gravity and atmospheric drag have little enough influence that they can be neglected. Therefore the only disturbance relevant to our model is solar pressure.

The maximum Solar Pressure is defined by $P_{max} = S_l/c * 2 = 9.07e(-6)\ N/m^2$

[1] Vectors are denoted in bold, in this case ω^{IR} and $\partial\omega$ are vectors.

,with S_I the solar intensity which is about 1361 W/m^2 at Earth's distance
and c the speed of light, c = 300e6 m/s.
The 2 factor comes from the fact the satellite is made out of a reflective material.

It can be modeled as a sinusoidal function of time as the satellite goes around the Earth and is more or less in the range of the Sun, where P ≈ 0 when the satellite is fully in the Earth's shadow and P_{max} happens at the opposite point.

Therefore we can model it by: $P(t) = P_{max} * \sin^2(\pi t/T)$,with T the orbital period.

That results in a force equal to the pressure over the exposed surface area of the satellite.
$F_P(t) = P(t)/A_{exposed}$

It then gives us a disturbance torque due to the solar pressure. τ_{SP}= d x F_P with d the distance between the area-weighted centroid of the exposed area and the center of mass of the satellite.

The state-space model then becomes:

$$\begin{bmatrix}\dot{\omega_1}\\ \dot{\omega_2}\\ \dot{\omega_3}\end{bmatrix} = \begin{bmatrix}0 & 0 & \frac{(I_3-I_2)\omega_0}{I_1}\\ 0 & 0 & 0\\ \frac{(I_2-I_1)\omega_0}{I_3} & 0 & 0\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} + \begin{bmatrix}\frac{1}{I_1} & 0 & 0\\ 0 & \frac{1}{I_2} & 0\\ 0 & 0 & \frac{1}{I_3}\end{bmatrix}\begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\end{bmatrix} + \begin{bmatrix}\frac{\tau_{sp1}}{I_1}\\ \frac{\tau_{sp2}}{I_2}\\ \frac{\tau_{sp3}}{I_3}\end{bmatrix}$$

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} + \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\end{bmatrix}$$

And we have the following block diagram:

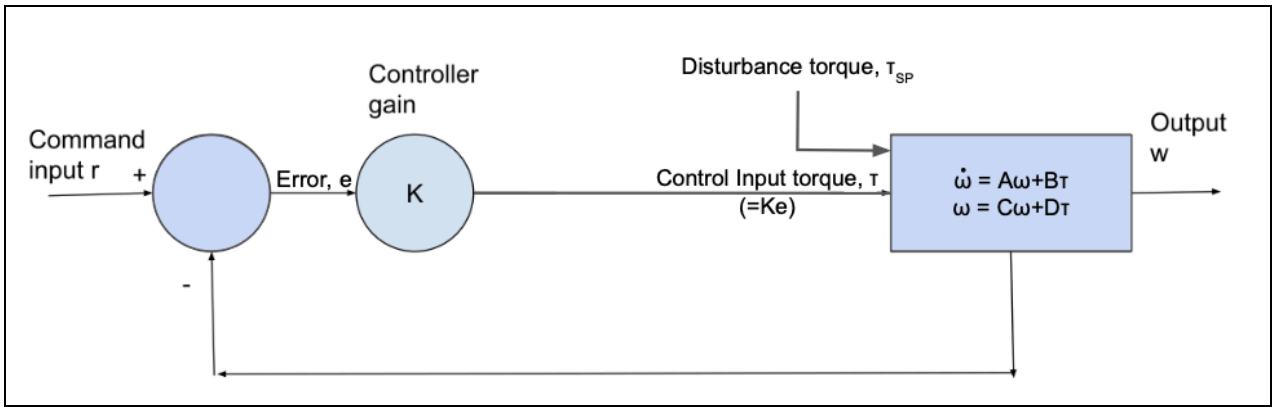

Fig.4 : Block Diagram for Satellite angular velocity control with disturbances

4) Performance Requirements

Several performance requirements have been defined in order to ensure proper functioning and optimal performance of the satellite attitude determination and control system. These parameters are based on current satellites performance standards and mission requirements for GEO orbit satellites.

The pointing accuracy error is e_{ss} <= 0.0175 rad or 1°. This parameter is in the range of typical satellites' pointing accuracy error [2]. This pointing accuracy provides sufficient precision for reliable signal transmission while remaining achievable by standard hardware.

The target or reference angular rate needs to be low enough to avoid tumbling, or uncontrolled spinning and rotating of the satellite [3]. For this satellite purposes, an angular velocity of ω_{ss} <= 0.01 rad/s or 0.095493 rpm is deemed to be sufficient for operations and does not impact the stability of the spacecraft.

Similar satellites have a t_{ss} between 90 s and 180 s. Therefore, the satellite under study has t_{ss} <= 180s. Choosing the higher bound for the satellite allows it to be a little more forgiving.

The maximum overshoot of the satellite is MO <= 0.0262 rad or 1.5°. This decision was based on the potential need for bigger angles pointing adjustments. Even though the satellite operates under the assumption that it only deals with small angles pointing adjustments, in the case where it would need a bigger adjustment, using a 15% maximum overshoot or anything significantly larger than a 1.5° maximum overshoot would give a completely off pointing position.

5) Matlab Model

This first figure models the pointing angle and angular velocity of the response with and without disturbance compared to the reference pointing angle. As it can be observed on the graphs, a disturbance of 0.1Nm significantly impacts the responses. A disturbance of this magnitude is a

bigger order of magnitude than the actual disturbance, but amplifying it allows the model to show better how it is impacted by a disturbance.

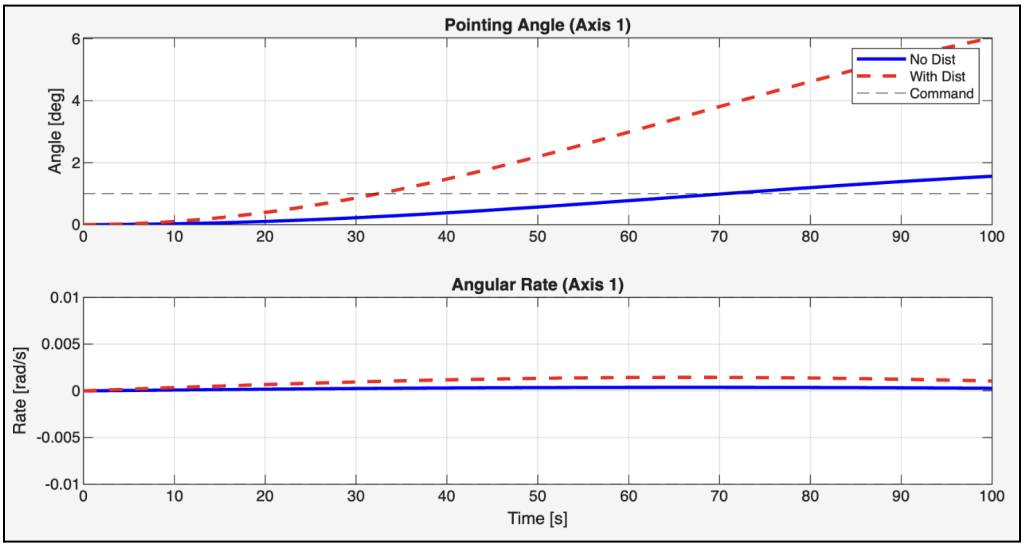

Fig.5 : Time response comparison with and without disturbance

The second figure generated by the code is a comparison of the different gain values for the satellites PD controller. It shows that a higher gain makes the response faster but with a higher overshoot.

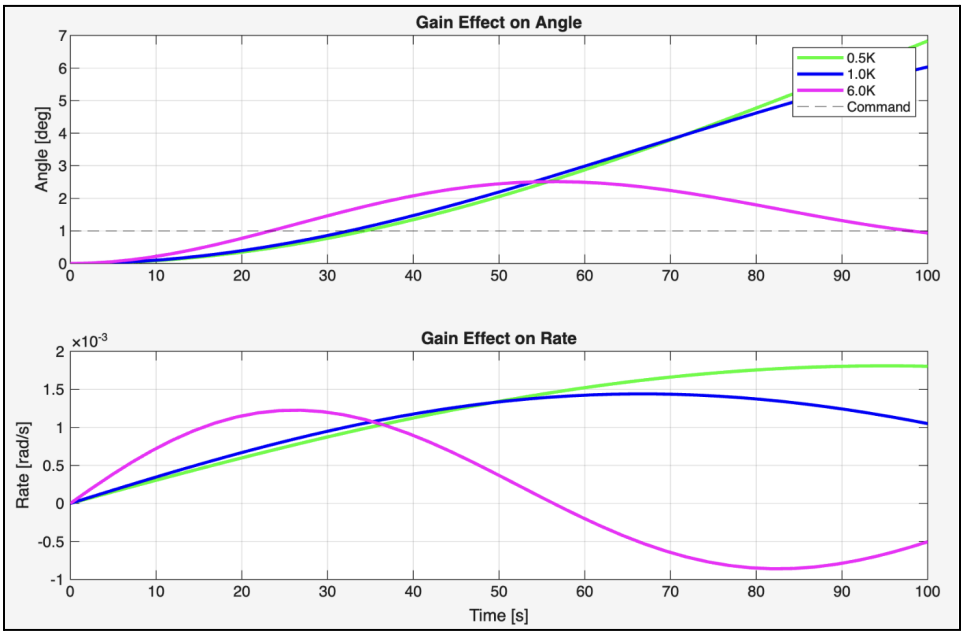

Fig.6 : Controller Gain Effects

The third figure generated and attached below shows the overall bode plot of the system. It shows how the position of the satellite relative to the Earth changes with increasing angular velocity for two systems: open-loop and closed-loop.

Benjamin Okoronkwo (bco26), Ethan Kim (ejk252), Olivia Lee (ol67), Aloyse Maille (aam299)

For the open loop system (no controller), at low torque frequency (angular velocity) the satellite drifts further away from Earth without changing its rotational orientation. As the frequency increases, the satellite is able to gain more control and stability.

For the closed loop system (PD controls), at low frequency the satellite stays stagnant and doesn't change its position due to the K_p and K_d rejecting external disturbances. It's also good to note that when the resonant frequency is met, the phase plot transitions from 0 to 180deg which is a mathematical shift and doesn't affect the rotational orientation of the satellite. At higher frequencies, it's able to gain more control of itself and orient itself toward its target.

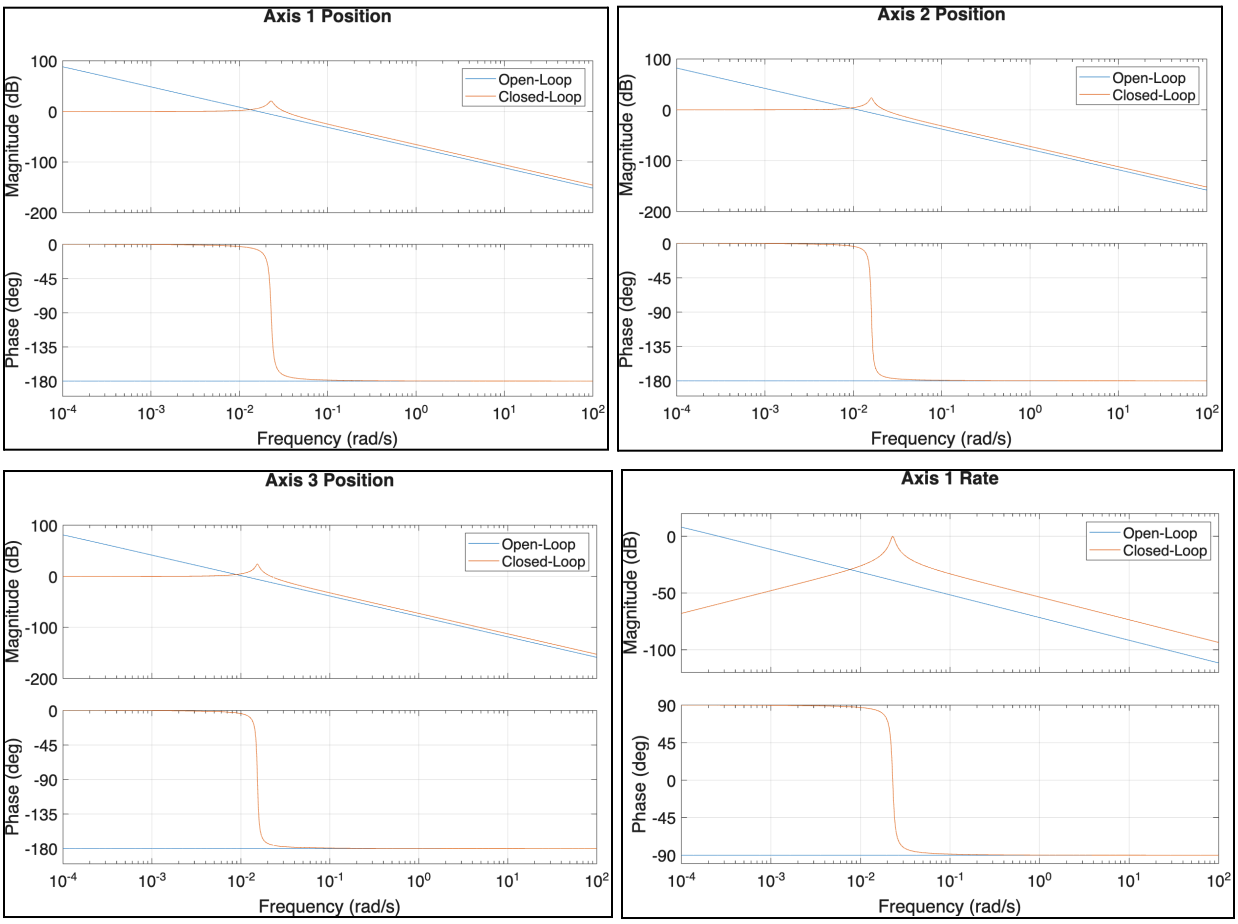

Fig.7 : Bode Plots for the angular position and rate along the different axis

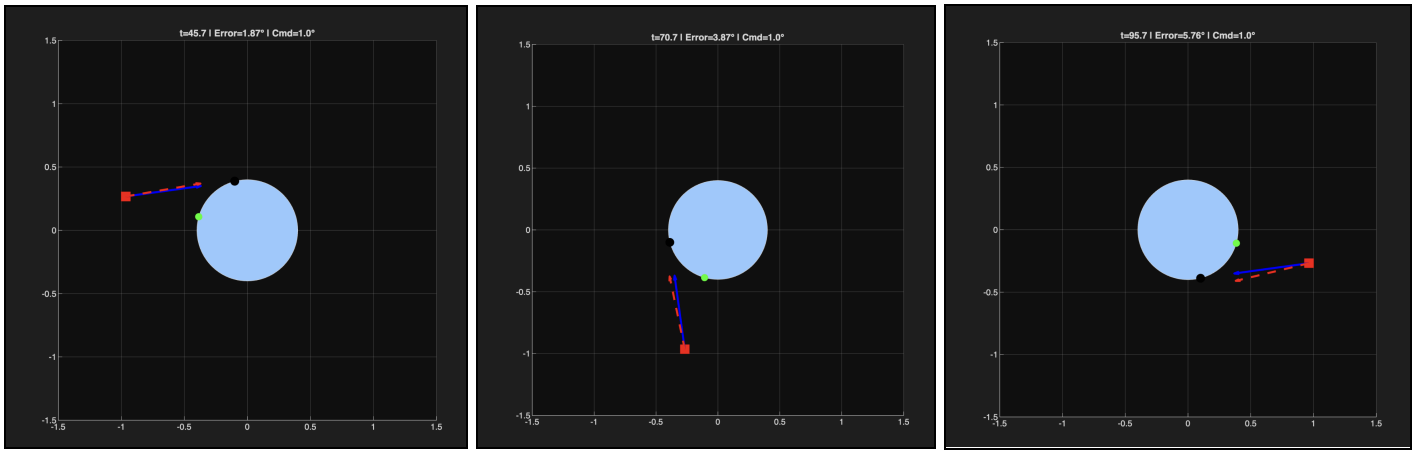

Fig.8 : Snapshots of the simulation of the satellite on its orbit (in red actual pointing - with error; in blue reference pointing input)

We created a simulation of the satellite on its orbit to visualize it pointing to a specific point on the Earth. It showed the actual pointing of the satellite in red and the reference pointing in blue. It allowed us to see the error while it was going around the orbit.

```
Satellite: 3000 kg, I=[3812, 7812, 8500] kg*m^2
Orbit: 42164 km, omega=0.000073 rad/s

=== PERFORMANCE (AXIS 1) ===

No Disturbance:
  Steady-state err: 0.0098 rad (0.56 deg) ✓
  Overshoot: 0.0098 rad (0.56 deg) ✓
  Settling time: 70.0 s ✓
  Final rate: 0.00027 rad/s ✓

Done!
```

Fig.9 : MATLAB code output

We created a satellite with a mass of 3000 kg, length of 5 m, width of 3 m, and height of 2.5 m. Based on our MATLAB code we obtained a steady state error of 0.0098 rad which is less than our performance requirements of 0.0175 rad, the overshoot was 0.0098 rad which is also less than our performance requirements of 0.0262 rad, the settling time is 70s which is less than our performance requirement of under 180s, and an angular velocity of 0.00027 rad/s which is also lower than our performance requirement of 0.01 rad/s.

6) Conclusion

In this project we created a state space model by using its inertia components and created a block diagram for satellite angular velocity control without disturbances. We also integrated the angular velocity outputs through MATLAB to obtain the angular position output. Then we modeled it with disturbances, solar pressure. We modeled it as a sinusoidal function of time as the satellite goes around the Earth. This gave us a disturbance torque which we modeled with a block diagram. We decided on performance requirements based on actual satellites that we researched. We then

created a MATLAB code that gave us graphs for time response, controller gain effects, bode plots for open and close loop, a simulation of the satellite on its orbit, and MATLAB output of the performance.

7) References

[2] “How to Pick up a Suitable CubeSat ADCS | Tensor Tech.” n.d. Accessed December 6, 2025. https://tensortech.co/updates/detail/How-to-pick-up-a-suitable-CubeSat-ADCS.

[3] Franquiz, Francisco J., "Attitude Determination & Control System Design and Implementation for a 6U CubeSat Proximity Operations Mission" (2015). Dissertations and Theses. 266. https://commons.erau.edu/edt/266

8) Appendix

Matlab Code[2]:

```
function geo_satellite_control
clc; clear; close all;
%% SATELLITE PARAMETERS
M = 3000; L = 5; W = 3; H = 2.5;  % Mass [kg], Dimensions [m]
% Moments of inertia
I1 = (1/12)*M*(W^2 + H^2);
I2 = (1/12)*M*(L^2 + H^2);
I3 = (1/12)*M*(L^2 + W^2);
% GEO orbit
R_orbit = 42164e3;  % m
mu_earth = 3.986e14;  % m^3/s^2
omega0 = sqrt(mu_earth / R_orbit^3);  % Orbital rate [rad/s]
fprintf('Satellite: %.0f kg, I=[%.0f, %.0f, %.0f] kg*m^2\n', M, I1, I2, I3);
fprintf('Orbit: %.0f km, omega=%.6f rad/s\n\n', R_orbit/1000, omega0);
%% STATE-SPACE MODEL
A_omega = [0, 0, (I3-I2)*omega0/I1;
           0, 0, 0;
           (I2-I1)*omega0/I3, 0, 0];
B_omega = diag([1/I1, 1/I2, 1/I3]);
A = [zeros(3), eye(3); zeros(3), A_omega];
B = [zeros(3); B_omega];
E = B;  % Disturbance input same as control
%% CONTROLLER GAINS
Kp = diag([2.0, 2.0, 2.0]);
Kd = diag([8.0, 8.0, 8.0]);
K = [Kp, Kd];
%% COMMAND & DISTURBANCE
theta_cmd_deg = 1;  % Command angle [deg]
theta_cmd = deg2rad(theta_cmd_deg)*[1;0;0];
x_ref = [theta_cmd; zeros(3,1)];
tau_dist = 0.1*[1;0;0];  % Disturbance torque [Nm] - increased to be visible
%% SIMULATIONS
t_end = 100;
tspan = [0 t_end];
x0 = zeros(6,1);
% Without disturbance
odefun_no = @(t,x) dynamics(t,x,A,B,E,K,x_ref,false,tau_dist);
[t_no, x_no] = ode45(odefun_no, tspan, x0);
% With disturbance
```

[2] *“This code was partially generated and modified by the AI Claude during the week of December 1st.”*

```
odefun_yes = @(t,x) dynamics(t,x,A,B,E,K,x_ref,true,tau_dist);
[t_yes, x_yes] = ode45(odefun_yes, tspan, x0);
%% PERFORMANCE ANALYSIS
fprintf('=== PERFORMANCE (AXIS 1) ===\n');
analyze_perf('No Disturbance', t_no, x_no(:,1), x_no(:,4), theta_cmd(1));
%% PLOTS
% Time response comparison
figure('Position', [100 100 1000 500]);
subplot(2,1,1);
plot(t_no, rad2deg(x_no(:,1)), 'b-', 'LineWidth', 2); hold on;
plot(t_yes, rad2deg(x_yes(:,1)), 'r--', 'LineWidth', 2);
yline(theta_cmd_deg, 'k--');
ylabel('Angle [deg]'); title('Pointing Angle (Axis 1)'); grid on;
legend('No Dist', 'With Dist', 'Command');
subplot(2,1,2);
plot(t_no, x_no(:,4), 'b-', 'LineWidth', 2); hold on;
plot(t_yes, x_yes(:,4), 'r--', 'LineWidth', 2);
yline([0.01 -0.01], 'k--');
xlabel('Time [s]'); ylabel('Rate [rad/s]'); title('Angular Rate (Axis 1)'); grid on;
% Gain sensitivity
figure('Position', [150 150 1000 600]);
gains = [0.5, 1.0, 6.0];
colors = {'g', 'b', 'm'};
for i = 1:3
   K_test = gains(i)*K;
   odefun = @(t,x) dynamics(t,x,A,B,E,K_test,x_ref,true,tau_dist);
   [t_test, x_test] = ode45(odefun, tspan, x0);

   subplot(2,1,1);
   plot(t_test, rad2deg(x_test(:,1)), colors{i}, 'LineWidth', 2); hold on;

   subplot(2,1,2);
   plot(t_test, x_test(:,4), colors{i}, 'LineWidth', 2); hold on;
end
subplot(2,1,1);
yline(theta_cmd_deg, 'k--');
ylabel('Angle [deg]'); title('Gain Effect on Angle'); grid on;
legend('0.5K', '1.0K', '6.0K', 'Command');
subplot(2,1,2);
xlabel('Time [s]'); ylabel('Rate [rad/s]'); title('Gain Effect on Rate'); grid on;
% Bode plots
sys_open = ss(A, B, eye(6), zeros(6,3));
A_cl = A - B*K;
sys_closed = ss(A_cl, B*K, eye(6), zeros(6,6));
figure('Position', [250 100 1200 800]);
for i = 1:3
   subplot(2,2,i);
   if i <= 2
       bode(sys_open(i,i), sys_closed(i,i), {1e-4, 1e2});
   else
       bode(sys_open(3,3), sys_closed(3,3), {1e-4, 1e2});
   end
   grid on;
   title(sprintf('Axis %d Position', i));
   legend('Open-Loop', 'Closed-Loop');
end
subplot(2,2,4);
bode(sys_open(4,1), sys_closed(4,4), {1e-4, 1e2});
grid on; title('Axis 1 Rate');
legend('Open-Loop', 'Closed-Loop');
%% HELPER FUNCTIONS
function dx = dynamics(~, x, A, B, E, K, x_ref, dist_on, tau_dist)
tau = -K*x + K*x_ref;
d = dist_on * tau_dist;
dx = A*x + B*tau + E*d;
end
```

Benjamin Okoronkwo (bco26), Ethan Kim (ejk252), Olivia Lee (ol67), Aloyse Maille (aam299)

```
function analyze_perf(name, t, theta, omega, cmd)
fprintf('\n%s:\n', name);
e_ss = abs(theta(end) - cmd);
overshoot = max(theta) - cmd;
tol = 0.02*abs(cmd);
idx = find(abs(theta-cmd) <= tol, 1);
t_settle = t(idx);
fprintf('  Steady-state err: %.4f rad (%.2f deg) %s\n', ...
   e_ss, rad2deg(e_ss), pass_fail(e_ss, 0.0175));
fprintf('  Overshoot: %.4f rad (%.2f deg) %s\n', ...
   overshoot, rad2deg(overshoot), pass_fail(overshoot, 0.0262));
fprintf('  Settling time: %.1f s %s\n', t_settle, pass_fail(t_settle, 180));
fprintf('  Final rate: %.5f rad/s %s\n', abs(omega(end)), pass_fail(abs(omega(end)), 0.01));
end
function str = pass_fail(val, limit)
if val <= limit, str = '✓'; else, str = '✗'; end
end
function animate_orbit(t, theta, cmd)
figure('Position', [200 200 800 800]);
R_e = 0.4; R_s = 1.0;
target_ang = -pi/3;
sat_ang = 0;
for k = 1:10:length(t)
   clf; hold on; axis equal;

   ang = 2*pi*t(k)/t(end);

   % Earth
   theta_e = linspace(0, 2*pi, 100);
   fill(R_e*cos(theta_e+ang), R_e*sin(theta_e+ang), [0.6 0.8 1]);
   plot(R_e*cos(ang), R_e*sin(ang), 'go', 'MarkerSize', 8, 'MarkerFaceColor', 'g');

   % Target & Satellite
   target = R_e*[cos(target_ang+ang); sin(target_ang+ang)];
   sat = R_s*[cos(sat_ang+ang); sin(sat_ang+ang)];
   plot(target(1), target(2), 'ko', 'MarkerSize', 10, 'MarkerFaceColor', 'k');
   plot(sat(1), sat(2), 'rs', 'MarkerSize', 14, 'MarkerFaceColor', 'r');

   % Pointing vectors
   cmd_dir = (target-sat)/norm(target-sat);
   R_err = [cos(theta(k)), -sin(theta(k)); sin(theta(k)), cos(theta(k))];
   act_dir = R_err*cmd_dir;

   quiver(sat(1), sat(2), cmd_dir(1)*0.6, cmd_dir(2)*0.6, 0, 'b', 'LineWidth', 2.5);
   quiver(sat(1), sat(2), act_dir(1)*0.6, act_dir(2)*0.6, 0, 'r--', 'LineWidth', 2.5);

   xlim([-1.5 1.5]); ylim([-1.5 1.5]);
   title(sprintf('t=%.1f | Error=%.2f° | Cmd=%.1f°', t(k), rad2deg(theta(k)), rad2deg(cmd)));
   grid on; drawnow;
end
end
```