Orbit Determination from a Remote Telescope

Group 8, Section 0101

Prashanta Aryal

Ronak Chawla

Nico Lagendyk

Ryan Quigley

Table of Contents

I. Abstract

Our group was given observation data of the 299°E Cluster of Echostar and Amazonas satellites. This data set includes information about azimuth, elevation, right ascension, site longitude, site latitude, and site altitude at a range of date times. Observations were made on the 29th and 30th of October 2020 from OAP - Chile (-30.142°N, -70.695°E, 1500 m). We were tasked with analyzing the subset “satBset4” which corresponds to 39008 – Echostar-16, determining the orbit of best fit and conducting studies using this orbit, examining how different observation sets and force models affect our prediction. The resources used in our orbit determination analysis include MATLAB, SNaG-App, and the course textbook: *Space Flight Dynamics and Navigation* by Liam Healy.

II. Initial Orbit Determination

Before starting our in-depth orbit determination analysis, we had to determine an initial orbit. In accordance with recommendations from the project description, we selected several sets of three vectors from four observation sets which are larger sections of data from the first night, “opt2satBset4.”

The best initial orbit was determined from finding the observation time in an observation set that had the smallest RMS value correlating to the smallest residuals once propagated to the second night and compared to the night 2 observations in “opt3satBset4.”

The four observation sets were chosen as follows: data points close to each other from early in the first night, from the middle of the first night ,and from late in the first night and one set of spread out data points throughout the first night. The “night 1 early” set includes observation indices from 10 to 110, the “night1 mid” set is from 350 to 450 and the late set is from 600 to 700. The first three observation sets incremented in steps of 4 giving 26 data points where the last observation set, “night 1 spread” indexed from 10 to 700 in steps of 10 giving 70 data points spread evenly throughout the entire first night.

Initially, the “night1 early” observation set was the first 100 observations (1-100) included in our “opt2satBset4” dataset. However, a gap in the data caused issues while running the code. This gap can be seen in the table below (**Figure 1**) and left a 26 minute period without observation data. The commercial service ExoAnalytic did not identify what caused this gap in coverage.

Obs #	Datetime	Az_ deg	El_ deg	RA_deg	Dec_ deg	Lat_ deg	Lon_deg	Alt_m
50	10/29/2020 0:52	17.6400	53.5108	350.3852	4.8420	-30.1428	-70.6945	1500
63	10/29/2020 0:52	17.6400	53.5109	350.5077	4.8418	-30.1428	-70.6945	1500
69	10/29/2020 0:53	17.6396	53.5108	350.5565	4.8420	-30.1428	-70.6945	1500
186	10/29/2020 1:19	17.6354	53.5139	357.1289	4.8382	-30.1428	-70.6945	1500
291	10/29/2020 1:21	17.6357	53.5141	357.6917	4.8380	-30.1428	-70.6945	1500
307	10/29/2020 1:21	17.6357	53.5143	357.7653	4.8377	-30.1428	-70.6945	1500

Figure 1. Rows 7 to 12 from opt2satBset4, the full data set has 788 lines of observation data.

Within each observation set three combinations of three indices (observation times) were selected such that one set was concentrated at the beginning of the observation set, one spread across the data set and one concentrated at the ends of each observation set. With all indices or observation times included we analyzed 36 combinations to determine the best initial orbit.

We created a MATLAB function `IOD.m` that takes the observation set timespan (`10:4:110`) and a vector of three indices (`[10 15 20]`) and returns the rms values for these three indices (observation times) and all the relevant information from the calculations which will be used at later points in the project. We learned the vector of three observation times requires indices that were separated by equal or greater increments than the timespan after seeing a dot indexing error for `[10 11 12]`. All the steps for initial orbit determination are wrapped into this function.

To begin the initial orbit determination process, the data needed to be reformatted which is done in a separate function called `Get_observations.m`. This function takes the observation data set and the indices and sets up three observation structures which have the date time, right ascension, declination and the site latitude, longitude and altitude. These three observations can then be passed through to the Gauss angles-only and Gibbs calculation methods to produce three sets of position and velocity vectors at unique epochs that can be used for initial orbit determination.

The `Gauss.m` function utilizes the two angle measurements (right ascension and declination) from three different points in time to solve for the cartesian position of the object at each respective epoch. The Gauss function requires the observations at three times since six quantities are needed to fully define an orbit. Using the procedure outlined in section 7.3.4 of *Space Flight Dynamics and Navigation*, three Earth Centered Inertial (ECI) positions are

calculated for each of the epochs. When constructing our Gauss function, we initially ran into the issue of having imaginary positions outputted. This was due to the Gauss method taking the roots of an 8th order polynomial where some roots are imaginary. To resolve the issue, we inserted an if statement in our Gauss.m function to use only the real root that is greater than the radius of earth as a logical position.

The `Gibbs.m` function takes the three ECI positions and calculates their corresponding velocities by assuming that the orbit lies in the same plane. The function itself follows the procedure from section 7.3.2 of the textbook to produce three velocities. When checking our Gibbs function with the example in the textbook, there was a negligible discrepancy between our output and the output in the text. This could be due to the text rounding the solution to the gibbs example or a slightly different process used for the gibbs method.

Once all the positions and velocities were calculated, we could combine them using the `pvt.m` function from SNaG-App. This function created three initial estimates of position and velocity at an epoch which were outputted to the main code so we could use them during the orbit estimation section of the project.

For each of the initial estimates we used the `propagate_to_times.m` function from SNaG-App to find predicted ephemeris data for night 2. The initial estimates were propagated to the same span of indices in the second night respectively. The output data was in terms of position and velocity so we used a simple function (`eci_to_azelrn.m`) from SNaG-App to convert the ECI positions to azimuth and elevation measurements which could be compared to the observation data from night 2 (“opt3satBset4”).

The predicted observation angles for azimuth and elevation were compared to their counterparts in the night 2 data set using the function `RMS.m`. This function performed a root mean square (RMS) calculation for the two angles using the equation provided in the project definition document (**Figure 2**). However due to the fact that the angles lie on a circle a simple subtraction couldn’t produce the smallest angle between the two values so a combination of MATLAB’s `mod` and `min` functions had to be used to ensure the angles are less than 180°. In addition to calculating the RMS value, the code prints out two plots which have the residual values of the azimuth and elevation angles and returns these values to the output.

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\zeta_i^{\text{obs}}-\zeta_i^{\text{pred}}\right)^2+\left(\epsilon_i^{\text{obs}}-\epsilon_i^{\text{pred}}\right)^2},$$

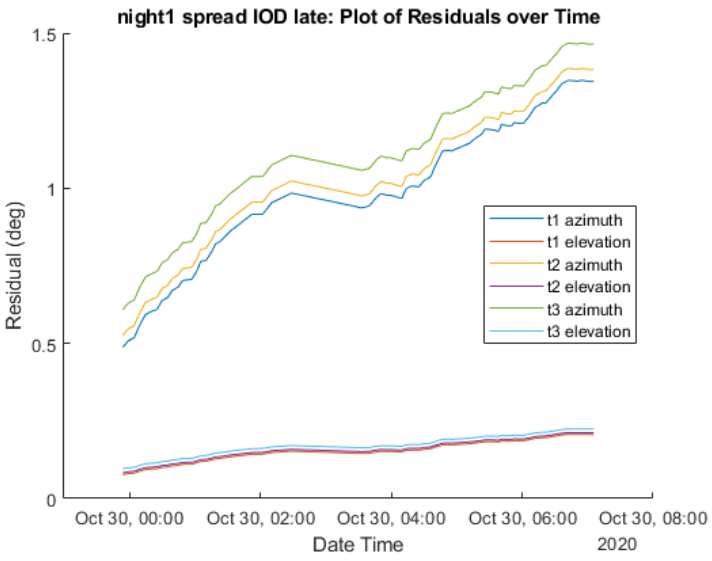
Figure 2. Root Mean Square formula from project definition document

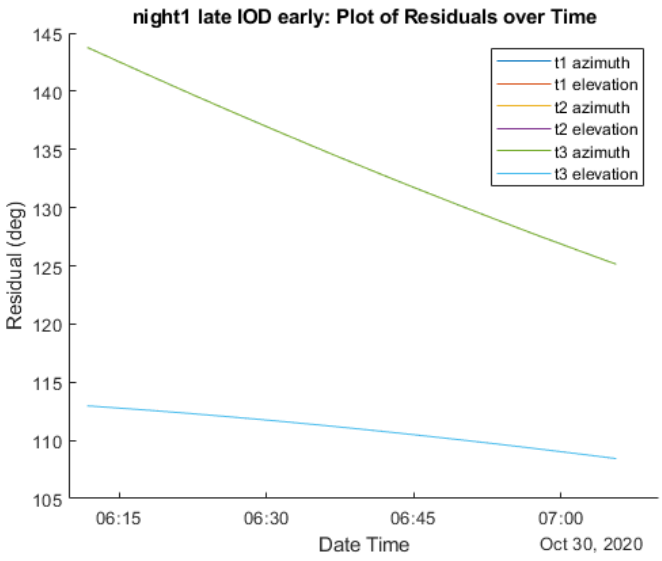
Once the RMS values for each initial estimate are calculated the graphs are combined so all the different epochs can be analyzed. This graph is then saved in a separate folder for future reference. We ran into an issue with this line of code since members of our team worked on both macOS and WindowsOS which have different syntax for file paths (/ vs. \\). To resolve this issue we included an if statement using MATLAB's functions `ismac` and `ispc`.

Although the initial orbit determination only requires one pair of position and velocity vectors, the IOD function calculates three initial orbits. The three corresponding RMS values are then compared to one another to determine the smallest one which corresponds to the best estimation from a vector set.

The last element of the `IOD.m` function is to output the three position and velocity pairs, the three respective observation time indices and the best RMS value with its corresponding observation time index.

We then repeated the `IOD.m` function 12 times, 3 for each observation set. We compared all the best RMS values to find the best initial estimate. This comparison can be seen in **Figure 5** below.





Figures 3 and 4. Plots of residuals over time from two iterations of the IOD function

The graphs from each iteration of the `IOD.m` function varied significantly, but there were a few trends. As seen in **Figure 5**, every early vector set from the 3 concentrated observation sets had drastically higher RMS values (130.3, 82.692, 174.21). Two examples of how the graphs varied can be seen above between the worst and best RMS. One interesting observation is that the residual for the observation set with the largest RMS is getting smaller over time but the one with the smallest RMS is growing over time. The night1 spread observation set having a growing RMS makes more sense as the predicted values for azimuth and elevation are getting worse over time which is expected when looking at observations in the past.

observation_set	best_rms_value
"night1_early.IOD.early"	130.3
"night1_early.IOD.spread"	1.4566
"night1_early.IOD.late"	1.9517
"night1_mid.IOD.early"	82.692
"night1_mid.IOD.spread"	4.4771
"night1_mid.IOD.late"	131.79
"night1_late.IOD.early"	174.21
"night1_late.IOD.spread"	1.353
"night1_late.IOD.late"	117.29
"night1_spread.IOD.early"	1.2093
"night1_spread.IOD.spread"	10.716
"night1_spread.IOD.late"	1.0366

Figure 5. Table showing best RMS values for each of the 12 iterations of the IOD function

The best RMS value of 1.0366 resulted from the best “night1_spread.IOD.late” initial estimate which corresponds to the date time of 29-Oct-2020 07:03:42 and a position and velocity of:

	x component	y component	z component
position (m)	5804217.83971894	41682618.8729034	-29848.9652210469
velocity (m/s)	-3051.34095115518	423.754085126723	6.10576186372653

Figure 6. Table showing the position and velocity xyz components for the best initial estimate “night1_spread.IOD.late”

III. Estimation

After determining the best initial orbit estimate as “night1_spread.IOD.late” from the initial orbit determination step described above, we began a more robust orbit determination that utilized the SNaG-App’s `determine_orbit.m` function. Note that the “night1_spread.IOD.late” best initial orbit was used for all subsequent orbit determination in the study.

The `determine_orbit.m` function works by inputting the best initial estimate, a “make station” or site longitude, latitude and altitude, a set of observations from the first night, and a force model to determine a better orbit from the initial estimate using the OraaS server and database. The default force model inputted into the `determine_orbit.m` function was third-body force_model_4x4_1m2_cr1 which used a “4x4” central body degree and order with the drag values being set to zero. The area and reflectivity coefficients were both set to 1 and the mass set to 1000 kg.

The function outputs a structure that contains the orbit estimate, “estimated” as well as other useful information such as the RMS and residuals of the observations. These “details” were used to analyze and validate the robustness of the orbit estimates from `determine_orbit.m` and compare the impact of the various observation sets from night 1 that were used.

This estimation step was repeated four times, feeding in a different observation set (night 1 early, mid, late and spread) while keeping the initial estimate, station site (OAP-Chile), and the force model constant. After we had verification of proper estimation techniques, we could proceed to the study portion of the project.The results of these 4 orbit determinations can be seen below.

	x component	y component	z component
position (m)	5806165.32418394	41755926.6903133	-22744.9782883768
velocity (m/s)	-3045.56962953875	422.184144934838	6.19695071736014

Figure 7. Table showing the position and velocity xyz components for “od_night1_early.estimated” which occurred at an epoch of '29-Oct-2020 07:03:42'

	x component	y component	z component
position (m)	5807826.32621718	41733031.9490404	-24797.480669033
velocity (m/s)	-3044.02519619028	420.753199252951	6.23429844793371

Figure 8. Table showing the position and velocity xyz components for “od_night1_mid.estimated” which occurred at an epoch of '29-Oct-2020 07:03:42'

	x component	y component	z component
position (m)	5813030.5910567	41762313.056733	-22970.9825551545
velocity (m/s)	-3044.98072514563	423.349189822864	6.15801717051056

Figure 9. Table showing the position and velocity xyz components for “od_night1_late.estimated” which occurred at an epoch of '29-Oct-2020 07:03:42'

	x component	y component	z component
position (m)	5813341.6339473	41765268.0757634	-22692.1220800742
velocity (m/s)	-3045.16761002176	423.26562675863	6.16799097852596

Figure 10. Table showing the position and velocity xyz components for “od_night1_spread.estimated” which occurred at an epoch of '29-Oct-2020 07:03:42'

The initial estimates calculated from the `determine_orbit.m` function were marginally different from the IOD calculated orbit shown in **Figure 6**. The slight variations seen in the position and velocities in **Figures 7-10** will give rise to slight variations in the data during part 1 of the study portion of the project.

IV. Study

As we began to examine our data further, there were two things we wished to analyze: the impact of varying the observation data set and the impact of varying the force model on orbit determination. To better analyze the impact that varying the observation sets and varying the force models had on the orbit estimate and residuals, we propagated the orbit estimates to the second night changing one only parameter while holding the rest constant each time for a controlled study.

For each study described below, the orbit estimation output from `determine_orbit.m` was propagated several times, with one parameter varied each time, to the second night using the `propagate_to_times.m` function similar to the procedure described in the Initial Orbit Determination section above. To make the conversion for each of the propagated position and velocities into respective azimuth and elevations simpler, we wrote the `ADD_aer.m` function that ran a "for" loop of the `eci_to_azelrn.m` from SNaG-App to produce a field in the propagate structure of predicted azimuths and elevations. These would then be compared to the observation data set from night 2 ("opt3satBset4") using the function `RMS.m`. Lastly, a table for each study was generated comparing the RMS values of the varied parameter (i.e observation set or force model). The residuals between the propagated/predicted azimuths and elevations and the observed were also plotted for each variation and compared in each study.

1. Vary the Observation Data Set

First, we examined the impact of changing observation times on the propagated orbit. We use the default third-body force model described earlier and provided in the example of the project description, keeping it constant for each propagation in this part of the study. The only parameter that was varied was the observation set inputted into the `propagate_to_times.m` function.

Similar to the initial orbit determination, we chose to analyze the same four observation sets. The first three observation sets incremented in steps of 4 giving 26 data points where the last observation set, "night 1 spread" indexed from 10 to 700 in steps of 10 giving 70 data points spread evenly throughout the entire first night. For data spread across the entire set, 70 points were taken into consideration, but for the smaller sets only 26 points were being examined. We agreed this would allow for the most robust analysis, and would provide a way to see how our estimations changed in time. Furthermore, this would give us a means to examine how the

quantity of data being selected impacts the estimation. Using the process described earlier, a propagation to the second night was analyzed for each observation set and a plot and table comparing the resulting residuals and RMS values between each variation was produced.

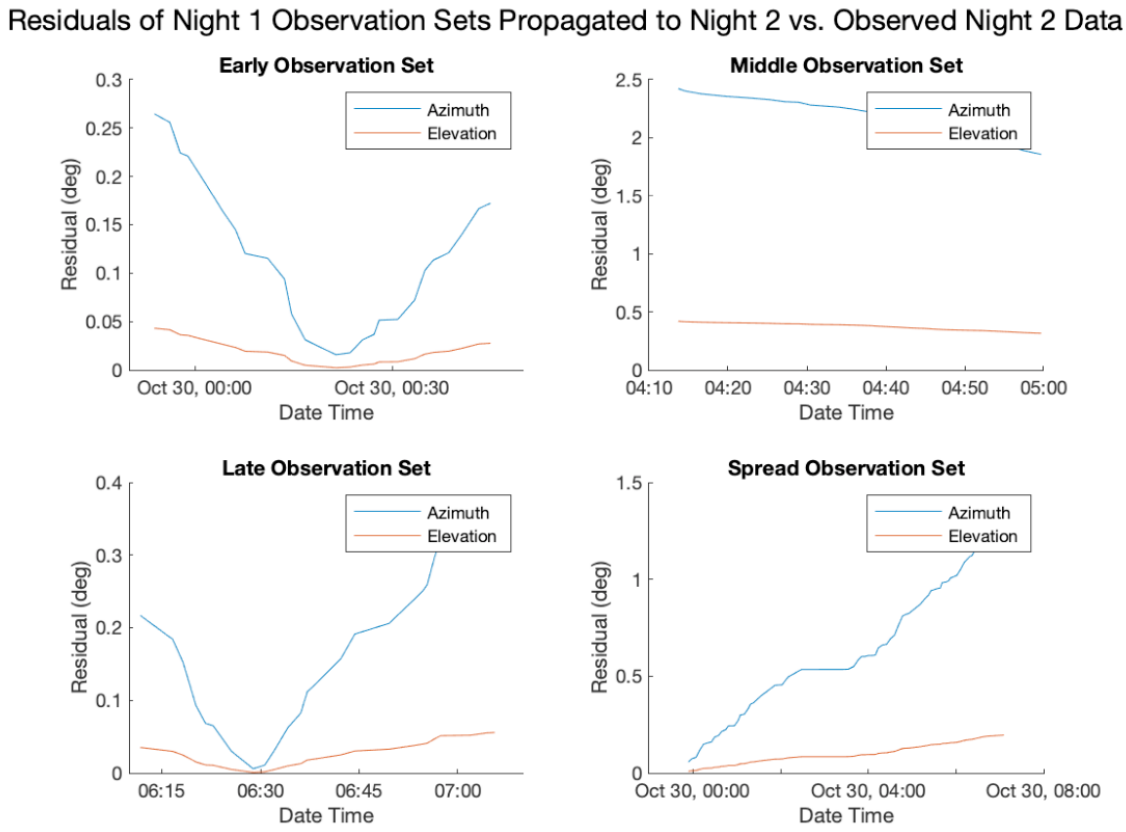


Figure 11. Plots showing the azimuth and elevation residuals for Study 1

observation_set	best_rms_value
"night1_early"	0.13902
"night1_mid"	2.1984
"night1_late"	0.19996
"night1_spread"	0.76663

Figure 12. Table showing best RMS values for the 4 observation sets

2. Vary the Force Model

After we picked our best observation set, we used it in our future propagations to find an optimal force model. The force model is a representation of the effect of the forces on the dynamics of the system. This is depicted by an array of numbers describing the central gravity degree, central gravity order, drag area, drag coefficient, cross sectional area, coefficient of reflectivity, and spacecraft mass respectively. For example, the default force model we use for our previous applications is '4 x 4 x 0 x 0 x 1 x 1 x 1000'. This indicates a central gravity degree and central gravity order of 4, negligible drag, cross sectional area of 1 m^2, a coefficient of reflectivity of 1, and spacecraft mass of 1000 kg. For our study, we want to see the impact of central gravity and solar pressure on propagation. We do not change the terms dealing with drag, as atmospheric drag is deemed negligible for geostationary satellites.

Varying the Gravity Force Parameter:

To start our study on the impact of varying the force model on orbit estimation, we first varied the gravity force gravity parameter. The pairs of gravity parameters that were tested, in reference to the project statement, were 2x0, 2x2, and 20x20. The other parameters of the force model were held constant where the mass was set to 6000 kg. This mass was an estimation based on the launch mass of our satellite which was 6650 kg (Technical Details for Satellite Echostar 16). Using the process described earlier, a propagation was analyzed for each variation of the gravity parameter and a plot and table comparing the resulting residuals and RMS values between each variation was produced.

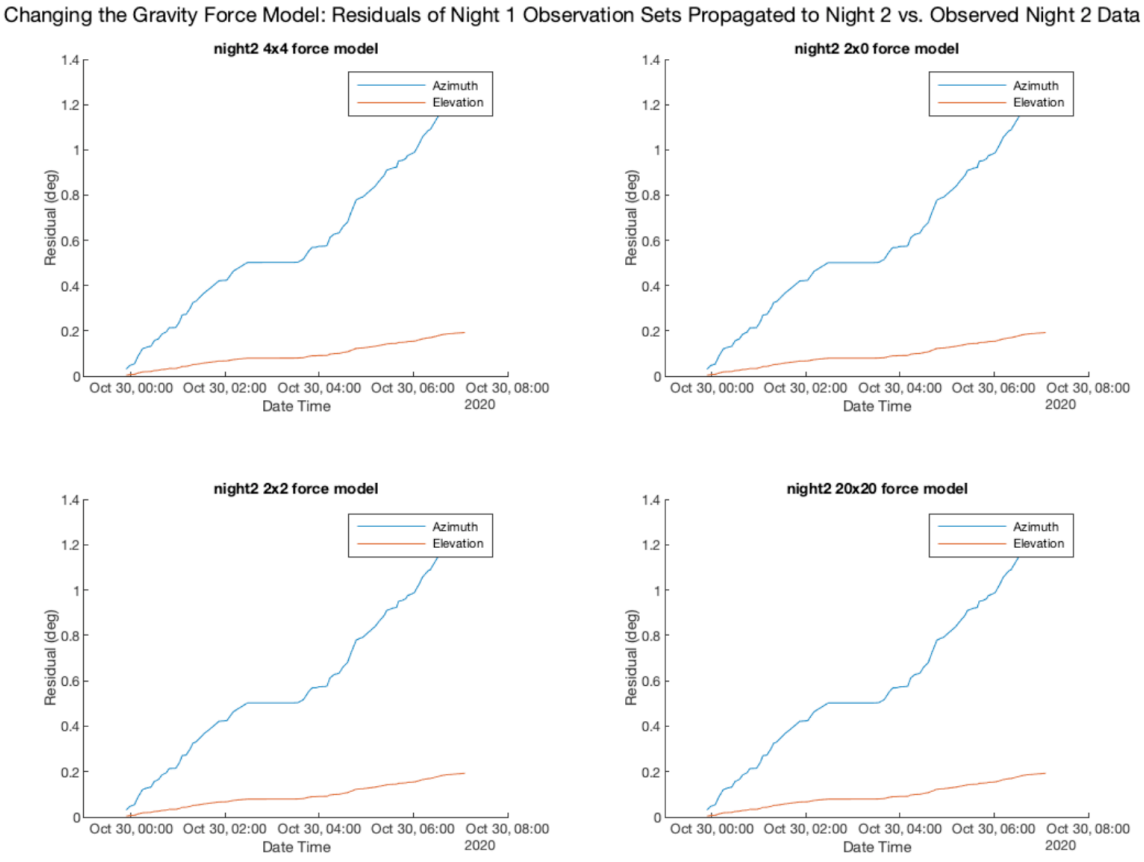


Figure 13. Plots showing the azimuth and elevation residuals for the changing gravity force model

observation_set	best_rms_value
"night2_4x4"	0.738507043949236
"night2_2x0"	0.737489515251461
"night2_2x2"	0.738671840386138
"night2_20x20"	0.738504661912253

Figure 14. Table showing the best RMS value for each varying gravity force model

Varying Solar Radiation Parameters

We then choose the gravitational model “2x0”, which produced the smallest RMS value, in order to optimize the solar pressure parameters. Solar pressure, as seen in the equation below, is parameterized by mass, coefficient of reflectivity, and surface area. We proceed to make 3 sets of 3 different propagations. Each of the three sets deal with each of our solar pressure parameters. Every propagation inside each set varies the respective parameter by some amount.

$$F_{SRP} = \upsilon \cdot \frac{S}{c} \cdot C_R \cdot A_s$$

Figure 15. Solar Force Equation (a.i. solutions, n.d.)

Varying Mass

For varying our mass, we chose 1000 kg, 5000 kg, and 9000 kg. These were chosen as 1000 kg and 9000 kg are extreme changes to our original 6000 kg, while 5000 kg is a smaller change. Through these we can see how different changes in mass affect propagation. Using the process described earlier, a propagation was analyzed for each variation of the mass of the spacecraft and a plot and table comparing the resulting residuals and RMS values between each variation was produced.

observation_set	best_rms_value
"night2_mass1: 1000 kg"	0.765598205786066
"night2_mass2: 5000 kg"	0.765651344911314
"night2_mass3: 9000 kg"	0.765657249323774

Figure 16. Table showing the best RMS value for varying the mass of the spacecraft

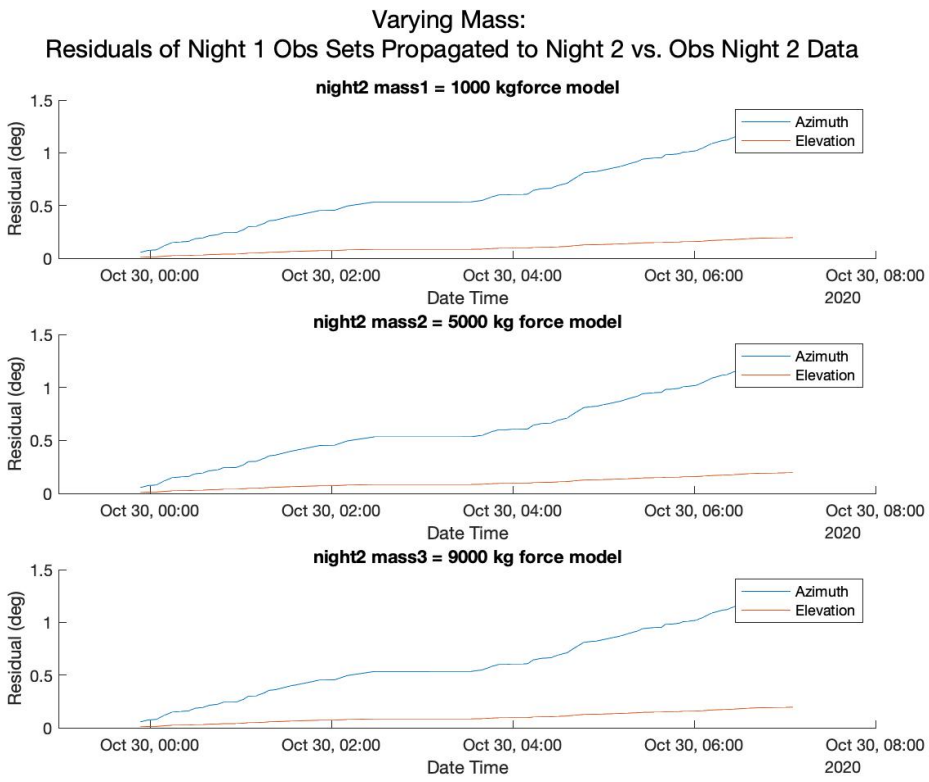


Figure 17. Plots showing the azimuth and elevation residuals for varying the mass of the spacecraft

Varying Surface Area

We then move on to our next set of propagations, varying the surface area of the satellites. We choose the variations .5 m², 2 m², and 4 m². A similar reasoning is used to choose the numbers: there is one lower value, one higher value, and one extreme value when compared to the original 1 m². Using the process described earlier, a propagation was analyzed for each variation of the surface area and a plot and table comparing the resulting residuals and RMS values between each variation was produced.

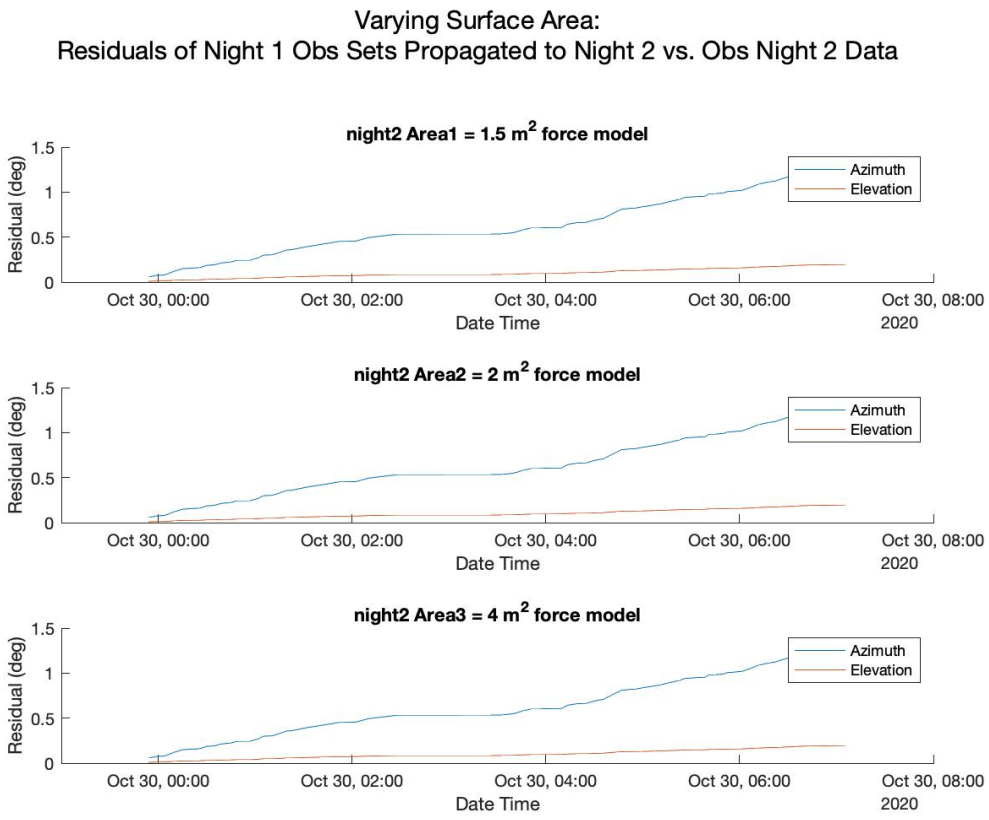


Figure 18. Plots showing the azimuth and elevation residuals for varying the surface area of the spacecraft

observation_set	best_rms_value
"night2_Area1: 0.5 m^2"	0.765659094454341
"night2_Area2: 2 m^2"	0.765642488316766
"night2_Area3: 4 m^2"	0.765620346960132

Figure 19. Table showing the best RMS value for varying the surface area of the spacecraft

Varying Coefficient of Reflectivity

We then move on to our final set of propagations, varying the coefficient of reflectivity. We choose the variations of .6, 1.2, and 1.8 for the coefficient. These are low, average, and high values of the coefficient of reflectivity. This is based on the definitions of an object with a coefficient of 0 not reflecting any light and an object with a coefficient of 2 completely reflecting light. Using the process described earlier, a propagation was analyzed for each variation of the coefficient of reflectivity and a plot and table comparing the resulting residuals and RMS values between each variation was produced.

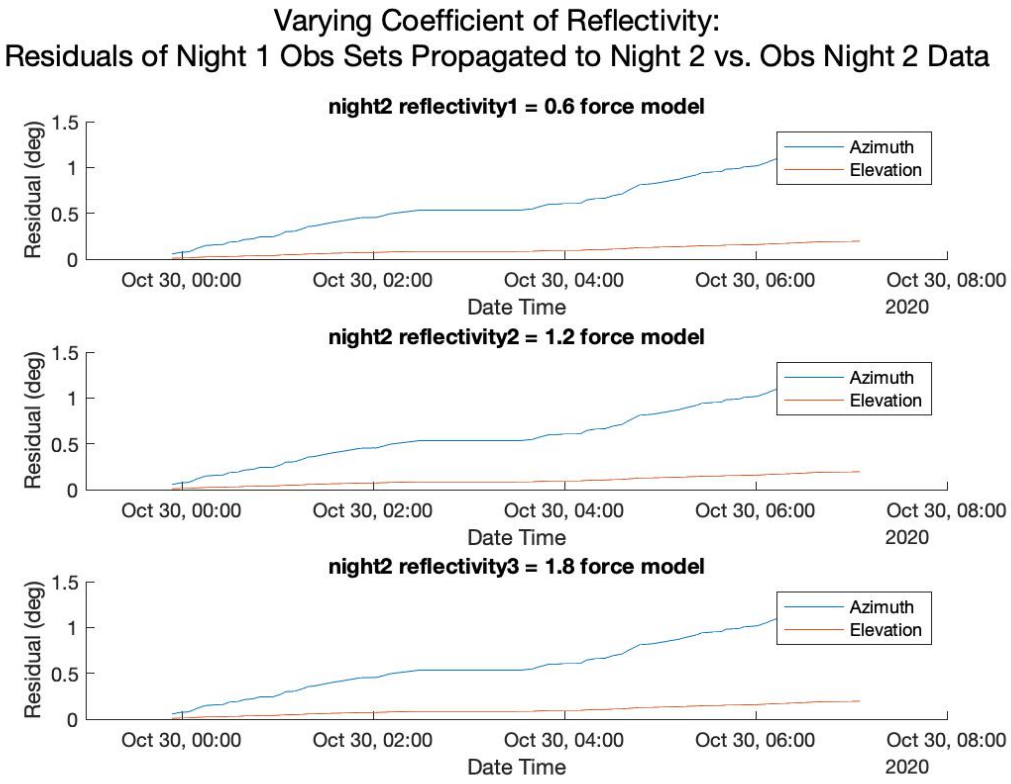


Figure 20. Plots showing the azimuth and elevation residuals for varying the reflectivity of the spacecraft

observation_set	best_rms_value
"night2_cr1: 0.6"	0.765659094454341
"night2_cr2: 1.2"	0.765653559062703
"night2_cr3: 1.8"	0.765648023684715

Figure 21. Table showing the best RMS value for varying the reflectivity of the spacecraft

V. Results and Analysis

The goal of varying the observation data was to determine the impact of different ranges of observations. By looking at the output graph (**Figure 11**) and RMS values (**Figure 12**), it is clear that the observation set taken in the middle of the data resulted in the least accurate estimations, as it produced residuals of about 2.5° and 0.5° for the azimuth and elevation, respectively. From the graph of residuals over time, the dataset containing “middle” points is shown to be mostly consistent in residual value. The observation set that is spread throughout the data set has a large RMS value, indicating lack of accuracy. The residuals graphs show the residuals start at their minimum at the first data entry of the set and increase for the following entries. The early and late datasets had the lowest residuals, but the early set was the best fit, as it had lower RMS and residuals. The early set residual graph shows the residuals minimum to be at approximately 00:20. The late set residual graph shows the residuals minimum to be at approximately 06:30.

The cause of discrepancies between the four sections of data can be identified by looking at how the residuals behave. Both early and late residuals act similarly, with the main difference being when they converge. Looking at `opt3satBset4`, we pull the observations from “29-Oct-2020 23:53:51” to “30-Oct-2020 00:44:58” for the early set and “30-Oct-2020 06:11:44” to “30-Oct-2020 07:05:37” for the late set. These correlate to entries 10 to 110 and 600 to 700. Inside these ranges, there are uneven time skips between observations. From inspection, the time steps between the observations seem to be smaller around the minimum residual values for our early and late sets than compared to other parts of each set. For instance, entry 664 and 665 of `opt3satBset4` shows a time skip greater than 4 minutes. Comparatively, the entries around 06:30 have an average time skip of approximately 45 seconds. Using a similar reasoning, the middle set’s residuals being kept constant throughout propagation are likely

caused by the time steps in the middle of the `opt3satBset4` to have a low variance. The spread residual graphs also support this reasoning. It shows the early times have lower residuals than the late times, which correlates well with the early and late residuals values we have outputted. By inspecting the `opt3satBset4`, one can see the time skips of the early set are a little less in magnitude than the late set. This highlights the idea that observation data sets with smaller time skips will allow for more accurate propagation.

When changing the gravitational parameters from 4x4 to 2x0, 2x2, and 20x20, there is a difference in RMS of -.00101753, .00016480, and -.00000238, respectively. This shows that increasing the central gravity orders and degrees cause less change in the RMS than decreasing these parameters. When changing the solar pressure parameters, how they correlate to change in RMS is clearly displayed by each table. In the mass RMS table, a positive correlation is seen between mass and RMS. In the surface area RMS table, a negative correlation is seen between surface area and RMS. In the coefficient of reflectivity RMS table, a negative correlation is seen between reflectivity and RMS. The reason why solar pressure parameters change RMS requires some thought. Logically, a satellite with a greater surface area and reflectivity will be easier to track and predict. Likewise, a satellite with a smaller mass is less affected by gravity, and hence its mass affects the orbit less.

The results of changing the force model were very minimal. Why the RMS does not change substantially is likely due to these gravitational models and radiation forces being near negligible. The geopotential model represents the iterations required to model the Earth's mass distribution. This is a correction of the wrong assumption that Earth can be modeled as a perfect sphere, with center of gravity at the center. As seen from the equation below, the geopotential model is changing the energy of the system marginally. When we put in the first two numbers of the force model, they actually represent "l x m" in this equation, controlling the amount of iterations and slightly changing gravity potential. According to the textbook, it is better for this model to be higher, as the Earth is said to be 720 x 720. This is not supported by our findings, as the lowest RMS value we found was 2x0. However, as earth is close to an ideal mass distribution, all changes in the central body parameters are nearly negligible to RMS. This discrepancy is possibly attributed to data error, computational error, or human error. As for the solar pressure term, the figure below shows that the acceleration of gravity due to the sun radiation is approximately 10^{-7} m/s^2. Again, because the effect of solar pressure is very miniscule, changing the value won't cause a great change to RMS. Also note that the residuals

are nearly identical for each scenario, indicating that the force model has a very small effect on propagation as a whole.

$$U(r,\lambda,\phi) = \frac{\mu}{r}\left[1 + \sum_{\ell=2}^{\infty}\left(\frac{R_\oplus}{r}\right)^{\ell}\sum_{m=0}^{\ell} P_{\ell m}(\sin\phi)[C_{\ell m}\cos m\lambda + S_{\ell m}\sin m\lambda]\right],$$

Figure 22. Geopotential Perturbation Energy Equation (Healy 84)

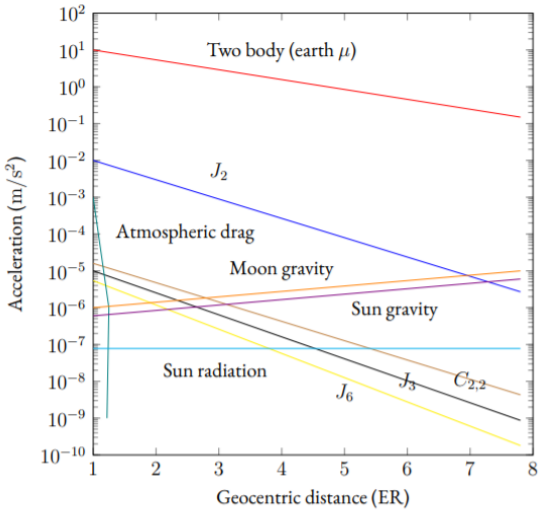


Figure 23. Acceleration due to various forces in low-earth orbit (Healy 80)

VI. Conclusion

When performing our estimations, studies, and analyses, several sources of errors may have come into play and limited the accuracy of our data. Here, we will be discussing the limitations of our analysis, sources of possible errors and their potential effects, as well as our mitigation strategies. Additionally, a discussion of our overall results will be present along with a summary of our methods.

One limitation in our analysis was that the data set had two gaps in collection, one occurring within the first 10 data points, and another occurring around the 2-hour mark. The first collection gap was unexplained by the collection company, and caused the largest issues with our code. We found that including points both before and after this gap resulted in errors and incorrect calculations, so we opted to exclude those points from our analysis. This is the reason all of our “early” data sets start at the tenth data entry rather than the first. The second collection gap was explained by the satellite moving in front of the moon, and the brightness preventing accurate measurements. This gap comes in between our “early” and “mid” sets, but is not included in either, so we do not see a noticeable error caused by this gap.

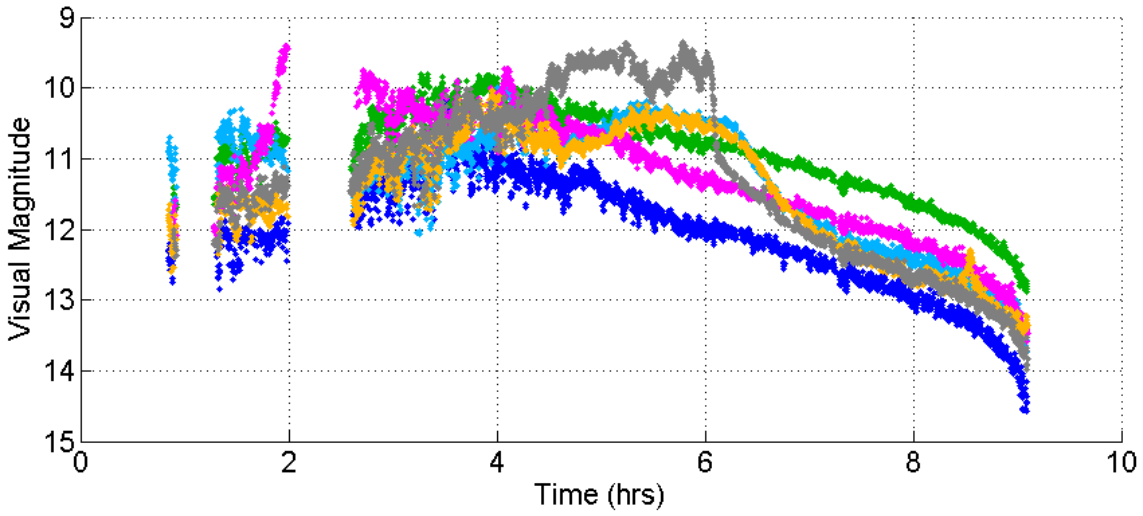


Figure 24. Visual Magnitude of observations over time from 29 October 2020 (Jefferies Slide 7)

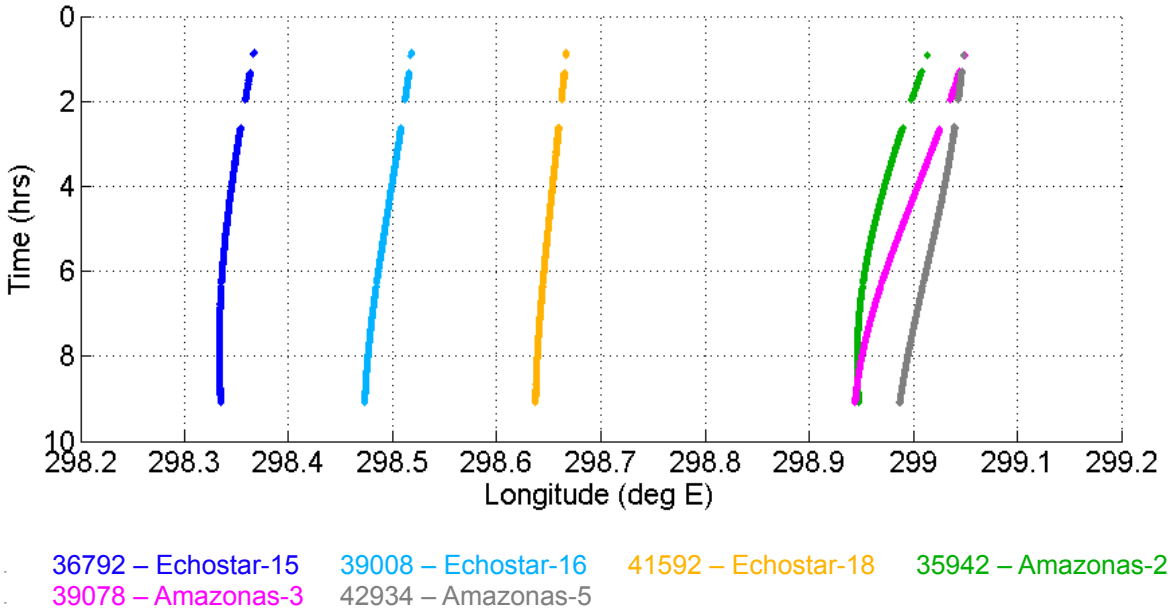


Figure 25. Longitude of observations over time from 29 October 2020 (Jefferies Slide 7)

One of the biggest limitations and potential source of error comes from our data selection for our Initial Orbit Determination. As we did not have the time or computing power to go through every initial data point and pick the best possible starting observation, our initial points were selected arbitrarily. This means the vectors selected from the early, middle, and late subsets and ultimately used in the `IOD` function were not necessarily the best options. Although our residuals and RMS values show accurate predictions, this initial generalization could have caused cascading inaccuracies and thrown off every part of our calculations.

The most unforeseen error we encountered was the ORaaS website being down. This led to several issues, as the `determine_orbit.m` pulls data from ORaaS to perform more accurate calculations. Without this valuable tool, we were unable to get highly accurate initial orbit estimations, as the Gauss and Gibbs methods we were using have inherent inaccuracies. Fortunately, the website came back online, and our orbit determination was able to proceed as planned.

Something we discovered early on while performing our orbit estimation and analysis was that the estimation became more accurate as we spread out the initial data points. When we first started performing our calculations, we took consecutive data points, as we believed this would produce the most accurate estimation. However, this method produced errors and very high residuals. Thus, we determined that the best estimations came when we chose four values

spread far throughout the time span. This allowed for a more comprehensive understanding of the data, and we ultimately realized this was the best approach. Additionally, the position and velocity vectors were closer to the observed values, as were the residuals and RMS values.

One of the main tasks we needed to perform involved varying the gravitational body parameters and examining how this impacted our predictions. The first part of the force model is the number of gravitational bodies affecting the spacecraft. Typically, it can be set to either a two or three-body approximation. We discovered that when we used the two-body model and varied other parameters, our estimations and residuals changed. However, when we shifted to a three-body approximation, our RMS and residual values did not change regardless of how we varied other parameters. Thus, we were able to conclude that two-body approximations place more emphasis on the parameters of the spacecraft, as it isolates the effect earth has versus the three-body which takes into account the effects of the sun and moon.

Through all of our variations, several conditions resulted in very small RMS values and residuals of less than a degree. However, we found that the best RMS values were produced using a mass body estimate of “2x0,” even though this should have been one of the least accurate. Additionally, we found almost no effect from changing solar pressure parameters, even though in practice they could have a bigger impact. Likewise, varying the satellite area resulted in very small impact on the calculated RMS. Ultimately, we realized that using the “late” window dataset and a 2x0 force model created the most accurate results, regardless of the other parameters.

Data

Night1 Spread Observation Data from opt2satBset4

Note this data was used for the best initial estimate calculation and hence used for the initial estimate throughout the study

observation_number	datetime	azimuth_deg	elevation_deg
291	10/29/2020 1:21	17.63565816	53.51405556
801	10/29/2020 1:35	17.63326095	53.51521845
1205	10/29/2020 1:51	17.63069094	53.51725149
1447	10/29/2020 2:37	17.62243934	53.52182814
1754	10/29/2020 2:44	17.621638	53.52238552
2046	10/29/2020 2:50	17.61955568	53.52257229
2315	10/29/2020 2:55	17.61951764	53.52314846
2624	10/29/2020 3:01	17.61833008	53.52371568
2848	10/29/2020 3:05	17.61818102	53.52396582
3186	10/29/2020 3:11	17.61690673	53.52456262
3416	10/29/2020 3:16	17.61652518	53.52494392
3973	10/29/2020 3:28	17.6142157	53.52580302
4164	10/29/2020 3:31	17.61380452	53.5259371
4352	10/29/2020 3:34	17.61329961	53.52626147
4643	10/29/2020 3:39	17.61229106	53.52682969
4884	10/29/2020 3:43	17.61165953	53.52741044
5234	10/29/2020 3:49	17.61067769	53.52746172
5500	10/29/2020 3:54	17.61019229	53.52800852
5792	10/29/2020 3:59	17.60919715	53.52858127
6128	10/29/2020 4:05	17.60837299	53.52858754
6341	10/29/2020 4:09	17.60778565	53.52879743
6602	10/29/2020 4:13	17.6071692	53.52909278
6889	10/29/2020 4:19	17.60631106	53.52945096
7170	10/29/2020 4:24	17.60536407	53.52997668
7470	10/29/2020 4:29	17.60478524	53.53025828
7674	10/29/2020 4:32	17.60429525	53.53037352
7925	10/29/2020 4:37	17.60367379	53.5307572
8101	10/29/2020 4:40	17.60341304	53.53092997
8332	10/29/2020 4:43	17.6031995	53.53109076
8576	10/29/2020 4:48	17.60220421	53.53139462
8687	10/29/2020 4:49	17.60181494	53.53137585
8945	10/29/2020 4:54	17.601513	53.53164293
9151	10/29/2020 4:58	17.60087624	53.5318131
9389	10/29/2020 5:02	17.60025958	53.53209258

9558	10/29/2020 5:05	17.600072	53.5322197
9846	10/29/2020 5:10	17.59895451	53.53245922
10013	10/29/2020 5:13	17.59889554	53.53259207
10202	10/29/2020 5:16	17.59833667	53.53271769
10451	10/29/2020 5:20	17.59814713	53.53288665
10729	10/29/2020 5:25	17.59729768	53.53318952
11157	10/29/2020 5:32	17.5964336	53.53322806
11466	10/29/2020 5:37	17.5956374	53.53363107
11750	10/29/2020 5:43	17.59523388	53.53375016
12013	10/29/2020 5:47	17.59458485	53.53392516
12386	10/29/2020 5:54	17.59377553	53.53386921
12562	10/29/2020 5:57	17.59337529	53.53398642
12794	10/29/2020 6:01	17.59317355	53.53413048
13052	10/29/2020 6:05	17.59239607	53.53439854
13272	10/29/2020 6:09	17.59211107	53.53431707
13464	10/29/2020 6:12	17.59155088	53.53438639
13688	10/29/2020 6:20	17.59081652	53.53455115
13842	10/29/2020 6:23	17.5904797	53.53475966
13960	10/29/2020 6:25	17.59044355	53.53463283
14152	10/29/2020 6:29	17.59010362	53.53472125
14308	10/29/2020 6:33	17.5898193	53.53490891
14410	10/29/2020 6:37	17.58929727	53.53481318
14610	10/29/2020 6:40	17.58890776	53.53497175
14933	10/29/2020 6:47	17.58827921	53.53478926
15243	10/29/2020 6:57	17.58751402	53.5349123
15547	10/29/2020 7:03	17.58701657	53.53484943
15739	10/29/2020 7:07	17.58700877	53.53482711
15905	10/29/2020 7:12	17.58623395	53.53491478
16139	10/29/2020 7:19	17.58603206	53.53487449
16434	10/29/2020 7:24	17.58579489	53.53474112
16761	10/29/2020 7:31	17.5854075	53.53461252
17062	10/29/2020 7:40	17.58467163	53.53450152
17363	10/29/2020 7:49	17.58421349	53.53429342
17546	10/29/2020 7:53	17.5842995	53.53425317
17822	10/29/2020 7:58	17.58309272	53.53362755
18131	10/29/2020 8:06	17.58361087	53.53389975

Contributions

Prashanta Aryal:

For this project, I wrote preliminary RMS and Gibbs we used for initial testing. Eventually this was replaced by code with better organization. Collaborated with writing `Gauss.m` function. Worked together with the orbit estimation sections of the project, focusing on outputting the graphs and choosing correct observation sets. I worked jointly with others during the debugging stage of our code, which was time consuming. For the study, I mainly focused on varying the solar pressure parameters to see the effect on propagation. For the report, I wrote the preliminary Initial Orbit Determination, which was later refined by other group members. I also wrote the Results and Analysis section and the second portion of the study, Vary the Force Model.

Ronak Chawla:

For this project, I had significant contributions to both the project code and the final report. In terms of the project code, I collaborated with the team on the majority of the main `Group8_final_project.m` file that included our procedure and studies. In addition, I wrote and/or collaborated on several functions that this main file used. For example, I wrote the `Get_observations.m`, `Gauss.m`, `Gauss.m`, and `ADD_aer.m` functions. I also collaborated on the `IOD.m` and `RMS.m` functions. Moreover, I helped the team with the Initial Orbit determination and Study portions of the project. For the study, I collaborated on the analysis of how varying the observation sets and force models impacted the orbit determination, especially after the estimates were propagated to the second night. I also assisted in plotting the residuals and writing rms values into tables for the comparison. For the report, I collaborated on writing the procedure and analysis for the Initial Orbit Determination, Estimation, and Study sections. For the Study section, I focused on the "Varying Observation" and "Varying the Gravity Force Parameter" subsection. Lastly, I attached several of the figures and tables and proofread and formatted the report.

Nico Lagendyk:

For this project, I began working a bit later than the rest of the group due to scheduling conflicts over the Thanksgiving break. I primarily worked on consolidating the work done by the other group members by creating the `IOD.m` function and then applying it to a range of different vector sets in the Group Project code. I worked on the `RMS.m` code to set up the angle exception so that our residuals were less than 180° and assisted in the debugging of the rest of the main code. In this report I wrote the Abstract, a majority of the Initial Orbit Determination section and helped overall with small edits throughout the report as well as adding figures and their descriptions. Lastly, I added and formatted the Data and Software Files at the end of the report.

Ryan Quigley:

For this project, my main contribution was the early version of the `RMS.m` function code, which I later helped edit into the final version used in the finished project along with Nico and Ronak. Additionally, I worked on a separate `Gauss.m` function that was used to check the accuracy of the one utilized in the final project code. As the code came together, I helped run, debug, and optimize the code along with all team members, which ended up being one of the most time consuming aspects of the project. Additionally, I worked on the Study to see the impact of different force body parameters on the estimation; i.e. the 2x2, 2x0, and 20x20 models. When it came to the report, I co-wrote the Estimation, wrote the Conclusion, and early portions of the Study sections, which involved parts I was closest to in the code.

Sources

a.i. solutions . (n.d.). *Solar Radiation Pressure*. FreeFlyer Guide. Retrieved December 13, 2021, from https://ai-solutions.com/_freeflyeruniversityguide/solar_radiation_pressure.htm.

Healy, Liam M.. *Space Flight Dynamics and Navigation*. 2020. PDF.

Healy, Liam M. (2021) SNaG-app (Version 11) [MATLAB Application] https://snag.orbitdynamics.space/

Jefferies, Mark and Bill Therien. *University of Maryland Data Collection 2: 299°E Cluster.* 10 Nov. 2020, 2020-11-10-Exo-Data-Summary-2.pptx. PowerPoint Presentation.

“Technical Details for Satellite Echostar 16.” *N2YO.Com - Real Time Satellite Tracking and Predictions*, https://www.n2yo.com/satellite/?s=39008.

“ORaaS.” OreKit. 1 April 2021. https://oraas.orekit.space/

Software Files

Get Observations Function

```
function [obs1,obs2,obs3] = Get_observations(dataset, obs_indices)
%Get Observations
%   This function reads in dataset from one of the satellites and three
%   desired observation indices and creates three structures for each of the
%   three observations. Each observation structure includes the set of
datetimes, lla, Right Ascension
%   and Declinations. These three observations structures are used by the
Gauss and Gibbs
%   functions for initial orbit determination.

%   Written by: Ronak Chawla
obs1.time = dataset.datetime(obs_indices(1));
obs2.time = dataset.datetime(obs_indices(2));
obs3.time = dataset.datetime(obs_indices(3));

obs1.lla = latlonalt_deg(dataset.site_latitude_deg(obs_indices(1)),
dataset.site_longitude_deg(obs_indices(1)),
dataset.site_altitude_m(obs_indices(1)));
obs2.lla = latlonalt_deg(dataset.site_latitude_deg(obs_indices(2)),
dataset.site_longitude_deg(obs_indices(2)),
dataset.site_altitude_m(obs_indices(2)));
obs3.lla = latlonalt_deg(dataset.site_latitude_deg(obs_indices(3)),
dataset.site_longitude_deg(obs_indices(3)),
dataset.site_altitude_m(obs_indices(3)));

obs1.RA = dataset.right_ascension_deg(obs_indices(1));
obs2.RA = dataset.right_ascension_deg(obs_indices(2));
obs3.RA = dataset.right_ascension_deg(obs_indices(3));

obs1.Declination = dataset.declination_deg(obs_indices(1));
obs2.Declination = dataset.declination_deg(obs_indices(2));
obs3.Declination = dataset.declination_deg(obs_indices(3));

end
```

Gauss Angles-Only Calculation

```
function [r1,r2,r3] = Gauss(ob1, ob2, ob3)
%Gauss Angles-Only function.
%   Takes in three sets of angles (right ascension and declination) and
%   their respective epochs to compute three position vectors r1, r2 and r3
%   in units of km used by the Gibbs function for orbit determination.

%   Written by: Ronak Chawla and Prashanta Aryal

t1 = ob1.time; % datetime 1
t2 = ob2.time; % datetime 2
t3 = ob3.time; % datetime 3

% Right Ascension (alpha) and Declination of three observations in degrees
alpha1t = ob1.RA;
delta1t = ob1.Declination;

alpha2t = ob2.RA;
delta2t = ob2.Declination;

alpha3t = ob3.RA;
delta3t = ob3.Declination;

Tau1 = seconds(t1-t2);
Tau3 = seconds(t3-t2);

% Site Vectors from three observations in meters:

Rsv_1 = lla_to_eci(ob1.time, ob1.lla);
Rsv_2 = lla_to_eci(ob2.time, ob2.lla);
Rsv_3 = lla_to_eci(ob3.time, ob3.lla);

R = [Rsv_1 Rsv_2 Rsv_3]/1000; % in km

% Observation direction matrix

rho1_hat = [cosd(delta1t)*cosd(alpha1t); cosd(delta1t)*sind(alpha1t);
sind(delta1t)];
rho2_hat = [cosd(delta2t)*cosd(alpha2t); cosd(delta2t)*sind(alpha2t);
sind(delta2t)];
rho3_hat = [cosd(delta3t)*cosd(alpha3t); cosd(delta3t)*sind(alpha3t);
sind(delta3t)];

L = [rho1_hat rho2_hat rho3_hat];

M = inv(L)*R;

a1 = Tau3/(Tau3 - Tau1);
```

```
a3 = -Tau1/(Tau3 - Tau1);

a1u = Tau3*((Tau3-Tau1)^2 - Tau3^2)/(6*(Tau3-Tau1));
a3u = -Tau1*((Tau3-Tau1)^2 - Tau1^2)/(6*(Tau3-Tau1));

A = M(2,1)*a1 - M(2,2) + M(2,3)*a3;
B = M(2,1)*a1u + M(2,3)*a3u;

R_2 = R(:,2);
E = dot(rho2_hat,R_2);

mu = 3.986*10^5;
coef3 = -mu^2*B^2;
coef2 = -2*mu*B*(A + E);
coef1 = -(A^2 + 2*A*E + norm(R_2)^2);

r2_roots = roots([1 0 coef1 0 0 coef2 0 0 coef3]);

 for i = 1:8

  if (r2_roots(i) > 6378) && (isreal(r2_roots(i)))
      r2 = r2_roots(i);
  end

 end

u = mu/r2^3;

c1 = a1 + a1u*u;
c2 = -1;
c3 = a3 + a3u*u;

C = [-c1; -c2; -c3];

% rho1,2,3 in km:
rho = -(M*C)./C;

r1 = R(:,1) + rho(1)*rho1_hat;
r2 = R(:,2) + rho(2)*rho2_hat;
r3 = R(:,3) + rho(3)*rho3_hat;

end
```

Gibbs Method

```
function [v1, v2, v3] = Gibbs(r1, r2, r3)
%Gibbs method:
%   This function takes in three position vectors in units of km of different
%   times and solves for the orbit by returning the velocity vectors in units
%   of km/s corresponding to each position vector. The three points are
coplanar.

%   Written by: Ronak Chawla

mu = 3.986*10^5;

D = cross(r2,r3) + cross(r3,r1) + cross(r1,r2);
N = norm(r1)*cross(r2,r3) + norm(r2)*cross(r3,r1) + norm(r3)*cross(r1,r2);
S = (norm(r1)-norm(r2))*r3 + (norm(r3)-norm(r1))*r2 + (norm(r2)-norm(r3))*r1;

B1 = cross(D,r1);
B2 = cross(D,r2);
B3 = cross(D,r3);

L = sqrt(mu/(norm(D)*norm(N)));

v1 = (L/norm(r1))*B1 + L*S;

v2 = (L/norm(r2))*B2 + L*S;

v3 = (L/norm(r3))*B3 + L*S;

end
```

Root Mean Square Function for angles

```
function [RMS] = RMS(obs_dataset, pred_dataset)
%RMS Root Mean Square Function for angles.
%   [RMS] = RMS(obs_dataset, pred_dataset) returns the root mean square
%   value and residuals for azimuth and elevation.
%
%   obs_dataset is a structure with fields azimuth_deg and elevation_deg
%   corresponding to an observed data set. pred_dataset is the exact same
%   except for a predicted dataset. Both variables should also have an
%   epoch field which is a column vector of datetimes. The length of both
%   obs_dataset and pred_dataset fields must all be the same length.
%
%   Written by: Ryan Quigley, Ronak Chawla, and Nico Lagendyk

azObs = obs_dataset.azimuth_deg;
elObs = obs_dataset.elevation_deg;

azPred = pred_dataset.azimuth_deg;
elPred = pred_dataset.elevation_deg;

timeObs = obs_dataset.epoch;

RMSsum = 0;

N = length(azObs);
azresidual = zeros(N,1);
elresidual = zeros(N,1);

      for i=1:N

        aztemp = mod(azObs(i) - azPred(i),360);
        azresidual(i) = min(360-aztemp,aztemp);
        eltemp = mod(elObs(i)-elPred(i),360);
        elresidual(i) = min(360-eltemp,eltemp);
        RMSsum = azresidual(i)^2 + elresidual(i)^2 + RMSsum;

      end

      RMS.value = sqrt((1/N)*RMSsum);
      RMS.azresidual = azresidual;
      RMS.elresidual = elresidual;

      plot(timeObs,azresidual)
      plot(timeObs,elresidual)
end
```

Add Azimuth, Elevation and Range to structure

```
function [dataset] = ADD_aer(dataset, observer_lla)
%ADD_aer: Add Azimuth, Elevation and Range to structure
%
%   This function uses the position vectors at each time and an observer lla
%   to convert the eci coordinates of a data into azimuth, elevation and
range.
%   The function uses a for loop to go through each time in the data set and
%   uses the Snag-App function eci_to_azelrn to obtain an azimuth, elevation
%   and range for each time. Then, the function returns fields of azimuth
%   and elevation to the structure of the input dataset.

%   Written by: Ronak Chawla

for i = 1:length(dataset.epoch)
      aer = eci_to_azelrn(dataset.epoch(i), dataset.position_m(i,:),
observer_lla);
      dataset.azimuth_deg(i,1) = aer.azimuth_deg;
      dataset.elevation_deg(i,1) = aer.elevation_deg;
end

end
```

Initial Orbit Determination function

```
function [output, night2_IOD_prop] =
IOD(night1_dataset,night2_dataset,timespan,vector,name)
%IOD  Initial Orbit Determination
%
%   This function compiles all the steps necessary to go from a single set
%   of three indexes (vector) which specify rows in the night2_dataset.
%   These indexes are then used to pull observed values of the datetimes,
%   llas, right ascension and declination using the Get_obervations
%   function.
%
%   Once the data is reformatted the Gauss function performs and angles
%   only calculation to find 3 position vectors. These vectors are then
%   plugged into the Gibbs function which solves for the corresponding
%   velocities to these positions thereby fully defining the orbit.
%
%   The three position velocity pairs are then converted to initial
%   estimates which are then propagated to times during the second night.
%   Then using the observed values from night2_dataset the RMS function is
%   used to compare the calculated. Then the rms values from the three
%   position velocity pairs are compared and the best (i.e. lowest
%   residual) is exported alongside all the individual rms values to the
%   output as fields in a structure. The RMS function also prints graphs
%   which are layered and exported as part of the output structure.
```

```
%
%   The night2_IOD_prop are all the values from the propagation steps and
%   are also sent out with the associated values from night2_dataset which
%   allows for easy comparison.
%
%   Written by: Nico Lagendyk, Ronak Chawla, and Prashanta Aryal


      % Redefine variable names to make code easier to read
      ds1 = night1_dataset;
      ds2 = night2_dataset;
      ts = timespan;
      vec = vector;

      % Reformat observation data so it is easier to handle using
      % Get_obervations function
      [obs1,obs2,obs3] = Get_observations(ds1, vec);

      % Run Gauss code to determine position from 3 angles
      [r1, r2, r3] = Gauss(obs1,obs2,obs3); % r in units of km

      % Run Gibbs code to determine correlating velocities
      [v1, v2, v3] = Gibbs(r1, r2, r3); % v in units of km/s

      % Convert to m and m/s
      r1 = r1*1000;
      r2 = r2*1000;
      r3 = r3*1000;
      v1 = v1*1000;
      v2 = v2*1000;
      v3 = v3*1000;

      % Create 3 initial estimate pvts from calculated values
      initial_est1 = pvt(ds1.datetime(vec(1)),r1,v1);
      initial_est2 = pvt(ds1.datetime(vec(2)),r2,v2);
      initial_est3 = pvt(ds1.datetime(vec(3)),r3,v3);

      % Copy data from night 2 into separate structure
      night2_IOD_prop.obs.azimuth_deg = ds2.azimuth_deg(ts);
      night2_IOD_prop.obs.elevation_deg = ds2.elevation_deg(ts);
      night2_IOD_prop.obs.epoch = ds2.datetime(ts);

      force_model_4x4_1m2_cr1 = force_model_third_body(4, 4, 0, 0, 1, 1,1000);
      observer_lla = latlonalt_deg(ds1.site_latitude_deg(1),
ds1.site_longitude_deg(1), ds1.site_altitude_m(1));

      % Propagate initial estimate 1 to the same indices of the second night
      night2_IOD_prop.pred1 = propagate_to_times(initial_est1,
ds2.datetime(ts), force_model_4x4_1m2_cr1);
```

```
      for i = 1:length(night2_IOD_prop.pred1.epoch)
      aer1 = eci_to_azelrn(night2_IOD_prop.pred1.epoch(i),
night2_IOD_prop.pred1.position_m(i,:), observer_lla);
      night2_IOD_prop.pred1.azimuth_deg(i,1) = aer1.azimuth_deg;
        night2_IOD_prop.pred1.elevation_deg(i,1) = aer1.elevation_deg;
      end

      % Propagate initial estimate 2 to the same indices of the second night
      night2_IOD_prop.pred2 = propagate_to_times(initial_est2,
ds2.datetime(ts), force_model_4x4_1m2_cr1);

      for i = 1:length(night2_IOD_prop.pred2.epoch)
      aer1 = eci_to_azelrn(night2_IOD_prop.pred2.epoch(i),
night2_IOD_prop.pred2.position_m(i,:), observer_lla);
        night2_IOD_prop.pred2.azimuth_deg(i,1) = aer1.azimuth_deg;
        night2_IOD_prop.pred2.elevation_deg(i,1) = aer1.elevation_deg;
      end

      % Propagate initial estimate 3 to the same indices of the second night
      night2_IOD_prop.pred3 = propagate_to_times(initial_est3,
ds2.datetime(ts), force_model_4x4_1m2_cr1);

      for i = 1:length(night2_IOD_prop.pred3.epoch)
      aer1 = eci_to_azelrn(night2_IOD_prop.pred3.epoch(i),
night2_IOD_prop.pred3.position_m(i,:), observer_lla);
      night2_IOD_prop.pred3.azimuth_deg(i,1) = aer1.azimuth_deg;
        night2_IOD_prop.pred3.elevation_deg(i,1) = aer1.elevation_deg;
      end

      % Setup figure for RMS plots
      output.rms_graph = figure;
      hold on

      % Determine RMS for each position velocity combination
      output.t1.rms = RMS(night2_IOD_prop.obs, night2_IOD_prop.pred1);
      output.t2.rms = RMS(night2_IOD_prop.obs, night2_IOD_prop.pred2);
      output.t3.rms = RMS(night2_IOD_prop.obs, night2_IOD_prop.pred3);

      title(sprintf('%s: Plot of Residuals over Time',name))
      xlabel('Date Time')
      ylabel('Residual (deg)')
      legend('t1 azimuth', 't1 elevation', 't2 azimuth', 't2 elevation', 't3
azimuth', 't3 elevation')

      if ismac
      saveas(gcf,[pwd sprintf('/ResidualGraphs_IOD/%s',name)]) % for mac
    elseif ispc
      saveas(gcf,[pwd sprintf('\\ResidualGraphs_IOD\\%s',name)]) % for win
```

```
      end


      % Return the initial estimates
      output.t1.initial_est = initial_est1;
      output.t2.initial_est = initial_est2;
      output.t3.initial_est = initial_est3;

      % Reformat vector sets into each sub-structure
      output.t1.posvel = [r1,v1];
      output.t2.posvel = [r2,v2];
      output.t3.posvel = [r3,v3];

      % Return Observation index into the structure
      output.t1.obs_index = vec(1);
      output.t2.obs_index = vec(2);
      output.t3.obs_index = vec(3);

      % Comparison of values
      rms_compare = [output.t1.rms.value, output.t2.rms.value,
output.t3.rms.value];
      [output.best.rms, k] = min(rms_compare);
      output.best.index = vec(k);

end
```