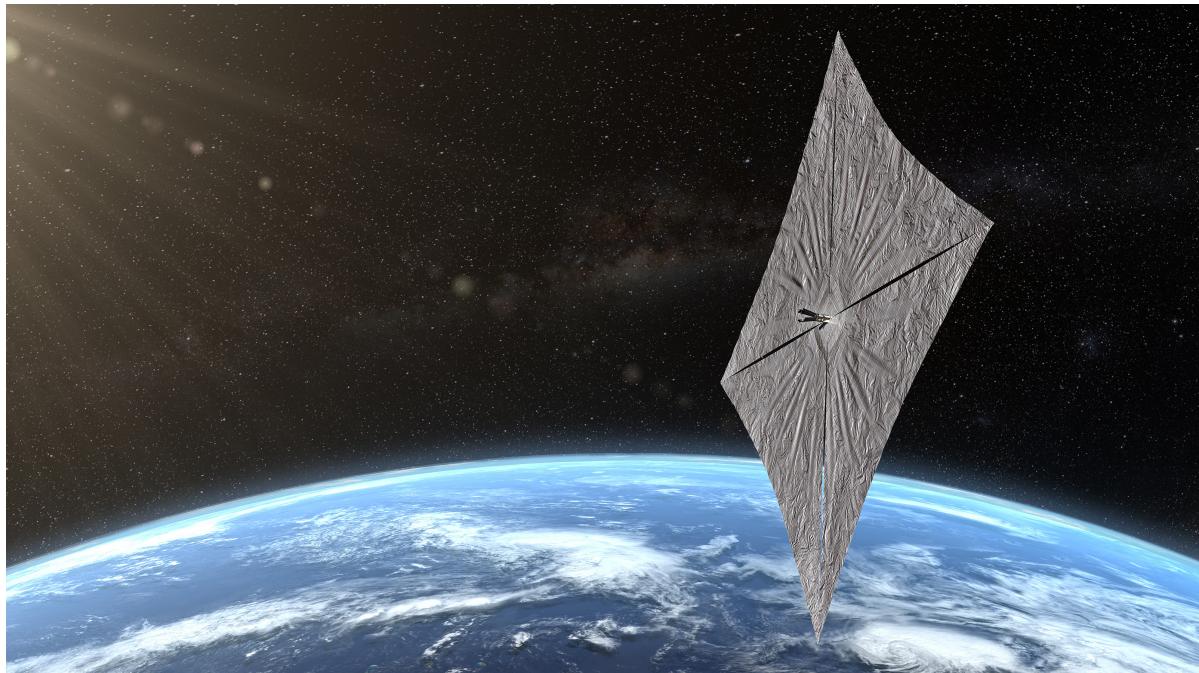


SATELLITE DYNAMICS AND ATTITUDE CONTROL

Max Newport, Valerie Pietrasz

02 June 2021



REVISION HISTORY

VERSION	REVISION NOTES
PS1	<ul style="list-style-type: none">- Created document- Added PS1 material: Mission specifications, satellite selection. Structure and mass distribution. Inertia matrix, body axes. Orbit.
PS2	<ul style="list-style-type: none">- Added PS2 material: Euler propagation, validation, axisymmetry.- Modified inertia to include asymmetric components (in PS1 and Appendix).
PS3	<ul style="list-style-type: none">- Added PS3 material: attitude propagation, integration.- Stability and equilibrium tests.- Added comparison of Euler 312 and quaternion results (Appendix).
PS4	<ul style="list-style-type: none">- Added PS4 material: dual spin, gravity gradient.
PS5	<ul style="list-style-type: none">- Renumbered section to align with PS numbers.- Modified formatting to constrain figures to their relevant sections.- Added PS5 material: magnetic field, SRP, and atmospheric drag perturbations. Attitude control errors.- Added gravity gradient stability analysis (in PS4).
PS6	<ul style="list-style-type: none">- Added PS6 material: Attitude determination, estimation error. Sensor measurements, sensor noise.
PS7	<ul style="list-style-type: none">- Added PS7 material: Extended Kalman Filter, sensor error representation.- Added estimated attitude control error (in PS6), further fleshed out comparison of attitude determination methods (Appendix).
PS8	<ul style="list-style-type: none">- Added PS8 material: Actuators, control laws.- Moved EKF discussion from appendix to PS6.

Table 1: Summary of project revisions.

TABLE OF CONTENTS

INTRODUCTION	5
1 PROBLEM SET 1	5
1.1 MISSION ADCS CHARACTERISTICS	5
1.2 SIMILAR MISSIONS	6
1.3 MECHANICAL LAYOUT	7
1.4 INERTIA PROPERTIES	9
1.5 OUTER SURFACE DISCRETIZATION	9
1.6 ORBITAL ELEMENTS & PROPAGATION	9
2 PROBLEM SET 2	12
2.1 PRINCIPAL AXES	12
2.2 NO-TORQUE PROPAGATION	13
2.3 AXIAL SYMMETRY	15
3 PROBLEM SET 3	17
3.1 KINEMATIC INTEGRATION	17
3.2 ATTITUDE PROPAGATION	18
3.3 EQUILIBRIUM TESTS	20
3.4 STABILITY TESTS	22
4 PROBLEM SET 4	25
4.1 DUAL SPIN	25
4.2 GRAVITY GRADIENT TORQUE	27
5 PROBLEM SET 5	30
5.1 PERTURBATIONS	30
5.2 ATTITUDE TARGETING & CONTROL ERROR	30
6 PROBLEM SET 6	33
6.1 ATTITUDE DETERMINATION FROM SENSOR DATA	33
6.2 SENSOR NOISE	34
6.3 SMALL SENSOR ERRORS	34
6.4 ESTIMATING ATTITUDE CONTROL ERROR	37
7 PROBLEM SET 7	37
7.1 UPDATED SENSOR MODELS	37
7.2 EXTENDED KALMAN FILTER IMPLEMENTATION	38
7.3 EKF WITH SENSOR BIAS	38
7.4 EKF WITHOUT SENSOR BIAS	39
8 PROBLEM SET 8	43
8.1 ATTITUDE CONTROL SYSTEM SIZING & SELECTION	43
8.2 ACTUATOR UNIT TESTING	44
8.3 STATE MACHINE	44
8.3.1 DETUMBLE MODE	45
8.3.2 STABILIZE MODE	46

8.3.3	SLEW MODE	48
8.3.4	CONTROL LAW IMPLEMENTATION	48
8.4	SATELLITE CONTROL TESTS	49
8.4.1	<i>q</i> -METHOD, NO SENSOR NOISE	49
8.4.2	<i>q</i> -METHOD, SENSOR NOISE & PERTURBATIONS	54
8.4.3	EKF, SENSOR NOISE, & PERTURBATIONS	58
9	FUTURE WORK	62
10	REFERENCES	62
Appendices		63
A	Mass Distribution Analysis	63
A.1	Solar Sail	63
A.2	Sail Booms	63
A.3	Forward Body	63
A.4	Solar Panels	63
A.5	Rear Body	63
B	LS2 System Drawings	65
C	VBA Solidworks Macro for Exporting Surface Normals	68
D	Verification of Inertia Computations	71
E	Additional Orbital Propagator Notes	73
F	Quaternion and 312 Euler Kinematic Comparison	74
G	Dual Spin Equilibrium & Stability Analysis	77
H	Gravity Gradient Stability	82
I	Attitude Estimation Comparisons	86
J	Extended Kalman Filter Derivations	89
K	Extended Kalman Filter Result Analysis	93

INTRODUCTION

We have chosen to study the LightSail 2 mission (LS2), a solar sailing proof-of-concept mission developed by The Planetary Society. Solar sails use solar radiation pressure (SRP) to propel the spacecraft, and JAXA's IKAROS mission in 2021 was the first mission to successfully do so in interplanetary space. In contrast, LS2 used solar sailing in Earth orbit to perform a controlled orbital maneuver: namely, raising spacecraft apogee.

1 PROBLEM SET 1

1.1 MISSION ADCS CHARACTERISTICS

We will model LightSail 2's mission objectives and hardware. The spacecraft was deployed into a circular, low-earth orbit at 720km altitude, 24 degrees inclination, where it then attempted to raise its apogee using solar sailing. For analysis, we will use the starting orbital parameters with arbitrary values selected for the unknown parameters, as noted in the orbital elements section.

The method by which LS2 raised its apogee is called the on/off control strategy. While the spacecraft was moving away from the sun, the sail normal vector pointed toward the sun ("on") in order to increase spacecraft velocity. While the spacecraft was moving toward the sun, the spacecraft normal pointed perpendicularly to the direction of the sun ("off"), to maintain orbit until the spacecraft was once again moving away from the sun. Thus, the spacecraft had two target attitudes: sun pointing and sun-perpendicular pointing. *The spacecraft attitude will be represented in this project using quaternions.*

In order to realize sun and sun-perpendicular pointing, the spacecraft will require at minimum sun sensors, but additional sensors for accuracy improvement and redundancy would be preferred. To move between the two target attitudes will require 90 degree slew maneuvers twice per orbit, which can be achieved with a variety of actuators. In this case, torque rods and momentum wheels were used.

The high level ADCS requirements are listed below [9]. To keep the project within a reasonable scope, we will focus on those above the line. Anything below the line will be assumed to be met.

- Provide attitude knowledge to within 5 degrees per axis during all mission phases.
- Sun sensors provide data on the angle of light incidence to the sensors to within ± 3 degrees accuracy.
- Magnetometers provide attitude knowledge of the body-fixed x-, y-, and z-axes to within ± 5 degrees relative to the Earth magnetic field.
- Utilize a momentum wheel to achieve 90-degree slew maneuvers about one axis in < 5 minutes.
- Following solar sail deployment, be capable of providing an angular acceleration of 0.0005 degrees/sec² per axis.
- Detumble from a maximum of 10 degrees per second per axis after sail deployment.
- Align +Z axis of the spacecraft with the magnetic field with maximum variation once settled of < 60 degrees.

-
- Damp attitude rates within 2 hours of P-POD deployment.
 - Accommodate a tip off rate of up to 25 degree/sec per axis from P-POD deployment.
 - Prior to sail deployment, utilize torque rods to achieve attitude control to within 10 degree per axis.
 - Prior to sail deployment, be capable of providing an angular acceleration of 0.1 degrees/s² per axis.
 - Prior to solar sail deployment, provide attitude control to within 10 degrees per axis.
 - Downlink telemetry for sensors, actuators and performance data.
 - Sample spacecraft angular rates using gyro sensors.
 - Be actively controllable in each of its three-axes.

To achieve these objectives, LS2 used the sensors and actuators listed in Tables 3 and 2. These components were selected for their low cost, low mass, and reliability.

Component	Number	Range	Accuracy
Sun sensors	5	[15, 165] deg/axis	± 3 deg/axis
Magnetometers	4	[40μ , 2] Gauss/axis	± 5 deg/axis
Primary Gyro	1		± 1 deg/axis
Intrepid Gyro	1		± 1 deg/axis

Table 2: LS2 Sensors

Component	Number	Specifications
Torque Rods	3	Max torque: $1 \text{ Am}^2 \times \vec{B}$
Momentum Wheel	1	Max torque: 0.06 Nm^2

Table 3: LS2 Actuators

1.2 SIMILAR MISSIONS

There are several satellites with similar mission objectives. LightSail 1 (LS1) was originally developed to meet the same objective, but was down-scoped to only achieve the goals listed prior to and including sail deployment. To reduce development costs, LS2 reused much of the LS1 design. Notable differences include a lower orbital altitude (400km for LS1 compared to 720km for LS2), removing the momentum wheel to simplify the spacecraft and reduce costs, and a simpler ADCS on LS1 [5].

IKAROS, the first successful solar sail mission, differed from LS2 in three major respects: first, it orbits the sun rather than Earth, and thus must survive the interplanetary, rather than near-Earth, environment. Secondly, it performs attitude control using liquid crystals in the sail that alter the diffusivity and reflectivity of the solar sail to allow for differential SRP across the sail [13]. Lastly, while LS1 and 2 used rigid booms to maintain the sail's shape, IKAROS removed these to save mass, and thus relies on centrifugal force from spacecraft spin to keep the sail

extended. All of these differences have driven ADCS requirements that are different from LS2 (such as maintaining a minimum spin about the axis normal to the sail).

NASA has launched missions similar to LS1 and LS2, NanoSail-D, to prove usage of solar sails as passive de-orbiting mechanisms. The first NanoSail-D was lost in launch, while the second performed many of the desired functions, de-orbiting after 240 days [4].

Another relevant mission is Prox-1, LS2's launch partner. Originally developed to demonstrate automated, close-proximity maneuvers relative to another CubeSat (with LS2 as its target), the proximity maneuvers were ultimately cut and the spacecraft now serves primarily to monitor and image LS2. Prox-1's ADCS mission requirements are very different (and more strict) than LS2's, requiring greater attitude knowledge and control to perform its maneuvers and imaging [10].

1.3 MECHANICAL LAYOUT

High-level drawings of the LS2 system in its various configurations are provided in Appendix B; note that this study will only consider the LIGHTSAIL DEPLOYED configuration. The basic layout of the 3U structure with the placement of primary ADCS components is provided in Figure 1. The body coordinates used to describe the mechanical layout are plotted in Figure 2. Note that the system is roughly axisymmetric over the x and y , with the $+z$ axis pointing away from the deployed solar panels.

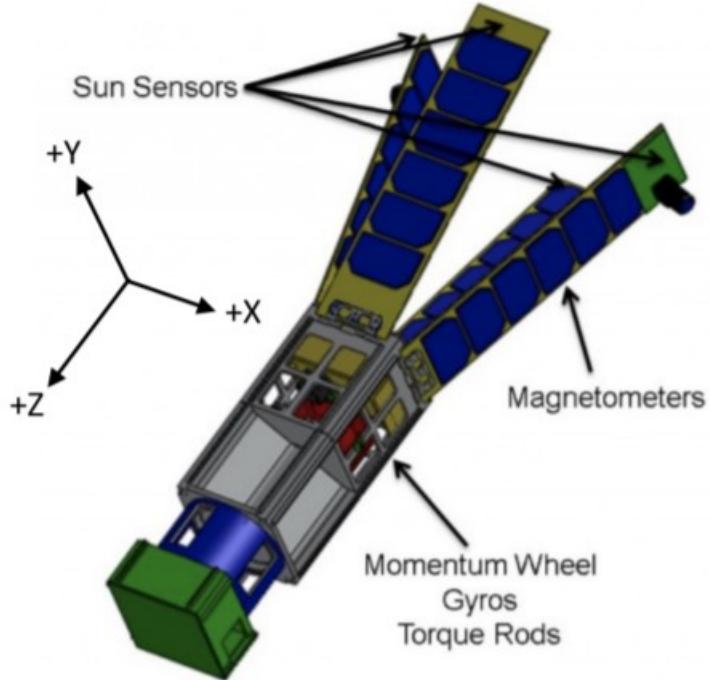
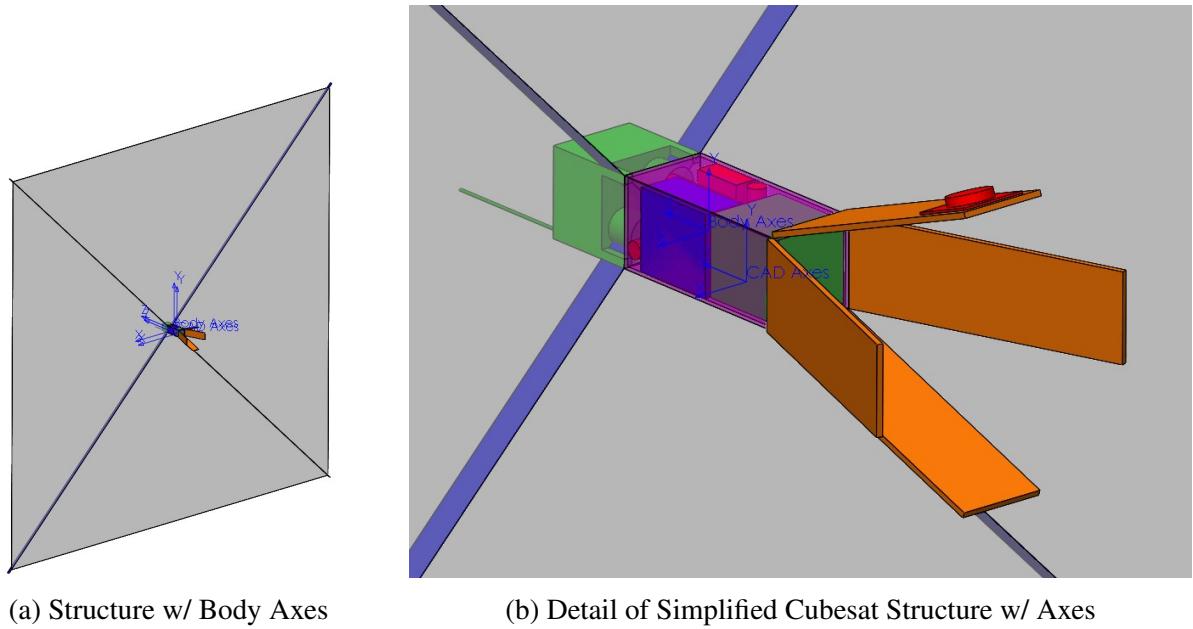


Figure 1: Annotated Full-Detail Layout of 3U Structure

LS2 uses a motorized TRAC (Triangular Roll-able And Collapsible) boom deployment system to unfurl its 5.57m square Mylar sail. During launch, the 4 primary solar panels are folded along the length of the satellite - once deployed, the panels extend along the $-z$ axis. After de-tumbling with magnetic torquing, the booms extend and unfold the 4.5 micron thick Mylar from a central "sail housing" unit. Due to limited knowledge about the exact mass distribution



(a) Structure w/ Body Axes

(b) Detail of Simplified Cubesat Structure w/ Axes

Figure 2: Simplified Mechanical Layout of LightSail 2

of the satellite and a desire for a simplified model, the mechanical layout has been reduced to five primary components, detailed in Table 4 below. The mass calculations are described in further detail in Appendix A; while some effort was taken to include asymmetric contributions to the total volume, the large contribution of the square solar sail creates a fairly axisymmetric system.

Name	Color in 3D Model	Subsystems	Mass (kg)
Forward Body	Green	Comms, Boom Motors (x2), Boom Housing	1.47
Booms	Blue	TRAC Booms (x4)	0.93
Solar Sail	Grey	Mylar sail	0.20
Rear Body	Purple	Magnotorquers, Mom. Wheel, Power, Avionics	1.73
Solar Panels	Orange	Solar Cells, Magnetometers, Cameras	0.60
TOTAL	-	-	4.93

Table 4: Primary Components of Simplified Mechanical Layout

1.4 INERTIA PROPERTIES

The inertia properties provided below were computed using the 3D model shown in Figure 2. An analytical verification of these values is provided in Appendix D. Along the principal axes, the moments of the inertia for the system are

$$I_{xx} = 3.10288 \text{ kg} \cdot \text{m}^2, I_{yy} = 3.10553 \text{ kg} \cdot \text{m}^2, I_{zz} = 5.98305 \text{ kg} \cdot \text{m}^2$$

Using our 3D model to calculate the full inertia matrix yields, in the body frame,

$$L = \begin{bmatrix} 3.10553 & -0.00011 & -0.00003 \\ -0.00011 & 3.10289 & -0.00005 \\ -0.00003 & -0.00005 & 5.98305 \end{bmatrix} \text{ kg} \cdot \text{m}^2$$

1.5 OUTER SURFACE DISCRETIZATION

To provide an easy way of analyzing environmental perturbative torques, we also simplify the outer geometry. The surface normals and centroids are pulled from the CAD model, where the solar panels and solar sail are taken to have a negligible thickness (effectively reducing those components to planes) and minor elements like cameras are not included in the discretization. Using the Solidworks VBA Macro in Appendix C, we export the centroid, unit normal, and area information for each outward-facing surface of interest. A MATLAB script (included in the same appendix) imports the information into MATLAB data structures for use in future simulations. Table 5 displays the centroid locations in body coordinates, the associated normal vector, and the associated area.

Location (m)	Normal Vec	Area (m ²)
(-0.003, -0.054, -0.049)	(0, -1, 0)	0.023
(0.047, -0.004, -0.049)	(1, 0, 0)	0.023
(-0.003, 0.046, -0.049)	(0, 1, 0)	0.023
(-0.053, -0.004, -0.049)	(-1, 0, 0)	0.023
(-0.003, -0.004, -0.164)	(0, 0, -1)	0.01
(-0.003, 0.1, -0.313)	(0, -0.94, -0.342)	0.032
(0.101, -0.004, -0.313)	(-0.94, 0, -0.342)	0.032
(-0.003, -0.108, -0.313)	(0, 0.94, -0.342)	0.032
(-0.107, -0.004, -0.313)	(0.94, 0, -0.342)	0.032
(-0.003, 0.1, -0.313)	(0, 0.94, 0.342)	0.032
(-0.107, -0.004, -0.313)	(-0.94, 0, 0.342)	0.032
(-0.003, -0.108, -0.313)	(0, -0.94, 0.342)	0.032
(0.101, -0.004, -0.313)	(0.94, 0, 0.342)	0.032
(-0.053, -0.004, 0.121)	(-1, 0, 0)	0.011
(-0.003, -0.054, 0.121)	(0, -1, 0)	0.011
(0.047, -0.004, 0.121)	(1, 0, 0)	0.011
(-0.003, 0.046, 0.121)	(0, 1, 0)	0.011
(-0.003, -0.004, 0.176)	(0, 0, 1)	0.01
(-0.003, -0.004, 0.066)	(0, 0, -1)	31.003
(-0.003, -0.004, 0.066)	(0, 0, 1)	31.003

Table 5: Centroid, normal, and area data pulled from CAD model.

1.6 ORBITAL ELEMENTS & PROPAGATION

The initial orbit propagator was developed for AA 279A in Winter 2019, and later further supplemented to include simple orbital perturbations (namely, atmospheric drag and J2, with

the infrastructure in place to add additional perturbations as relevant). SRP will be added to model the change in apogee that is one of the mission objectives. It will be incorporated into the ADCS simulation, and an overview is presented in Figure 3.

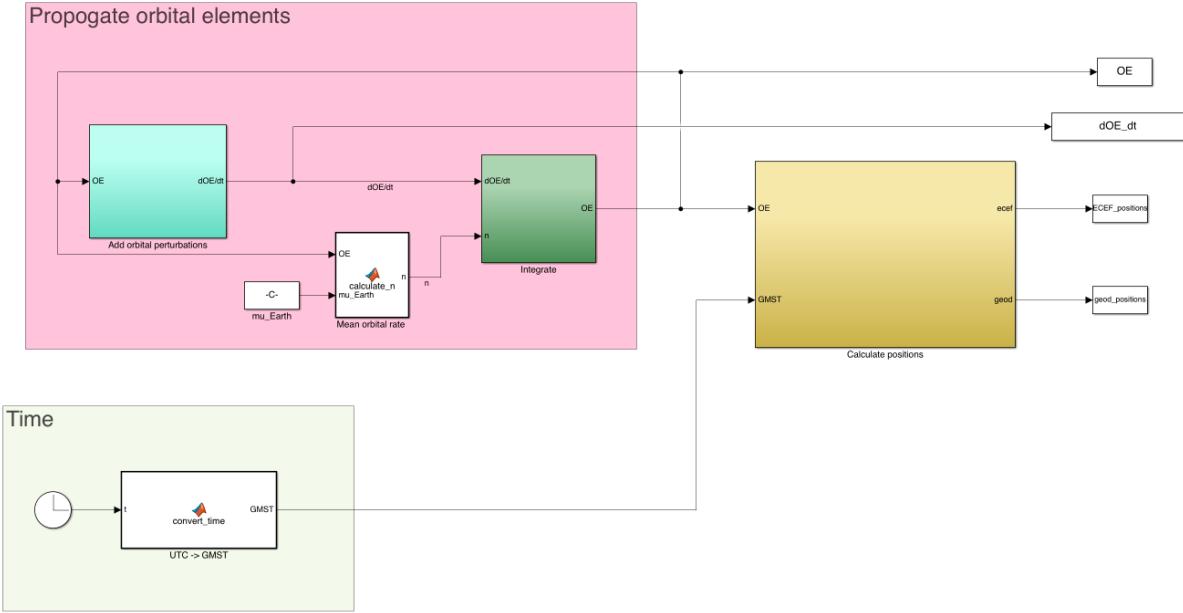


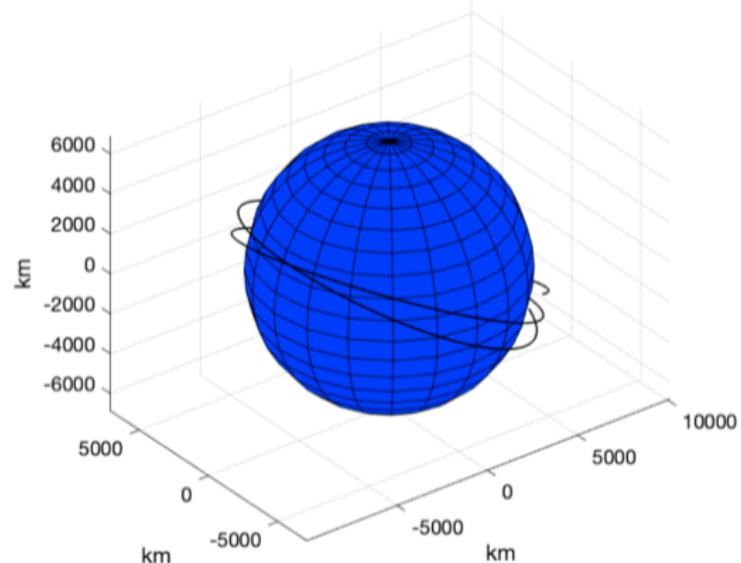
Figure 3: Top-level overview of the initial orbit propagator.

Initial values for the orbital elements are given in Table 6. These were used to produce the sample orbits given in Figure 4.

Orbital element	Symbol	Initial value	Units	Notes
Epoch	t_0	2019-07-08.20	YY-MM-DD.dd	
Semi-major axis	a	7095.553	km	$R_E + 717.4175\text{km}$
Eccentricity	e	0.0010951		
Inclination	i	24	deg	
RAAN	Ω	0	deg	Selected arbitrarily
Argument of Periapsis	ω	0	deg	Selected arbitrarily
Eccentric anomaly	E	0	rad	Selected arbitrarily

Table 6: Initial orbital elements, pulled from [12]. The arbitrarily selected values are those for which we do not have data, and have little effect on our modeling.

Orbit of LS2 in ECEF reference frame



Groundtrack of LS2 in the Mercator lat/long projection

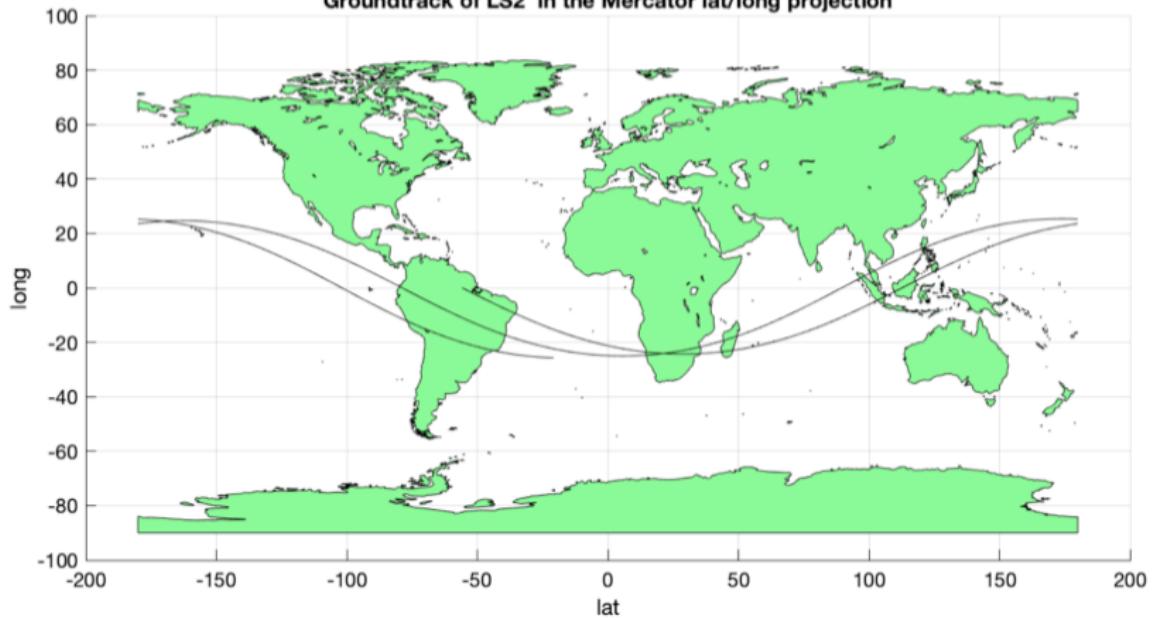


Figure 4: Initial sample orbits of LS2 given the orbital parameters in Table 6.

2 PROBLEM SET 2

2.1 PRINCIPAL AXES

The principal axes are shown alongside the body axes in Figure 5 - note that due to the overwhelming inertial contribution of the solar sail and booms, the primary difference between the axes systems is the flipping of the x and y axes to ensure well-ordered eigenvalues. Using the MATLAB script below, we can calculate the rotation matrix from the principal axes to body axes (whose columns represent the expression of the principal axes in body coordinates).

$$R = \begin{bmatrix} 0.04156 & 0.99911 & -1.042 \times 10^{-5} \\ 0.9991 & -0.04156 & -1.736 \times 10^{-5} \\ 1.778 \times 10^{-5} & 9.695 \times 10^{-6} & 1.000 \end{bmatrix}$$

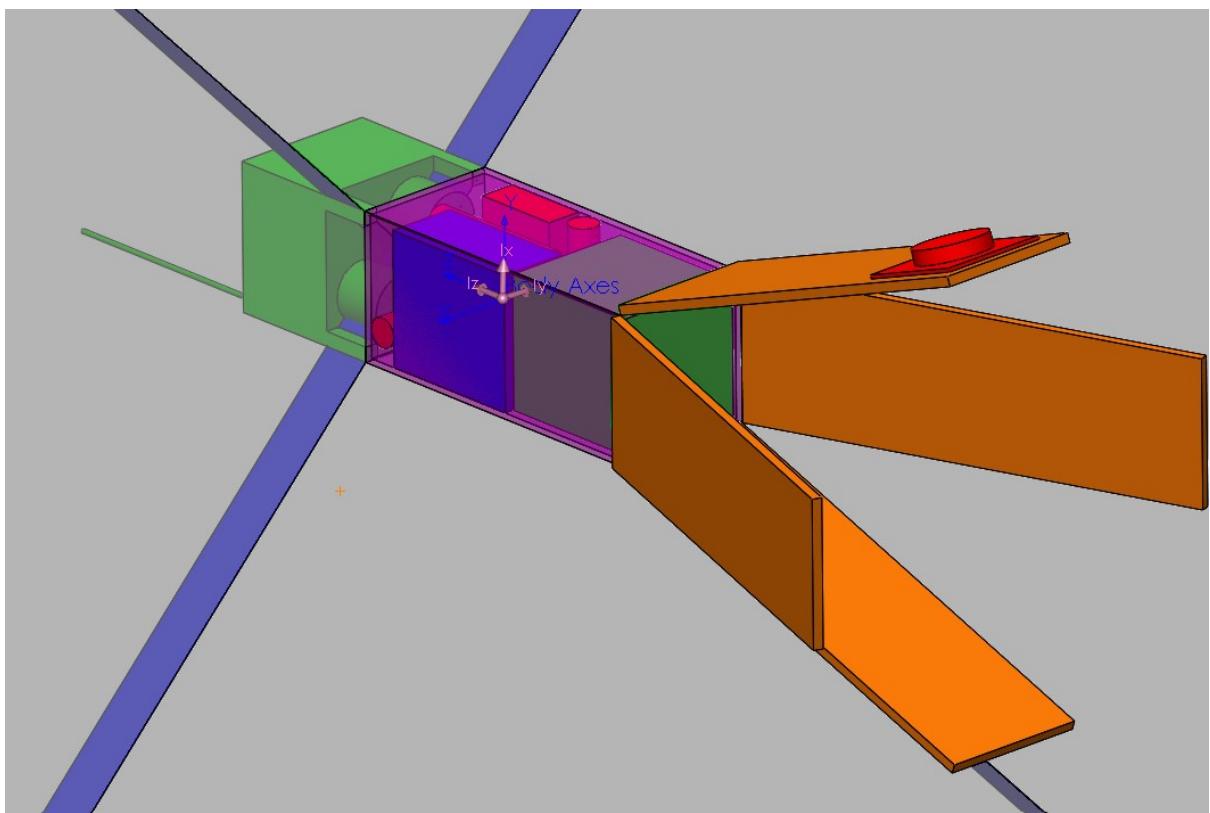


Figure 5: Body axes (blue) and principal inertia axes (pink), overlaying the simplified space-craft CAD model.

2.2 NO-TORQUE PROPAGATION

Using the Simulink block-diagram in Figure 6, we can propagate the angular velocity without the contribution of torque with respect to the principal axes. Plotting the resulting polhode over top of the momentum and energy ellipses that constrain the motion of the spacecraft in Figure 7, we can clearly see how our angular velocity vector is constrained to lay along the intersection of the two ellipses. Additionally, we can see from the planar projections of the polhode that while the x and z projections are (sections of) ellipses, the y projection is a hyperbola. Note that we use the following (arbitrary) initial angular velocity (expressed in principal axes) for our spacecraft:

$$\omega_0 = \begin{bmatrix} -6 \\ 8 \\ 0.1 \end{bmatrix} \text{ deg/sec}$$

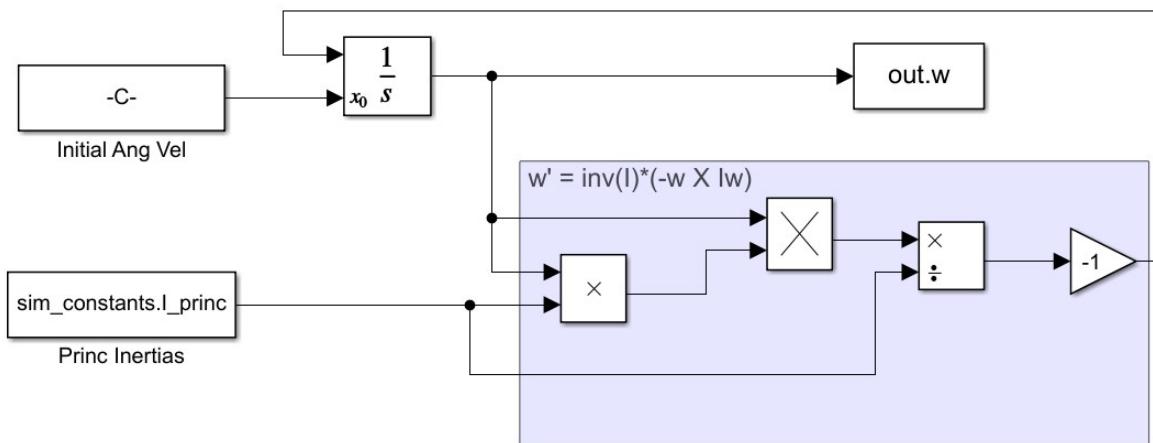


Figure 6: Simulink block diagram for propagating no-torque Euler equations in principal axes.

To verify that our solution works in other cases, we consider an angular velocity along a principal axis to test out our propagator:

$$\omega_{0,2} = \begin{bmatrix} 0 \\ -10 \\ 0 \end{bmatrix} \text{ deg/sec}$$

The results of this propagation are shown in Figure 8. Again, our results are as expected - our angular velocity remains constant, but appears to lay at an unstable equilibrium (which makes sense, as our spin is about the intermediate axis - any off-axis angular velocity, however small, would result in tumbling).

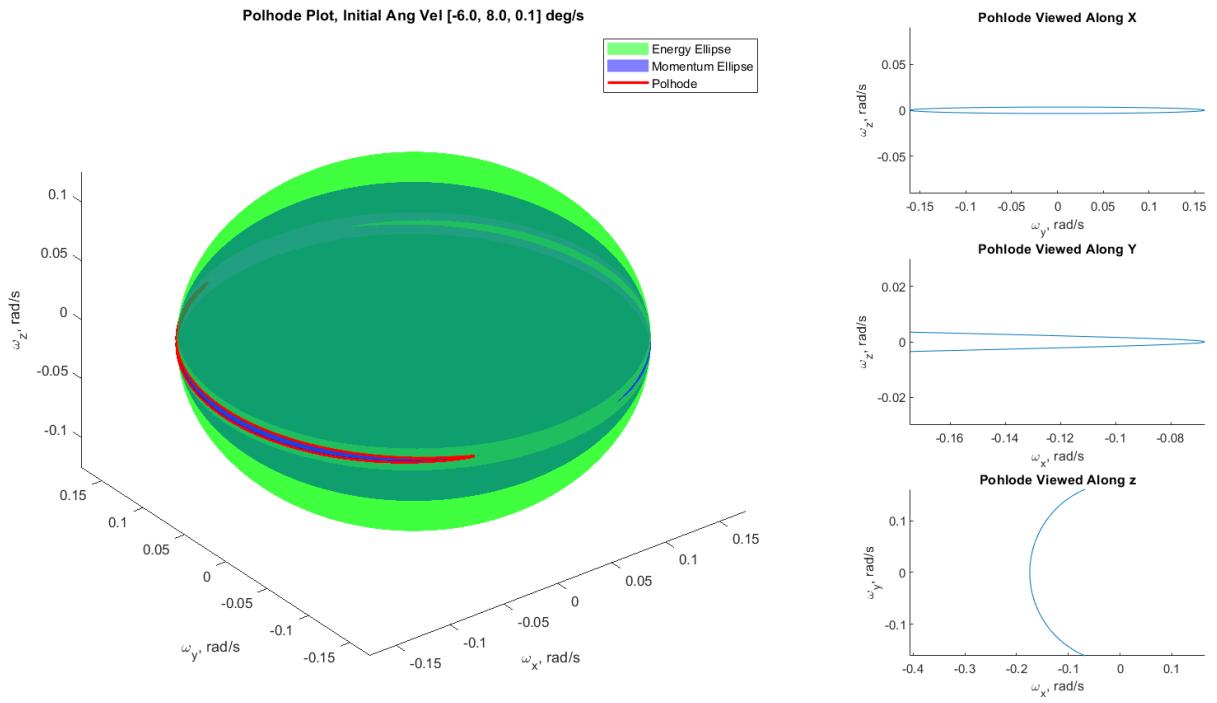


Figure 7: Plot of spacecraft polhode propagation in principal axes, assuming no-torque and non-trivial initial conditions.

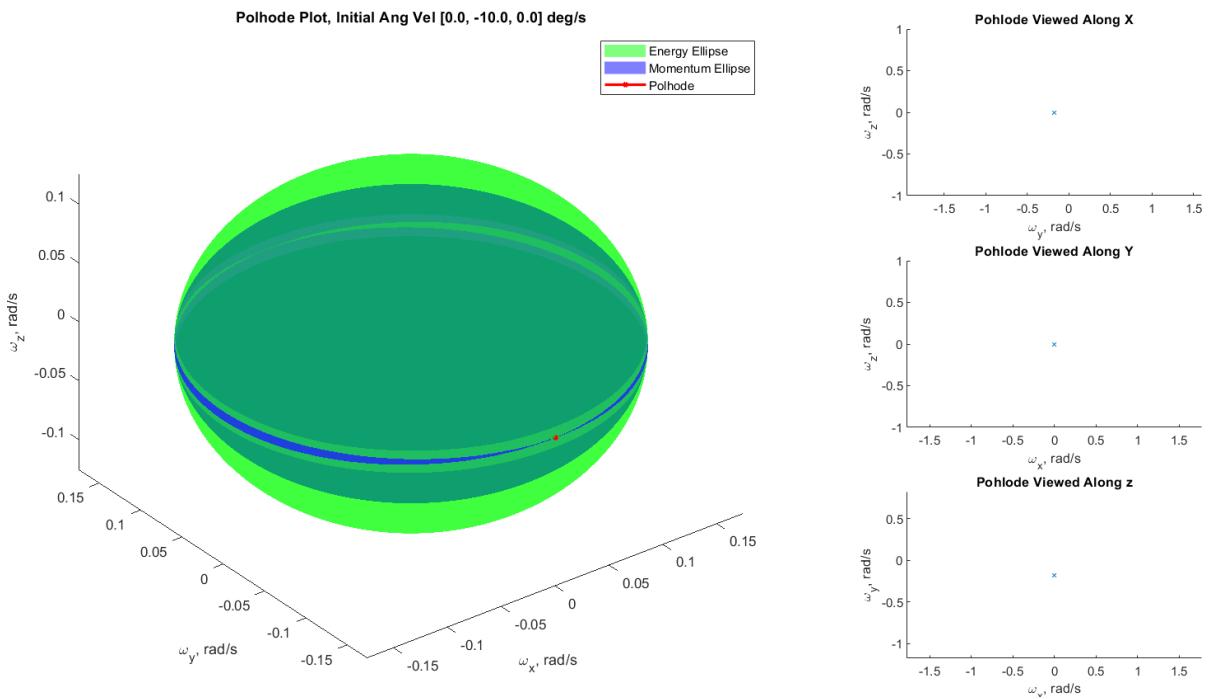


Figure 8: Plot of polhode propagation in principal axes, assuming no-torque and an initial angular velocity along the intermediate axis only.

2.3 AXIAL SYMMETRY

Although our spacecraft is nearly axisymmetric, we would like to analyze the case of torqueless true axial symmetry. For this, we set $I_{xx} = I_{yy} = 3.10288 \text{ kg} \cdot \text{m}^2$. The solution to Euler's equations in the axisymmetric case, where $\lambda = \frac{I_z - I_x}{I_x} \omega_z$, is simply the circle

$$\omega_{xy} = (\omega_{x0} + i\omega_{y0}) \exp(i\lambda t) \quad (1)$$

As usual for this representation of circles, ω_x is the real component while ω_y is the imaginary. Using the initial conditions that generated Figure 7, the analytic result across time is shown in Figure 9.

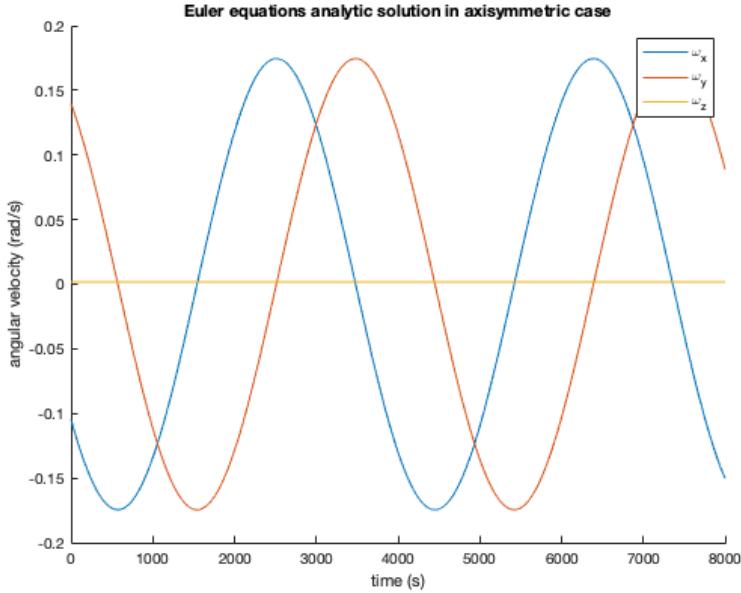


Figure 9: Time-domain solution of axisymmetric Euler equations using $\omega_0 = [-6, 8, 0.1]$.

Comparing to the numeric results achieved using numeric propagation of the Euler equations, the error evolves as in Figure 10. First, we observe that there is no ω_z error. This is expected as in this system, it is constant and uncoupled from ω_x, ω_y , so there should be no numeric integration error. For ω_x, ω_y , the error starts small – on the order of 10^{-10} rad/s. The error behaves as sinusoids with mean 0 (and period similar to analytic result) bounded by exponential growth – the amplitudes are growing and will continue to grow unbounded as time progresses, although it will take many revolutions to reach any substantial error. Further, there is a phase delay between ω_x, ω_y in the numeric propagator.

Thus, within the integration error, the angular velocity vectors exchange momentum as expected, maintaining a constant ω_{xy} . However, there will be an apparent fluctuation of angular momentum about the mean due to integration error.

Finally, we compare the numeric results of the no-torque pseudo-axisymmetric case, as we have modelled the satellite, to the true axisymmetric case in Figure 11. Although the differences in inertia in the two cases are small ($I_{yy} = 3.10288 \text{ kg} \cdot \text{m}^2$ in the ideal case versus $I_{yy} = 3.10553 \text{ kg} \cdot \text{m}^2$ in the real case), the error is not small for ω_x, ω_y . Because ω_z is largely decoupled and constant, this error is small, although noticeable. From this, we can see it is important for us to simulate the coupled motion of the nearly symmetric axes of our spacecraft and not make the true axisymmetric assumption.

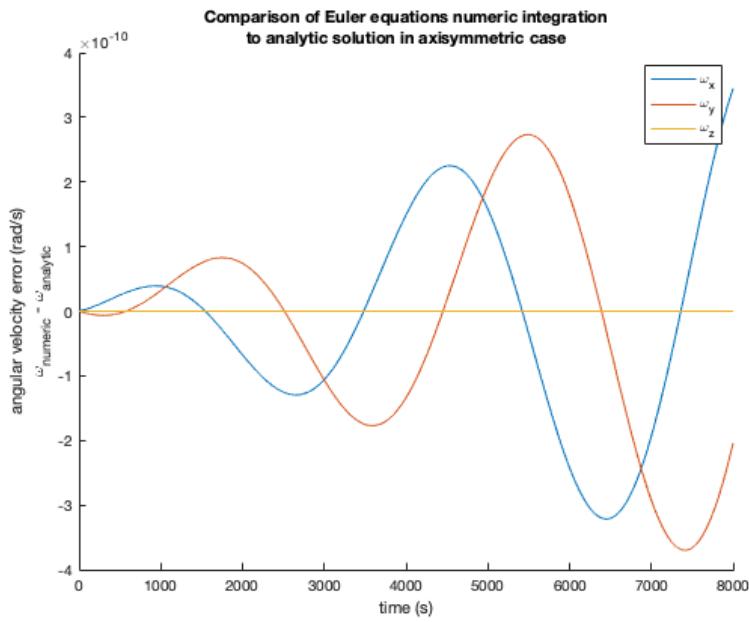


Figure 10: Evolution of error of numeric integration of Euler equations in the axisymmetric case ($\omega_0 = [-6, 8, 0.1]$.)

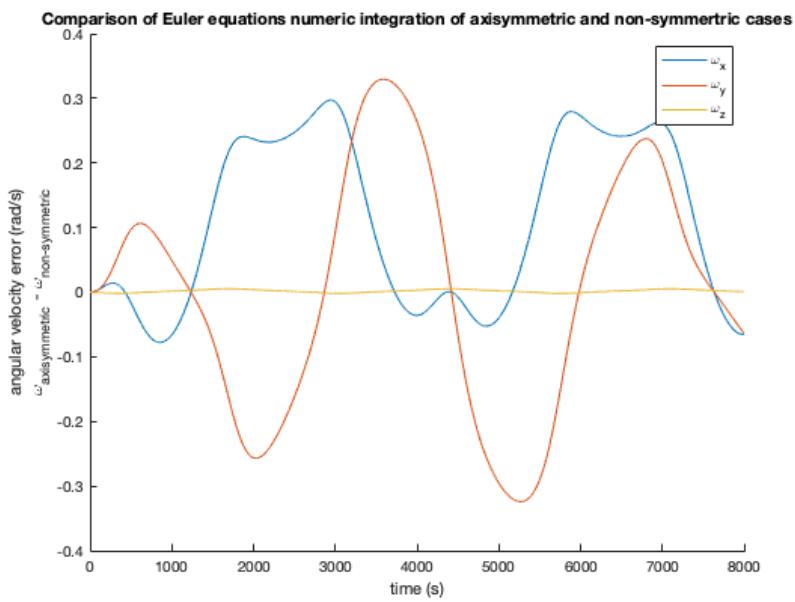


Figure 11: Evolution of difference between numeric integration of Euler equations in the pseudo-axisymmetric and true axisymmetric cases ($\omega_0 = [-6, 8, 0.1]$.)

3 PROBLEM SET 3

3.1 KINEMATIC INTEGRATION

To build toward an easily validated system, we build in kinematics integration that supports two types of attitude parameterization: quaternions (our primary method) and 312 Euler angles (a secondary method for validation). By toggling an input boolean, the simulation's attitude propagation can be switched between the two attitude parameterizations with ease.

The kinematics equation for quaternions was developed in class as

$$\frac{d\vec{q}}{dt} = \frac{1}{2}\Omega\vec{q} \quad (2)$$

for Ω as a function of the angular velocity, $\vec{\omega}$,

$$\Omega = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix}$$

For 312 Euler angles, we derive the kinematics equation starting with the angular velocity in principal axes

$$\vec{\omega} = \dot{\phi}\hat{3} + \dot{\theta}\hat{1}' + \dot{\psi}\hat{y}$$

We get the components in principal frame by projecting onto the principal axes

$$\begin{aligned} \omega_x &= \vec{\omega} \cdot \hat{x} = \dot{\phi}(\hat{3} \cdot \hat{x}) + \dot{\theta}(\hat{1}' \cdot \hat{x}) \\ &= \dot{\phi} \cos \theta \sin \psi + \dot{\theta} \cos \psi \end{aligned}$$

$$\begin{aligned} \omega_y &= \vec{\omega} \cdot \hat{y} = \dot{\phi}(\hat{3} \cdot \hat{y}) + \dot{\theta}(\hat{1}' \cdot \hat{y}) + \dot{\psi} \\ &= \dot{\phi} \sin \theta + \dot{\psi} \end{aligned}$$

$$\begin{aligned} \omega_z &= \vec{\omega} \cdot \hat{z} = \dot{\phi}(\hat{3} \cdot \hat{z}) + \dot{\theta}(\hat{1}' \cdot \hat{z}) \\ &= \dot{\phi} \cos \theta \cos \psi - \dot{\theta} \sin \psi \end{aligned}$$

Solving for our angular rates, we get

$$\begin{aligned} \dot{\phi} &= \frac{1}{\cos \theta} (\omega_x \sin \psi + \omega_z \cos \psi) \\ \dot{\theta} &= \omega_x \cos \psi - \omega_z \sin \psi \\ \dot{\psi} &= \omega_y + \tan \theta (\omega_x \sin \psi + \omega_z \cos \psi) \end{aligned} \quad (3)$$

Note the singularity at $\theta = \frac{\pi}{2}$ for this parameterization.

Integrating our kinematics equations with the Euler equations from the previous problem set gives a full determination of our attitude in our orbital propagation. Using the bi-directional conversions between quaternions, 312 Euler angles, and direction cosine matrices, we can compare our solutions and visualize our attitude throughout the orbit. All solution below are generated using the quaternion kinematics, but a brief comparison with the 312 Euler results is provided in Appendix F.

3.2 ATTITUDE PROPAGATION

Given our previously-defined initial conditions, we propagate our spacecraft through a single orbit period. We can then verify that our simulated attitude and angular velocity match our expectations. In Fig 12, we can see that the components of angular momentum remain very nearly constant in the inertial frame throughout the duration of simulation (note that the small errors are due to numerical propagation effects). In Fig 13, we see that the angular velocity trace in inertial coordinates, the "herpolhode," lays in a plane normal to the (constant) angular momentum vector.

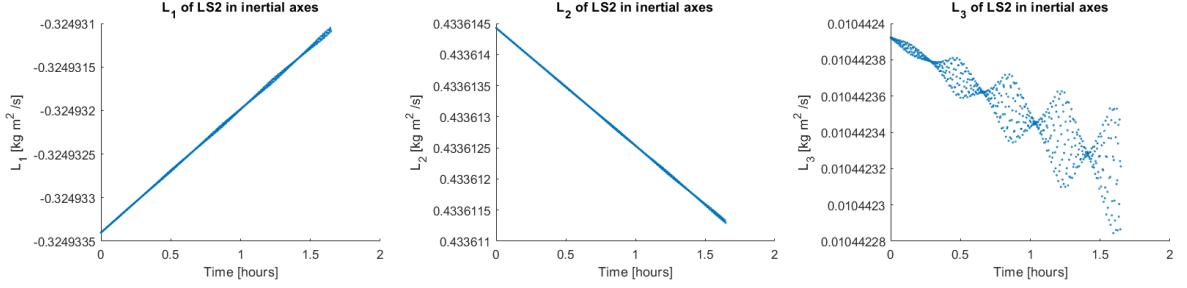


Figure 12: Components of Angular Momentum in Inertial Coordinates

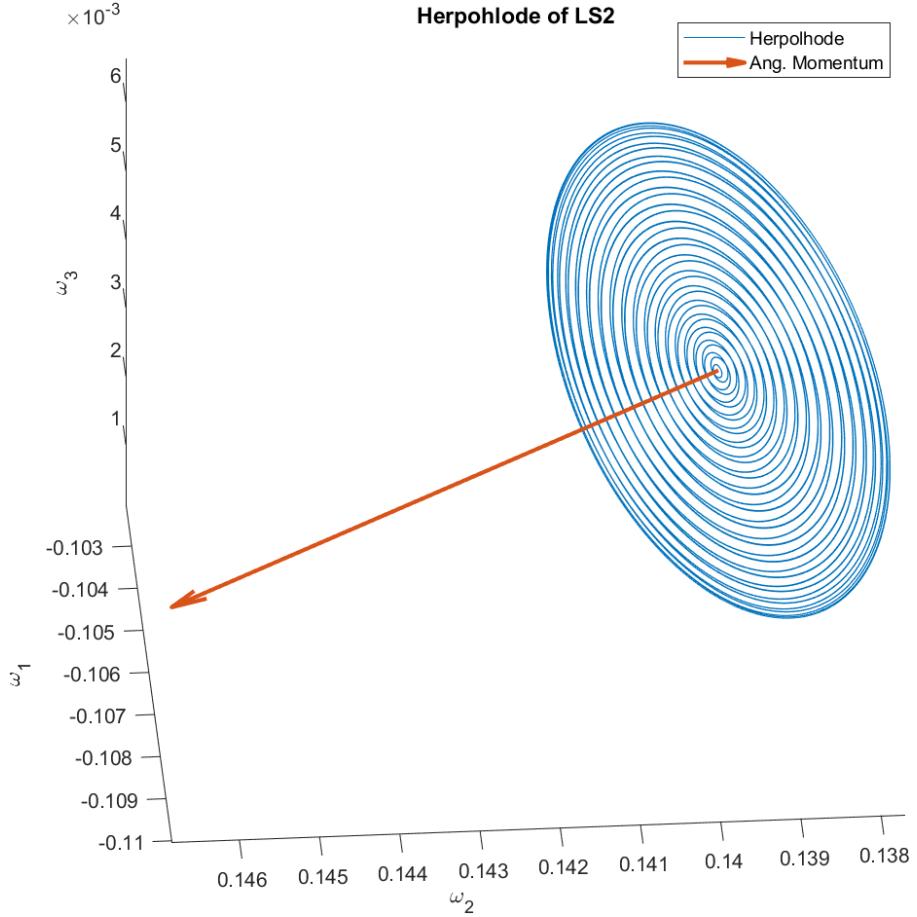


Figure 13: Herpolhode of LS2 spacecraft in plane

In Figs 14 and 15, we can see various coordinate systems of interest (RTN, body, and principal

axes) plotted in time. The former plot places the coordinate triads in the appropriate place in ECI, for visualization of the coordinate system with respect to the orbital motion. The latter show the trace of the coordinate axes' tips, their color changing from the initial color to yellow as time progresses.

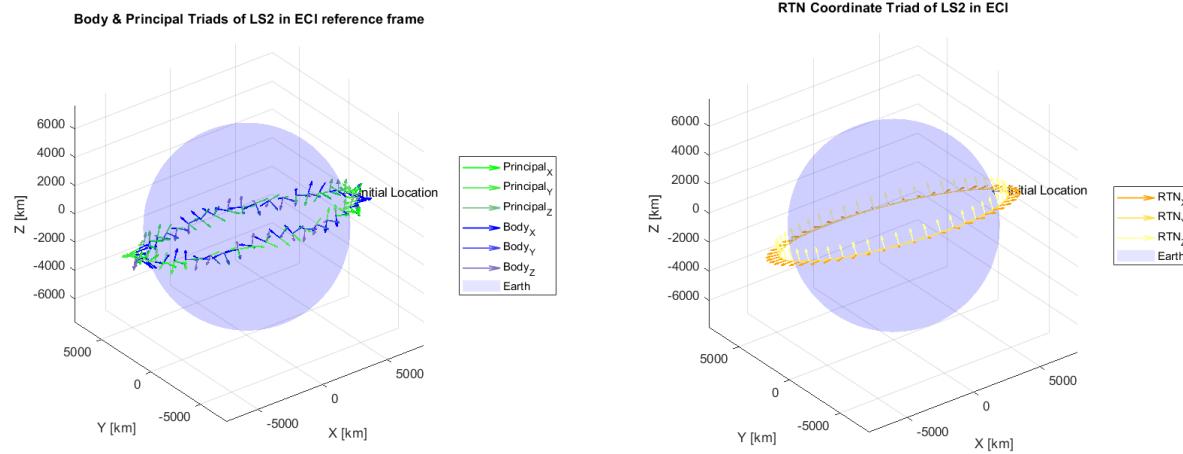


Figure 14: Coordinate triads motion in ECI

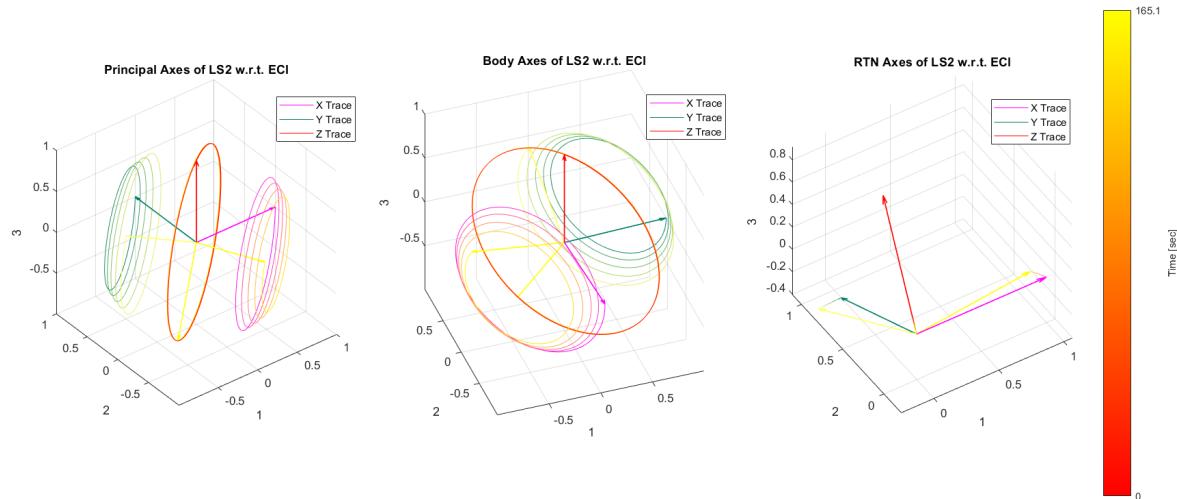


Figure 15: Coordinate triad motion in time

3.3 EQUILIBRIUM TESTS

Placing our initial spin along the principal axis of maximum inertia and assuming that the principal axes initially correspond with the inertial axes, we can test if our attitude propagation maintains stability - we see in Fig 16 that the components of angular velocity remain constant in principal axes, as expected. Converting our quaternion output to 312 Euler angles in Fig 17, we can see that θ and ψ remain constant while ϕ increases linearly (note that the sawtooth shape comes from the angle being wrapped to 2π).

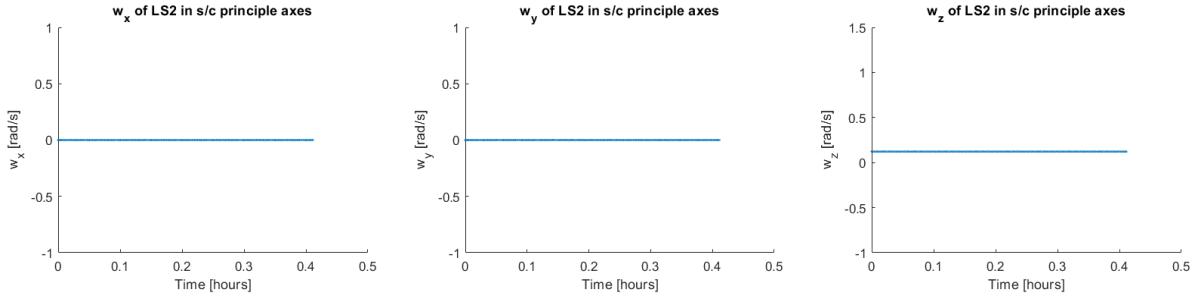


Figure 16: Components of angular velocity for spin along principal Z axis

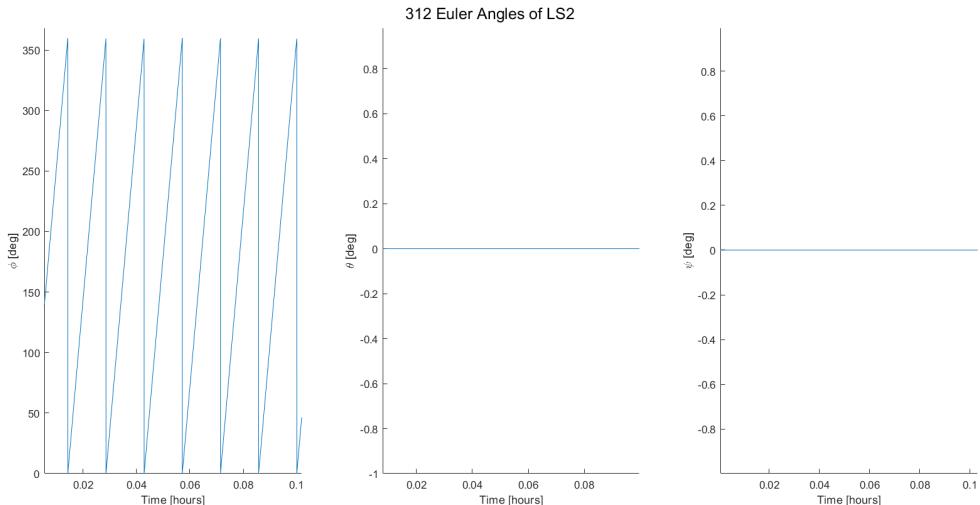


Figure 17: 312 Euler angles for spin along principal Z axis

If we instead assume that our principal axes align with the RTN frame initially, with the angular velocity oriented along the N axis, our angular velocity components once again remain constant (so long as we ignore any orbital perturbations), as seen in Fig 18. This behavior initially may seem strange, as the RTN frame is non-inertial, but with spin around the principal z-axis, our angular velocity is stable; furthermore, the N axis remains constant due to the lack of orbital perturbations (no torques keep the angular momentum vector, \vec{h} , constant).

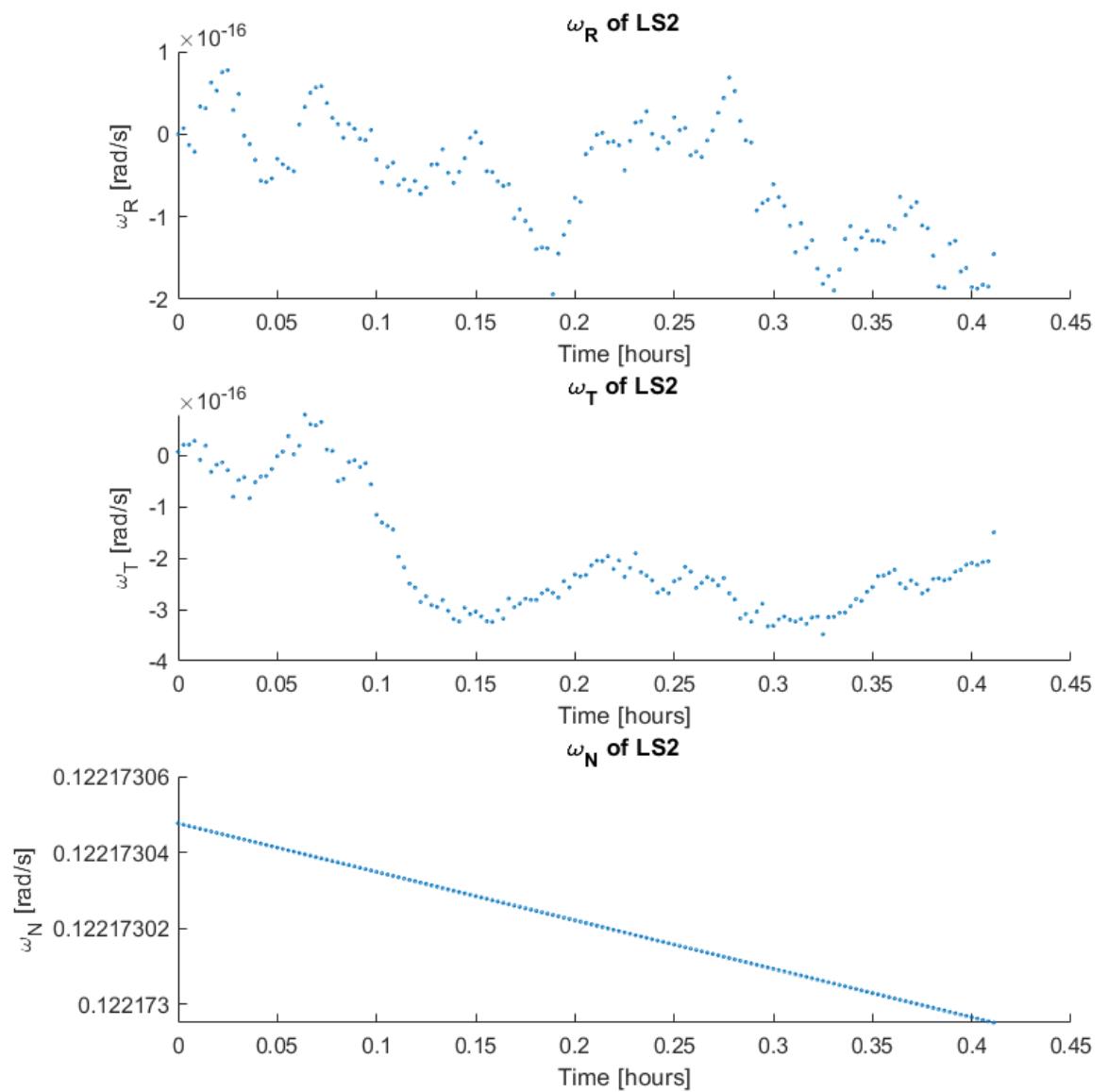


Figure 18: Components of angular velocity in RTN frame, with spin along N axis

3.4 STABILITY TESTS

Once again initially aligning the principal axes with the inertial, we test the stability of spin about the three principal axes with minor perturbations. This effect is generated by providing a spin of 7 deg/s about the principal axis in question, and giving a small spin of 0.01 deg/s about the other two principal axes. The results of these stability simulations (principal-axes angular velocity, momentum, and Euler angles) are provided in Figs 19 - 21 - note that in general, the rapid switching between complementary values for Euler angles is due to the inverse trigonometric functions used in converting from quaternions to 312 Euler angles.

For an initial spin about the X-axis, the axis of least inertia, we see *periodic stability* in angular velocity and instability in angles in Fig 19.

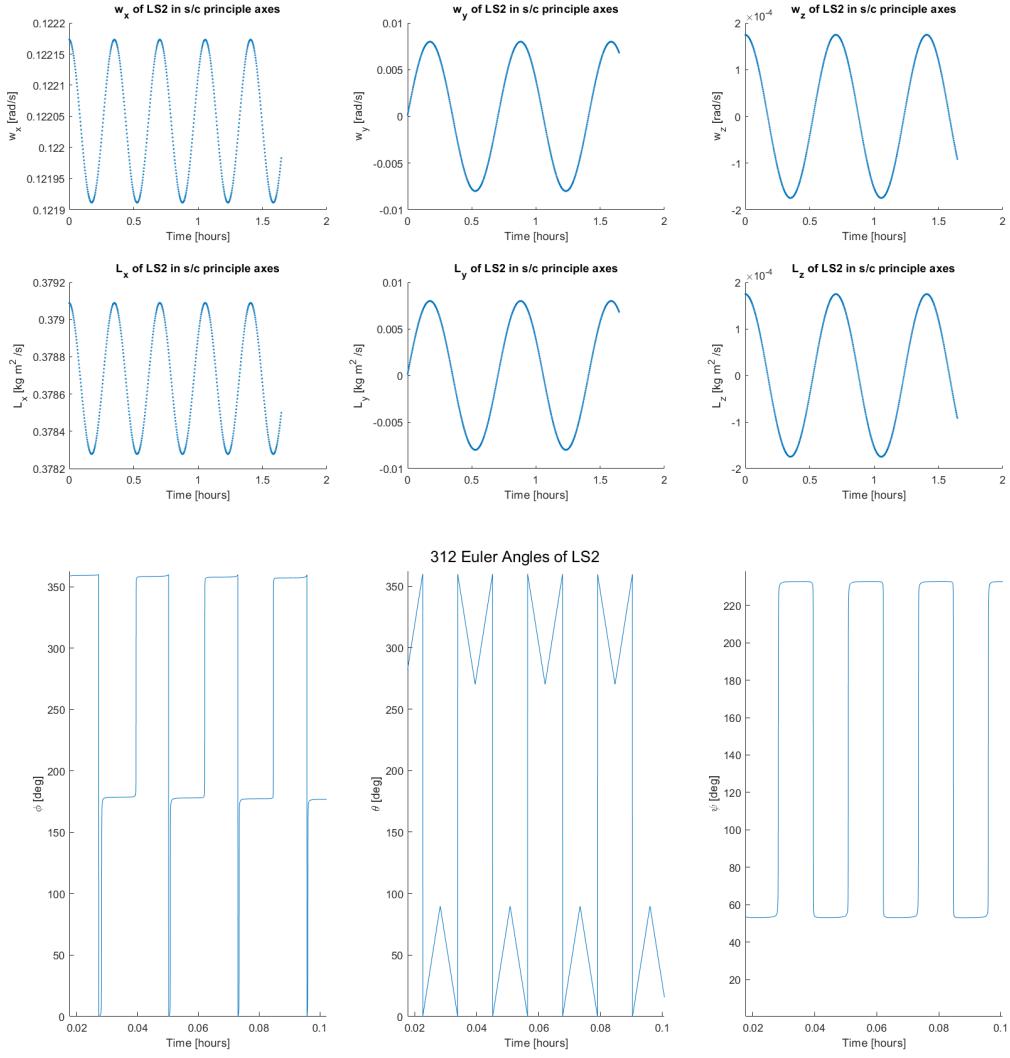


Figure 19: Velocities, angular momentum, and 312 Euler angles for principal X-axis spin.

For an initial spin about the Y-axis, the axis of intermediate inertia, we see *instability* in angular velocity and instability in angles in Fig 20.

For an initial spin about the Z-axis, the axis of maximum inertia, we see *periodic stability* with a much higher frequency in angular velocity and instability in angles in Fig 21.

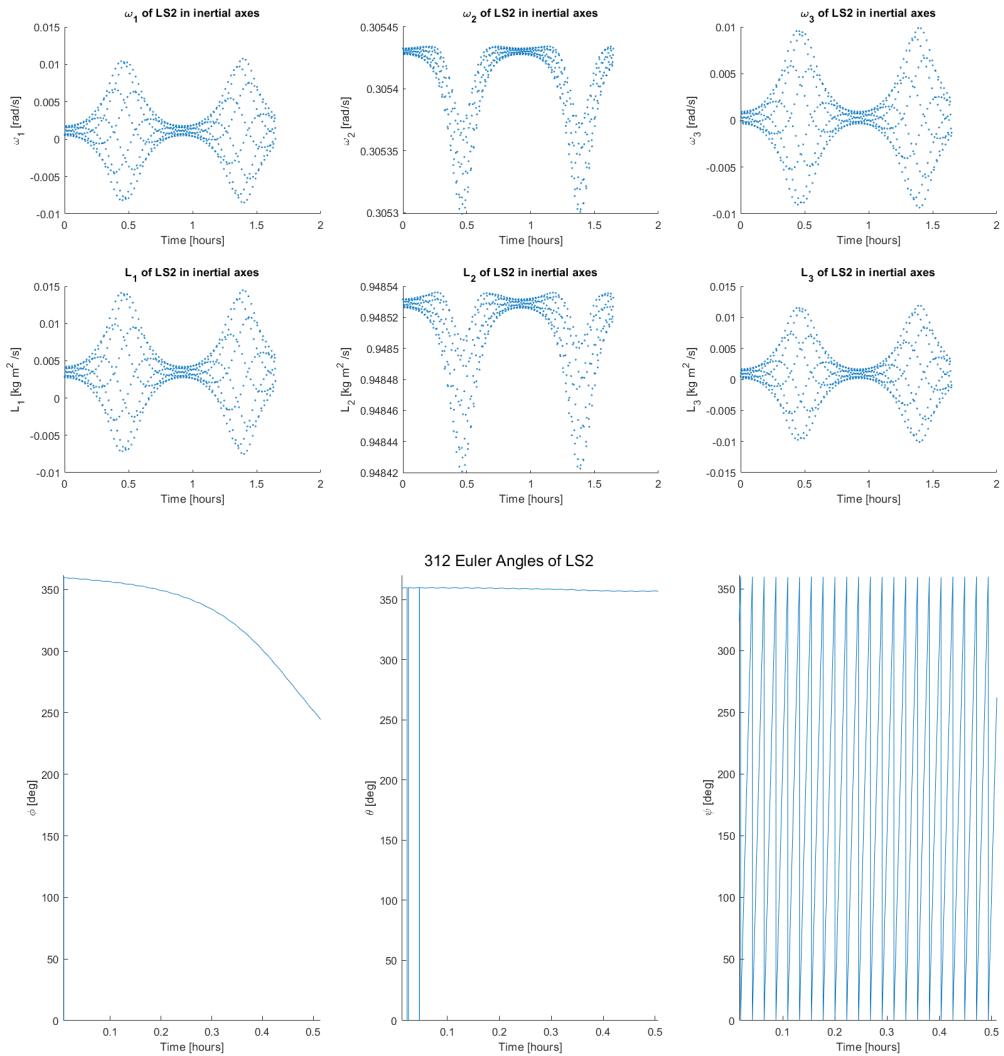


Figure 20: Velocities, angular momentum, and 312 Euler angles for principal Y-axis spin.

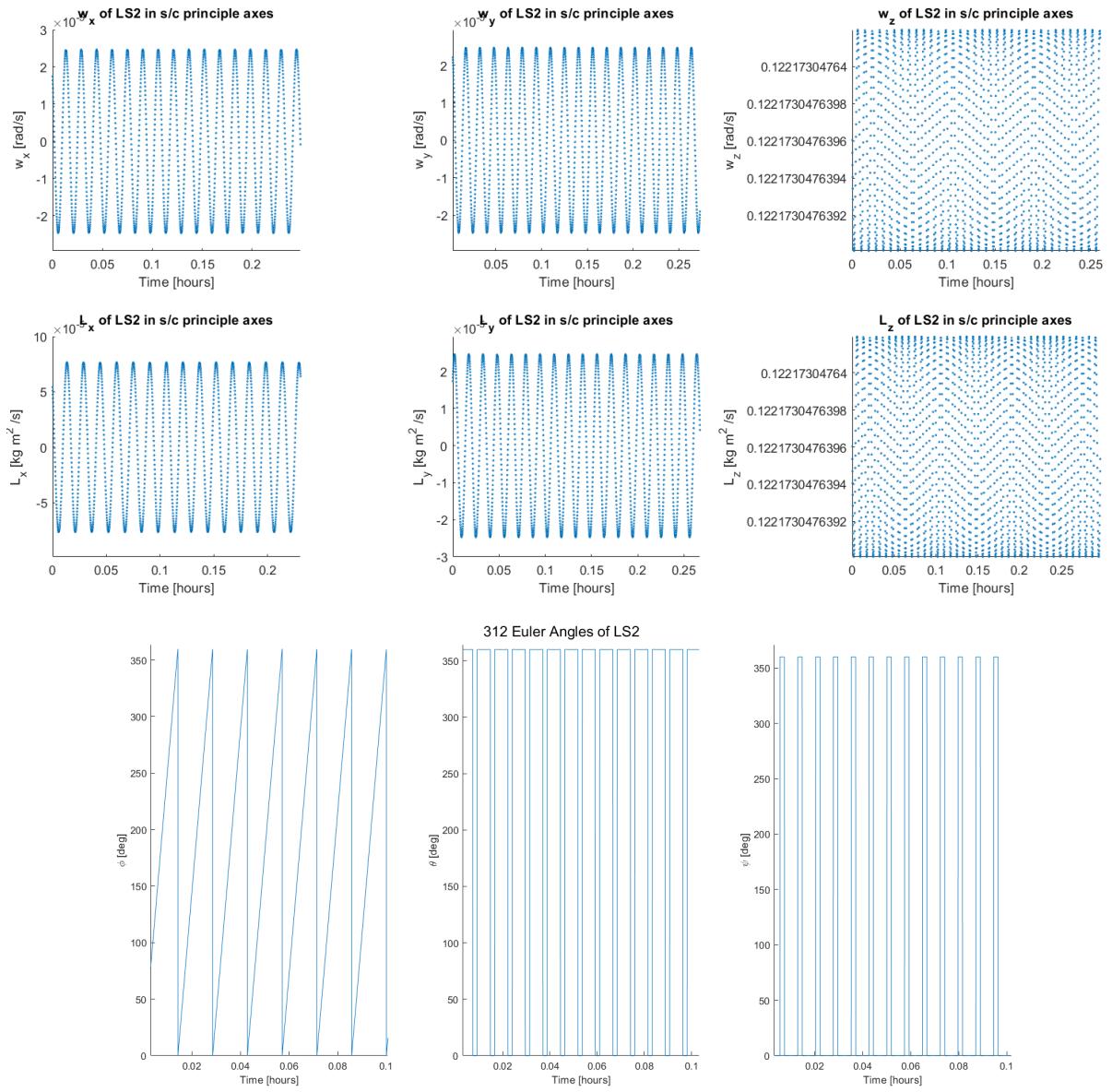


Figure 21: Velocities, angular momentum, and 312 Euler angles for principal Z-axis spin.

4 PROBLEM SET 4

4.1 DUAL SPIN

By adding a momentum wheel into our simulation (as seen in Fig 22), we give the spacecraft the capacity to perform maneuvers and ensure stability along an axis of choice. In Appendix G, we validate our new dynamics by testing the equilibrium and stability cases from the previous problem set, but now with an applied angular momentum in the momentum wheel. The data used for our momentum wheel's moment of inertia and maximum spin rate is pulled from the LS2 parts list [8], which indicates that LS2 uses a RW3-0.060 momentum wheel from Sinclair Interplanetary.

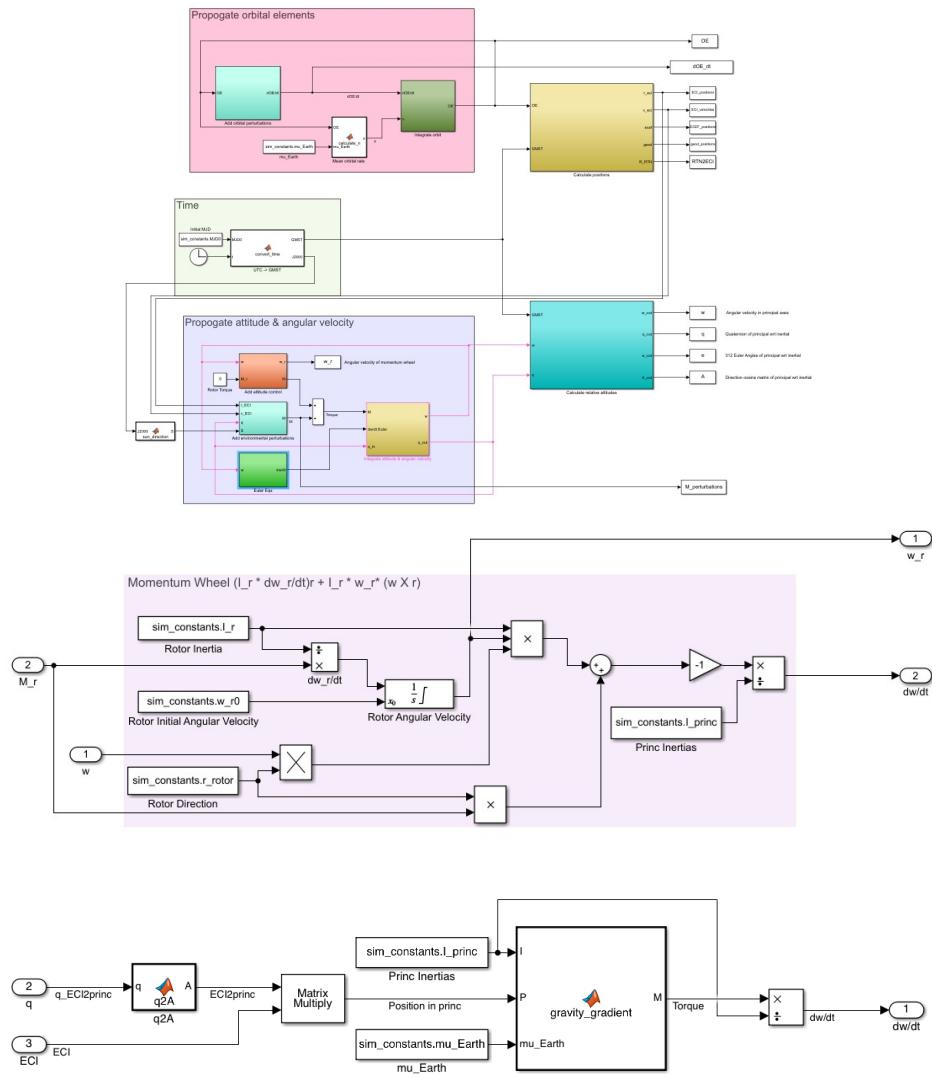


Figure 22: Updated Overview of Simulink Propagator with Momentum Wheel and Gravity Gradient Details

To give stability along our intermediate axis (the principal Y), we can place the rotor's axis along Y and give it a spin such that

$$I_r \omega_r > (I_z - I_y) \bar{\omega}_y$$

where I_r is the rotor's moment of inertia and $\bar{\omega}_y$ is the desired equilibrium spin rate. By setting our initial angular velocity to be equivalent to that used in our Y -axis stability test, we can cross-compare our results. In Fig 23, we can see the resulting inertial-axes angular velocity components and the 312 Euler angles that result from our simulation. Unlike in our stability test for the intermediate axis, we now see that our spin is stable with the added help of a momentum wheel along principal Y axis (note that oscillations in ω_y are very small in magnitude, and that only ψ increases linearly while the other two Euler angles remain near 0° or 360°). This is incredibly useful given that our mission requirements require Y -axis slewing for Sun-pointing, and maintaining stable spin about this axis can help to prevent tumbling.

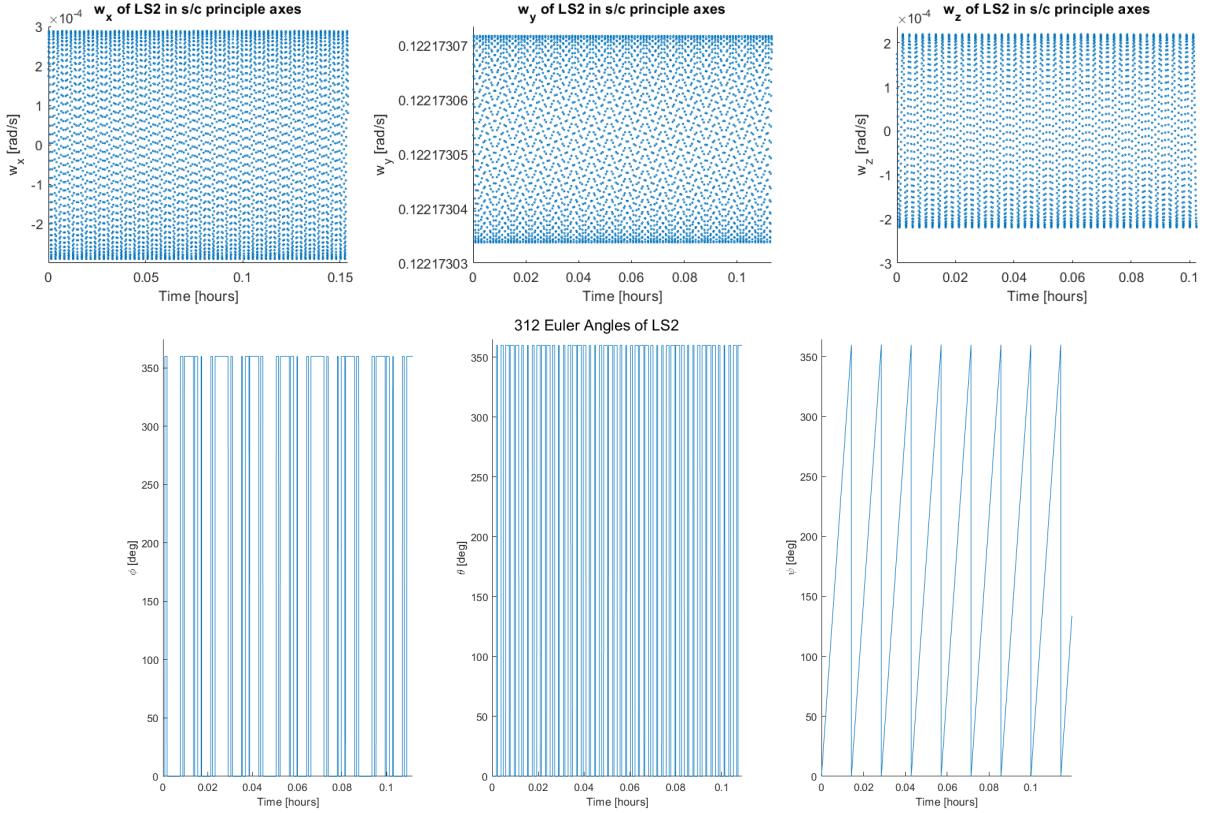


Figure 23: Angular velocity components and 312 Euler angles for intermediate axis spin, with momentum wheel for stability.

We can additionally attempt to ensure stability about the principal Z axis without spinning the spacecraft about the Z axis. This would be useful directly before Y axis slew maneuvers, allowing a precise understanding of how momentum wheel changes will change the inertial alignment of the Z axis. Giving the spacecraft an initial angular velocity of 0.01 deg/s about each principal axis, we put the momentum wheel axis along Z and spin it to 100 rad/s (with such a small Z spin, just about any non-trivial momentum wheel spin will suffice to ensure stability). The results of this trial can be seen in Fig 24 - note that just as in our stability test with spin primarily along the Z axis, we see periodic stability of the spacecraft angular velocity but without significant Z spin!

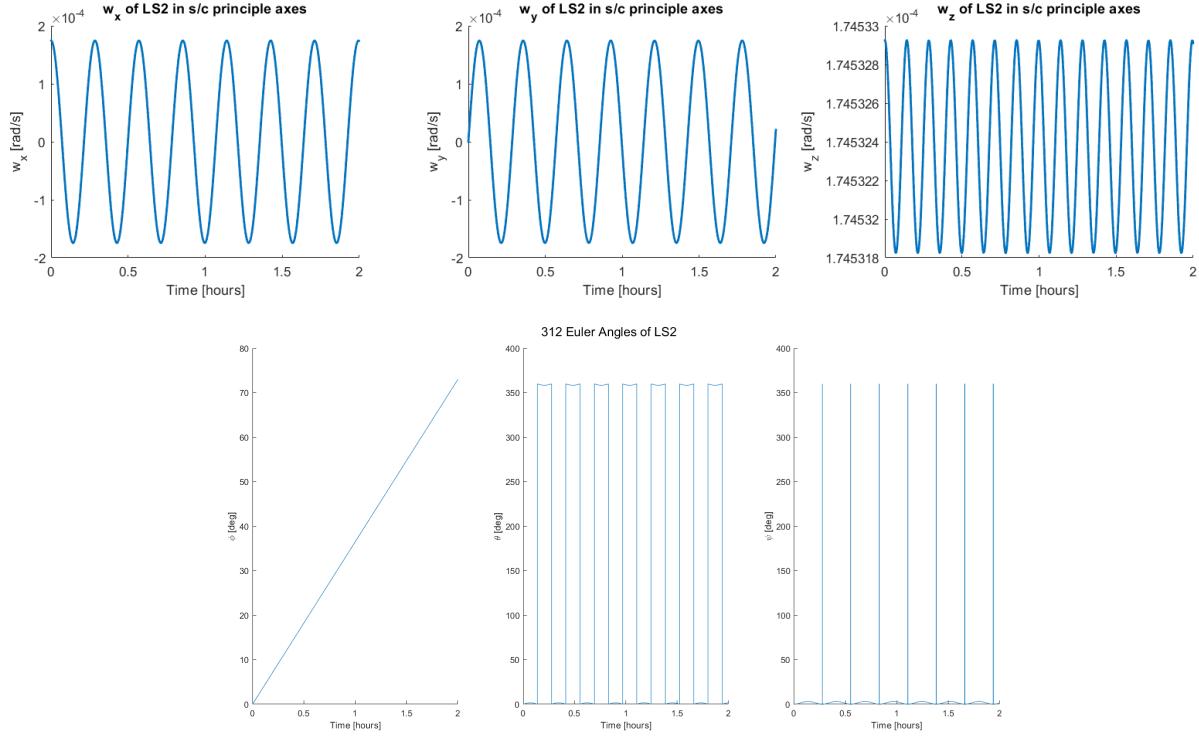


Figure 24: Angular velocity components and 312 Euler angles for low spin, with momentum wheel for Z stability.

4.2 GRAVITY GRADIENT TORQUE

The spacecraft torque due to a gravity gradient is modeled using Eq. 4 (included within the "Add environmental perturbations" block as seen in Fig 22).

$$\vec{M} = \frac{3GM}{R^3} \begin{bmatrix} (I_z - I_y)c_y c_z \\ (I_x - I_z)c_z c_x \\ (I_y - I_x)c_x c_y \end{bmatrix} \quad (4)$$

Using $a_0 = 7080.6$ km in an arbitrary direction not aligned with the RTN frame, a back-of-the-envelope calculation yields gravity gradient torques on the order of 10^{-6} in Y , Y and 10^{-9} in Z . When simulated with the same initial conditions as in Section 3.2, we see torques that match this calculation (Fig. 26).

To assess gravity gradient stability, we plot the spacecraft's moment of inertia coefficients K_i against regions of stable (white) and unstable gravity gradient torque in Fig. 25. LS2 can be oriented such that it is just barely within the stable region (top right corner). However, because of its proximity to the stability limit, small perturbations are enough to create a gravity gradient instability. The stability of the various configurations is assessed in Appendix H. Further, because the target attitudes are with respect to the sun and not Earth-pointing, the satellite will not often be in a stable configuration, so the gravity gradient torque must be actively compensated for.

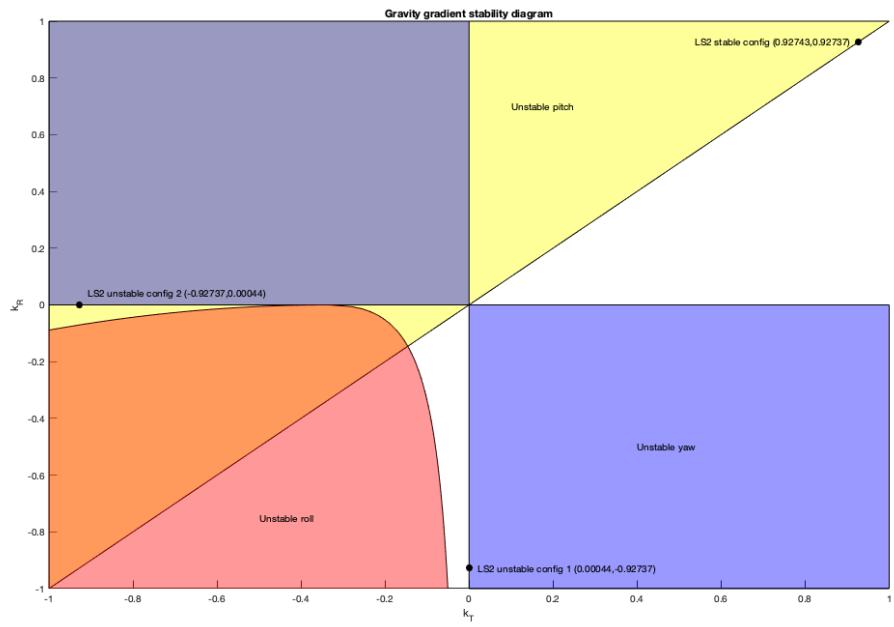


Figure 25: Assessment of spacecraft configurations that lead to stable and unstable gravity gradient torques based on LS2's mass properties.

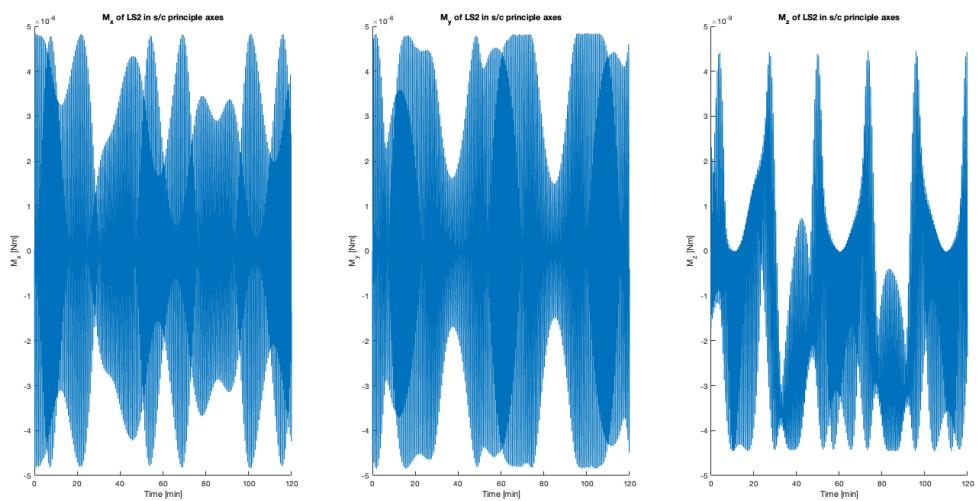


Figure 26: Torques on spacecraft principal axes due to gravity gradient.

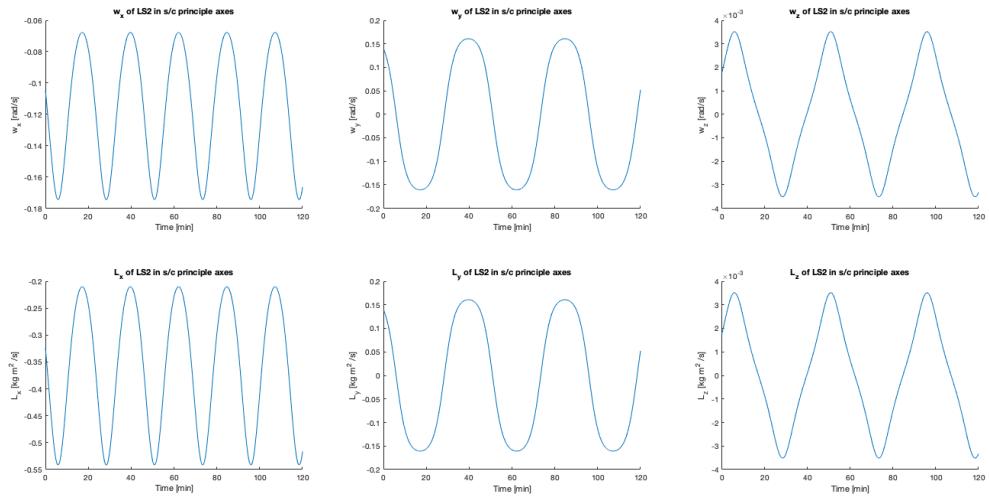


Figure 27: Angular velocity, momentum spacecraft principal axes due to gravity gradient.

5 PROBLEM SET 5

5.1 PERTURBATIONS

To include further environmental attitude disturbances, we implement torques as a result of solar radiation pressure (SRP), aerodynamic drag, and magnetic torque. The output of all environmental perturbation blocks is summed into a perturbation torque, which is subsequently fed into the Euler equations.

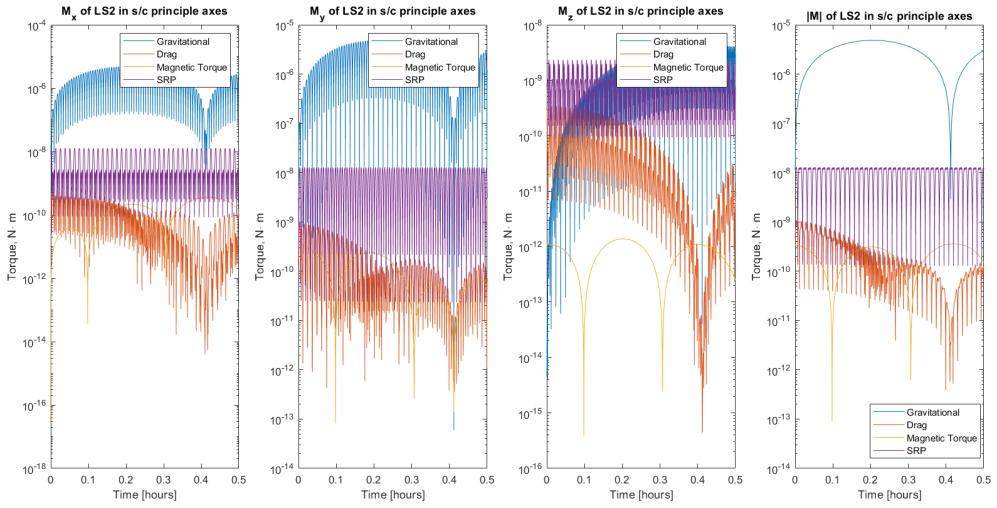


Figure 28: Comparison of disturbance torque magnitudes.

In Fig 28, we plot the logarithm of the torque magnitudes for relative comparison and validation using the conservative estimates from Chapter 11 of Larson & Wertz's Space Mission Analysis and Design (SMAD) [7]. The computed conservative estimates from Wertz's analysis are provided below in Table 7. Note that while the gravity gradient and magnetic torque estimates are very close to our calculated values, our simulated values of SRP and aerodynamic drag are much lower than the estimated value from SMAD. After checking our simulation, it seems likely that this discrepancy is due to the fact that the majority of the area used in calculating the conservative estimates is contributed by the solar sail. However, the solar sail does not have a very large moment arm for most attitudes, greatly reducing the effective areas.

Disturbance	Max. Simulated Torque, $N \cdot m$	SMAD Estimate, $N \cdot m$
Aero. Drag	$1E - 9$	$1.4E - 6$
Gravitational	$4E - 6$	$4.2E - 6$
Magnetic	$2E - 10$	$1.9E - 10$
SRP	$2E - 8$	$2.9E - 5$

Table 7: Comparison of simulated torque values to conservative estimates from SMAD [7].

5.2 ATTITUDE TARGETING & CONTROL ERROR

To be able to quantify our attitude error, we must define a target attitude. In line with mission requirements, we want our orientation to provide SRP thrust when moving away from the Sun, and want to minimize SRP effects when moving towards the Sun. Furthermore, we set our body Y -axis to always lie normal to the orbital plane, in line with the orbital angular momentum

vector. When flying away from the Sun, placing our body $-Z$ -axis along the Sun-pointing vector projected into the orbital plane provides in-plane solar sail thrusting. When flying toward the sun, we place our body X -axis along the Sun-pointing vector projected into the orbital plane to minimize cross-sectional area normal to the Sun direction. With this full definition of our target attitude, we can quantify the attitude control error for later implementation in active attitude control.

We define the attitude error direction cosine matrix (DCM), which we call A_{err} as

$$A_{ECI \rightarrow princ,target} = A_{err} A_{ECI \rightarrow princ,actual} \quad (5)$$

where $A_{ECI \rightarrow princ,target}$ is the desired rotation matrix to achieve target attitude and $A_{ECI \rightarrow princ,actual}$ is the actual rotation matrix of the satellite. We can solve this equation to see that

$$A_{err} = A_{ECI \rightarrow princ,target} A_{ECI \rightarrow princ,actual}^T$$

The resulting A_{err} matrix maps a vector in principal coordinates from its actual position to the desired, target position.

We verify the functionality of our attitude control system by turning off perturbations (both orbital and attitude) and setting our desired orientation to be identical to principal coordinates ($A_{ECI \rightarrow princ,target} = I$). Initially, our satellite's principal axes are aligned with the inertial frame and the spacecraft spins at 5 deg/s about its principal Z axis. A comparison of the resulting actual Euler angles and corresponding target and error angles is provided in Fig 29. Note that ϕ attitude and error angles are exactly opposite, as expected (the inverse rotation would need to be applied to reach the target, $\phi = 0$).

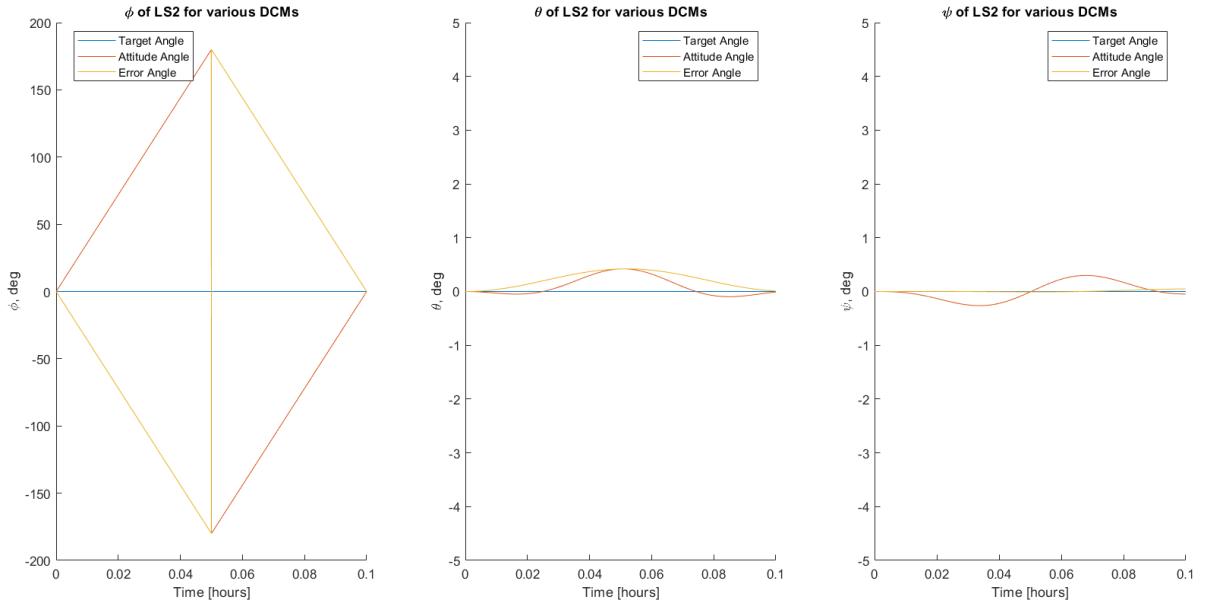


Figure 29: Euler angles, target, and error for target attitude identical to inertial frame, Z axis spin.

If we instead turn attitude perturbations on, keeping our target attitude to be the inertial reference system but with the satellite initially at rest, our resulting angular errors are purely a result of perturbative torques. The error angles provided in 30 represent the necessary rotation to take the satellite from its current attitude to the desired attitude.

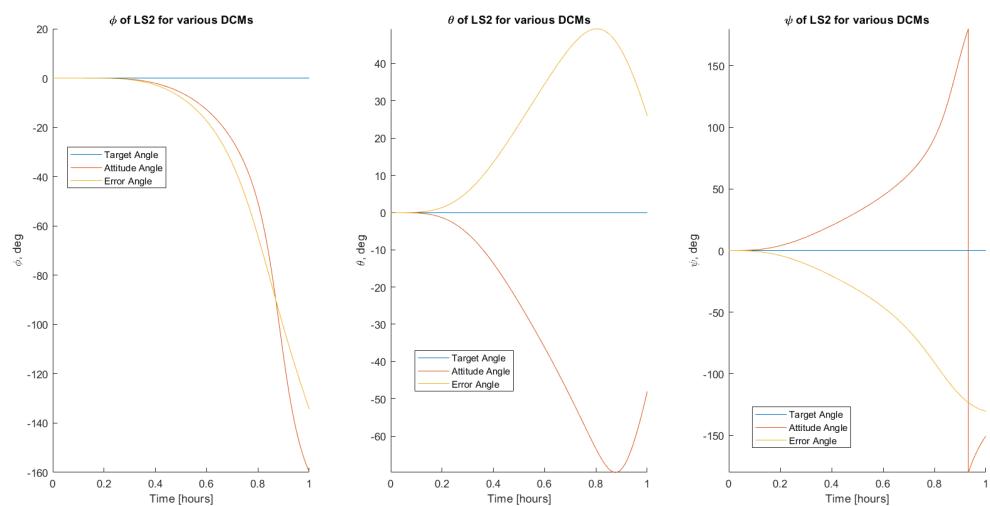


Figure 30: Euler angles, target, and error for target attitude identical to inertial frame, no initial spin with environmental perturbations.

6 PROBLEM SET 6

6.1 ATTITUDE DETERMINATION FROM SENSOR DATA

For our spacecraft, as discussed previously, we assume two sets of sensors that each produce a single vector measurement. Aboard LS2, a collection of 5 sun sensors measure the azimuth and elevation of sunlight in their respective fields of view. At any time, two or three sun sensors are able to generate measurements, and their data are transformed into the S/C body axes and fused to calculate an average Sun direction vector. Additionally, three orthogonal magnetometers provide a measurement of the magnetic field in the spacecraft body frame. Furthermore, a set of gyroscopic sensors provide measurements of angular rotation rates. In our simulation, we use the "ground truth" given by the propagator developed in previous problem sets to create measurements - in the next section, we discuss how we add random noise to these measurements to make the system realistic.

Before implementing sensor noise, however, we develop our attitude estimation tools to ensure functionality - by comparing the results of our attitude estimation with the ground truth, including zero sensor noise, we can verify that the spacecraft's estimation matches the propagator truth. For a set of measurements in the principal frame, \vec{M} , and a set of corresponding modeled vectors in ECI, \vec{V} , we implemented three attitude determination methods:

- **Deterministic method** - the estimated attitude is calculated directly from the measurements, $A = MV^{-1}$.
- **q-method** - the estimated attitude is determined using the q -method, which minimizes the least-squares error between a measurement and its corresponding, rotated model vector. Though this method allows sensor weighting, we give all measurements in \vec{M} equal weighting.
- **Angular velocity kinematics** - the estimated attitude is integrated using quaternion kinematics and gyroscopic measurements of the current angular velocity.

Our \vec{M} and \vec{V} matrices are generated using our measured Sun vector in the principal axes, \vec{S}_{meas} , the measured magnetic field vector, \vec{B}_{meas} , and their corresponding model vectors in ECI (taken from the ground truth), \vec{S}_{ECI} and \vec{B}_{ECI} . Given that the magnetometer readings have lower error, we generate our 3x3 matrices as

$$\vec{M} = \begin{bmatrix} | & | & | \\ \vec{p}_{meas} & \vec{q}_{meas} & \vec{r}_{meas} \\ | & | & | \end{bmatrix}$$

and

$$\vec{V} = \begin{bmatrix} | & | & | \\ \vec{p}_{ECI} & \vec{q}_{ECI} & \vec{r}_{ECI} \\ | & | & | \end{bmatrix}$$

where $\vec{p} = \vec{B}$, $\vec{q} = \frac{\vec{p} \times \vec{S}}{|\vec{p} \times \vec{S}|}$, and $\vec{r} = \vec{p} \times \vec{q}$.

In Fig 31, we see the q -method and the integrated angular velocity attitude estimations compared to the ground truth in the absence of sensor error. As expected, they agree very closely (note the minor deviation seen in the q -method is likely due to a near-singular matrix, resulting in larger errors in the computation of eigenvalues). The deterministic method, though not included in this plot, concurs exactly with the q -method in the absence of sensor error.

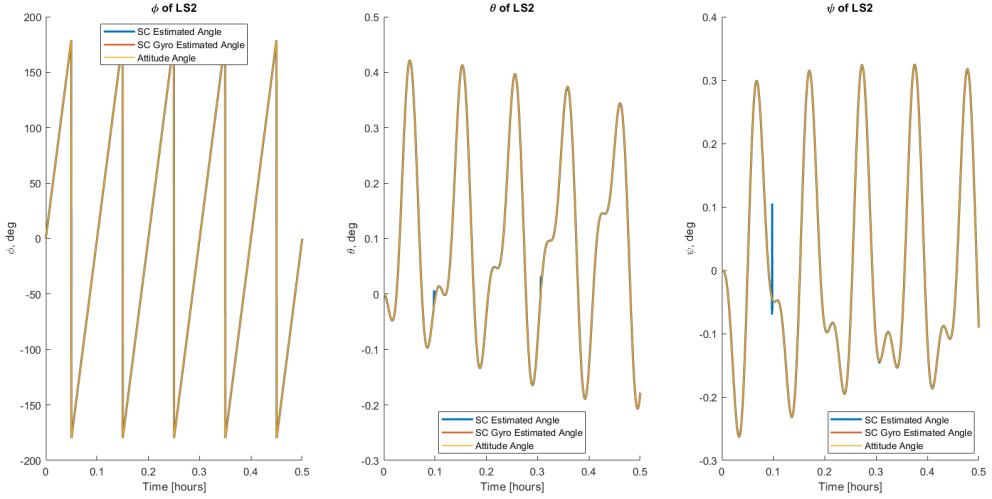


Figure 31: Comparison of q -method and gyroscope attitude estimates without sensor noise.

6.2 SENSOR NOISE

Sensor noise, based on the values in Table 2, was modelled by adding Gaussian white noise to each sensor measurement vector. For the sun sensors, which measure sun direction in azimuth and elevation, independent noise components are added to the true azimuth and elevation, then transformed to Cartesian for the reported sun direction measurement. For the magnetometer, the noise components in each axis are used to construct a 3-1-2 Euler angle rotation matrix that is then applied to the magnetic field direction vector.

The variance of the noise is calculated from the reported errors, e_{max} as $\sigma^2 = \frac{e_{max}}{\sqrt{3}}$, so that $> 99\%$ of errors fall within e_{max} . Further, a small, constant bias is added, $\approx e_{max}/10$ on each axis. The noise is seeded to enable repeatable simulations.

In Fig 32, we see the comparison of our q -method and deterministic attitude estimates with the ground truth after the addition of sensor noise. Fig 33 shows the error rotation between the q -method and the true estimate (note that the error for the deterministic method is very close to that of the q -method). The results show noisy measurements that oscillate ± 2 deg around the true attitude. In Fig 34, we see the comparison of the angular velocity estimate to the ground truth - note that the estimated attitude from gyroscopic measurements deviates severely from the true attitude. This is expected, as the integration of error accumulates as the simulation continues.

Additional analysis of deterministic attitude estimate and its comparison to the q -method are provided in Appendix I, including an analysis of the effect of "error distribution" techniques that seek to reduce error effects by averaging measurements.

6.3 SMALL SENSOR ERRORS

The sensor error DCM represents a rotation of sensor measurements from their "true" measurement due to sensor noise and bias. Because of this error, the attitude estimate will be rotated slightly from the true attitude. We define it as the rotation from the actual orientation to the believed orientation, so that:

$$A_{ECI \rightarrow princ,est} = A_{sens,err} A_{ECI \rightarrow princ,actual} \quad (6)$$

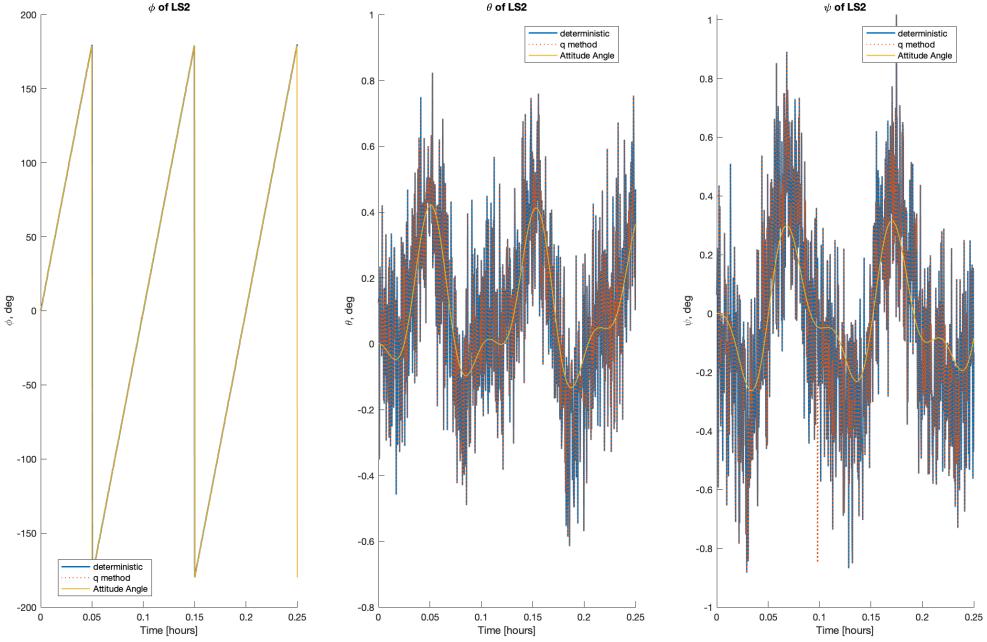


Figure 32: Comparison of q -method and deterministic attitude estimates with added sensor noise.

For an error DCM with small values, we note that this corresponds to a small rotation. For our 312 Euler angles, a small rotation error would produce an error DCM that looks like:

$$A \approx \begin{bmatrix} 1 & \phi & -\psi \\ -\phi & 1 & \theta \\ \psi & -\theta & 1 \end{bmatrix} \quad (7)$$

Clearly, given small ψ , θ , and ϕ , this matrix is nearly equivalent to the identity matrix, with the off-diagonal elements indicating the magnitude of the error rotation angles.

For a quaternion, which can be expressed in terms of 312 Euler angles as

$$q = \begin{bmatrix} \cos(\phi/2) \sin(\theta/2) \cos(\psi/2) + \sin(\phi/2) \cos(\theta/2) \sin(\psi/2) \\ \cos(\phi/2) \cos(\theta/2) \sin(\psi/2) - \sin(\phi/2) \sin(\theta/2) \cos(\psi/2) \\ -\cos(\phi/2) \sin(\theta/2) \sin(\psi/2) + \sin(\phi/2) \cos(\theta/2) \cos(\psi/2) \\ \cos(\phi/2) \cos(\theta/2) \cos(\psi/2) + \sin(\phi/2) \sin(\theta/2) \sin(\psi/2) \end{bmatrix}, \quad (8)$$

we note that the small angle approximation (throwing away terms with more than one angle) gives

$$q \approx \begin{bmatrix} \theta/2 \\ \psi/2 \\ \phi/2 \\ 1 \end{bmatrix}, \quad (9)$$

Thus, the vector component of the quaternion is directly proportional to Euler angles. Therefore, we can consider the off-diagonal elements of our error DCM or, alternatively, the vector component of our quaternion output (from our q -method, for example) to quantify the small Euler angles that rotate our estimated frame to the actual attitude.

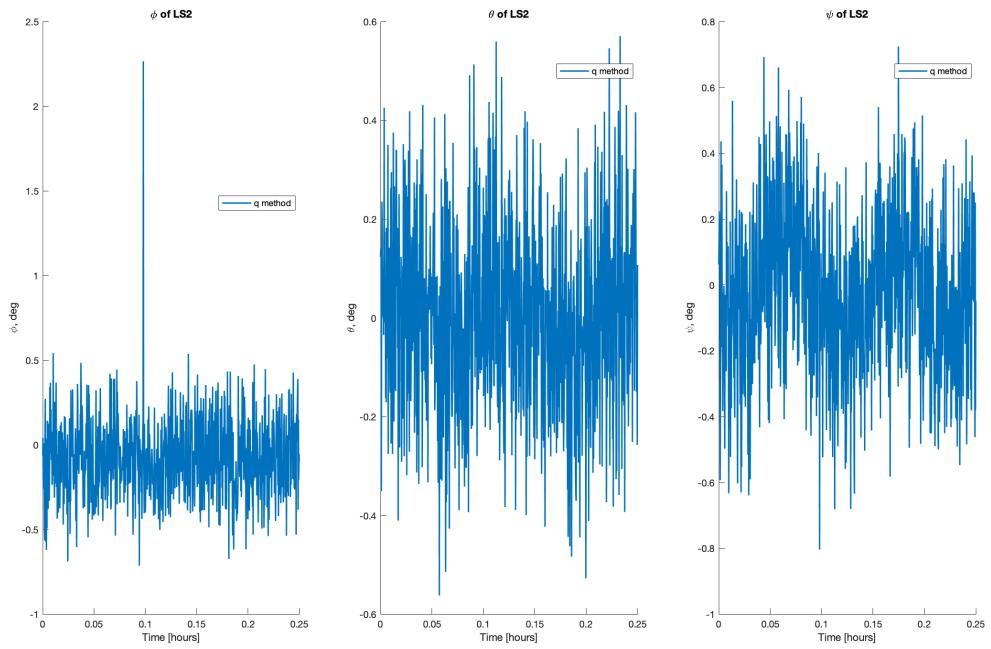


Figure 33: Error of q -method with respect to the true attitude.

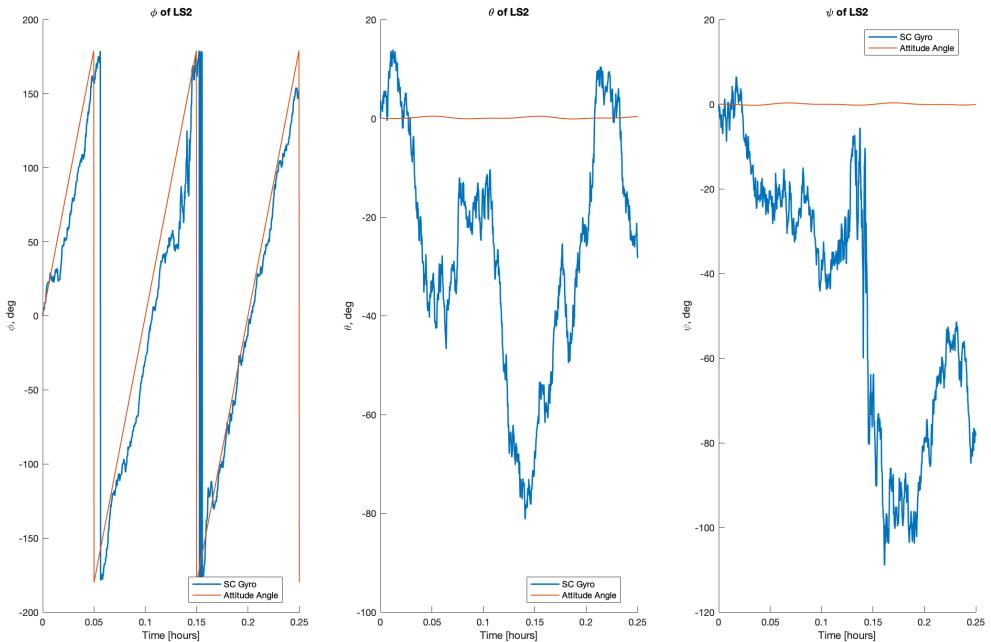


Figure 34: Integrated gyroscope estimation with sensor noise.

6.4 ESTIMATING ATTITUDE CONTROL ERROR

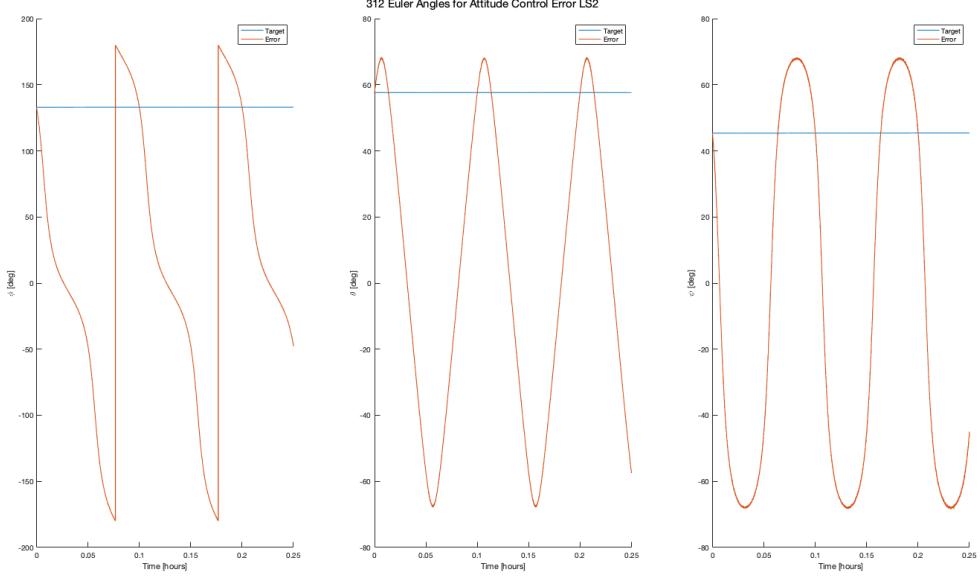


Figure 35: Attitude control error using estimated attitude from noisy data (using q-method).

Similar to how we have defined the target error in Section 5, this estimated target error represents the rotation (in principal axes) from the estimated spacecraft attitude to the desired, target attitude. Instead of $A_{ECI \rightarrow princ,target} = A_{err} A_{ECI \rightarrow princ,actual}$, we have

$$A_{ECI \rightarrow princ,target} = A_{err,est} A_{ECI \rightarrow princ,est}$$

and thus

$$A_{err,est} = A_{ECI \rightarrow princ,target} A_{ECI \rightarrow princ,est}^T$$

If $A_{ECI \rightarrow princ,est} = A_{sens,err} A_{ECI \rightarrow princ,actual}$, then

$$A_{err,est} = A_{ECI \rightarrow princ,target} (A_{sens,err} A_{ECI \rightarrow princ,actual})^T = A_{ECI \rightarrow princ,target} A_{ECI \rightarrow princ,actual}^T A_{sens,err}^T$$

We have plotted our resulting attitude control error in 35, with sensor noise included - note that the sensor noise is very small, and noise is barely visible on macroscopic contours of the attitude curves.

7 PROBLEM SET 7

7.1 UPDATED SENSOR MODELS

To further increase model fidelity, the magnetometer model was updated to add the independent noise components to each axis, rather than creating a rotation matrix, which was unnecessarily complex. The sun sensor noise, added to the azimuth and elevation of the sun direction vector, was determined to be fairly precisely modelled, and thus was not changed. In the previous section, once noise was added to the sun sensor, the measurement was transformed to Cartesian for deterministic and q -method attitude estimation. However, for the Extended Kalman Filter in this section, the sun sensor measurements are used directly in azimuth and elevation.

7.2 EXTENDED KALMAN FILTER IMPLEMENTATION

To improve our estimate of current state aboard the spacecraft, we implement an Extended Kalman Filter (EKF) to estimate key attitude state parameters. A full derivation of the associated non-linear functions and their linearizations for implementation is available in Appendix J. Note that we offer two formulations in this derivation: a state representation including sensor bias estimates and another without sensor biases. The latter is simply a subset of the former, and represents a very similar system. Based on the modification of our sensors discussed in the previous section, we define our measurement error matrix, R , to simply be the diagonal matrix with our associated sensor covariances along the diagonal (these same covariances are used to generate the Gaussian sensor noise in our simulation).

$$R = \begin{bmatrix} 7.6 \times 10^{-3} \mathbf{I}_3 \text{ rad}^2/\text{sec}^2 & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 2} \\ \mathbf{0}_{3 \times 3} & 2.5 \times 10^{-13} \mathbf{I}_3 \text{ Tesla}^2 & \mathbf{0}_{3 \times 2} \\ \mathbf{0}_{2 \times 3} & \mathbf{0}_{2 \times 3} & 6.854 \times 10^{-4} \mathbf{I}_{2 \times 2} \text{ rad}^2 \end{bmatrix} \quad (10)$$

We also add process noise to our system, such that the process noise matrix Q is a diagonal matrix whose diagonal components are equal to $0.1\delta t$, for a simulation time step of δt (in our case, we use 0.1 seconds).

Note in our derivation that we implement the full non-linear EKF, not the linearized Kalman Filter associated with our system. This means that when possible, we use the full non-linear functions to assess our expected measurements and state predictions. We test our EKF using simulation data identical to that from the last problem set - with attitude perturbations turned off, we initially align the spacecraft with the principal axes and spin it about the Z axis at 3 deg/s. Our initial state estimate is somewhat poor, estimating our initial quaternion as $(0, 0.5, 0, 0.866)^T$ compared to the true value $(0, 0, 0, 1)^T$ and our initial angular velocity as $(0.1, 0.1, 2.5)^T$ deg/s compared to the true value of $(0, 0, 3)^T$ deg/s (bias estimates, when included in the analysis below, are initially estimated to be zero). Our initial covariance estimate, P , is accordingly taken to be somewhat large, initialized as a diagonal matrix with a value of 4 along the diagonal (giving a 95% confidence interval of $\approx \pm 4$).

7.3 EKF WITH SENSOR BIAS

After implementing the formulation of the EKF with sensor bias included in the state estimation, our tests showed some rather severe issues. While angular velocity measurements behave in a fairly expected manner, the estimates of bias in the azimuth and elevation measurements of our Sun sensor run away. This causes severe issues in our attitude estimation. These results are provided in Figures 36-40. At this moment, we suspect that the implementation may be fundamentally flawed due to the fact the system is likely not totally observable - if the observation matrix of the system at each time step is not full rank, the EKF is not well suited to predicting the state given new measurements. Because of the size of the state ($N=15$), the observability matrix is quite complex and difficult to analyze, we have not been able to verify whether or not that is the case. A further question is whether the linearization of the measurement function lost some crucial component of the sensor model logic (such as angle wrapping from 0 to 2π). We will continue to work to debug this system, in case the error is in our implementation, but for the moment we move on to an implementation without sensor bias estimation.

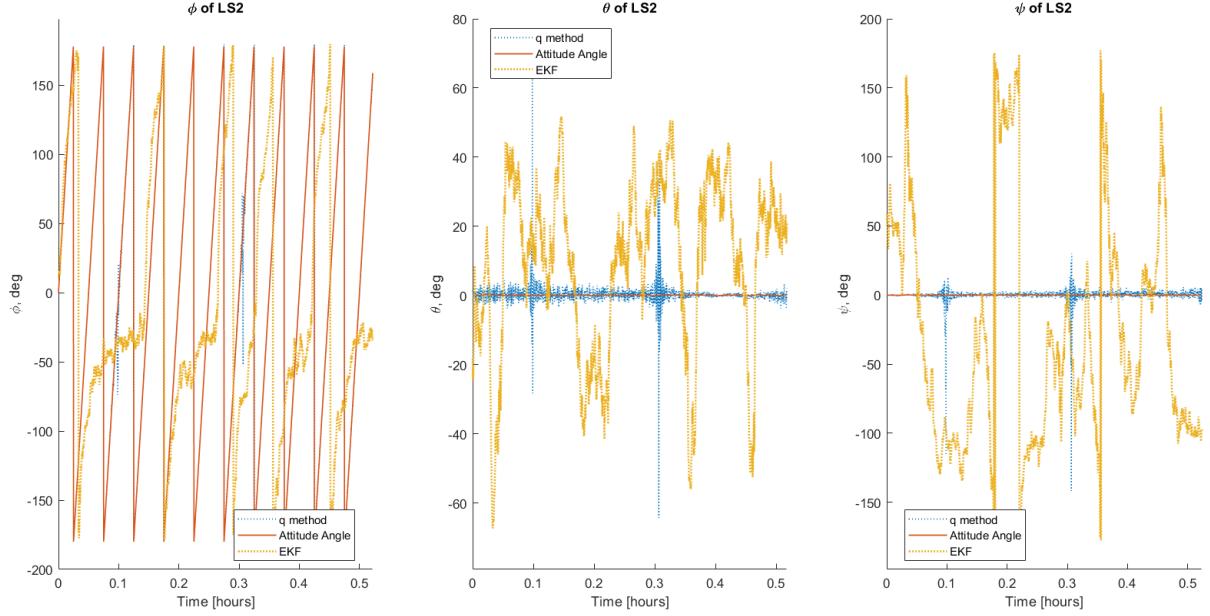


Figure 36: Attitude estimate from EKF with sensor bias estimation

7.4 EKF WITHOUT SENSOR BIAS

The implementation of the EKF without sensor bias included in the estimation was far more fruitful. The resulting estimates are provided below in Figures 41 and 42. A discussion of the error, residual, and covariance results is provided in Appendix K. In general, note that the performance of the EKF is noticeably better than the q -method after a short initial period of attitude uncertainty, in addition to providing an estimate of error at each time step.

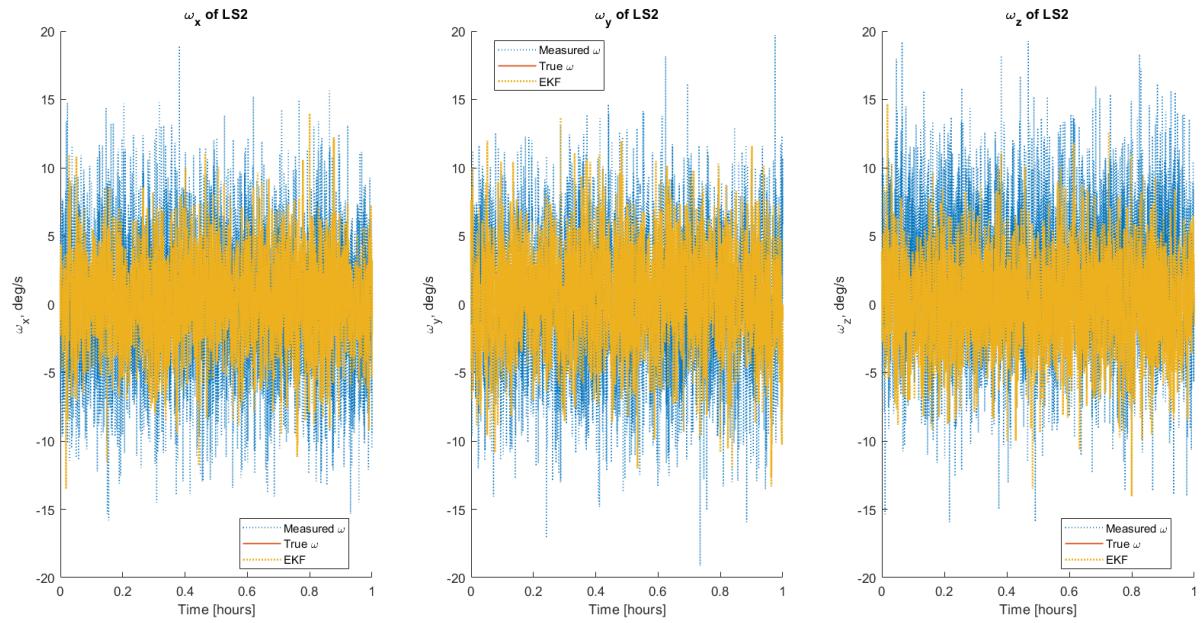


Figure 37: Angular velocity estimate from EKF with sensor bias estimation

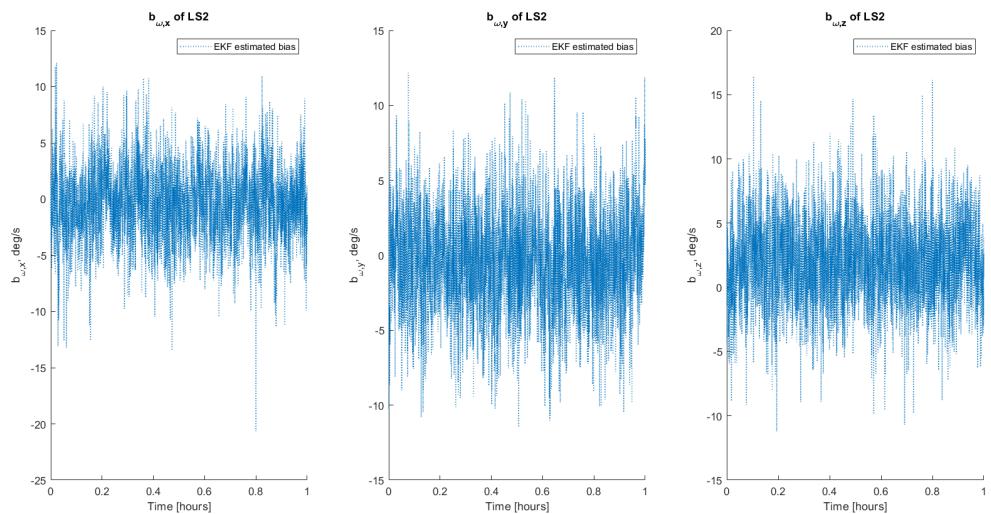


Figure 38: Angular velocity bias estimate from EKF

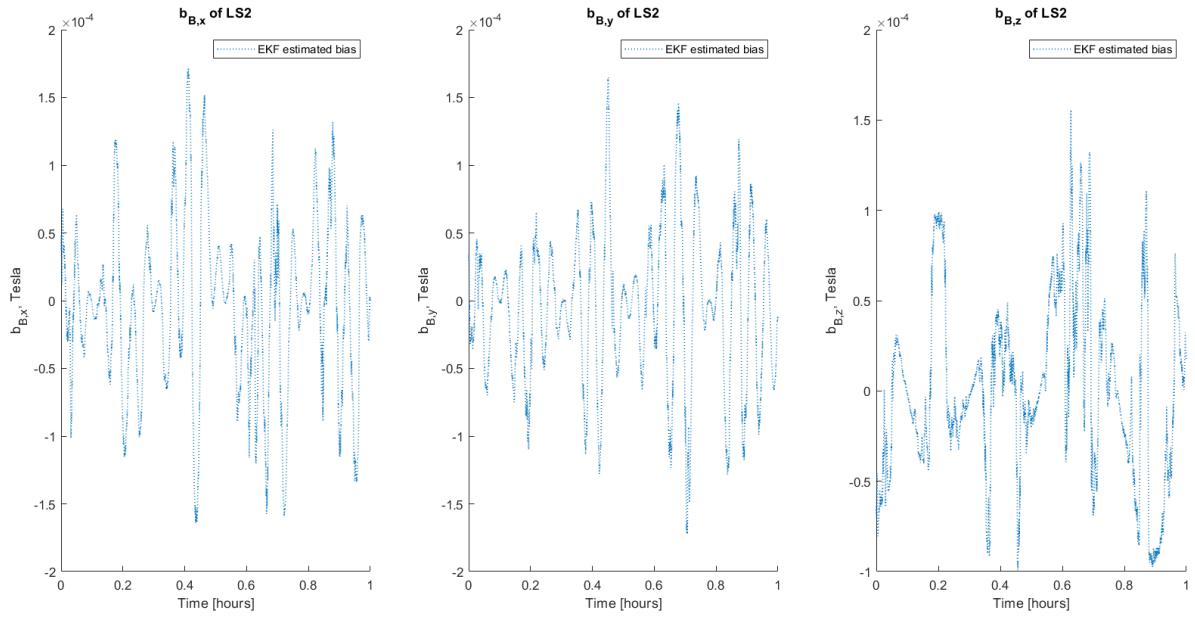


Figure 39: Magnetic sensor bias estimate from EKF

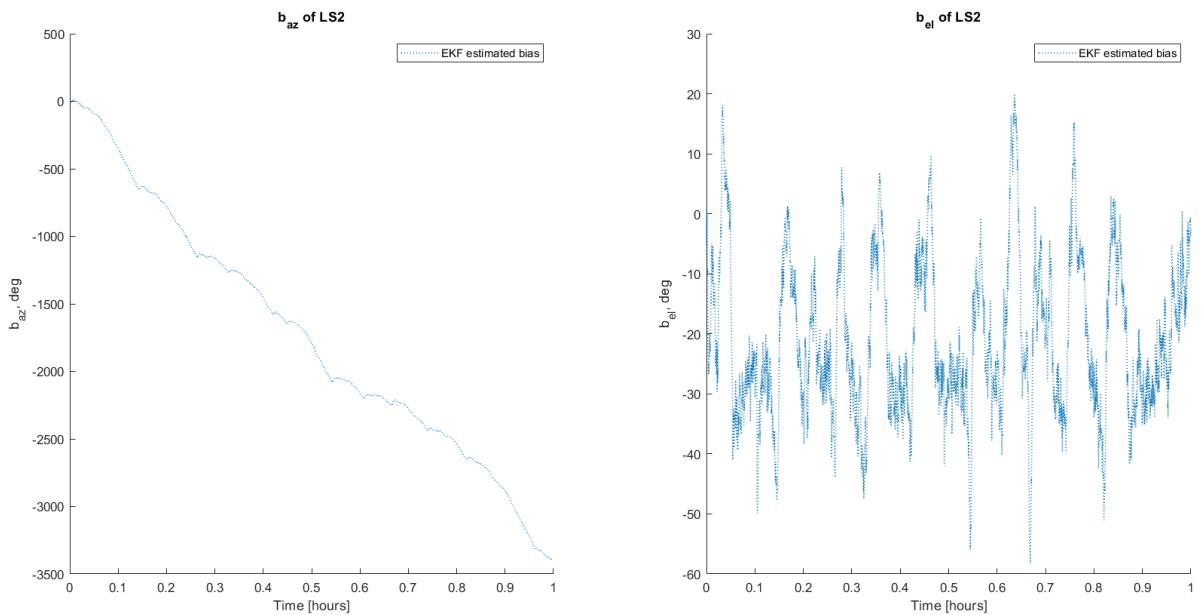


Figure 40: Sun Sensor azimuth and elevation bias estimate from EKF with major issues

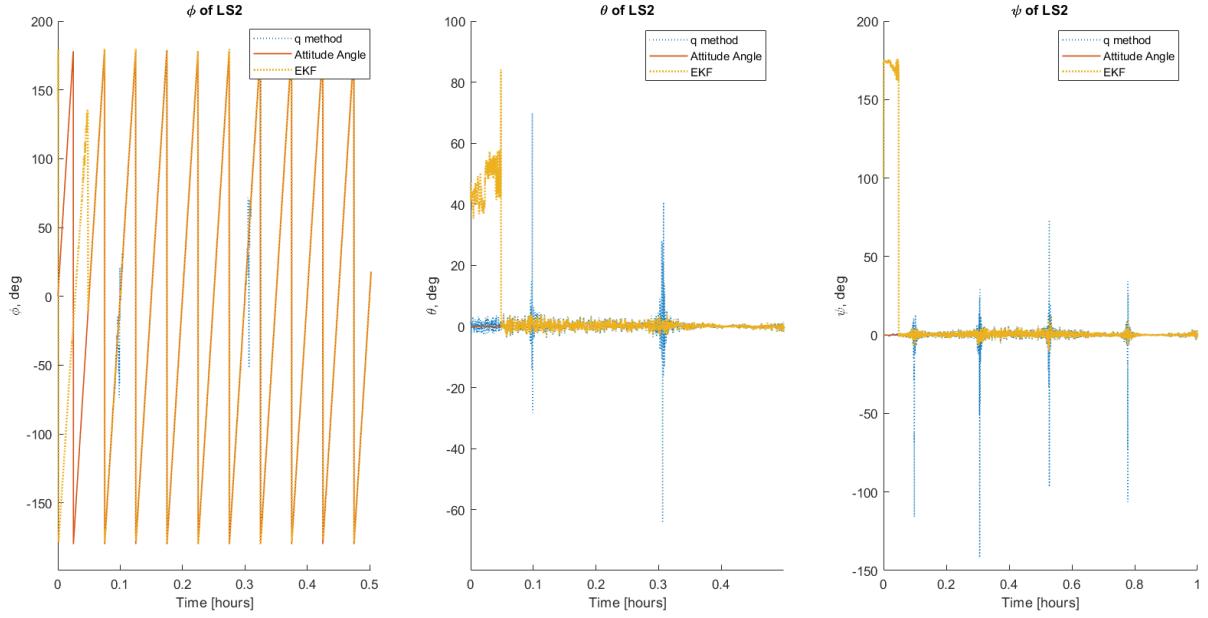


Figure 41: Attitude estimate from EKF without sensor bias estimation

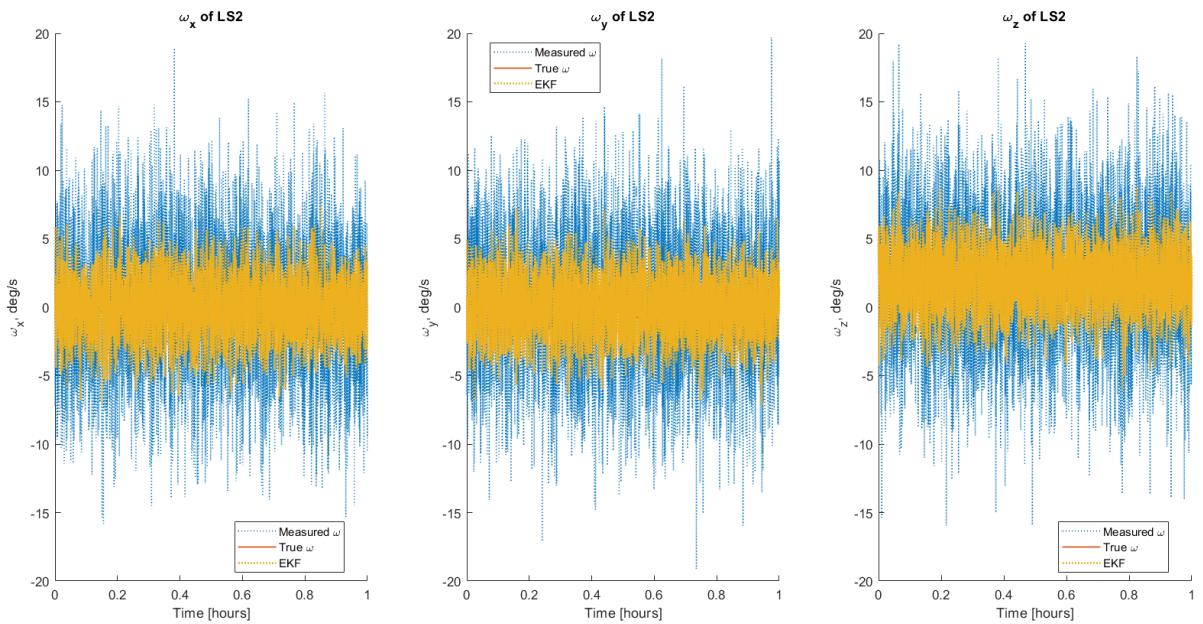


Figure 42: Angular velocity estimate from EKF without sensor bias estimation

8 PROBLEM SET 8

The simplified EKF methods applied in the previous problem set will continue to be used in the following sections - while it lacks estimation of sensor bias and attitude control states (such as momentum wheel spin rate), it provides a realistic state estimation device to base control methods on. Early issues implementing control laws on top of the EKF (issues in control law would cause the filter to diverge, making debugging difficult) inspired us to use q -method estimation to develop the systems discussed below. However, we will present results from EKF state estimation after development, with some success.

While not implemented yet, we hope to add estimation of momentum wheel state into our EKF to more accurately describe the spacecraft's knowledge of the control systems. With our sun sensor output already in azimuth/elevation form and component-wise magnetic field measurements, we feel that our measurement models are sufficiently realistic to provide practical simulations of LS2.

8.1 ATTITUDE CONTROL SYSTEM SIZING & SELECTION

As a small, low-mass cubesat, LS2 has the fairly standard actuator pair of 3 orthogonal magnetorquers and a single momentum wheel. As a solar sailing proof-of-concept mission, thrusters were explicitly not used, and control moment gyros were too costly, heavy, and complex. Additionally, in other solar sailing missions, control torques have been generated using SRP [13]. On these missions, more complex sails had color-changing liquid crystals that regulated the reflectivity of the surface across the sail. By having a darker side and a lighter side of the sail, a pressure difference can be created on the sail. However, LS2 did not have this feature.

Parameter	Simplified Eq.	Result	Note
Torque needed for disturbance rejection	$M_{c,mag} = M_d\epsilon$	1.5×10^{-5} Nm	Max disturbance 10^{-5} from gravity gradient
Moment from magnetorquers for disturbance rejection	$D = M_{c,mag}/B$	$0.15 \text{ A}\cdot\text{m}^2$	$B_{min} = 1 \times 10^{-4}$ T
Moment from magnetorquers for momentum wheel off-slew axis compensation	$D = M_{c,w}/B$	$50 \text{ A}\cdot\text{m}^2$	Less than available on LS2.
Slew torque for momentum wheel	$M_{c,w} = 4I\theta/t^2$	$0.005 \text{ N}\cdot\text{m}$	Much greater than available on LS2.
Momentum storage in momentum wheel	$L = M_d P^{\frac{0.707}{4}}$	$0.01 \text{ kg}\cdot\text{m}^2/\text{s}$	$I_{rot} = 10^{-4} \text{ kg}\cdot\text{m}^2 \rightarrow \omega_{max} = 10^2 \text{ rad/s}$

Table 8: Attitude actuator sizing, as presented in class.

To determine the sizing of the actuators, we use the results summarized in Lecture 14. We choose a safety margin of $\varepsilon = 1.5$ and use the maximum principal moment of inertia, $I \approx 6 \text{ kg}\cdot\text{m}^2$. Further, the max anticipated disturbance torque is $M_d = 10^{-5}$ Nm and is from gravity gradient acting on the high moments of the solar sail (see Fig 28). The slew maneuvers, from sun-pointing to sun-perpendicular, will be 90 deg, with a goal slew time of 1/70th of the orbital period. For our orbit, with an orbital period of 5939s, (approximately 1.6 hrs), this yields $t = 85$ s. These results are presented in Table 8.

Component	Number	LS2 Max Torque	Sim Max Torque
Torque Rods	3	$1 \text{ Am}^2 \times \vec{\mathbf{B}}$	$2000 \text{ Am}^2 \times \vec{\mathbf{B}}$
Momentum Wheel	1	0.06 Nm^2	0.6 Nm^2

Table 9: LS2 versus simulation actuators

As noted in Table 3, on the real spacecraft, each magnetorquer can generate a max moment of 1 Am^2 , while the momentum wheel can create a torque of up to 0.06 Nm^2 (with a max acceleration of 100 rad/s^2 and max speed of roughly 10^3 rad/s). Interestingly, the actual spacecraft has much more available torque in the momentum wheel than needed for slew maneuvers – although not enough from the magnetorquers for slew maneuvers as we’ve implemented (see control law section below). As a result, we loosened saturation limits of the magnetorquers to make testing the control system easier. In addition, we also increased the momentum wheel capability to allow the spacecraft to stabilize and slew faster: the momentum wheel can reach a max acceleration of 10^3 rad/s^2 and speed of 10^5 rad/s , corresponding to a max torque of 0.6 Nm^2 .

While in the future, we hope to re-impose the lower saturation limits, as of now, if the ADCS requests a torque from the actuators, but the magnetorquers are saturated before they cancel out the undesired off-axis effects of the momentum wheel, the system will control itself away from the target attitude and may oscillate around or deviate entirely from the target. Thus, very high saturation limits allow us to test out the functionality of these very simple control laws before considering the more complex, saturated system.

Further, on orbit, LS2 partially mitigated this problem by performing daily momentum dumps (over the course of two orbits), during which the magnetorquers were used to stabilize the spacecraft while the momentum wheels were allowed to spin down. This logic was not implemented in the simulation.

8.2 ACTUATOR UNIT TESTING

After determining desired control torque, the spacecraft must calculate the desired momentum wheel acceleration and magnetorquer dipole moment; random and systematic error are added to these outputs to represent realistic systems, and the former is integrated to calculate momentum wheel speed. These outputs are fed into the physics simulator, which uses the “true” magnetic field, “true” spacecraft orientation, and noisy inputs to calculate the resulting torque, which is nearly always different than the desired torque calculated by the ADCS system. To test the physical subsystems representing the momentum wheel, we place them in a separate model and perform a unit test with simple “truth” inputs: a constant magnetic field aligned with the inertial Z direction, a constant spacecraft angular velocity, and a spacecraft attitude that is held constant (initially aligned with the principal axes, the attitude rotates about the Y -axis by 90° halfway through the simulation to demonstrate the functionality of the B -field rotation). After feeding in a desired input, we add noise to that input and compare the resulting torque to the ADCS expected torque. These results are visible in Figures 43 and 44 - note the bias and Gaussian error applied to the input results in a “true” torque that is different than that requested by the ADCS.

8.3 STATE MACHINE

To handle different control situations, we define a state machine for our spacecraft, presented in Fig 45, to switch between three primary control modes: detumbling, stabilization, and slew-

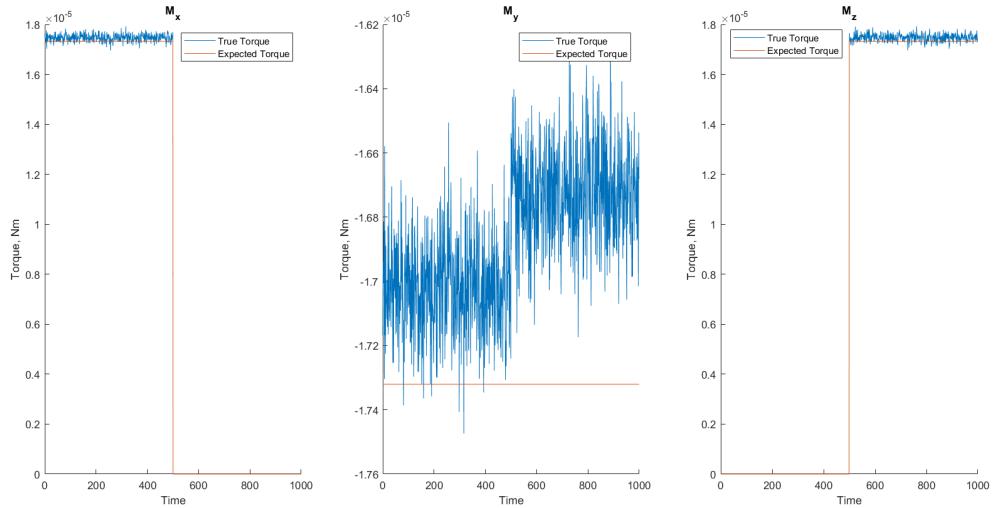


Figure 43: Magnetorquer output from attitude control unit testing

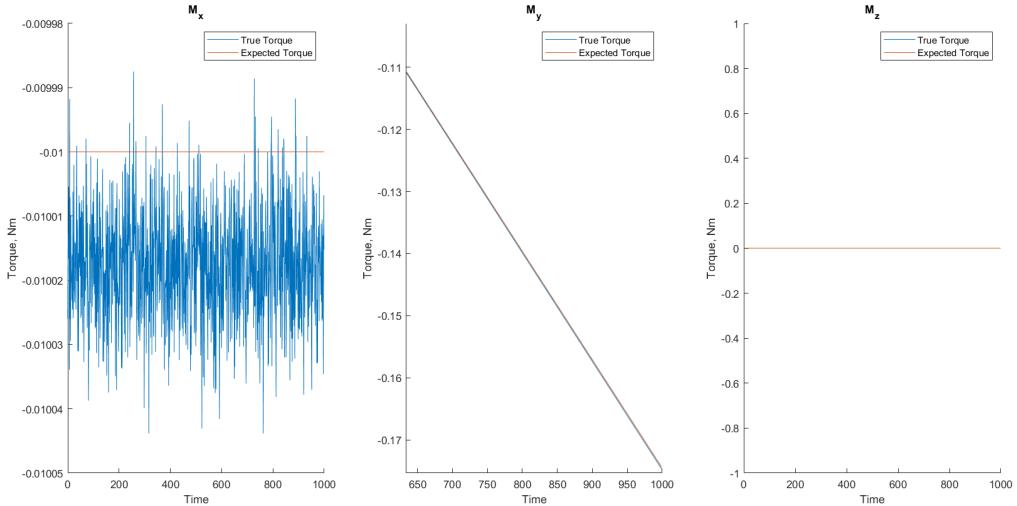


Figure 44: Momentum wheel output from attitude control unit testing

ing. Detumbling is the spacecraft "safe" mode, for high angular rotation rates - this mode uses B-dot control methods with onboard magnetorquers to stabilize the system around a single rotation axis. Stabilization mode is used for small-error 3-axis adjustments using the momentum wheel and 2 orthogonal magnetorquers. Slew mode uses the same attitude control systems, but implements a non-linear control method to handle large control errors - this mode is primarily intended for swinging the spacecraft between the "on" and "off" states for SRP thrusting, which occurs twice per orbit. The state machine was implemented in Simulink using Mathwork's StateFlow along with enumerated mode types. A more detailed description of each mode's control and estimation methods is provided below.

8.3.1 DETUMBLE MODE

Detumble mode, the spacecraft "safe" mode, takes over if the spacecraft's total angular velocity ever exceeds some "tumble limit" for longer than a specified period of time (called the

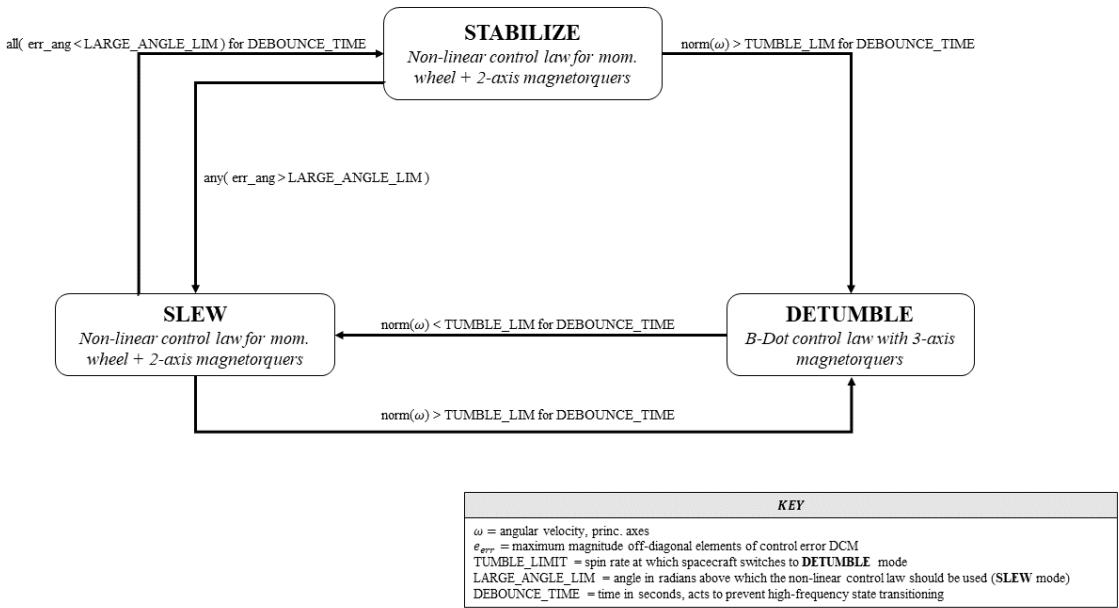


Figure 45: Spacecraft Mode State Machine Diagram

”debounce time”). This debounce protection prevents the spacecraft from switching into detumble mode when doing aggressive slew maneuvers, so long as those slew maneuvers don’t take longer than the debounce time. Currently, the spacecraft leaves detumble mode after the total angular velocity falls below the tumble limit - in the future, a better exit method would be checking if the spacecraft’s spin is stable about a single axis in the RTN frame (which is the stable result of the B-Dot control law).

In this mode, the spacecraft uses q -method to determine its state - this state estimation method was chosen due to its low computational requirements and inability to diverge in the way that EKFs do when experiencing high rate-of-change in the spacecraft state. The B-Dot control law is implemented using the magnetic field measurements from on-board sensors - the magnetorquer dipoles are set according to

$$\vec{m} = \vec{m}_{max} \text{sign}(\dot{\vec{B}}_{XYZ}) \quad (11)$$

where m_{max} is the maximum dipole moment able to be provided by the magnetorquers and $\dot{\vec{B}}_{xyz}$ is the time-derivative of the magnetic field in principal axes.

In later system testing, we primarily focus on the implementation of the linear control laws in the stabilization and slew modes. However, Figures 46 and 47 show attitude and angular velocity results from a detumble test with no sensor noise or attitude perturbations; after a short period, the satellite successfully stabilizes its spin about a single principal axis (with minor oscillations around the other two axes). This is clear in Fig 47, where the angular velocity along x and y approach zero, while ω_z approaches a stable spin rate.

8.3.2 STABILIZE MODE

Stabilize mode is intended to adjust minor attitude control errors using the momentum wheel (oriented along the body X axis) and two orthogonal magnetorquers to adjust the spacecraft

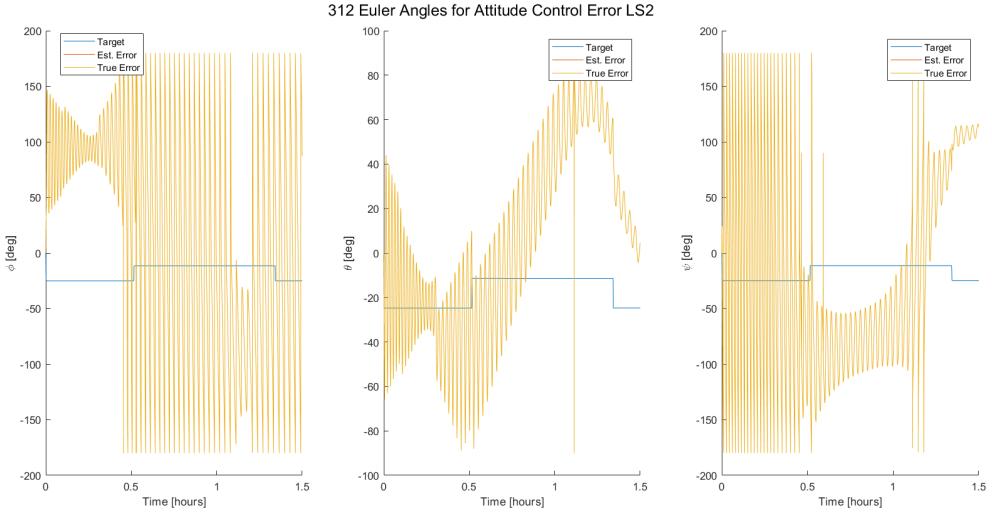


Figure 46: Attitude Results from B-Dot detumbling test

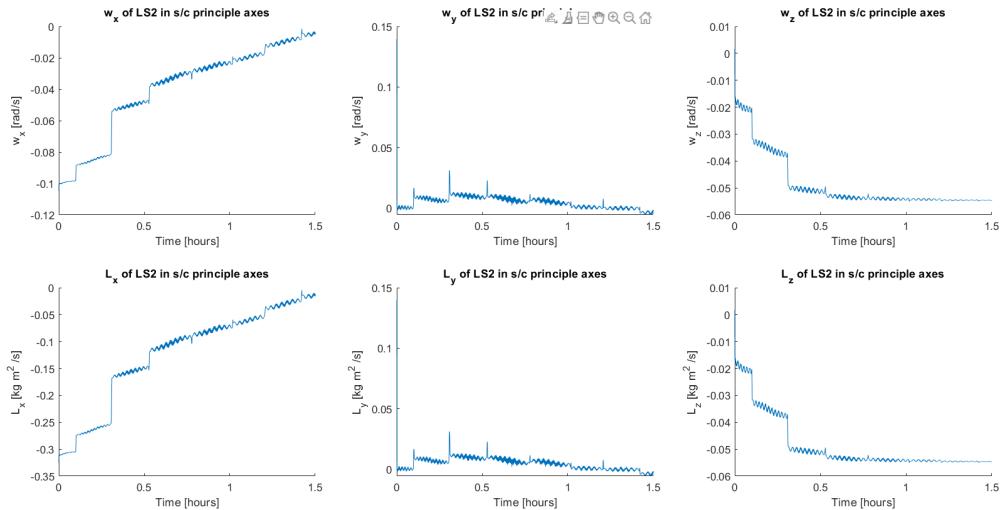


Figure 47: Angular Velocity Results from B-Dot detumbling test

orientation. This method uses the linearized attitude error matrix form

$$A_{err} \approx \begin{bmatrix} 1 & \alpha_z & -\alpha_y \\ -\alpha_z & 1 & \alpha_x \\ \alpha_y & -\alpha_z & 1 \end{bmatrix} \quad (12)$$

to compute the linearized control error, $\vec{\alpha} = (\alpha_x, \alpha_y, \alpha_z)^T$.

We then compute the brute-force control torque, M_{ccc} , using a PD control system,

$$M_{ccc} = I \ddot{\vec{\alpha}} = -k_d \dot{\vec{\alpha}} - k_p \vec{\alpha} \quad (13)$$

where $\dot{\vec{\alpha}}$ is simply $\vec{\omega}_{XYZ}$, the principal axes angular velocity, when the desired spin rate is zero. The selection of the proportional and derivative gains is discussed in detail below.

From the brute-force control torque, we calculate the corrected control torque, M_c , as

$$M_c = M_{ccc} + (\vec{\omega}_{XYZ} \times I \vec{\omega}_{XYZ}) + I_r \omega_r (\vec{\omega}_{XYZ} \times \vec{r}) \quad (14)$$

where I_r is the moment of the momentum wheel, ω_r is the wheel's speed, and \vec{r} is the wheel's orientation. From this value, we can calculate our desired magnetic dipole moments and momentum wheel acceleration from

$$\begin{bmatrix} -I_r\omega_r \\ m_y \\ m_z \end{bmatrix} = \begin{bmatrix} | & B_z & -B_y \\ \vec{r} & 0 & B_x \\ | & -B_x & 0 \end{bmatrix}^{-1} M_c \quad (15)$$

We take extra care to enforce the reality of our system by first converting the control torque, rotor orientation, and magnetic field to body frame and calculating the desired control tactic. Before returning the magnetic dipole moment to principal coordinates, we impose saturation limits (to respect the true capabilities of the magnetorquer system).

The stabilization mode for initial testing uses the same q -method approach as the detumble mode - however, for full implementation, we replace this with our EKF formulation to provide improved state estimation and error bound predictions.

8.3.3 SLEW MODE

Slew mode is very similar in its control tactics to the stabilization mode described above - the only difference between the two modes is the implementation of a non-linear control law that uses averages of the off-diagonal elements in the A_{err} matrix defined in Eq 12 to compute the brute-force torque

$$M_{ccc} = I\ddot{\vec{\alpha}} = -k_d\dot{\vec{\alpha}} - k_p\vec{\alpha}^* \quad (16)$$

where

$$\vec{\alpha}^* = \frac{1}{2} \begin{bmatrix} A_{2,3} - A_{3,2} \\ A_{3,1} - A_{1,3} \\ A_{1,2} - A_{2,1} \end{bmatrix} \quad (17)$$

and our control gains are the same as selected for the stability mode.

This method handles large control errors, which is useful when the spacecraft's target direction changes suddenly (i.e. when the spacecraft must slew between its "on" and "off" configurations).

Like the stabilization mode, slew mode initially used q -method for development, but we have replaced this in the full implementation with our EKF formulation from the previous problem set.

8.3.4 CONTROL LAW IMPLEMENTATION

For the stabilization and slew modes, we need a selection of our proportional and derivative gains - we do so via standard PD placement for a damped harmonic oscillator system,

$$I\ddot{\vec{\alpha}} + k_d\dot{\vec{\alpha}} + k_p\vec{\alpha} = 0 \quad (18)$$

We note that we can re-write this equation as

$$\ddot{\vec{\alpha}} + I^{-1}k_d\dot{\vec{\alpha}} + I^{-1}k_p\vec{\alpha} = \ddot{\vec{\alpha}} + 2\omega_n\zeta\dot{\vec{\alpha}} + \omega_n^2\vec{\alpha} = 0 \quad (19)$$

where ω_n is the desired response frequency and ζ is the damping factor. By selecting these two parameters, we can calculate our gains as

$$k_p = I\omega_n^2 \quad \text{and} \quad k_d = 2\zeta\sqrt{Ik_p} \quad (20)$$

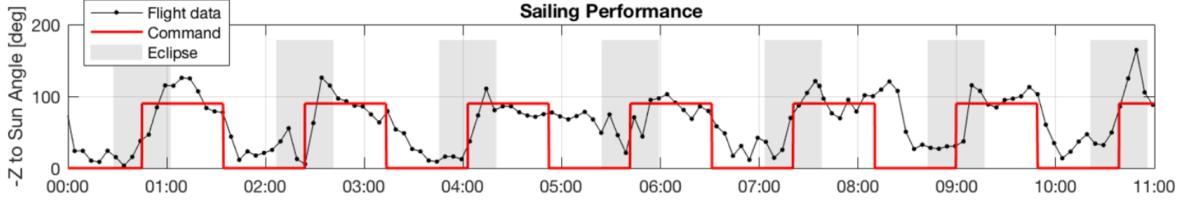


Figure 48: An example day (August 4, 2019 - UTC) of nominal LS2 operation, with several slew maneuvers to and from sun-pointing [1]

In the testing below, we select a heavy damping factor of $\zeta = 0.95$ and a response frequency of 0.075 rad/s (roughly 70 times the mean motion of our initial orbit). The high response frequency was chosen to ensure that the spacecraft stabilized its attitude quickly to achieve the longest optimal sail configuration possible during the half-orbit "on" phase.

In LS2's actual operations, seen in Fig 48, it appears that the satellite moved a bit more slowly than our simulation and overshoots its target position quite significantly when slewing. The slower slew is as expected, since we have significantly more control power than the actual spacecraft. The overshoot, meanwhile, implies that the derivative gains on LS2 are lower than the gains we have selected here. Future work includes reducing the actuator power to the proper levels and finding the gains that allow us to simulate the behavior seen in Fig. 48.

8.4 SATELLITE CONTROL TESTS

After implementing the above systems, we tested a variety of configurations to validate our estimation and control techniques. As mentioned above, initial configurations uses the q -method for all state estimation to simplify debugging; after successful control tests with this method (with and without sensor noise and attitude perturbations), we re-implemented the EKF for the stability and slew modes to include more advanced state estimation in our system.

For all tests, our attitude control systems are given non-physical capabilities (very high saturation limits and rate-of-change limits for both the momentum wheel and magnetorquers) and lack process noise, to prevent the added complexity that would have made the initial development less tractable. Furthermore, the added strength of the attitude control systems allows the spacecraft to find equilibrium faster, reducing the need for long, computationally-costly simulations. Each test started with a very low angular rate (< 0.5 deg/s on each axis) to reduce the required detumbling time and focus on stabilization and slew maneuvering, though the initial orientation is significantly different than the initial target.

The results for each test are provided in the sections below in a variety of figures - note that the spacecraft generally shows a capability to re-orient itself to the target attitude in a reasonable amount of time, though initial alignment is made more difficult when sensor noise is present.

8.4.1 q -METHOD, NO SENSOR NOISE

With sensor noise and attitude perturbations turned off, the state estimation is nearly exact using q -method (Fig 49). As a result, our control methods are simple to test, as the spacecraft knows the true attitude control error (rather than the attitude control error based off a slightly erroneous estimated attitude).

As shown in Figures 49-54, the spacecraft is able to quickly re-orient itself to the desired

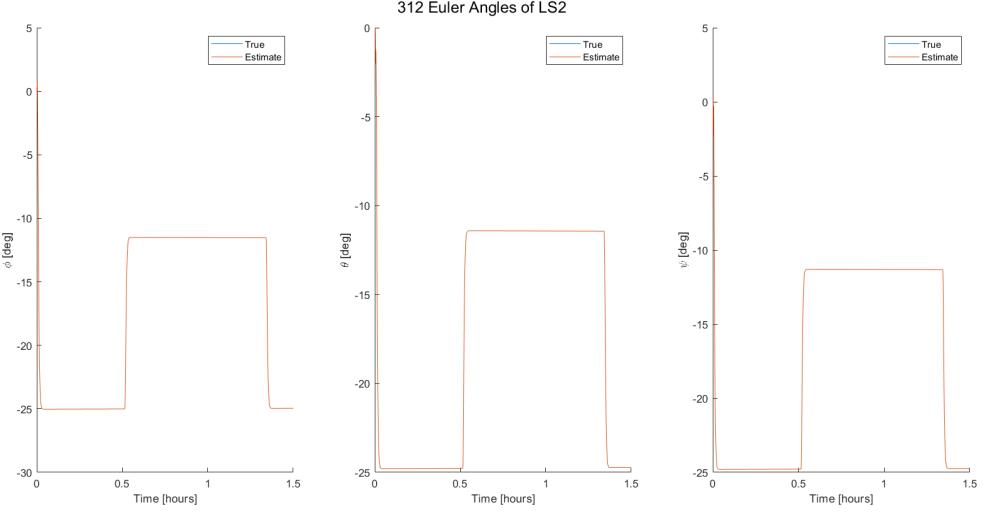


Figure 49: Attitude results from control testing using q -method estimation without sensor noise.

attitude within a time span $< 10\%$ of an orbital period. The high damping factor clearly removes oscillations around the target and the steady-state error is negligible (Fig 50).

Further, the spacecraft's estimates are generally identical to the ground truth - the only exception is the ADCS requested torque vs. the actual torque in Fig 51. Here, our imposed saturation limits (though large) limit the system output; most notably, the limit in the rate-of-change of the momentum wheel speed prevents overly-fast speed changes (which also tend to cause issues in the numerical integration). The result are torques that disagree non-trivially, but the results of the attitude control tactics are still quite impressive. We expect this problem would be eliminated by decreasing the control gains as previously discussed.

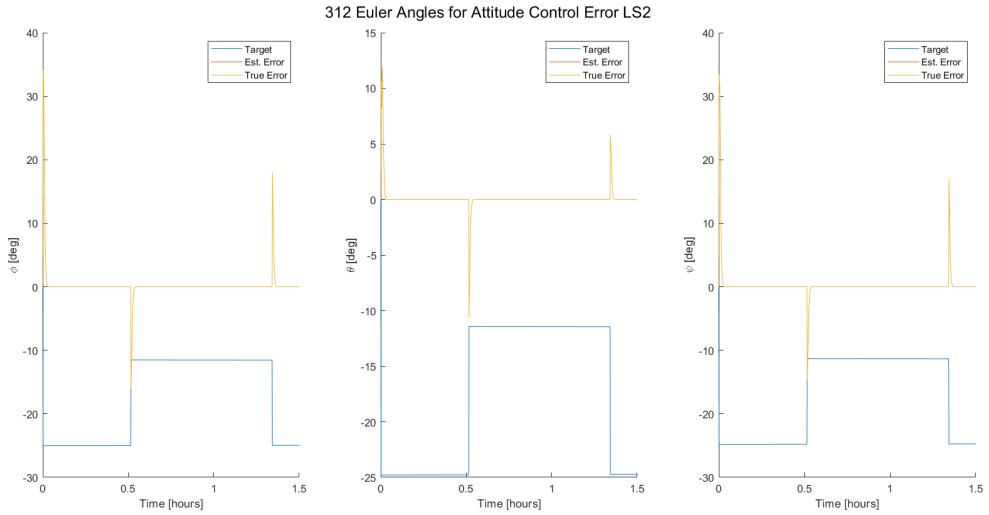


Figure 50: Control error results from control testing using q -method estimation without sensor noise. The spikes in the error are expected as the target orientation changes and the spacecraft slews to the new target.

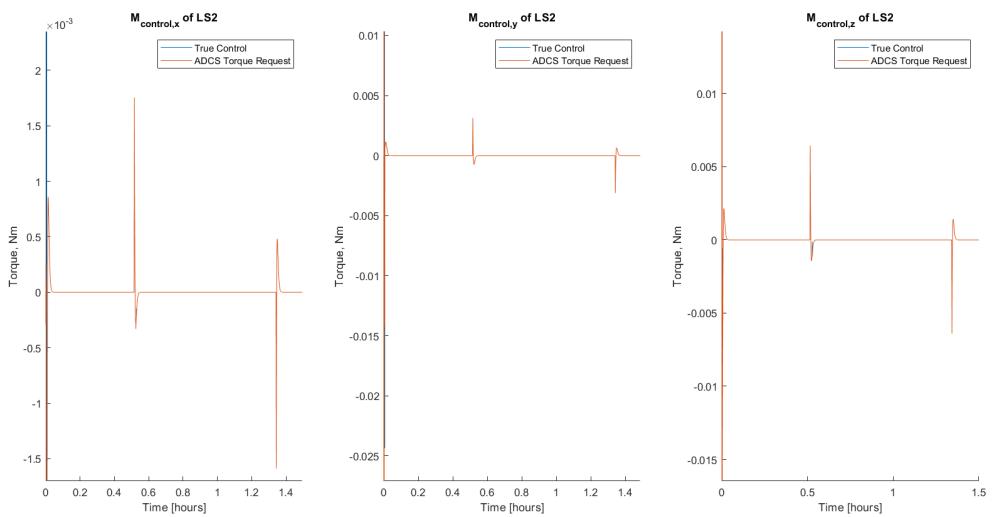


Figure 51: Control torque results from control testing using q -method estimation without sensor noise.

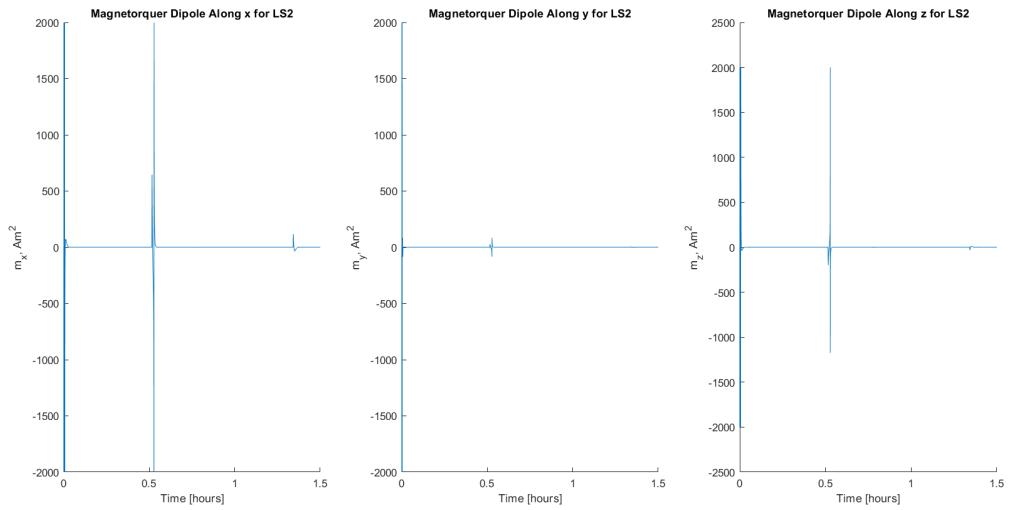


Figure 52: Magnetorquer dipole moment from control testing using q -method estimation without sensor noise.

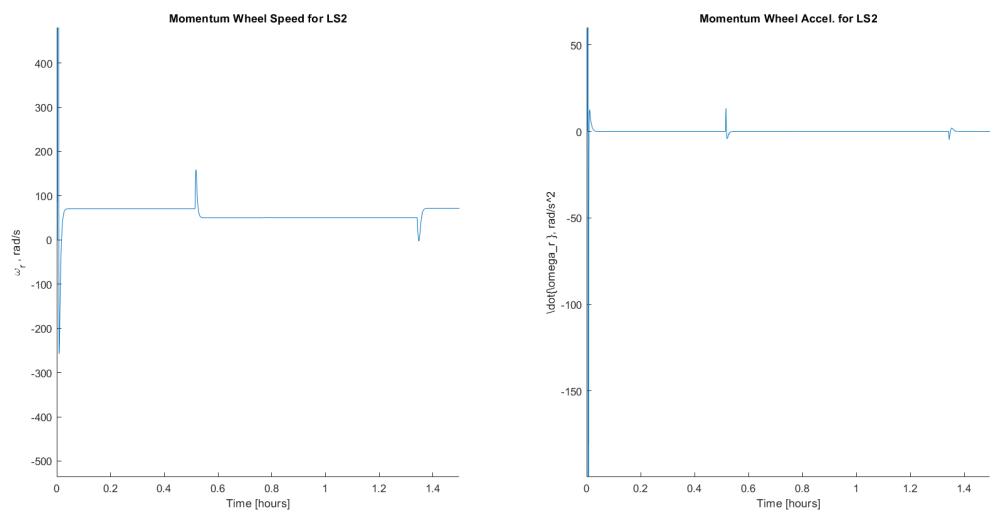


Figure 53: Momentum wheel results from control testing using q -method estimation without sensor noise.

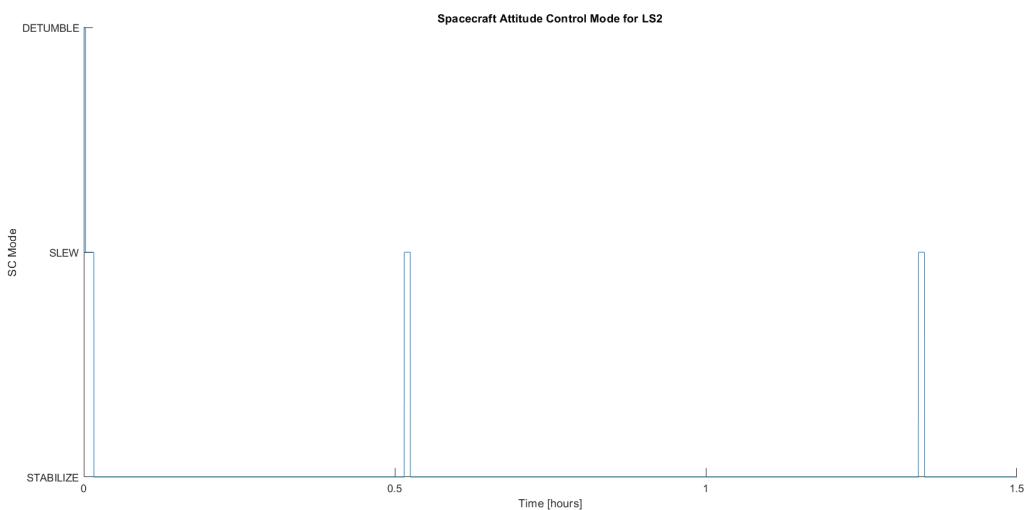


Figure 54: Spacecraft mode from control testing using q -method estimation without sensor noise.

8.4.2 q -METHOD, SENSOR NOISE & PERTURBATIONS

After validating the system without sensor noise, we now add in both sensor error and environmental torques. In doing so, we cause our spacecraft estimate to deviate from the true attitude, and the result is worse control (as the spacecraft's actions are now based on slightly erroneous knowledge of the spacecraft state). Our resulting torques are far noisier and our attitude shows more deviations from the target (where sensor error causes the system to take unnecessary action that shifts the attitude away from the target), but performance is still appreciably good as the spacecraft slews to the proper positions every half-orbit.

We do note that the spacecraft more often switches between the slew and stabilization modes - this is expected, as the spacecraft will always switch to slew mode whenever the estimated control error is outside of the range deemed acceptable for use in the linearized control equation ($\sim 5^\circ$, also the approximate pointing budget for LS2). Further, there are additional, fairly regular spikes in the estimated 312 Euler angle representation of LS2's attitude (Fig 55), which we did not see before – we have not yet determined the exact cause of these, but we observed this behavior previously in Section 6.

Notably, seen in Fig 60 even with noise, once the spacecraft has detumbled, we operate solely in slew or stabilize mode, so our controller is fairly robust to noise and perturbations, although improvements in the slewing logic could be made to prevent the rapid switching back and forth between slew and stabilize.

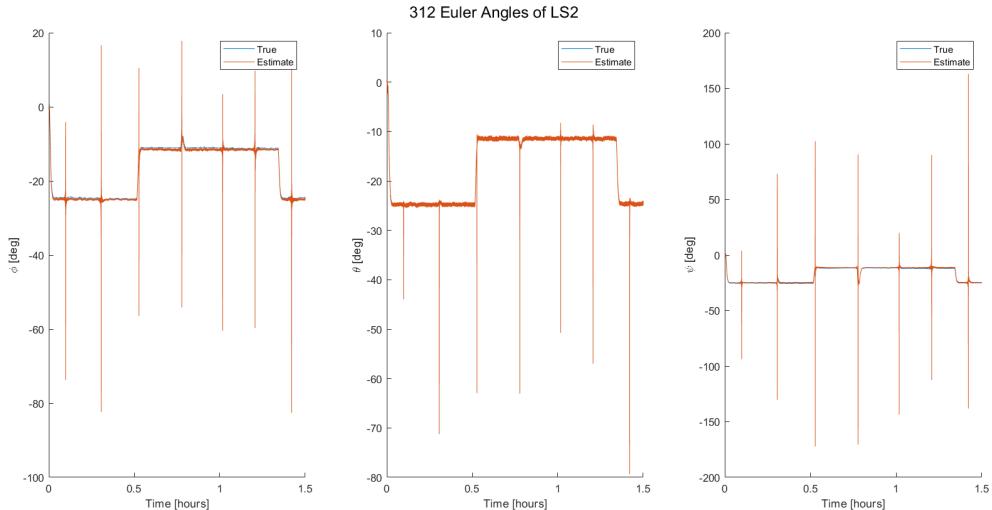


Figure 55: Attitude results from control testing using q -method estimation with sensor noise and attitude perturbations.

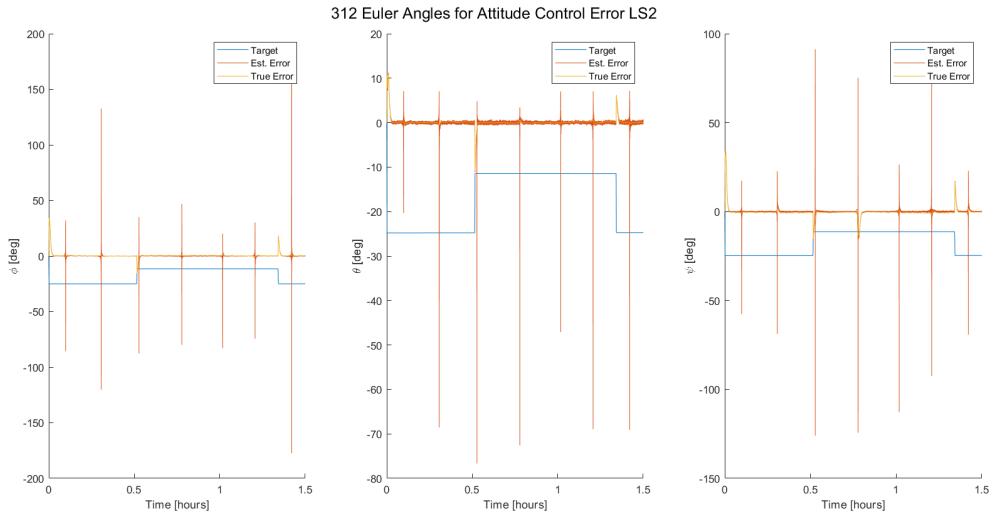


Figure 56: Control error results from control testing using q -method estimation with sensor noise and attitude perturbations.

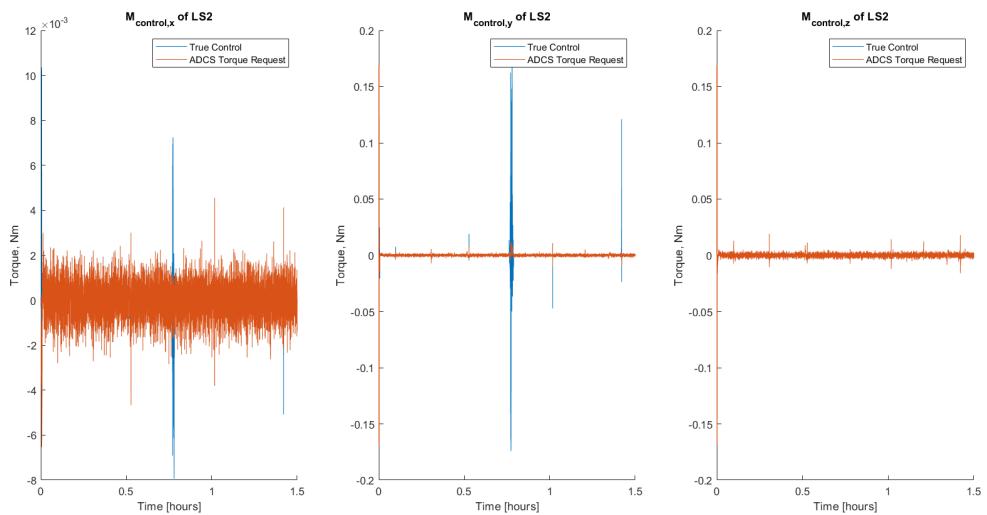


Figure 57: Control torque results from control testing using q -method estimation with sensor noise and attitude perturbations.

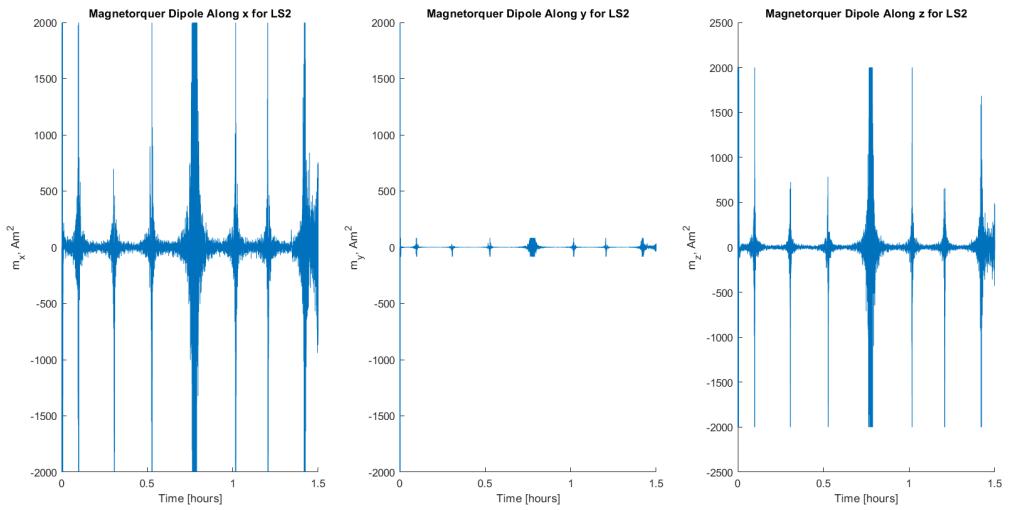


Figure 58: Magnetorquer dipole moment from control testing using q -method estimation with sensor noise and attitude perturbations.

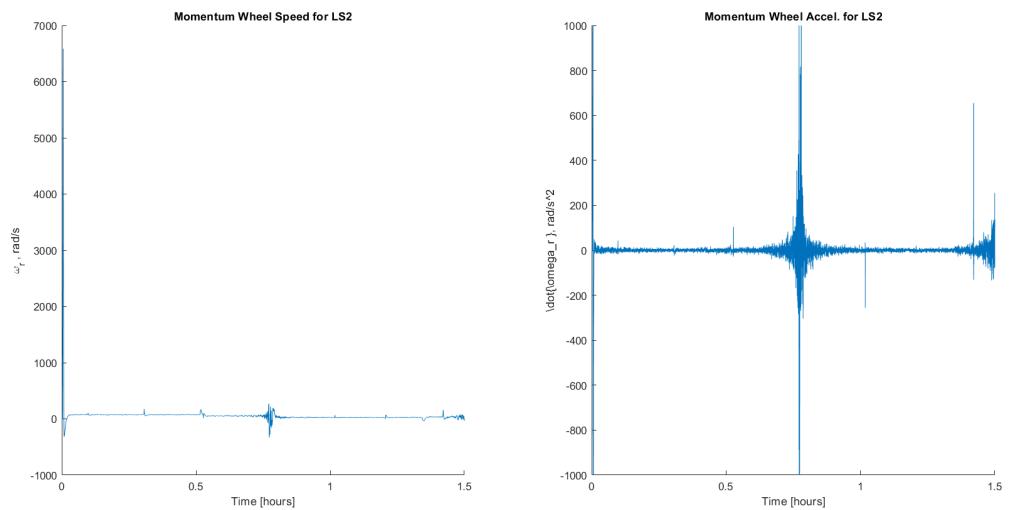


Figure 59: Momentum wheel results from control testing using q -method estimation with sensor noise and attitude perturbations.

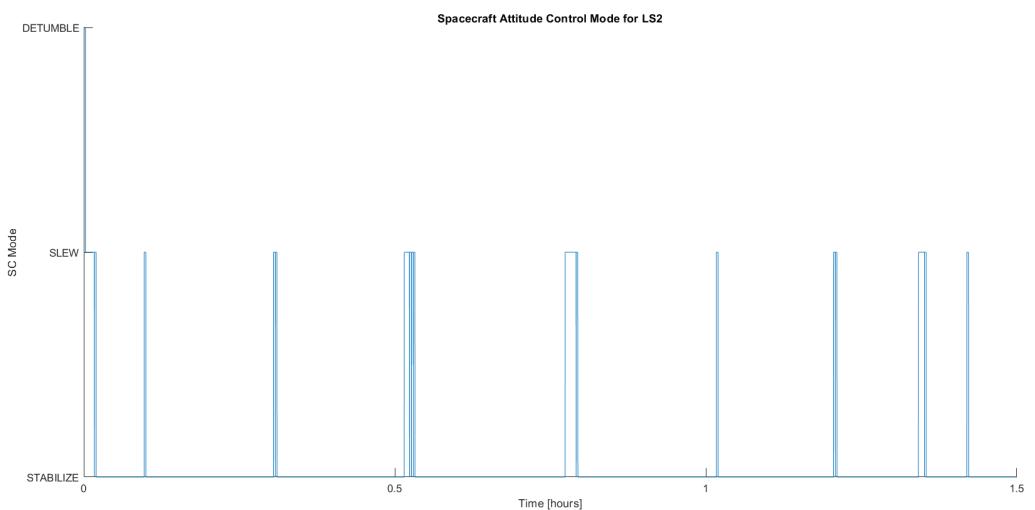


Figure 60: Spacecraft mode from control testing using q -method estimation with sensor noise and attitude perturbations.

8.4.3 EKF, SENSOR NOISE, & PERTURBATIONS

Finally, we implement state estimation via EKF for the slew and stability modes. The results are presented in the figures below - our initially high covariance in our state estimate causes initial control performance to be sub-par, but after approximately one quarter-orbit, the spacecraft state estimate is sufficiently accurate to produce effective control maneuvers - we see a nice near-square-wave output in the attitude figure and generally reasonable torque requests from the ADCS system towards the later end of the simulation. Note the addition of the attitude estimation error in the figure collection, Fig 67, which shows the initial large deviation in the state estimate from ground truth - otherwise, attitude estimates are generally good except for minor deviations that are likely due to Euler error singularities (which are either improperly handled by our control law or by the EKF itself).

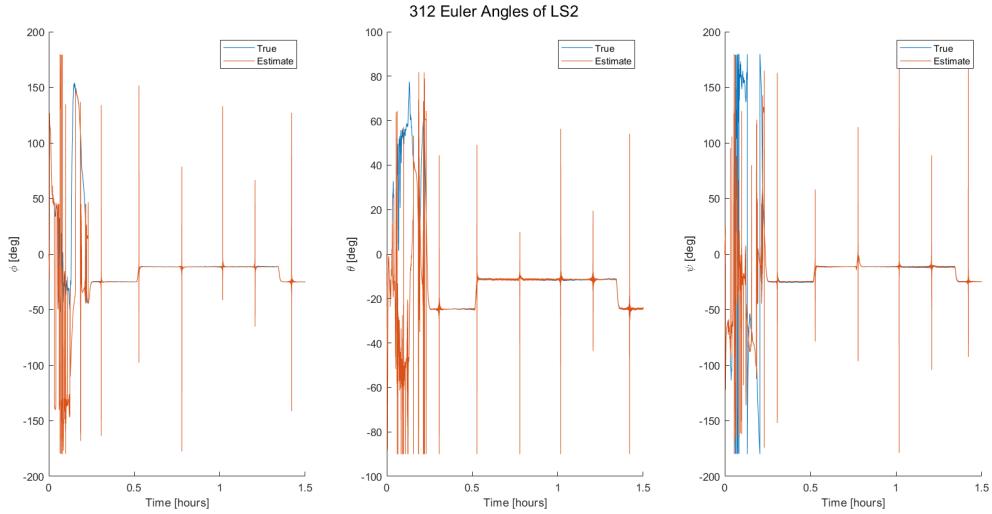


Figure 61: Attitude results from control testing using EKF estimation with sensor noise and attitude perturbations.

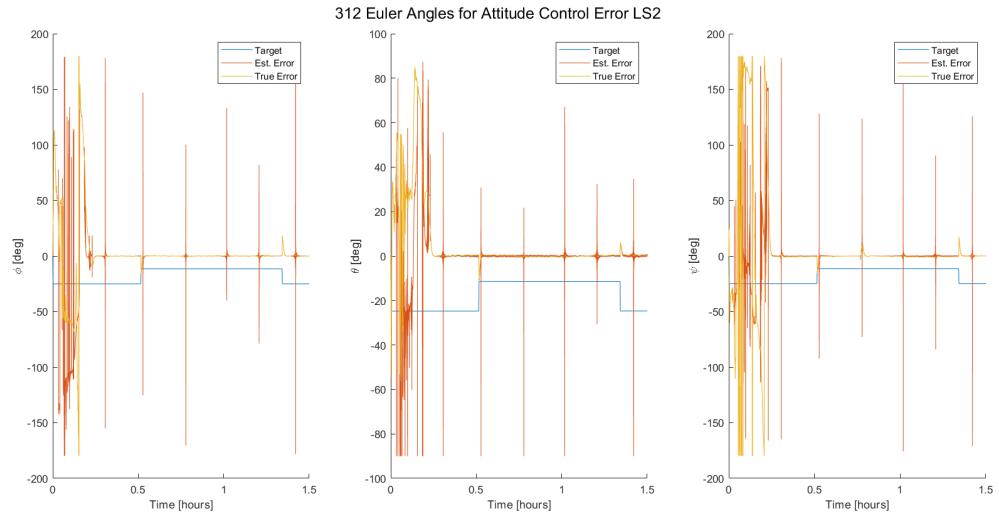


Figure 62: Control error results from control testing using EKF estimation with sensor noise and attitude perturbations.

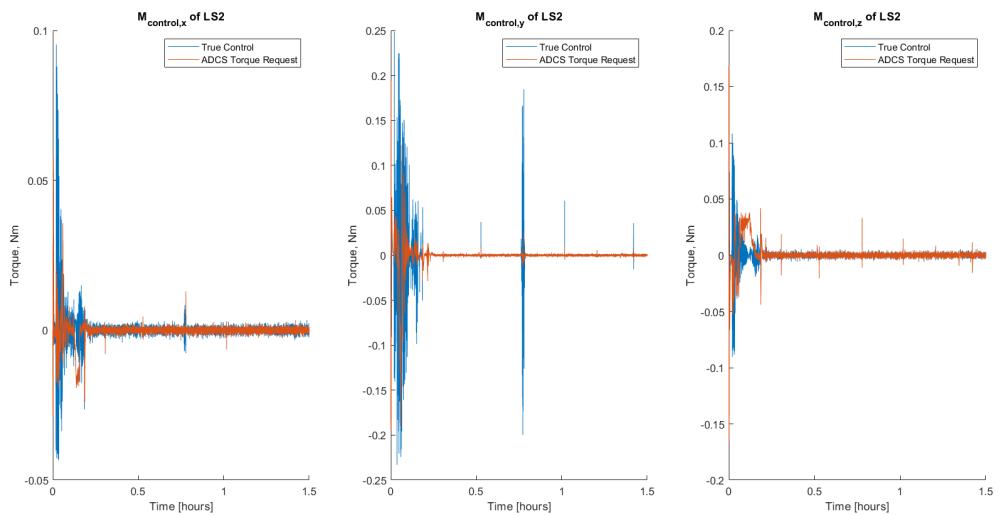


Figure 63: Control torque results from control testing using EKF estimation with sensor noise and attitude perturbations.

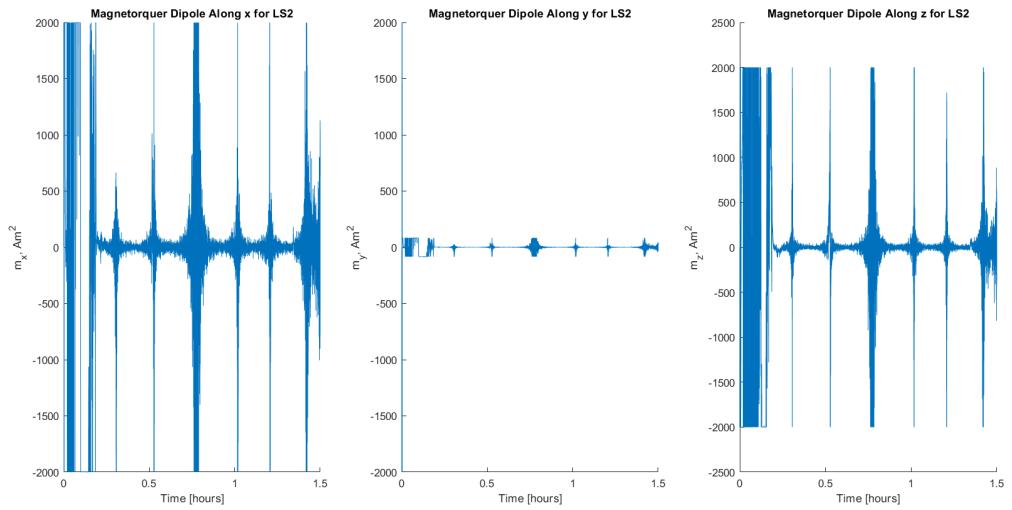


Figure 64: Magnetorquer dipole moment from control testing using EKF estimation with sensor noise and attitude perturbations.

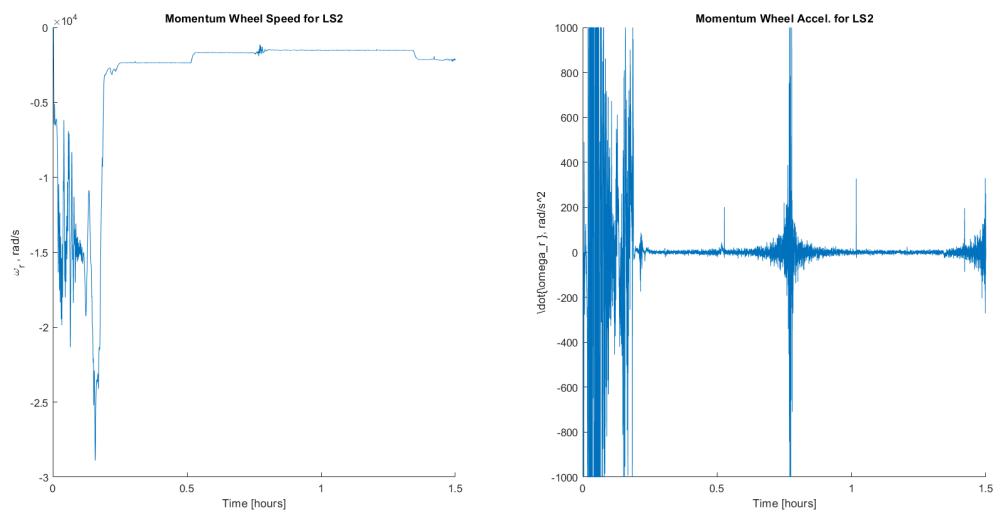


Figure 65: Momentum wheel results from control testing using EKF estimation with sensor noise and attitude perturbations.

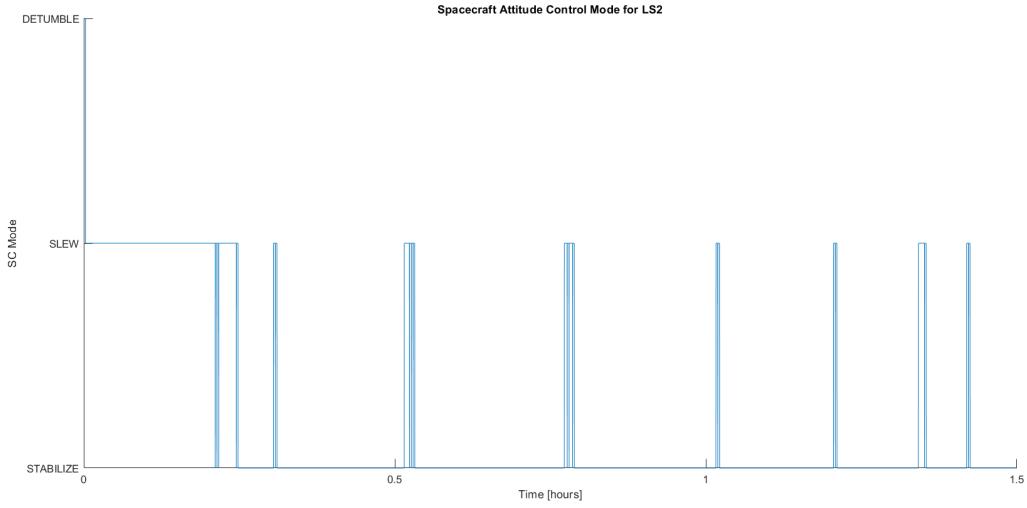


Figure 66: Spacecraft mode from control testing using EKF estimation with sensor noise and attitude perturbations.

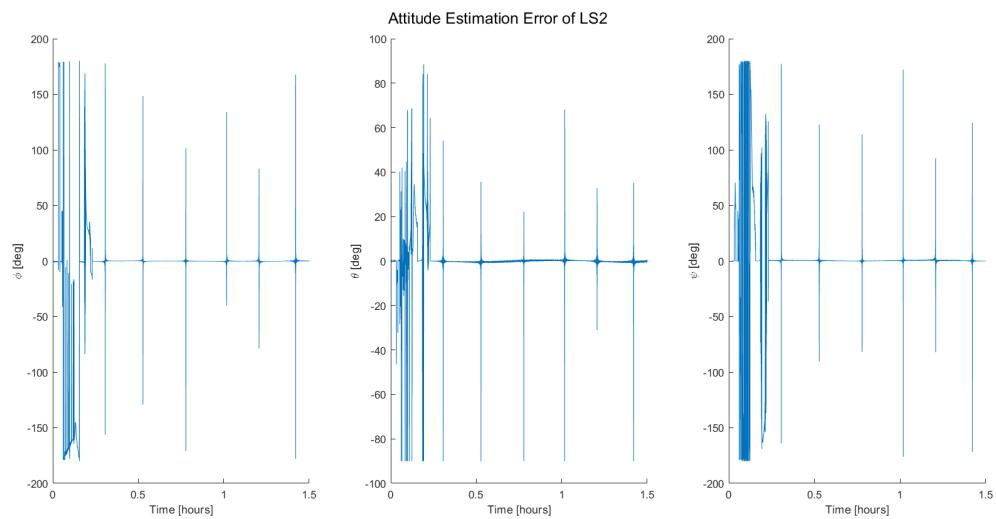


Figure 67: Attitude estimation error from control testing using EKF estimation with sensor noise and attitude perturbations.

9 FUTURE WORK

In addition to EKF (successfully implementing bias and momentum wheel speed estimation), actuator (re-imposing stricter saturation limits), and control law improvements (tuning gains, slew/stabilize switching logic), we also hope to implement SRP in the orbital dynamics model to show the effect of the solar sail with attitude control on the orbit.

10 REFERENCES

- [1] Bruce Betts et al. “LIGHTSAIL 2: CONTROLLED SOLAR SAIL PROPULSION USING A CUBESAT”. In: 2019.
- [2] Craig J. Van Beusekom and Ron Lisowski. “THREE-AXES ATTITUDE DETERMINATION FOR FALCONSAT-3”. In: AIAA (2003).
- [3] Chris Biddy and Tomas Svitek. “LightSail-1 Solar Sail Design and Qualification”. In: 2012.
- [4] George C. Marshall Space Flight Center. *NASA Facts: NanoSail-D*. https://www.nasa.gov/centers/marshall/pdf/484314main_NASAfactsNanoSail-D.pdf. Accessed: 4-6-2021.
- [5] Jason Davis. *What's the Difference between LightSail 1 and LightSail 2?* <https://www.planetary.org/articles/difference-between-lightsails>. Accessed: 4-6-2021. June 2019.
- [6] *Endurosat 3U Solar Panel*. <https://www.endurosat.com/cubesat-store/cubesat-solar-panels/3u-solar-panel-xy/>. Accessed: 4-3-2021.
- [7] Wiley J. Larson and James R. Wertz. *Space Mission Analysis and Design*. Space Technology Series. Microcosm Press, 1999. ISBN: 1-881883-10-8.
- [8] *Lightsail 2: Parts List*. Accessed: 4-3-2021. 2017.
- [9] Barbara Plante et al. “LightSail 2 ADCS : From Simulation to Mission Readiness”. In: 2016.
- [10] Rex W. Ridenoure et al. “Testing The LightSail Program: Advancing Solar Sailing Technology Using a CubeSat Platform”. In: 2016.
- [11] John B. Schleppe. “Development of a Real-Time Attitude System Using a Quaternion Parameterization and Non-Dedicated GPS Receivers”. In: *UCGE Reports* (1996).
- [12] The Planetary Society. *LIGHTSAIL 2 MISSION CONTROL*. https://secure.planetary.org/site/SPageNavigator/mission_control.html. Accessed: 4-6-2021.
- [13] Yuichi Tsuda et al. “Achievement of IKAROS – Japanese deep space solar sail demonstration mission”. In: *Acta Astronautica* 82.2 (2013), pp. 183–188. ISSN: 0094-5765. DOI: <https://doi.org/10.1016/j.actaastro.2012.03.032>. URL: <https://www.sciencedirect.com/science/article/pii/S0094576512001348>.

Appendices

A Mass Distribution Analysis

The total mass of the LS2 system is quoted as 4.93kg [10] - using various published resources from the Planetary Society and the drawings in Appendix B, we are able to make rough estimates for the masses of the five primary components: the solar sail, the sail booms, the forward body, the solar panels, and the rear body.

A.1 Solar Sail

We estimate the full mass of the solar sail system (solar sails, sail booms, boom extension mechanism, and sail housing) to be 2.9kg, based off a note from a deployment package development summary stating that the whole deployment system was contained in a package with mass " <3 kg" [3]. The sail housing, which is located in the rear body of the system, we estimate as ≈ 500 g. The solar sails, made of mylar with density $1.38 \frac{\text{g}}{\text{cm}^3}$, have a total volume of $5.57\text{m} \times 5.57\text{m} \times 4.5\mu\text{m}$ per the LS2 drawings. The total mass of the solar sails is therefore estimated as 0.19872 kg .

A.2 Sail Booms

The sail booms, made of elgiloy with density $8.3 \frac{\text{g}}{\text{cm}^3}$ [10], have a volume of $3.5\text{cm} \times 0.02\text{cm} \times 4\text{m}$ per the LS2 drawings. The total mass of each sail boom is therefore estimated as 0.2324 kg .

A.3 Forward Body

The forward body contains the communications avionics (transceiver board and antenna) as well as the boom extension mechanism. We can estimate the mass of the boom extension mechanism by subtracting the assumed 500g mass of the sail housing, the 198.72g of solar sail mass, and the 232.4g mass of the four sail booms from the estimated solar sail system mass of 2.9kg. By adding to that an estimated ≈ 200 g for communications components, we get the estimated mass of the forward body as 1.472 kg .

A.4 Solar Panels

The mass of each 3U-long solar panel was estimated to be 0.150 kg based off the similar mass and form factor of Endurosat's 3U solar panel system [6].

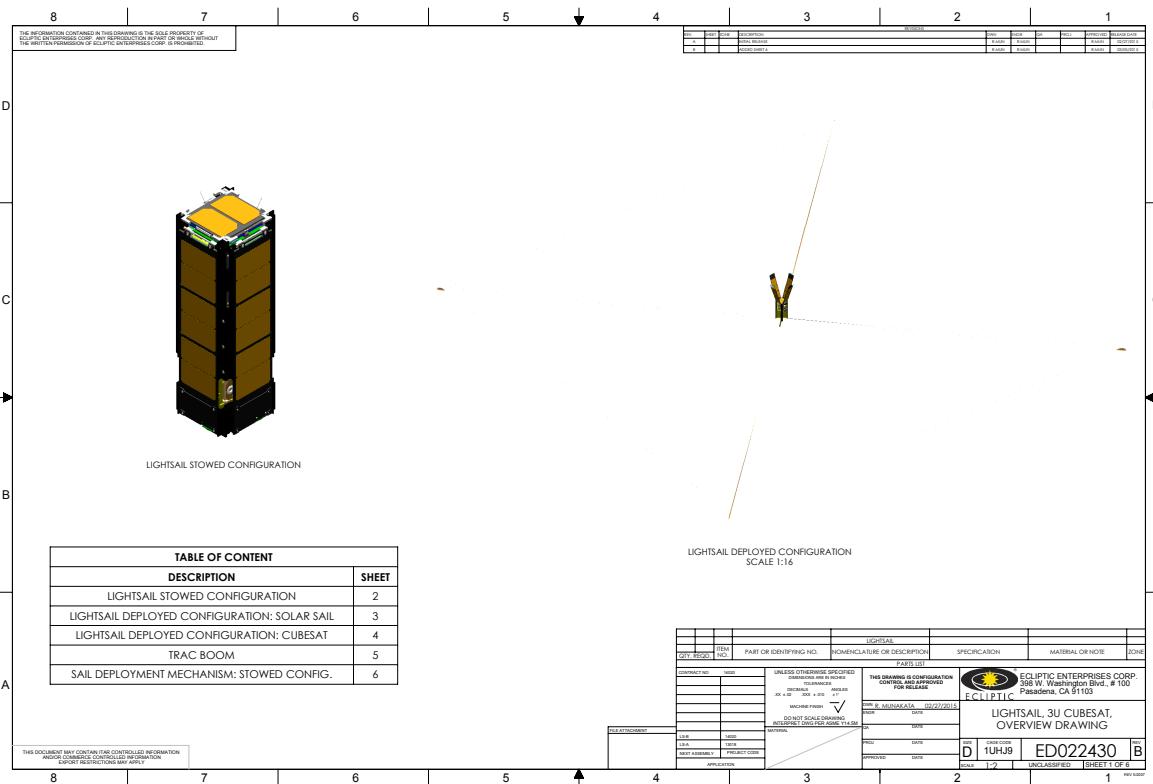
A.5 Rear Body

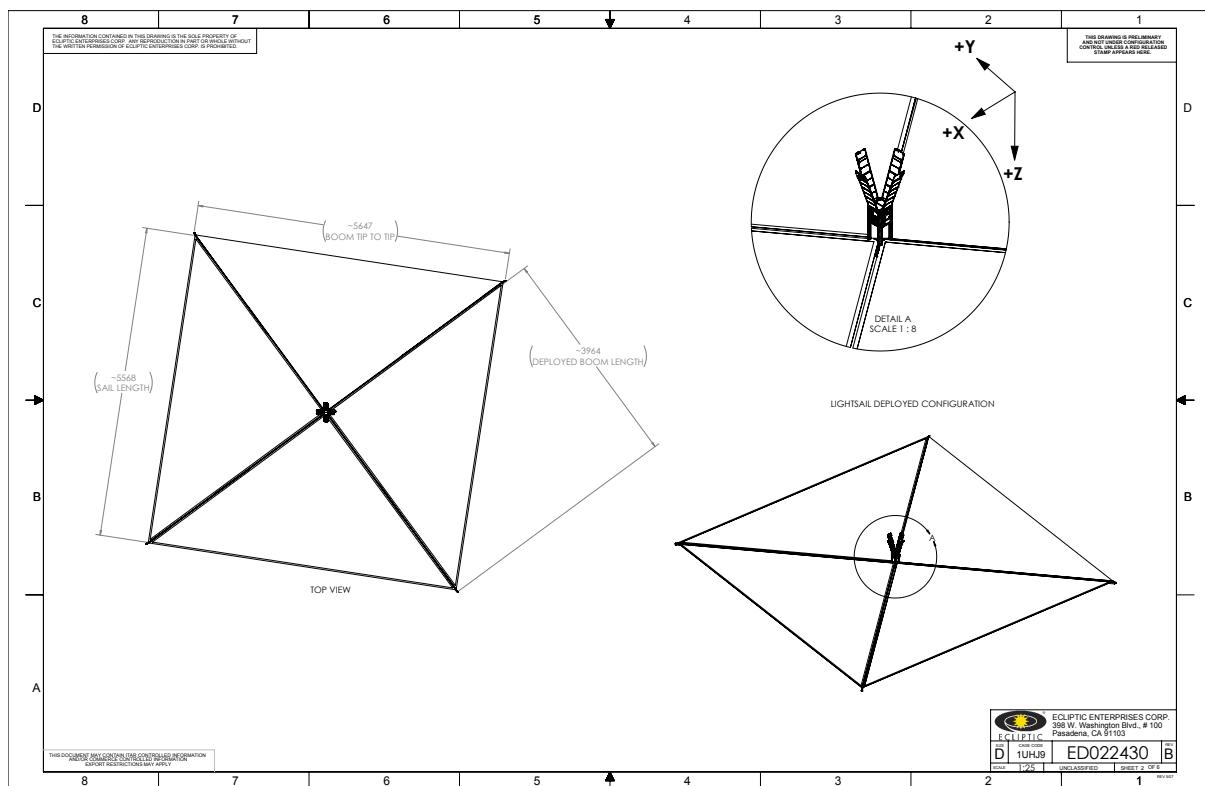
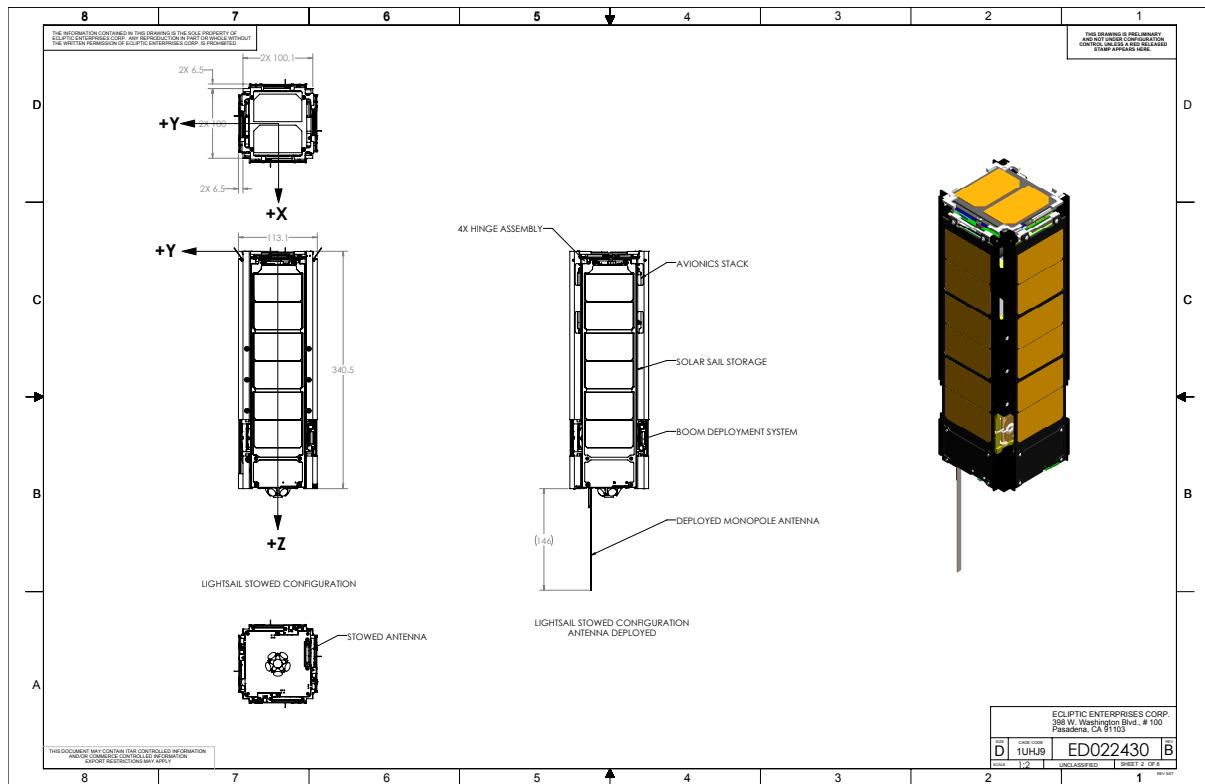
The mass of the rear body was calculated implicitly by subtracting the calculated masses of the above systems from the total satellite mass. This gives an estimated rear body mass of 1.73 kg . We give some validity to this estimate by noting that the breakdown in Table 10 provides reasonable estimates for rear body component masses that total to 1.74 kg. Note that the table lists some approximate values (indicated with \approx) and some values taken from the links on the Lightsail 2 parts list [8] where available .

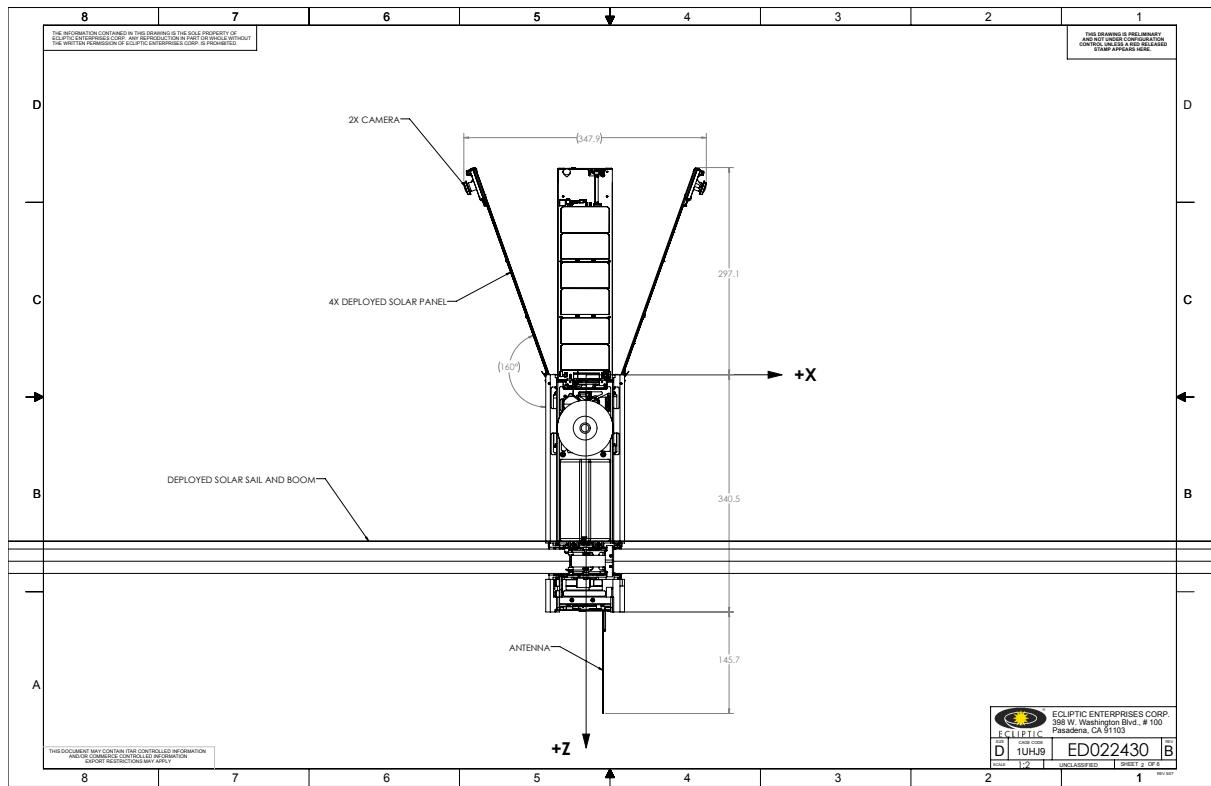
Description	Mass, kg
Momentum Wheel	0.226
Magnetorquers (x3)	0.690
Battery Cells (x8)	0.128
Avionics	≈ 0.1
Structures	≈ 0.1
Sail Housing	≈ 0.5
TOTAL	1.744

Table 10: Estimated Mass Breakdown of Rear Body

B LS2 System Drawings







C VBA Solidworks Macro for Exporting Surface Normals

The Solidworks VBA macro below uses a sketch to generate information about the outer geometry of a shape - the starting location of a line is taken to be the barycenter of the surface, and the unit direction is taken from the vector from the start point to the end point (normalized by length). The area of the associated surface (in m^2) is encoded in the length of the line (in m) - this is particularly easy to do in Solidworks with parameterized dimensions, and makes updating the centroids and normals an easy process, even for complicated shapes. The MATLAB script following then imports the information to MATLAB.

```
1 Dim swApp As Object
2 Sub main()
3 Dim swApp As SldWorks.SldWorks
4 Dim doc As SldWorks.ModelDoc2
5 Dim assembly As SldWorks.AssemblyDoc
6 Dim sm As SldWorks.SelectionMgr
7 Dim eq As SldWorks.EquationMgr
8 Dim feat As SldWorks.Feature
9 Dim sketch As SldWorks.sketch
10 Dim v As Variant
11 Dim i As Long
12 Dim sseg As SldWorks.SketchSegment
13 Dim sline As SldWorks.SketchLine
14 Dim sp As SldWorks.SketchPoint
15 Dim ep As SldWorks.SketchPoint
16 Dim s As String
17 Dim CMx, CMy, CMz As Double
18
19
20
21 Set exApp = CreateObject("Excel.Application")
22 If Not exApp Is Nothing Then
23     exApp.Visible = True
24     If Not exApp Is Nothing Then
25         exApp.Workbooks.Add
26         Set sheet = exApp.ActiveSheet
27         If Not sheet Is Nothing Then
28             sheet.Cells(1, 1).Value = "X (m)"
29             sheet.Cells(1, 2).Value = "Y (m)"
30             sheet.Cells(1, 3).Value = "Z (m)"
31             sheet.Cells(1, 4).Value = "Normal X"
32             sheet.Cells(1, 5).Value = "Normal Y"
33             sheet.Cells(1, 6).Value = "Normal Z"
34             sheet.Cells(1, 7).Value = "Area (m*m)"
35         End If
36     End If
37 End If
38
39
40 Set swApp = GetObject(, "sldworks.application")
41 If Not swApp Is Nothing Then
42     Set doc = swApp.ActiveDoc
43     If Not doc Is Nothing Then
44         If doc.GetType = swDocASSEMBLY Then
45             Set assembly = doc
46             Set sm = doc.SelectionManager
```

```

47     Set eq = doc.GetEquationMgr
48     If Not assembly Is Nothing And Not sm Is Nothing Then
49         Dim j, m As Long
50         m = eq.GetCount
51         For j = 0 To m - 1
52             If InStr(1, eq.Equation(j), "SW-CenterofMassX") Then
53                 CMx = eq.Value(j)
54             ElseIf InStr(1, eq.Equation(j), "SW-CenterofMassY") Then
55                 CMy = eq.Value(j)
56             ElseIf InStr(1, eq.Equation(j), "SW-CenterofMassZ") Then
57                 CMz = eq.Value(j)
58             End If
59         Next j
60
61     If sm.GetSelectedObjectType2(1) = swSelSKETCHES Then
62         Set feat = sm.GetSelectedObject6(1, -1)
63         Set sketch = feat.GetSpecificFeature2
64         If Not sketch Is Nothing Then
65             v = sketch.GetLines2(1)
66             numlines = sketch.GetLineCount2(1)
67             Dim c As Long
68             c = 0
69             For i = 0 To numlines - 1
70                 If Not sheet Is Nothing And Not exApp Is Nothing Then
71
72                     Dim myType As Double
73                     myType = (v(12 * i + 2))
74
75                     If myType < 1 Then
76
77                         Dim startX, startY, startZ, endX, endY, endZ, myL As Double
78                         startX = (v(12 * i + 6))
79                         startY = (v(12 * i + 7))
80                         startZ = (v(12 * i + 8))
81                         endX = (v(12 * i + 9))
82                         endY = (v(12 * i + 10))
83                         endZ = (v(12 * i + 11))
84                         myL = Sqr(((startX - endX) ^ 2) + ((startY - endY) ^ 2) ...
85                                     + ((startZ - endZ) ^ 2))
86                         sheet.Cells(2 + c, 1).Value = startX - CMx
87                         sheet.Cells(2 + c, 2).Value = startY - CMy
88                         sheet.Cells(2 + c, 3).Value = startZ - CMz
89                         sheet.Cells(2 + c, 4).Value = (endX - startX) / myL
90                         sheet.Cells(2 + c, 5).Value = (endY - startY) / myL
91                         sheet.Cells(2 + c, 6).Value = (endZ - startZ) / myL
92                         sheet.Cells(2 + c, 7).Value = myL
93                         exApp.Columns.AutoFit
94                         c = c + 1
95                     End If
96                 Next i
97             End If
98         End If
99     End If
100    End If
101    End If
102    End If
103 End Sub

```

```

1 %% get_surfacedata()
2 % Reads surface data from CSV file.
3 % No input.
4 % Output:
5 %   - C : an (n x 3) matrix, where n is the number of surfaces, denoting
6 %         the location in body coordinates of each surface's ...
7 %         centroid in m.
8 %   - N : an (n x 3) matrix denoting the unit outward-facing normal for
9 %         each surface.
10 %   - A : an (n x 1) matrix denoting the area for each surface, in m^2.
11 function [C, N, A] = get_surfacedata()
12     centroid_file = 'CentroidData.csv';
13     data = readtable(centroid_file, 'ReadVariableNames', false);
14     C = [data.Var1 , data.Var2 , data.Var3];
15     N = [data.Var4 , data.Var5 , data.Var6];
16     A = data.Var7;
17 end

```

D Verification of Inertia Computations

For our analytical calculations, we will ignore asymmetric components, and place the associated mass of each primary component at its geometric centroid (this is a fairly safe assumption, given the layout of our satellite). This will greatly simplify our calculations.

We begin with two basic equations; first, the moments of inertia of a rectangular prism with length a along its principal x axis, length b along its principal y axis, length c along its principal z axis, and mass m :

$$\begin{aligned} I_{xx} &= \frac{1}{12}m(b^2 + c^2) \\ I_{yy} &= \frac{1}{12}m(a^2 + c^2) \\ I_{zz} &= \frac{1}{12}m(a^2 + b^2) \end{aligned}$$

We will also need the formula for a rectangular prism tilted about an axis, as the sail booms are at a 45° angle w.r.t the z axis and the solar panels are tilted at 20° w.r.t to the x/y axes. For the sail booms, I_{zz} is the same as above, but our formula for I_{yy} will need to be modified to account for this rotation. I_{yy} for this rectangular prism (assuming the tilt angle is $\beta = 45^\circ$) is

$$I_{yy} = \frac{1}{12}m(acos\beta + bsin\beta + c)$$

For the solar panels, we will also need to use (depending on the orientation of the panel)

$$\begin{aligned} I_{zz} &= \frac{1}{12}m(acos\beta + b + csin\beta) \\ &\quad \text{if tilted w.r.t to } x \text{ or} \\ I_{zz} &= \frac{1}{12}m(a + bcos\beta + csin\beta) \\ &\quad \text{if tilted w.r.t to } y. \end{aligned}$$

We also will make use of the parallel axis theorem, which states that the moment of inertia of a body about an axis B parallel to an axis, A , can be written as

$$I_B = I_A + mr^2$$

where m is the mass of the body and r is the distance between the parallel axes.

We also recognize that due to our simplified $x-y$ symmetry, $I_{xx} = I_{yy}$ for our purposes, and by definition, our x and y axes are aligned with the principal axes. Furthermore, due to the alignment of our body axes along the line of symmetry, we know that our full inertia matrix w.r.t the body axes is roughly

$$L = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{xx} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

In Table 11, we tabulate the mass, a , b , c , and the distance between the principal axis and y/z body axes for each component (measured using the CAD model). Applying our equation for

moment of inertia and parallel axis theorem, we are able to calculate the contribution of each item to I_{xx} and I_{zz} . Our final results are $I_{xx} = I_{yy} = 3.125 \text{ kg} \cdot \text{m}^2$ and $I_{zz} = 5.983 \text{ kg} \cdot \text{m}^2$, which have relative errors compared to the CAD-computed values of 0.7% and 0.02%, respectively.

Given these negligibly small errors (likely caused due to the fact that the distances between the parallel axes were measured assuming that the solar panels and booms were planes, rather than prisms with finite thickness, in addition to our simplifying assumptions about mass distribution), it is clear that the CAD-computed values for the inertia characteristics of our system are trustworthy. The off-diagonal components calculate by the 3D model are trusted, as they are very close to zero.

Component	Mass, kg	Width a, m	Length b, m	Height c, m	// to Y-Axis, m	// to Z-Axis, m	Quantity	$I_{yy}, \text{kg m}^2$	$I_{zz}, \text{kg m}^2$
Forward Body	1.47168	0.1	0.1	0.11	0.12055	0	1	0.0241	0.0024528
Sail Booms	0.2324	2.00E-04	4	0.035	1.3249	1.9957	4	6.27E-01	1.24E+00
Sail	0.19872	5.57	5.57	4.50E-06	0.06555	0	1	5.15E-01	1.0275
Rear Body	1.73	0.1	0.1	0.23	0.04945	0	1	0.01330	0.002883
Solar Panel X-Tilt	0.15	6.50E-03	0.1	0.316	0.31294	0.1040	2	1.52E-02	2.18E-03
Solar Panel Y-Tilt	0.15	0.1	6.50E-03	0.316	0.3298	0.1040	2	1.76E-02	2.18E-03
TOTAL	4.93	-	-	-	-	-	-	3.125909335	5.983479674

Table 11: Tabulated Inertia Calculation Data.

E Additional Orbital Propagator Notes

Plots of evolution of the orbital elements used to generate Figure 4. Atmospheric drag contributes to the decrease in semimajor axis, while J2 contributes to the change in RAAN and argument of periapsis. The perturbation calculations make use of the $e \approx 0$ approximation, which is fine given the initial $e \approx 0.001$, but will need to be modified to increase fidelity as the spacecraft orbit elongates due to SRP.

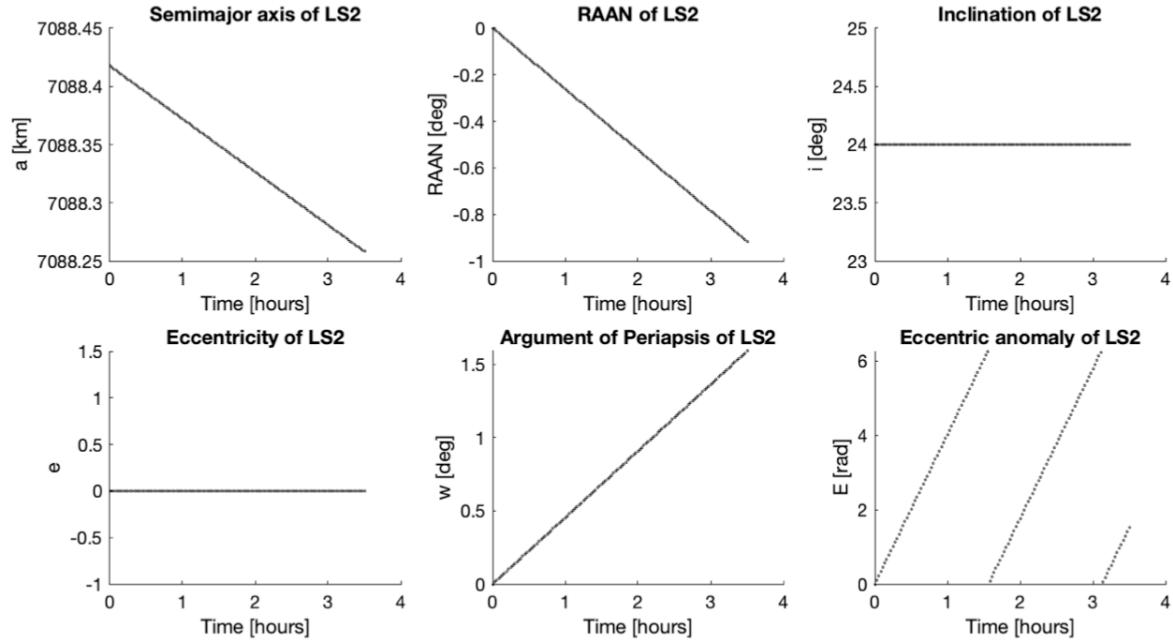


Figure 68: Keplerian orbital elements

F Quaternion and 312 Euler Kinematic Comparison

To validate our attitude propagation system, we compare the propagation of identical initial configurations for both quaternion kinematics and 312 Euler angle kinematics. In the series of figures below, we see the attitude propagation for both kinematics based on spin initially about the principal Y-axis (a principal axis was chosen to avoid the singularity of the 312 Euler sequence at $\Theta = \frac{\pi}{2}$). Note that all plots appear nearly exactly identical, validating the quaternion propagation method.

Furthermore, Fig 72 and 73 show the direct comparisons of the quaternion and 312 Euler output (each being converted to the other attitude for comparison, along with the raw difference error for comparison). Note that the 312 Euler strays from expected behavior the most, but error stay reasonable between the two parameterizations (note that a difference of 360 degrees is equivalent to a difference of zero).

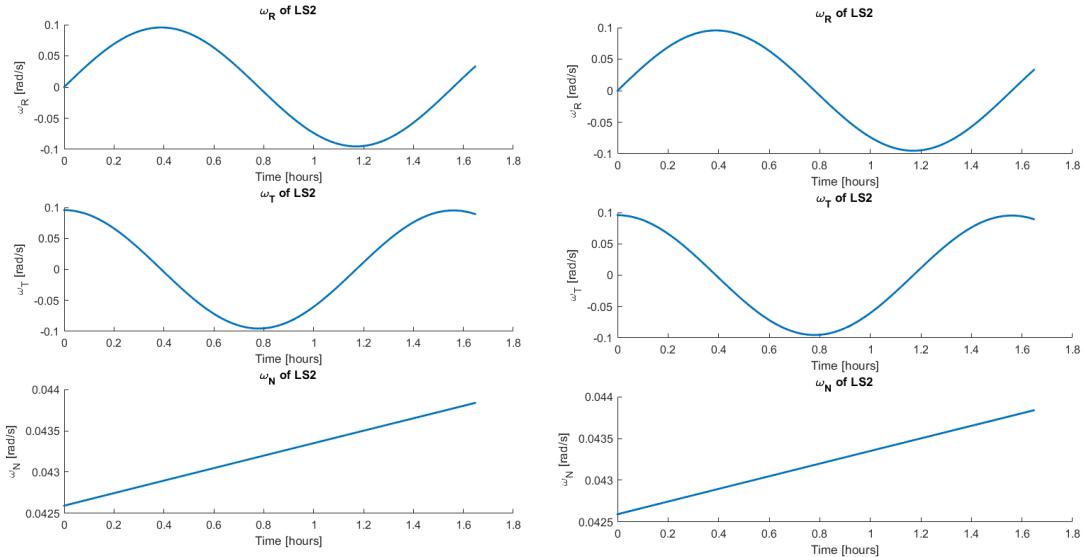


Figure 69: Angular vel. in RTN frame for Quaternion (left) and Euler 312 (right) propagation.

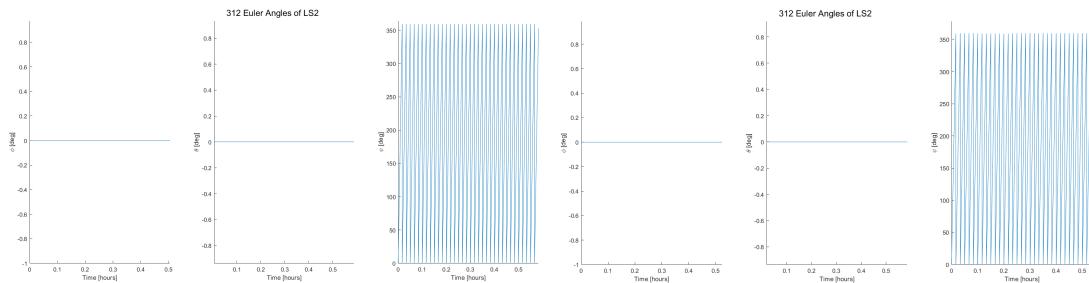


Figure 70: 312 Euler Angles for Quaternion (left) and Euler 312 (right) propagation.

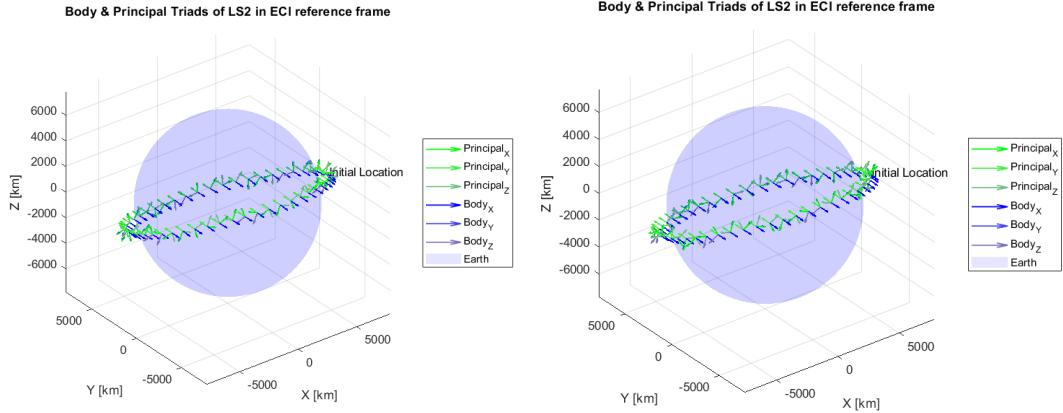


Figure 71: Coordinate triads in ECI frame for Quaternion (left) and Euler 312 (right) propagation.

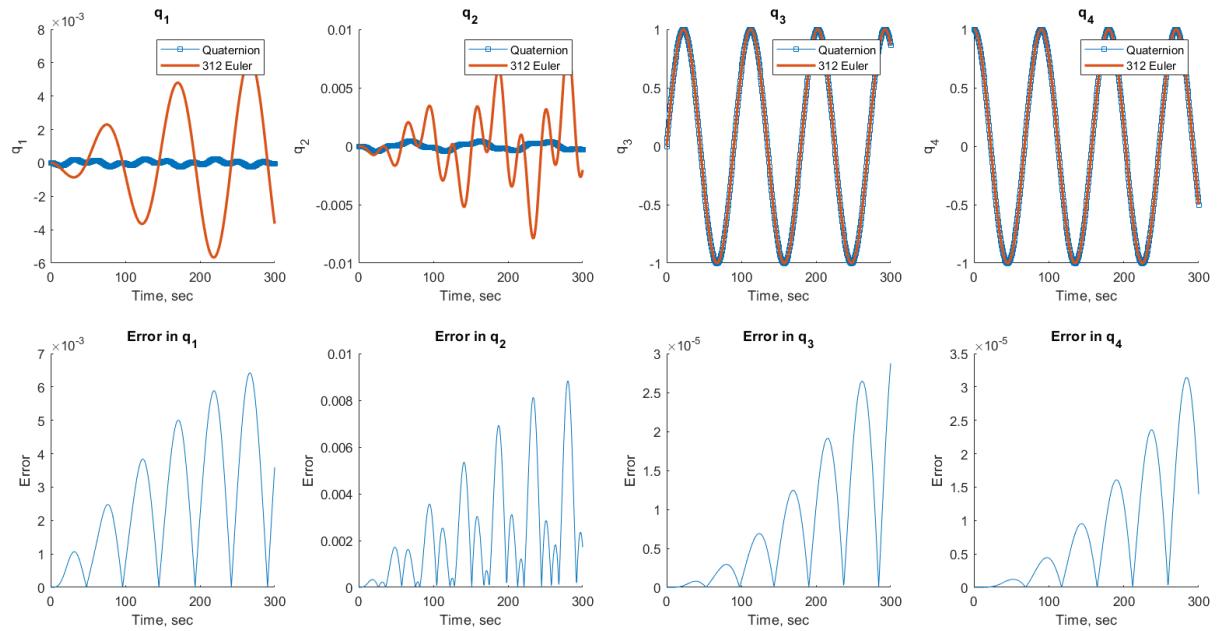


Figure 72: Comparison of quaternions from both attitude parameterizations.

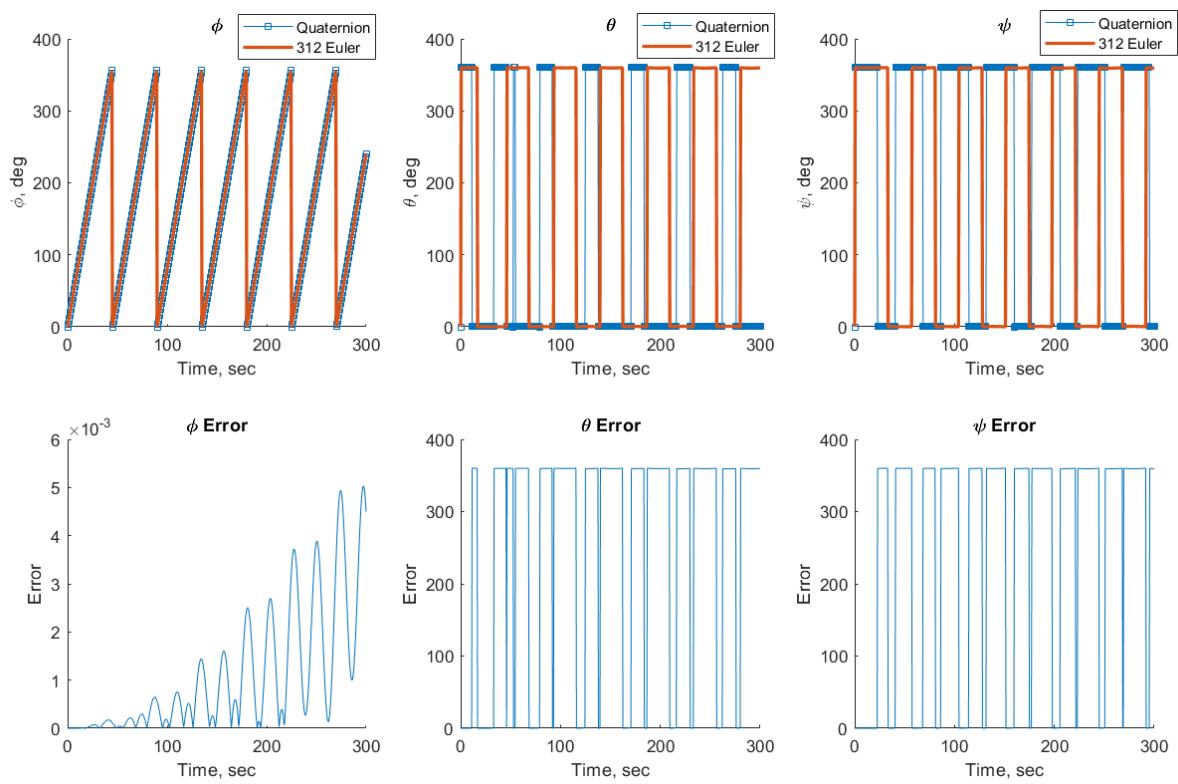


Figure 73: Comparison of Euler angles from both attitude parameterizations.

G Dual Spin Equilibrium & Stability Analysis

To ensure that our addition of a momentum wheel has not modified our simulation dynamics, we repeat the stability and equilibrium analyses required for Problem Set 3, but now with a spinning momentum wheel.

We test equilibrium in 4 cases - setting the momentum wheel's axis to be along a principal axis, we spin the system about the same principal axis and turn the momentum wheel on at 1 rad/s. Fig 74-75 show the resulting inertial-axes angular velocity and the associated 312 Euler angles. In the fourth case, we set the principal axes to be initially aligned with the RTN frame, then spin initially about the N axis. The resulting RTN frame angular velocity and 312 Euler angles are shown in 77. Note that as expected, our angular velocities remain constant and one Euler angle increases linearly.

We test stability in 3 cases - setting the momentum wheel's axis to be along a principal axis, we spin the system primarily about the same axis at 7 deg/s with the momentum wheel spinning at 1 rad/s, and give minor 0.01 deg/s perturbative spins about the other two principal axes. Fig 78-79 show the resulting inertial-axes angular velocity and the associated 312 Euler angles. Note that as expected, spinning about X and Z result in periodically stable spin, while initially spinning about Y with perturbation resulting in instability.

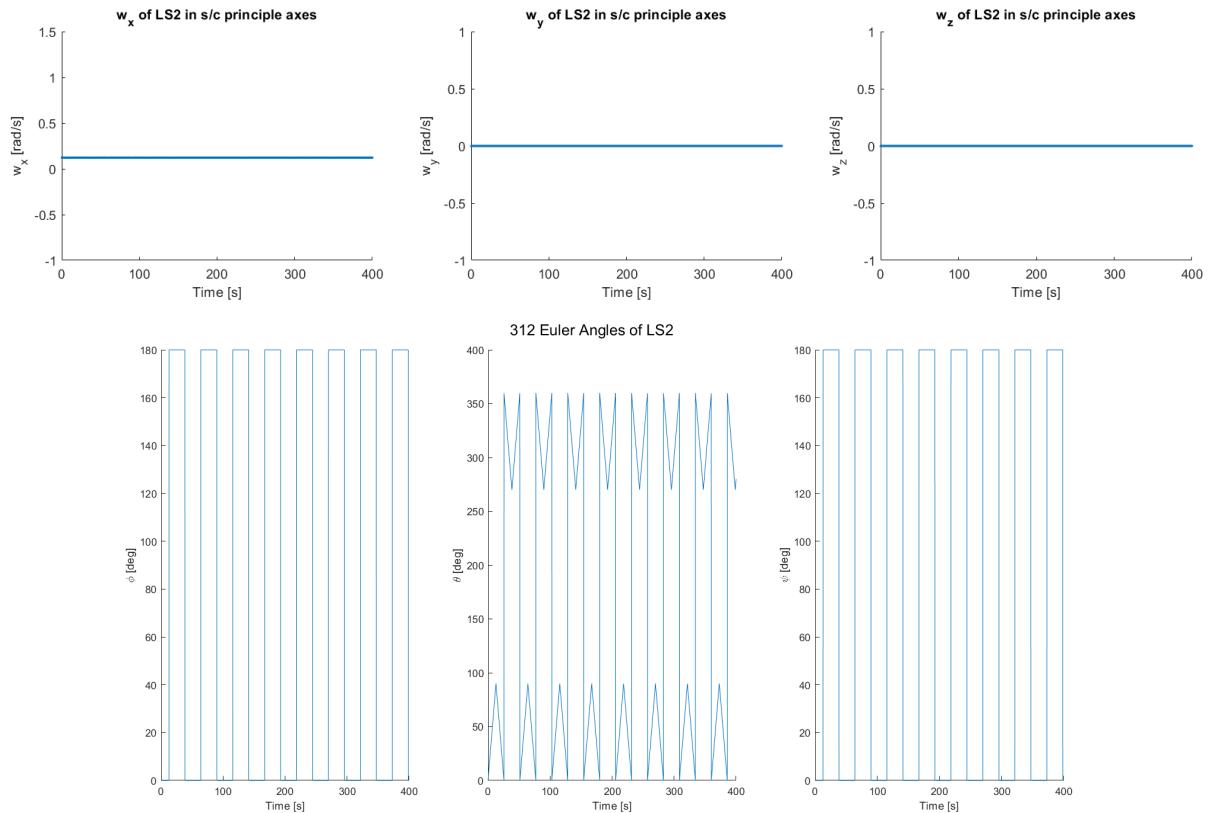


Figure 74: Dual spin equilibrium test with initial spin around X axis.

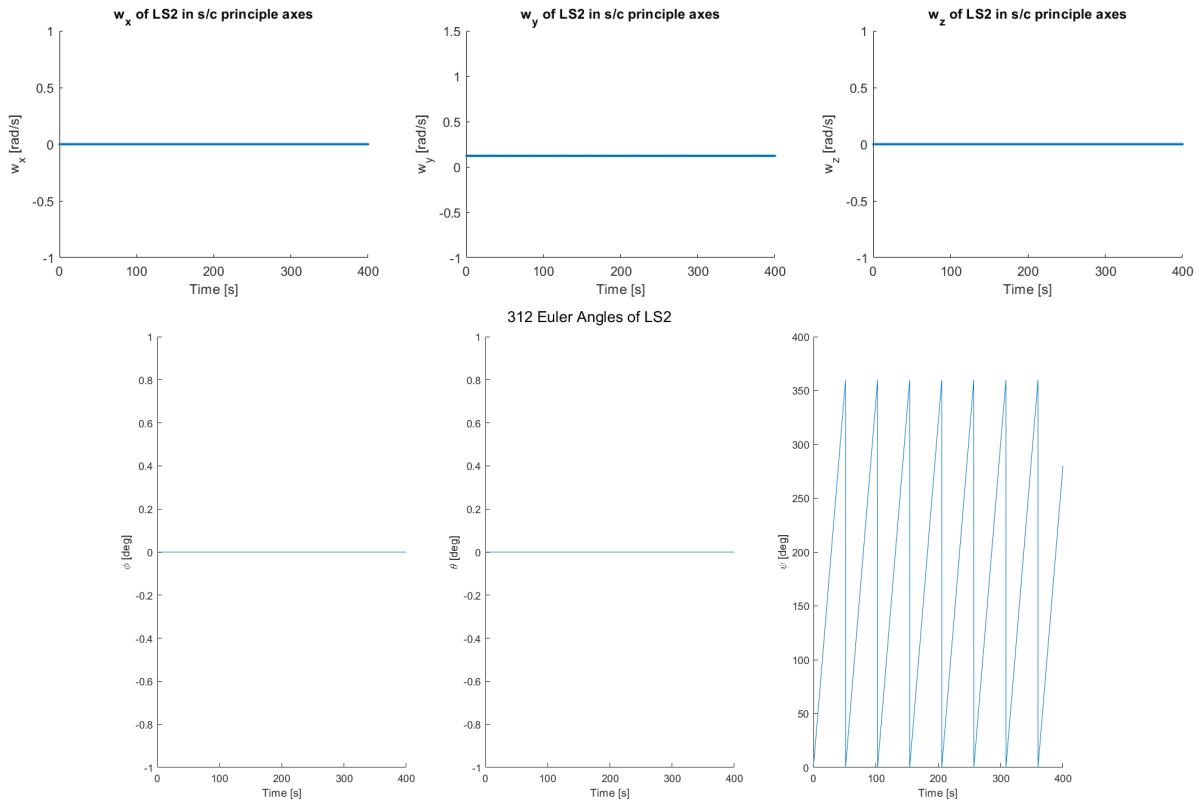


Figure 75: Dual spin equilibrium test with initial spin around Y axis.

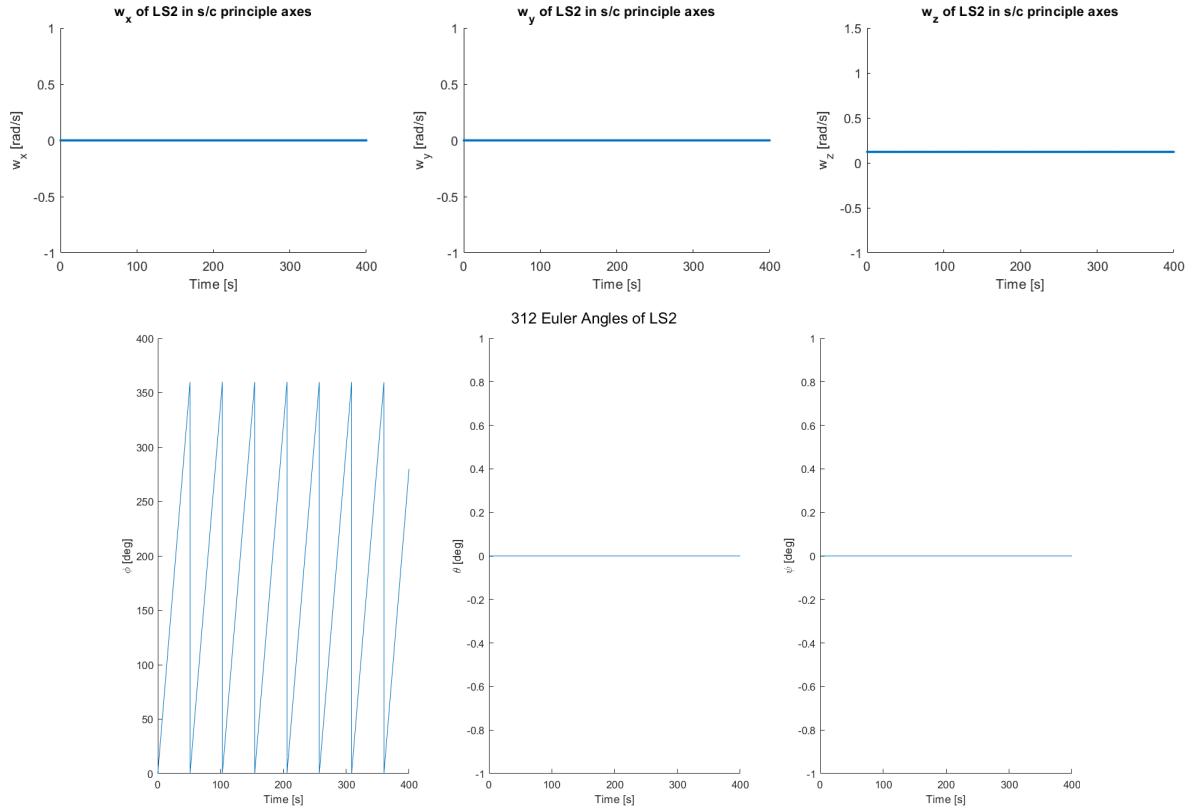


Figure 76: Dual spin equilibrium test with initial spin around Z axis.

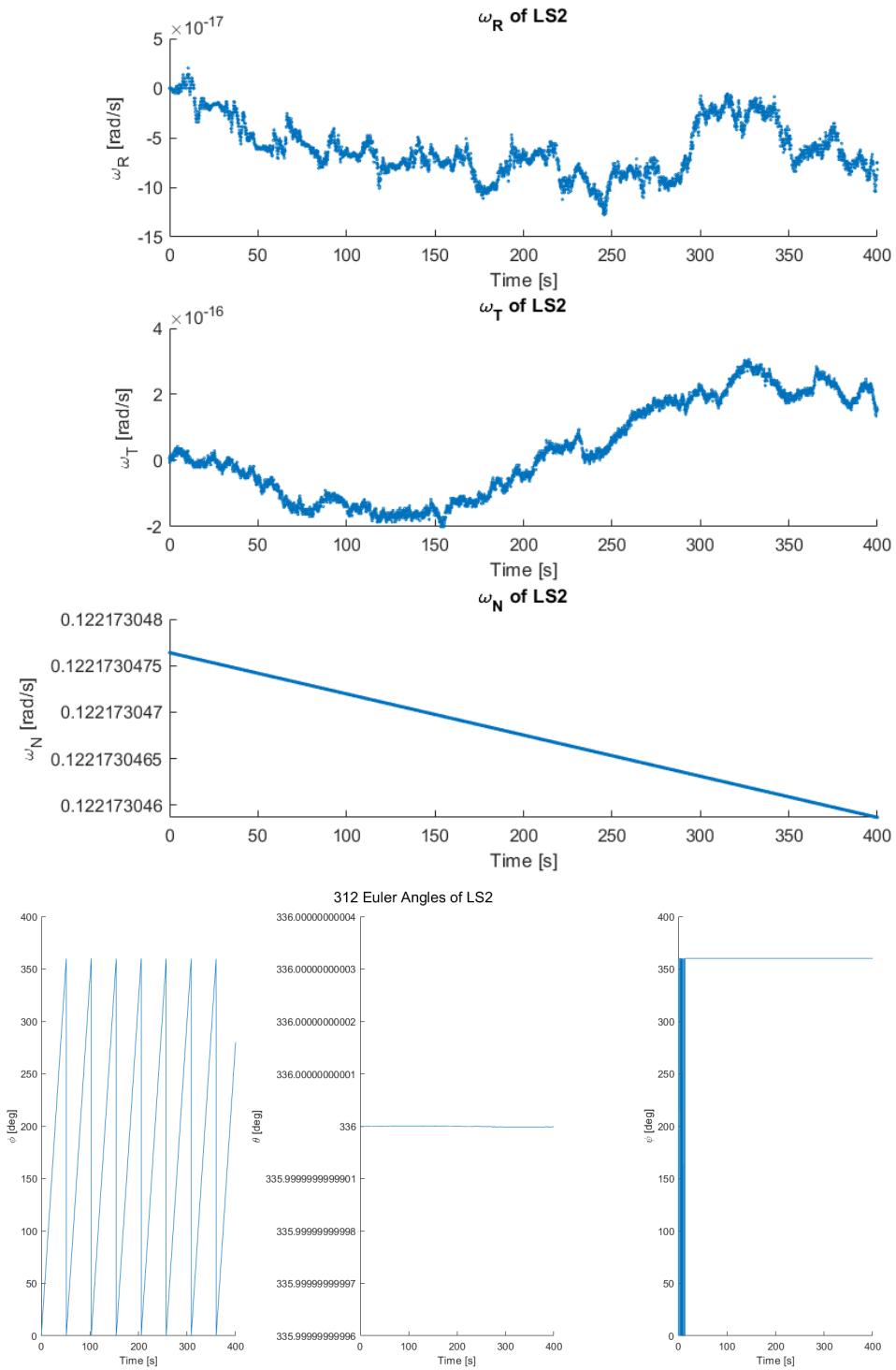


Figure 77: Dual spin equilibrium test with principal axes aligned with RTN, initial spin around N axis.

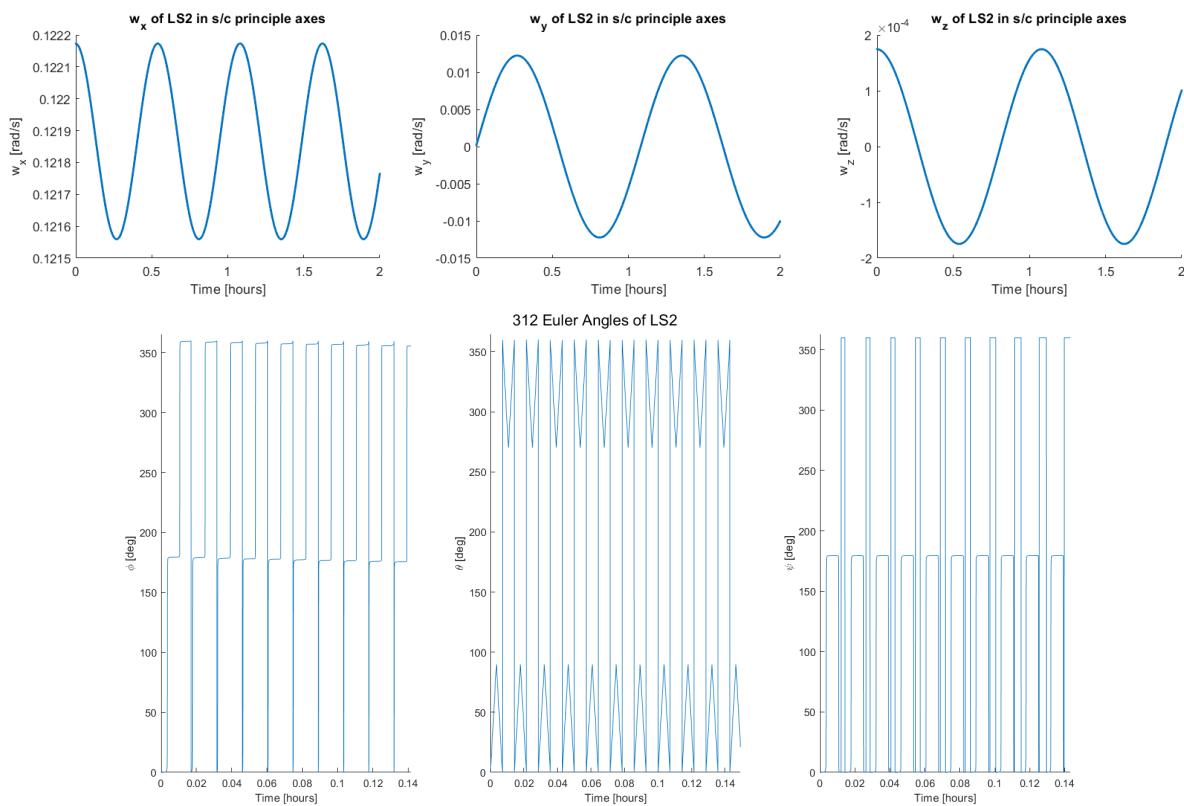


Figure 78: Dual spin stability test with initial spin mostly around X axis.

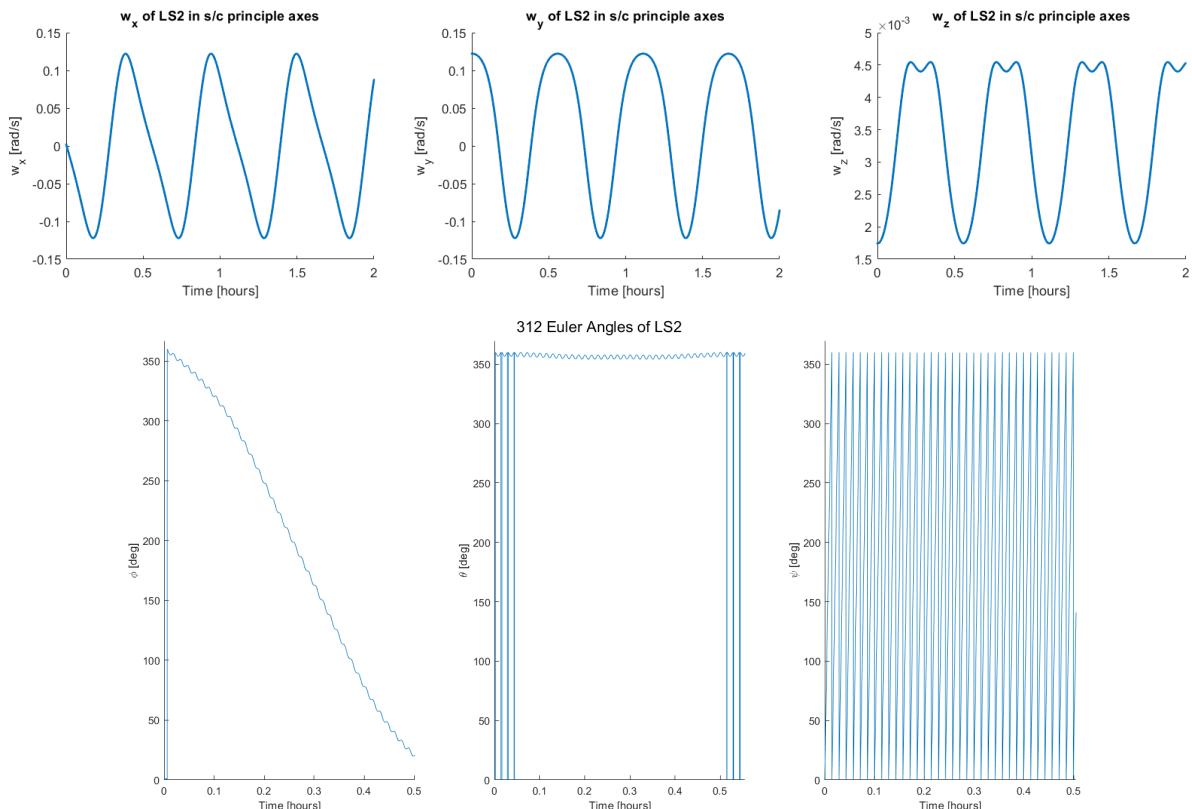


Figure 79: Dual spin stability test with initial spin mostly around Y axis.

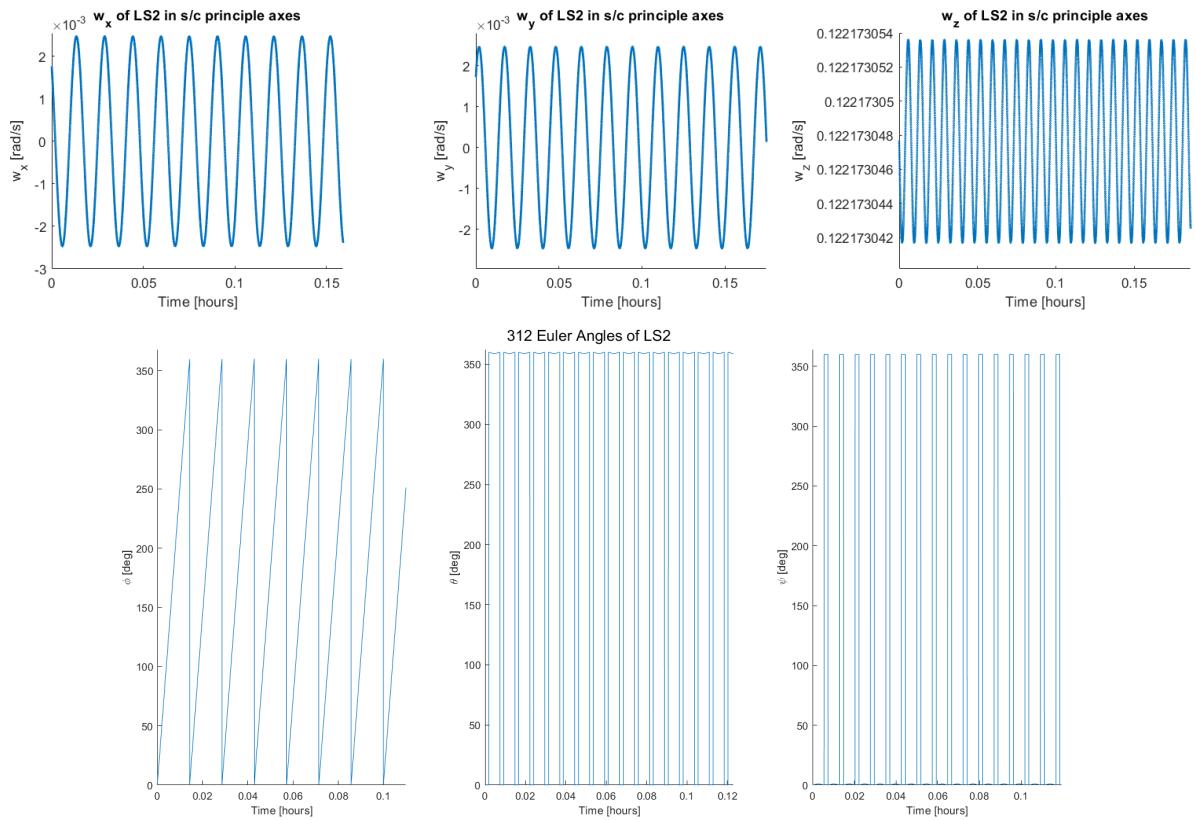


Figure 80: Dual spin stability test with initial spin mostly around Z axis.

H Gravity Gradient Stability

The gravity gradient stability is assessed in 3 cases in 2 configurations: stable (unperturbed and perturbed) and unstable. On Figure 25, these correspond to the LS2 stable config and LS2 unstable config 2, respectively.

Figure 81 shows that the spacecraft indeed remains at equilibrium if it begins in a stable orbit, with the principal axes aligned with the orbit *RTN* frame. The spacecraft performs small oscillations about the equilibrium position, but they remain bounded. However, as we can see in figure 82, even a small perturbation (of 10^{-4} rad/s) pushes the spacecraft into an unstable configuration, introducing a small oscillation that grows slowly with time. Lastly, Figure 83 shows the motion created by the gravity gradient in an unstable configuration, which behaves unpredictably.

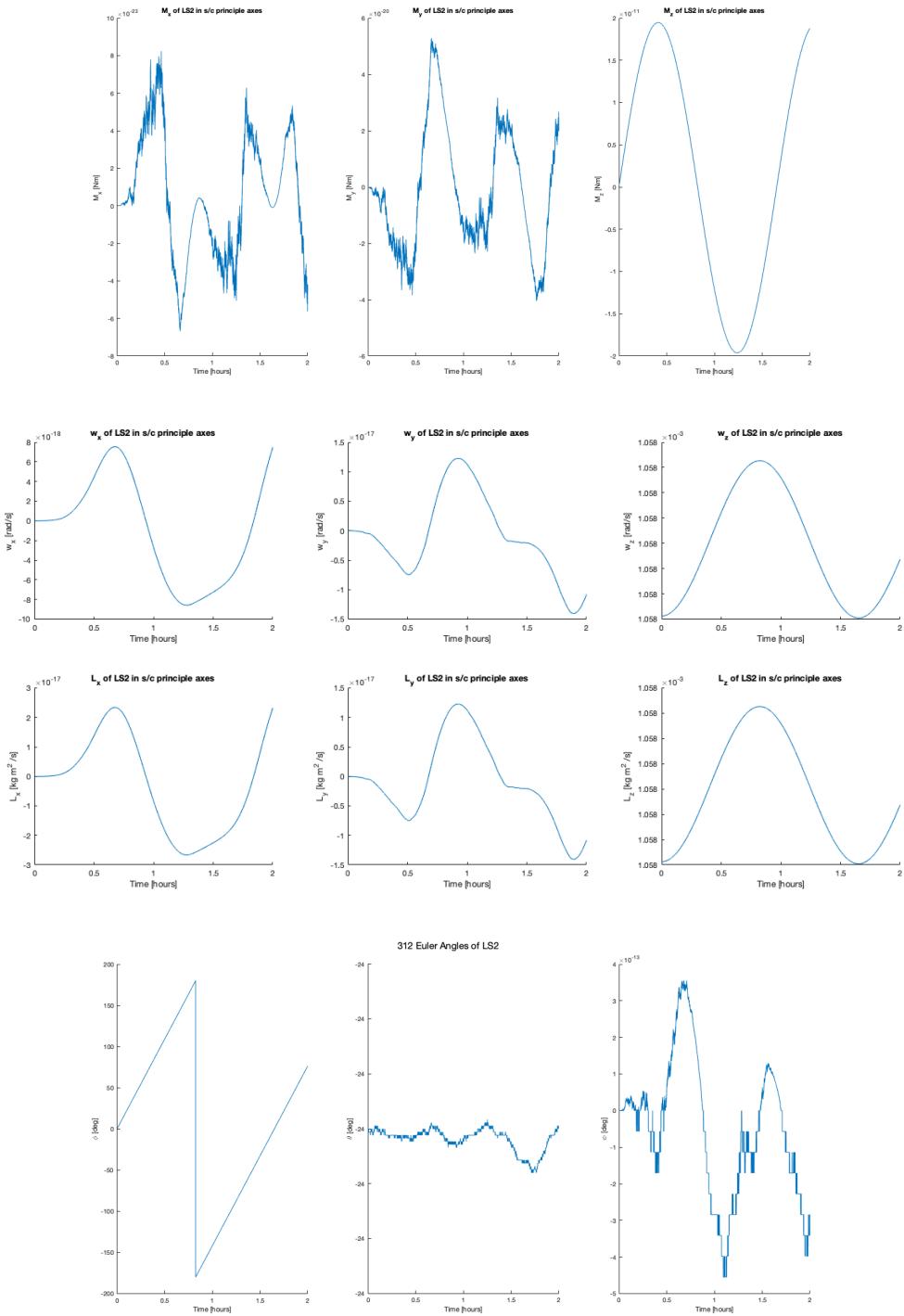


Figure 81: Gravity gradient stability assessment: stable unperturbed configuration.

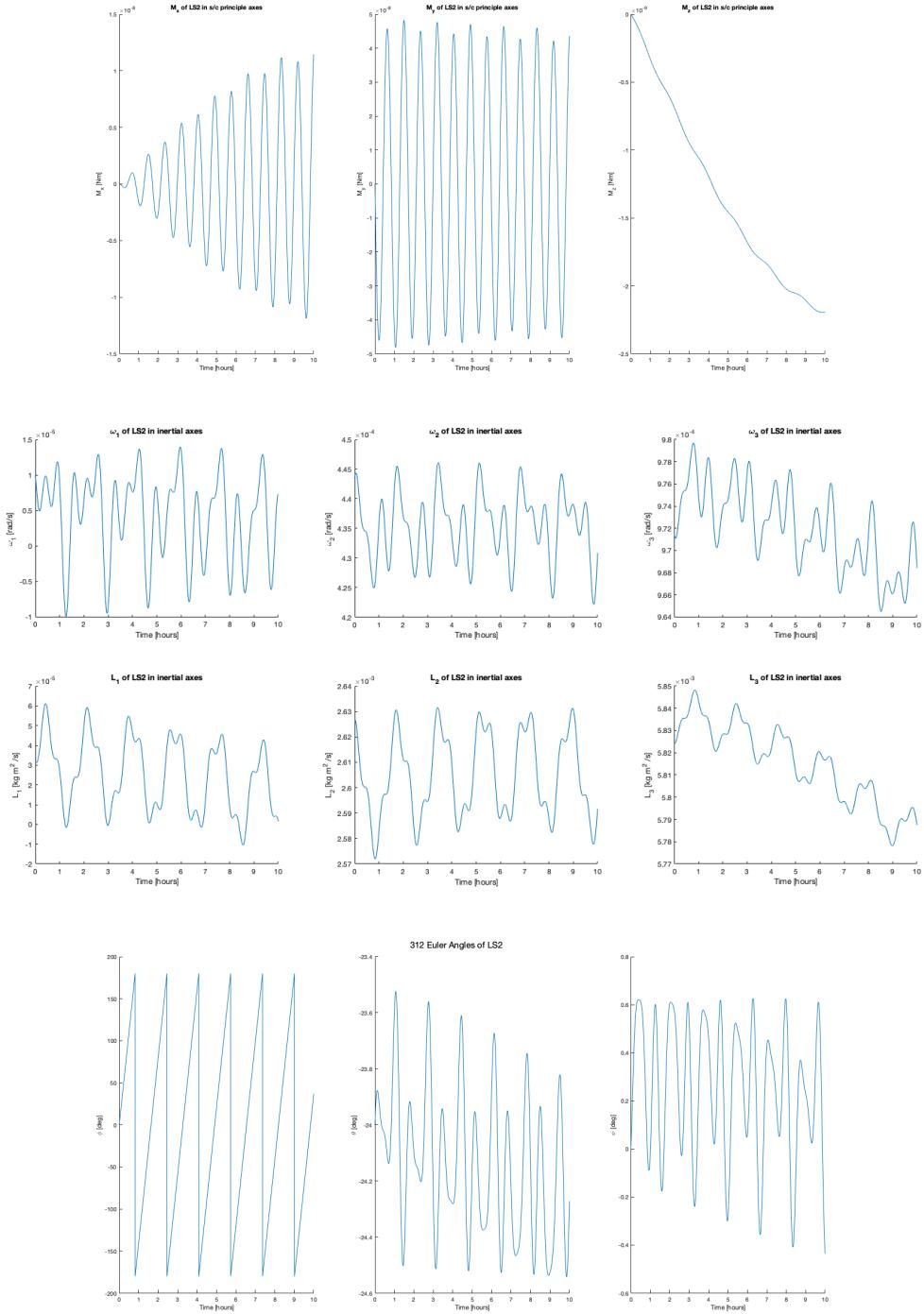


Figure 82: Gravity gradient stability assessment: stable perturbed configuration.

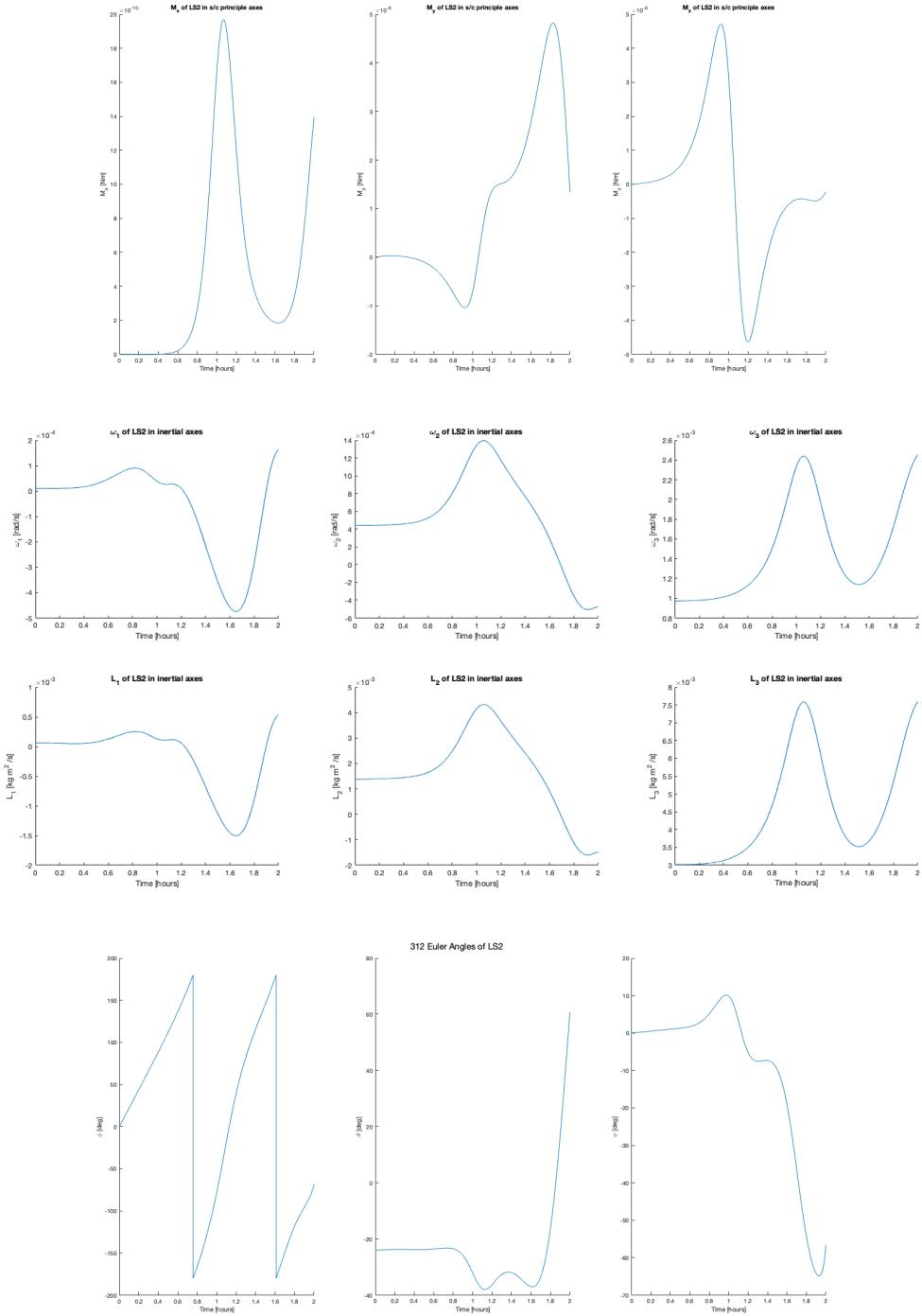


Figure 83: Gravity gradient stability assessment: unstable configuration.

I Attitude Estimation Comparisons

In the figures below, we compare our attitude estimations tactics - in Fig 84, we show the error between the q -method and deterministic method with sensor noise included. Notice that the magnitude of the error is very small, and generally shows close agreement between these two methods with the magnitude of sensor errors used.

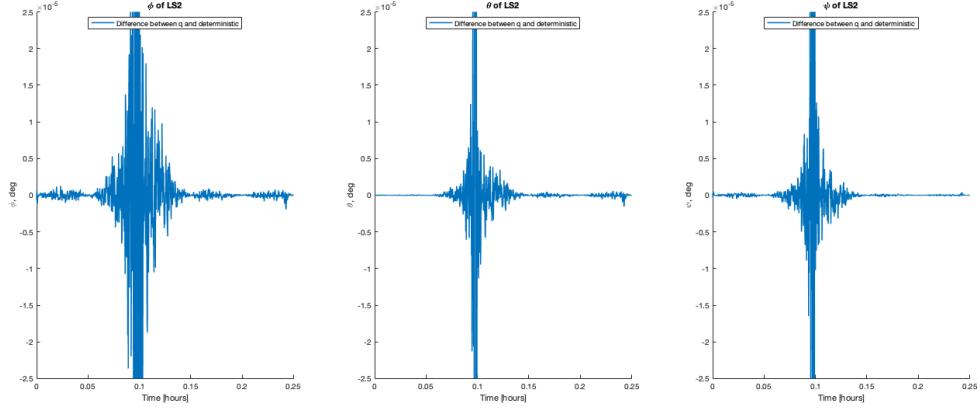


Figure 84: Error between q -method and deterministic method with sensor noise.

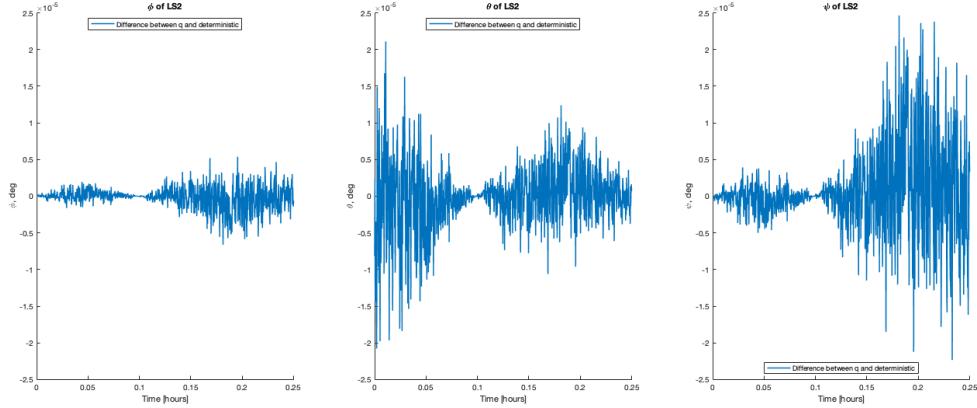


Figure 85: Error between q -method and deterministic method with sensor noise and error distribution. For comparison, Fig 84 was without error distribution.

We also analyze the effect of "distributing the error" of our measurements by re-writing our measurements and model as

$$\vec{m}_1 = \frac{\vec{B}_{meas} + \vec{S}_{meas}}{2} \text{ and } \vec{m}_2 = \frac{\vec{B}_{meas} - \vec{S}_{meas}}{2}$$

$$\vec{v}_1 = \frac{\vec{B}_{ECI} + \vec{S}_{ECI}}{2} \text{ and } \vec{v}_2 = \frac{\vec{B}_{ECI} - \vec{S}_{ECI}}{2}$$

such that

$$\vec{M} = \begin{bmatrix} | & | & | \\ \vec{p}_{meas} & \vec{q}_{meas} & \vec{r}_{meas} \\ | & | & | \end{bmatrix} \quad \text{and} \quad \vec{V} = \begin{bmatrix} | & | & | \\ \vec{p}_{ECI} & \vec{q}_{ECI} & \vec{r}_{ECI} \\ | & | & | \end{bmatrix}$$

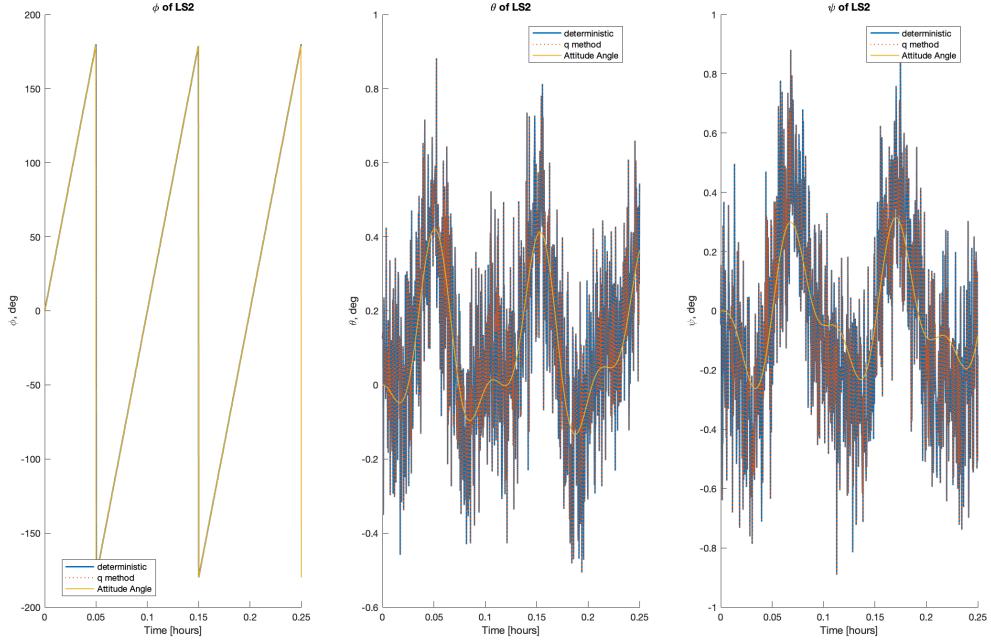


Figure 86: Attitude estimates with distributed error – non-distributed error can be seen in Section 6, Fig 32.

where $\vec{p} = \vec{m}_1$, $\vec{q} = \frac{\vec{p} \times \vec{m}_2}{|\vec{p} \times \vec{m}_2|}$, and $\vec{r} = \vec{p} \times \vec{q}$.

The effect of this distribution of error can be seen in Fig 85, 86, and 87. Note that the error redistribution generally reduces error (Fig 87), though the effect is not extremely pronounced. (Because the differences between q and deterministic are orders of magnitude less than the error between estimated and true, the error between deterministic and true with error distribution is not shown). Accordingly, the estimates provided by distributed measurements, seen in Fig 86, are very similar to those provided by Fig 32 in Section 6 of the main report. Interestingly, the difference between the two methods (q and deterministic) seems to increase at complementary times when the measurements are distributed versus not – without distribution, the difference is greatest around 0.1 hours, while with distribution, the difference is the smallest around 0.1 hours.

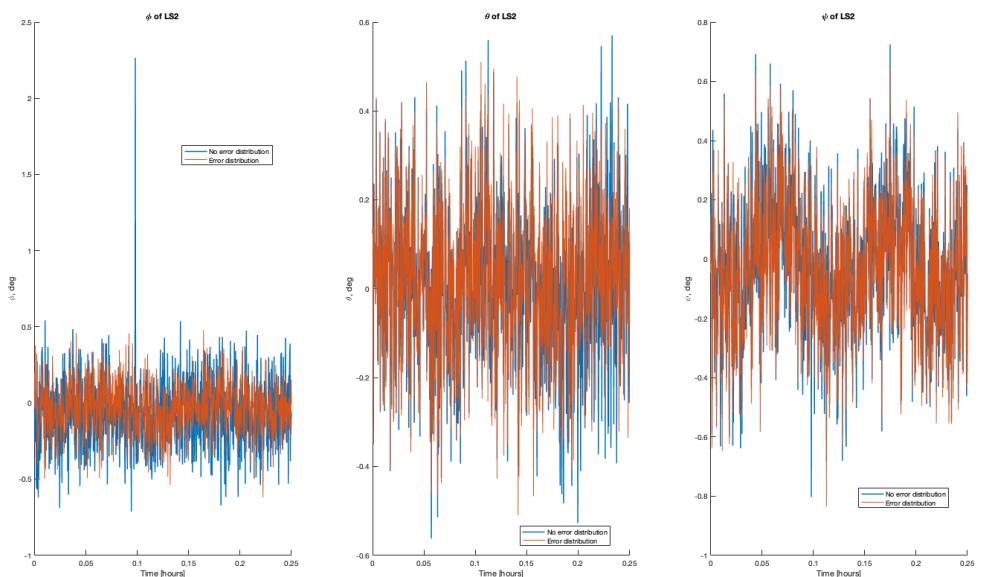


Figure 87: Error between *q*-method and true attitude with error distribution. The error without error distribution, previously shown in Fig 33, is shown again for comparison.

J Extended Kalman Filter Derivations

In this appendix, we document the derivation of key formulations for the implementation of our state estimation system, the Extended Kalman Filter (EKF). We represent our attitude state as

$$\vec{x} = \begin{bmatrix} \vec{q} \\ \vec{\omega} \\ \vec{b}_\omega \\ \vec{b}_B \\ \vec{b}_S \end{bmatrix} \quad (21)$$

where \vec{q} is our quaternion, $\vec{\omega}$ is our angular velocity expressed in principal axes, \vec{b}_ω is the gyroscope bias, \vec{b}_B is the magnetometer bias, and \vec{b}_S is the azimuth/elevation bias of the sun sensors. Note that the latter three vectors are unknown constants, which we fold into our state estimator to better make use of our measurement input.

We note that the time derivative of our state can be written as

$$\dot{\vec{x}} = \begin{bmatrix} \frac{1}{2}\Omega(\vec{\omega})\vec{q} = \frac{1}{2}\Lambda(\vec{q})\vec{\omega} \\ \frac{d\vec{\omega}}{dt} \\ \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{2 \times 1} \end{bmatrix} \quad (22)$$

Where

$$\Omega(\vec{\omega}) = \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \quad (23)$$

$$\Lambda(\vec{q}) = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \quad (24)$$

$$\frac{d\vec{\omega}}{dt} = I^{-1}(\vec{T} - (\vec{\omega} \times I\vec{\omega})) \quad (25)$$

Where 24 comes from Lecture 6, 23 comes from an ADCS summary for FALCONSAT-3 [2], and 25 comes from Euler's equation of rigid-body motion with torque \vec{T} .

Using forward Euler formulation for the discrete-time derivative approximation, we can write our state at a time index $t + 1$ as a function of a control input, u_t , the previous state at time index t ,

$$\vec{x}_{t+1} = f(x_t, u_t) = x_t + \delta t \dot{\vec{x}}_t \quad (26)$$

where w_t is white additive Gaussian (WAG) process noise with covariance Q and δt is the size of the time step. Note that our control input is simply the input torque, \vec{T} , seen in Eq 25.

We now calculate the Jacobian of our state transition function, $f(x_t, u_t)$, for use in our EKF where we are unable to apply the true, non-linear function (what has been called in lecture the

”state transition matrix”):

$$A_t = \frac{\partial f(\vec{x}, u_t)}{\partial \vec{x}}|_{\vec{x}=\vec{x}_t} \\ = \mathbf{I}_{15 \times 15} + \delta t \begin{bmatrix} \frac{1}{2}\Omega(\vec{\omega}) & \frac{1}{2}\Lambda(\vec{q}) & \mathbf{0}_{4 \times 3} & \mathbf{0}_{4 \times 3} & \mathbf{0}_{4 \times 2} \\ \mathbf{0}_{3 \times 4} & \Delta(\vec{\omega}) & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 2} \\ \mathbf{0}_{8 \times 4} & \mathbf{0}_{8 \times 3} & \mathbf{0}_{8 \times 3} & \mathbf{0}_{8 \times 3} & \mathbf{0}_{8 \times 2} \end{bmatrix} \quad (27)$$

where $\mathbf{I}_{15 \times 15}$ is the identity matrix and we've defined

$$\begin{aligned} \Delta(\vec{\omega}) &= \frac{\partial \dot{\vec{\omega}}}{\partial \vec{\omega}} \\ &= \frac{\partial}{\partial \vec{\omega}}(I^{-1}(\vec{M} - (\vec{\omega} \times I\vec{\omega}))) \\ &= I^{-1} \left([I\vec{\omega}^\times] \frac{\partial \vec{\omega}}{\partial \vec{\omega}} - [\vec{\omega}^\times] \frac{\partial I\vec{\omega}}{\partial \vec{\omega}} \right) \\ &= I^{-1} \left(\begin{bmatrix} 0 & -I_3\omega_3 & I_2\omega_2 \\ I_3\omega_3 & 0 & -I_1\omega_1 \\ -I_2\omega_2 & I_1\omega_1 & 0 \end{bmatrix} - \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix} \right) \\ &= I^{-1} \left(\begin{bmatrix} 0 & -I_3\omega_3 & I_2\omega_2 \\ I_3\omega_3 & 0 & -I_1\omega_1 \\ -I_2\omega_2 & I_1\omega_1 & 0 \end{bmatrix} - \begin{bmatrix} 0 & -I_2\omega_3 & I_3\omega_3 \\ I_1\omega_3 & 0 & -I_3\omega_1 \\ -I_1\omega_2 & I_2\omega_1 & 0 \end{bmatrix} \right) \\ &= \begin{bmatrix} \frac{1}{I_1} & 0 & 0 \\ 0 & \frac{1}{I_2} & 0 \\ 0 & 0 & \frac{1}{I_3} \end{bmatrix} \begin{bmatrix} 0 & (I_2 - I_3)\omega_3 & (I_2 - I_3)\omega_2 \\ (I_3 - I_1)\omega_3 & 0 & (I_3 - I_1)\omega_1 \\ (I_1 - I_2)\omega_2 & (I_1 - I_2)\omega_1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & \frac{I_2 - I_3}{I_1}\omega_3 & \frac{I_2 - I_3}{I_1}\omega_2 \\ \frac{I_3 - I_1}{I_2}\omega_3 & 0 & \frac{I_3 - I_1}{I_2}\omega_1 \\ \frac{I_1 - I_2}{I_3}\omega_2 & \frac{I_1 - I_2}{I_3}\omega_1 & 0 \end{bmatrix} \end{aligned} \quad (28)$$

having leveraged the fact that our angular velocity is expressed in principal axes, making the inertia tensor, I , diagonal. We consider now our measurement vector,

$$\vec{y} = \begin{bmatrix} \vec{\omega}_{meas} \\ \vec{B}_{XYZ,meas} \\ \vec{S}_{az/el,meas} \end{bmatrix} \quad (29)$$

where $\vec{\omega}_{meas}$ is the measured angular velocity (in principal axes), $\vec{B}_{XYZ,meas}$ is the measured magnetic field (in principal axes), and $\vec{S}_{az/el,meas}$ is the measured azimuth and elevation of the Sun in the sun-sensor frame. We can write this measurement at time step t as a function of our state at the same time step,

$$\begin{aligned} \vec{y}_t &= g(\vec{x}_t) + v_t \\ &= \begin{bmatrix} \omega + \vec{b}_\omega \\ A(\vec{q})\vec{B}_{ECI} + \vec{b}_B \\ M(\tilde{A}A(\vec{q})\vec{S}_{ECI}) + \vec{b}_S \end{bmatrix} + v_t \end{aligned} \quad (30)$$

where v_t is WAG measurement noise with covariance R , ω is the true angular velocity in principal axes, \vec{B}_{ECI} is the magnetic field in inertial coordinates, \vec{S}_{ECI} is the sun direction in inertial

coordinates, $A(\vec{q})$ is the direction cosine matrix from inertial to principal coordinates, and \tilde{A} is the rotation from principal frame to sun-sensor frame. Finally, we define the function $M(\vec{S})$ to be the function that transforms a unit vector to its azimuth/elevation representation,

$$M(\vec{S}) = \begin{bmatrix} \text{atan2}(S_2, S_1) \\ \text{acos}(S_3) \end{bmatrix} = \begin{bmatrix} \text{azimuth} \\ \text{elevation} \end{bmatrix} \quad (31)$$

Again, we will need the Jacobian of our measurement function, $g(x_t)$, for use in our EKF where we are unable to apply the true, non-linear function (what has been called in lecture the "sensitivity matrix"):

$$\begin{aligned} C_t &= \frac{\partial g(\vec{x})}{\partial \vec{x}}|_{\vec{x}=\vec{x}_t} \\ &= \begin{bmatrix} \mathbf{0}_{3 \times 4} & \mathbf{I}_{3 \times 3} & \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 2} \\ \Gamma(\vec{q}, \vec{B}_{ECI}) & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 2} \\ \tilde{M}(\tilde{A}A(\vec{q})\vec{S}_{ECI})\tilde{A}\Gamma(\vec{q}, A(\vec{q})\vec{S}_{ECI}) & \mathbf{0}_{2 \times 3} & \mathbf{0}_{2 \times 3} & \mathbf{0}_{2 \times 3} & \mathbf{I}_{2 \times 2} \end{bmatrix} \end{aligned} \quad (32)$$

where we define

$$\begin{aligned} \Gamma(\vec{q}, \vec{V}) &= \frac{\partial(A(\vec{q})\vec{V})}{\partial \vec{q}} \\ &= \frac{\partial}{\partial \vec{q}} \begin{bmatrix} (q_4^2 + q_1^2 - q_2^2 - q_3^2)V_1 + (2q_1q_2 + 2q_3q_4)V_2 + (2q_1q_3 - 2q_2q_4)V_3 \\ (2q_1q_2 - 2q_3q_4)V_1 + (q_4^2 - q_1^2 + q_2^2 - q_3^2)V_2 + (2q_2q_3 + 2q_1q_4)V_3 \\ (2q_1q_3 + 2q_2q_4)V_1 + (2q_2q_3 - 2q_1q_4)V_2 + (q_4^2 - q_1^2 - q_2^2 + q_3^2)V_3 \end{bmatrix} \\ &= 2 \begin{bmatrix} \vec{a} \cdot \vec{V} & \vec{b} \cdot \vec{V} & \vec{c} \cdot \vec{V} & \vec{d} \cdot \vec{V} \\ -\vec{b} \cdot \vec{V} & \vec{a} \cdot \vec{V} & -\vec{d} \cdot \vec{V} & \vec{c} \cdot \vec{V} \\ -\vec{c} \cdot \vec{V} & \vec{d} \cdot \vec{V} & \vec{a} \cdot \vec{V} & -\vec{b} \cdot \vec{V} \end{bmatrix} \\ \text{for } \vec{a} &= \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}, \vec{b} = \begin{pmatrix} -q_2 \\ q_1 \\ -q_4 \end{pmatrix}, \vec{c} = \begin{pmatrix} -q_3 \\ q_4 \\ q_1 \end{pmatrix}, \vec{d} = \begin{pmatrix} q_4 \\ q_3 \\ -q_2 \end{pmatrix}, \end{aligned} \quad (33)$$

and further define the function

$$\begin{aligned} \tilde{M}(\vec{V}) &= \frac{\partial M(\vec{V})}{\partial \vec{V}} \\ &= \begin{bmatrix} \frac{-V_2}{V_1^2 + V_2^2} & \frac{V_1}{V_1^2 + V_2^2} & 0 \\ 0 & 0 & \frac{-1}{\sqrt{1 - V_3^2}} \end{bmatrix} \end{aligned} \quad (34)$$

With these definitions in hand, we're prepared to implement our EKF. Given a prior state distribution defined by the prior mean, μ_t and the prior covariance, Σ_t , we can write the EKF algorithm as

$$\begin{aligned} A_t &= \frac{\partial f(\vec{x}, u_t)}{\partial \vec{x}}|_{\vec{x}=\mu_t} \\ \mu_p &= f(\mu_t, u_t) \\ \Sigma_p &= A_t \Sigma_t A_t^T + Q \\ C_t &= \frac{\partial g(\vec{x})}{\partial \vec{x}}|_{\vec{x}=\mu_p} \\ K_t &= \Sigma_p C_t^T (C_t \Sigma_p C_t^T + R)^{-1} \\ \mu_{t+1} &= \mu_t + K_t(y_{t+1} - g(\mu_p)) \\ \Sigma_{t+1} &= (I - K_t C_t) \Sigma_p (I - K_t C_t)^T + K_t R K_t^T \end{aligned} \quad (35)$$

Where our pre- and post-fit residuals are

$$\begin{aligned} z_{t+1,pre} &= y_{t+1} - \mu_p \\ z_{t+1,post} &= y_{t+1} - \mu_{t+1} \end{aligned} \quad (36)$$

Additional EKF Notes

We can further consider the partial derivative of our state transition function, $f(x_t, u_t)$, with respect to the control input, u_t . As noted previously, our input to the system is simply the input torque (one could also add disturbance torques into this input, or model them as uncertainty on the input itself). The resulting "control input matrix" for a control torque input in principal axes u_t is simply

$$\begin{aligned} B_t &= \frac{\partial f(x_T, u)}{\partial u} \Big|_{u=u_t} \\ &= dt I^{-1} \end{aligned} \quad (37)$$

where I is the inertia tensor (here in principal axes, i.e. diagonal).

Quaternion normalization can become an issue when propagating attitude states, as our first-order derivative implementation does not preserve quaternion length. Therefore, we can build in quaternion normalization to our update step: consider the matrix Q , which is identical to \mathbf{I}_{15} except that the upper left (4×4) submatrix is $\frac{1}{|q|} \mathbf{I}_4$ - we can then apply this matrix to our state transition matrix after the forward-Euler step to normalize the quaternion:

$$\vec{x}_{t+1} = f(x_t, u_t) = Q(x_t + \delta t \vec{v} \dot{c} x_t) \quad (38)$$

This modification changes our state transition matrix, to be

$$\begin{aligned} A_t &= \frac{\partial f(\vec{x}, u_t)}{\partial \vec{x}} \Big|_{\vec{x}=\vec{x}_t} \\ &= \tilde{Q} + \delta t \begin{bmatrix} \frac{1}{2}\Omega(\vec{\omega}) \odot \mathcal{Q} & \frac{1}{2|q|}\Lambda(\vec{q}) & \mathbf{0}_{4 \times 3} & \mathbf{0}_{4 \times 3} & \mathbf{0}_{4 \times 2} \\ \mathbf{0}_{3 \times 4} & \Delta(\vec{\omega}) & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 2} \\ \mathbf{0}_{8 \times 4} & \mathbf{0}_{8 \times 3} & \mathbf{0}_{8 \times 3} & \mathbf{0}_{8 \times 3} & \mathbf{0}_{8 \times 2} \end{bmatrix} \end{aligned} \quad (39)$$

where \odot is the element-wise multiplication operator, and we define

$$\begin{aligned} \mathcal{Q} &= \frac{\partial}{\partial q} \left(\frac{q}{|q|} \right) \\ &= \frac{1}{|q|^3} \begin{bmatrix} |q|^2 - q_1^2 & -q_1 q_2 & -q_1 q_3 & -q_1 q_4 \\ -q_1 q_2 & |q|^2 - q_2^2 & -q_2 q_3 & -q_2 q_4 \\ -q_1 q_3 & -q_3 q_2 & |q|^2 - q_3^2 & -q_3 q_4 \\ -q_1 q_4 & -q_2 q_4 & -q_3 q_4 & |q|^2 - q_4^2 \end{bmatrix} \end{aligned} \quad (40)$$

and \tilde{Q} is identical to \mathbf{I}_{15} except that the upper left (4×4) submatrix is \mathcal{Q} .

Finally, we note that an EKF formulation without the inclusion of sensor bias can be achieved by simply taking submatrices of our derivations above: the (7×7) upper left sub-matrix of A_t , and the first seven columns of the C_t sensitivity matrix.

K Extended Kalman Filter Result Analysis

In the figures below, we analyze the performance of the EKF implemented without the estimation of sensor bias, as discussed in Problem Set 7. Figures 88 and 89 show the attitude error and the absolute angular velocity error, respectively, from our EKF. The attitude errors are initially quite large (as expected, given our poor initial estimate and high uncertainty). The approximate range of these errors is encapsulated well by the associate covariance output by the EKF, presented in Figures 90 and 91; note that in general, the error of our estimate is within the 95% confidence interval suggested by the associated covariance. We've used the formulation developed in John B. Schlepe's thesis [11] to transform the covariance of our quaternions to that of the associated Euler angles.

Finally, we consider our residuals (both the pre-fit and post-fit residuals whose computation is described in Appendix J) for our various measurements, as plotted in Figures 92-94. The plots include the 95% confidence interval based on the covariance of the last 1000 data points - the resulting values are very similar to the sensor errors that we use for generating sensor noise (10 deg/s for the gyroscope, 1 μ T for the magnetic sensors, and 3 deg for the azimuth and elevation), as expected.

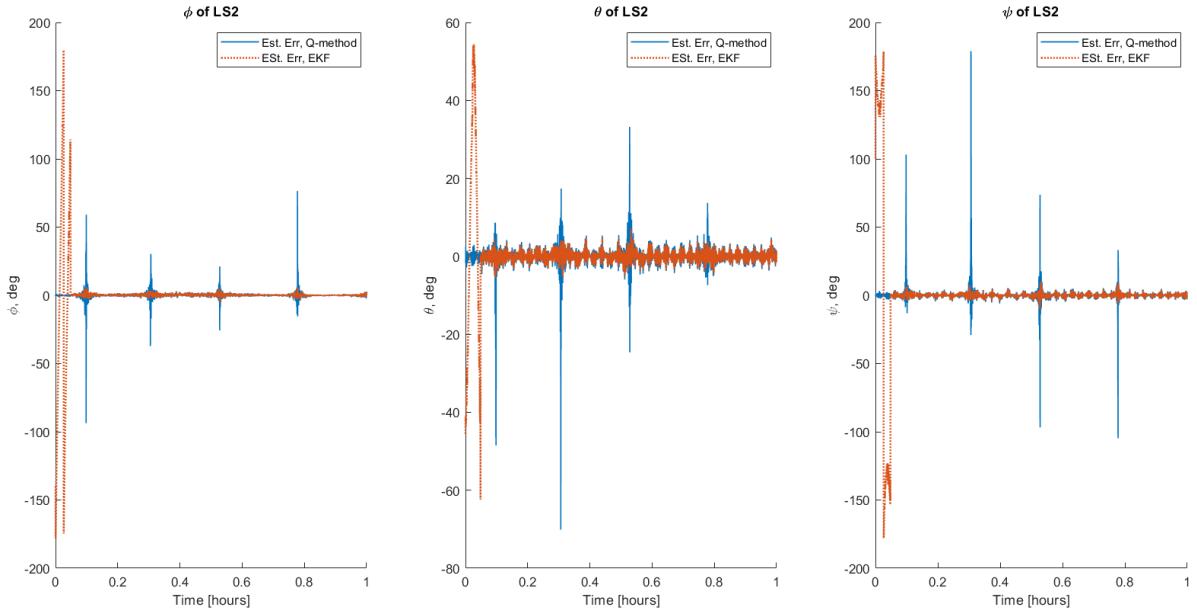


Figure 88: Attitude error from EKF state estimation.

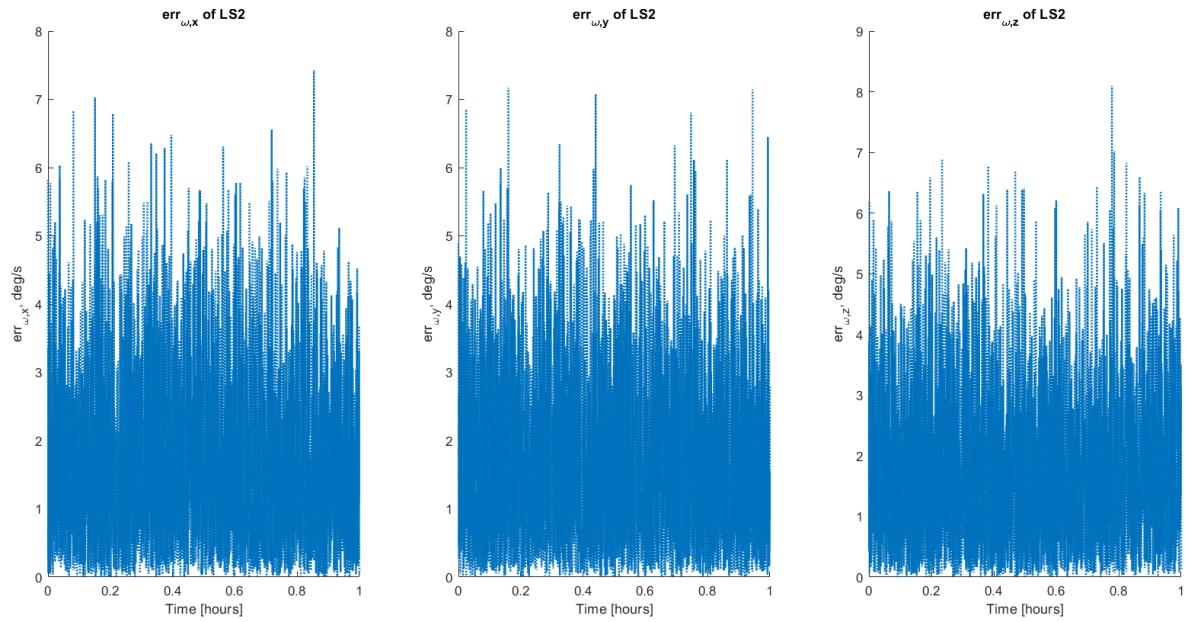


Figure 89: Angular velocity error from EKF state estimation.

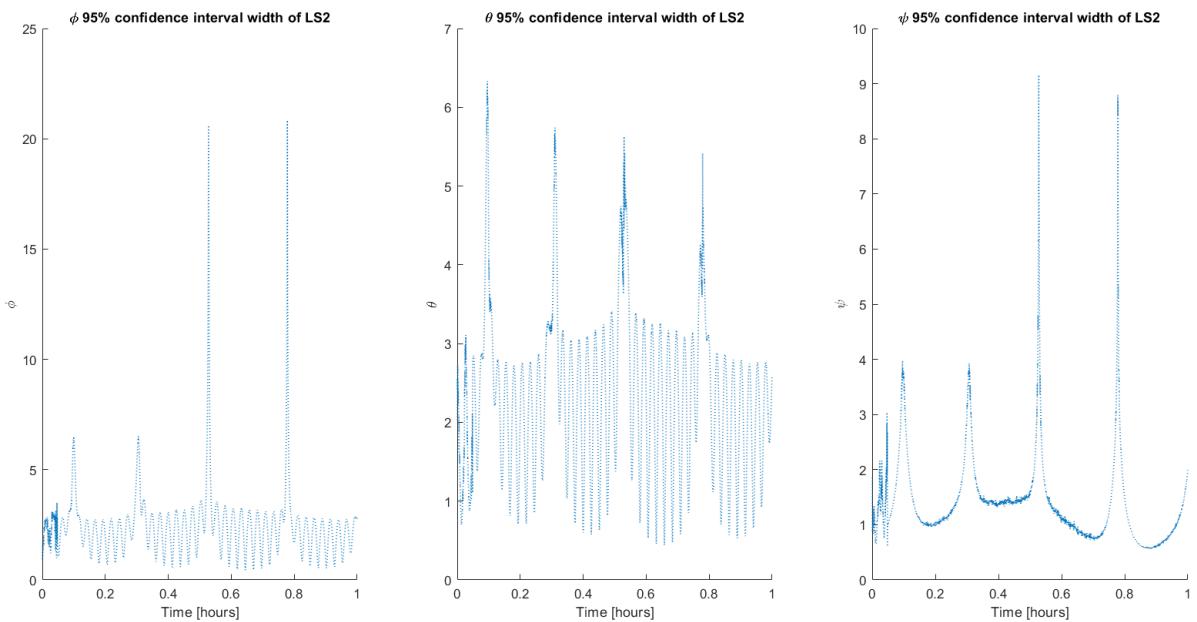


Figure 90: Attitude confidence interval from EKF state estimation.

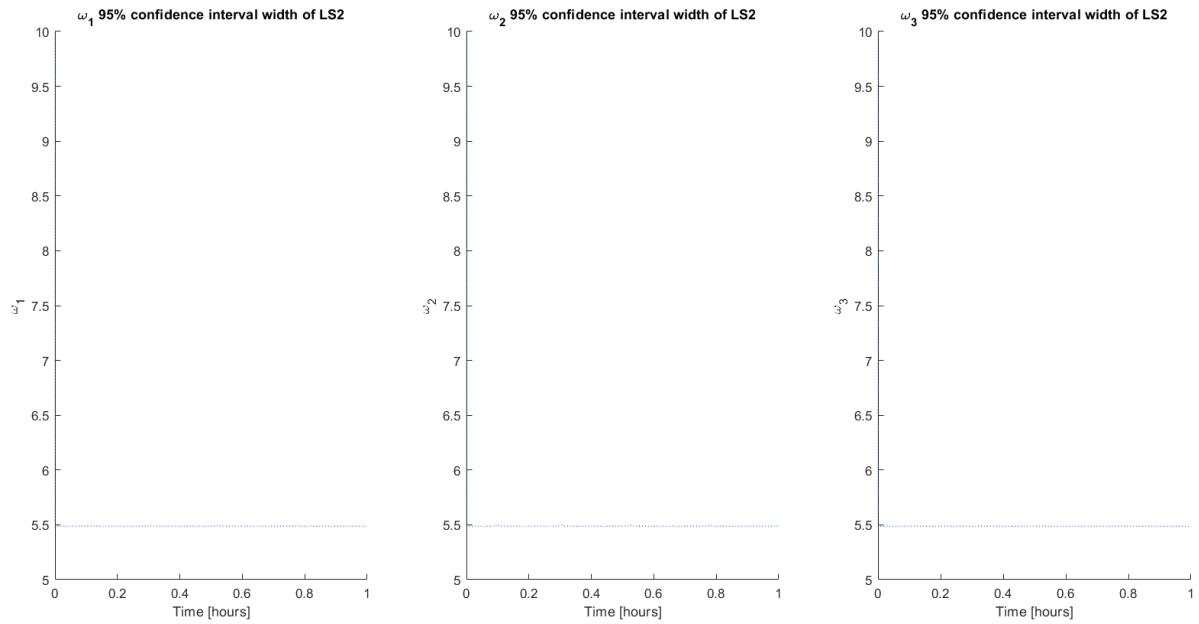


Figure 91: Angular velocity confidence interval from EKF state estimation.

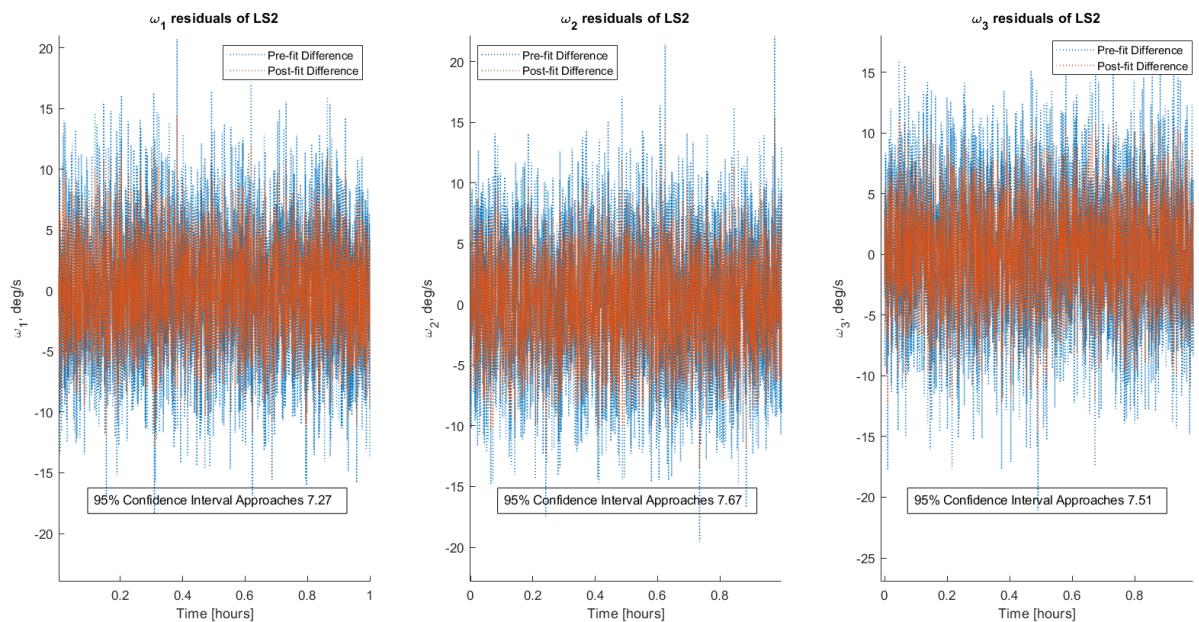


Figure 92: Angular velocity pre- and post-fit residuals from EKF state estimation.

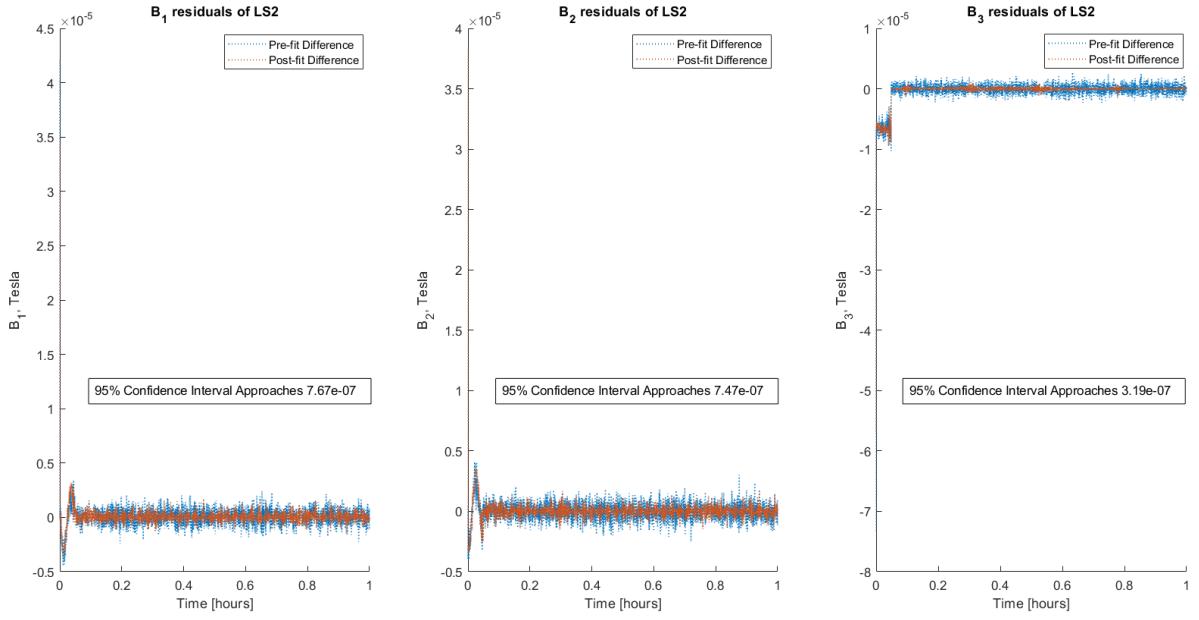


Figure 93: Magnetic sensor pre- and post-fit residuals from EKF state estimation.

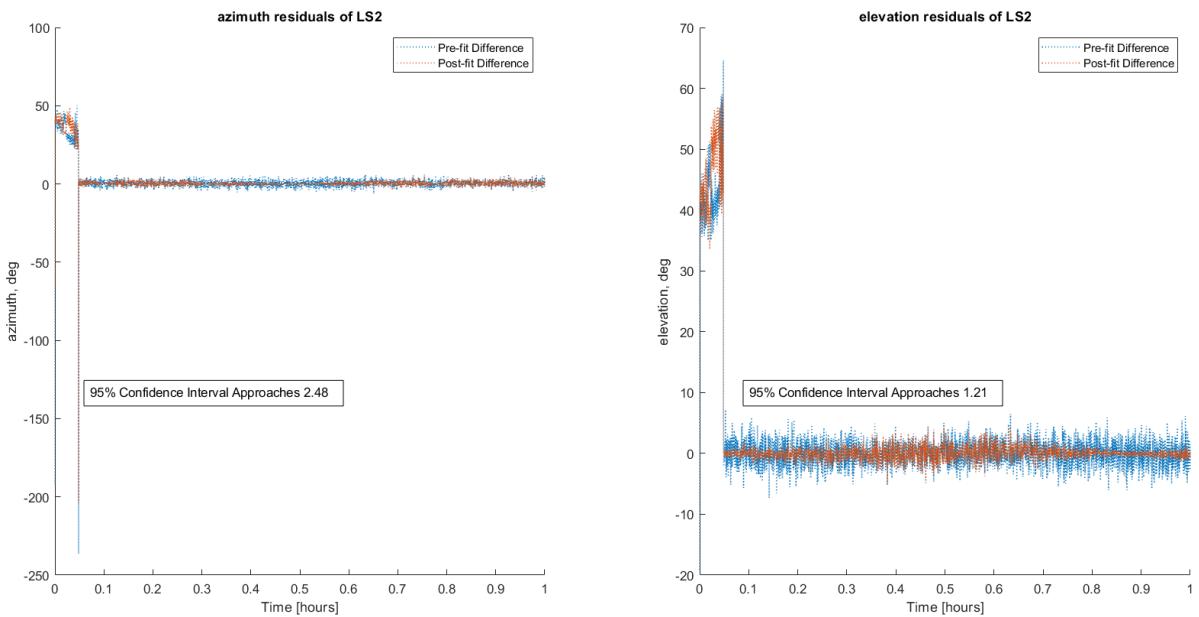


Figure 94: Sun sensor pre- and post-fit residuals from EKF state estimation.