University of Colorado - Boulder

ASEN 6080
Statistical Orbit Determination | Spring 2025

ASEN 6080 Statistical Orbit Determination: Project 2

Author:
Ian Faber[a]

[a]SID: 108577813

Instructor:
Jay McMahon

Thursday, May 1, 2025

College of Engineering & Applied Science
UNIVERSITY OF COLORADO BOULDER

Writeup of Project 2 for ASEN 6080: Statistical Orbit Determination.

I. Problem Overview

Orbit determination is an important field with many facets and applications. From monitoring global sea levels to providing GPS capabilities, determining where a satellite is at any given point in time is crucial to our modern society.

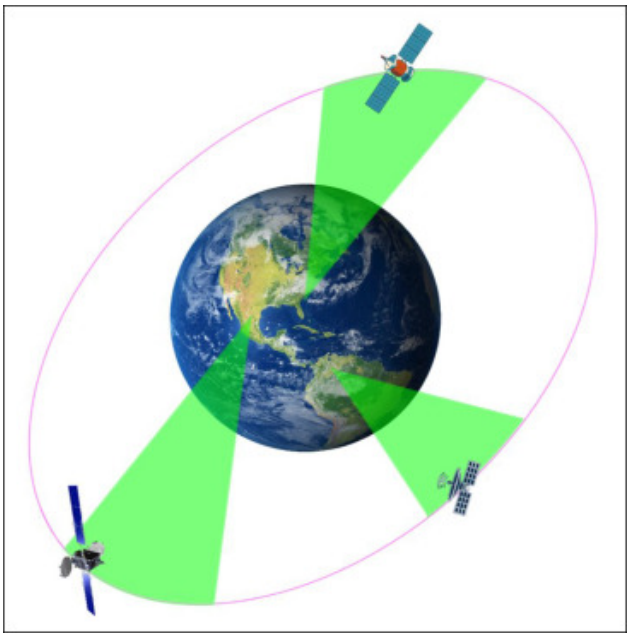

Fig. 1 Cartoon of a typical statistical orbit determination problem [1]

In this project, we were tasked with determining the trajectory of a satellite using a log of observations at specific times from 3 different ground stations: DSS 34 (Canberra Station), DSS 65 (Madrid Station), and DSS 13 (Goldstone Station). Each station provides a range (ρ) and range rate ($\dot{\rho}$) measurement of the satellite, which are modeled as follows [2]:

$$\rho = ||\vec{R} - \vec{R}_s|| \tag{1}$$

$$\dot{\rho} = \frac{(\vec{R} - \vec{R}_s) \cdot (\vec{V} - \vec{V}_s)}{\rho} \tag{2}$$

Where $\vec{R}$ is the satellite position, $\vec{R}_s$ is the ground station position, $\vec{V}$ is the satellite velocity, and $\vec{V}_s$ is the ground station velocity. In particular, we were given 2 data sets, one where we know the ground truth and one where the data log as an unknown number of issues/unmodeled dynamics.

Our ultimate goal was to determine the B-plane target of the satellite, which we were told is modeled after one of the Juno spacecraft's Earth flybys. This required modeling 3rd body gravity and solar radiation pressure in the satellite dynamics, a model of ground station measurements for a hyperbolic orbit, and calculation of the B-plane itself.

II. Scenario Dynamics Model, Measurement Model, and B-plane Calculation Verification

The first part of this project was to make sure we understood the dynamics environment and where the measurements come from, as well as how to calculate a B-plane.

A. Dynamics Model Verification

The first step of the process was to ensure that the new dynamics of the problem had been modeled correctly. In particular, the main dynamics included in this scenario are Earth point mass gravity, Sun third body gravity, and solar radiation pressure (SRP) using the cannonball spacecraft model. The accelerations for each are detailed below [2]:

$$\vec{a}_{\text{Earth, PM}} = \frac{-\mu_{\text{Earth}}}{|\vec{r}|^3}\vec{r}$$

$$\vec{a}_{\text{Sun, 3B}} = \mu_{Sun}\left(\frac{\vec{r}_{\text{sun/sc}}}{|\vec{r}_{\text{sun/sc}}|} - \frac{\vec{R}_{\text{sun/earth}}}{|\vec{R}_{\text{sun/earth}}|^3}\right)$$

$$\vec{a}_{\text{SRP}} = -C_R\frac{P_\Phi}{R^2}\frac{A}{m}\frac{\vec{r}_{\text{sun/sc}}}{|\vec{r}_{\text{sun/sc}}|}$$

$$\vec{a}_{\text{total}} = \vec{a}_{\text{Earth, PM}} + \vec{a}_{\text{Sun, 3B}} + \vec{a}_{\text{SRP}}$$

Where

- $\vec{r}$ is the vector from the center of Earth to the spacecraft.
- $\vec{r}_{\text{sun/sc}}$ is the vector from the spacecraft to the center of the Sun.
- $\vec{R}_{\text{sun/earth}}$ is the vector from the center of Earth to the center of the Sun.
- μ_{Earth} and μ_{Sun} are the gravitational parameters for the respective bodies.
- C_R is the coefficient of reflectivity for the spacecraft.
- $P_\Phi = \frac{\Phi}{c}$ is the solar momentum flux at the spacecraft's distance from the sun, where Φ is the solar pressure flux at the same distance and c is the speed of light.
- R is the distance of the spacecraft to the sun, non-dimensionalized by 1 AU.
- $\frac{A}{m}$ is the area to mass ratio of the spacecraft.

For verifying the dynamics, we were told that the spacecraft had an area to mass ratio of 0.01 m^2/kg and C_R of 1.2, and that Φ is 1357 W/m^2 at 1 AU, which is a good approximation for the distance of the spacecraft to the sun over this scenario. Further, we utilized an ephemeris model for the Earth around the Sun in order to get $\vec{R}_{\text{sun/earth}}$, which required knowing that the initial epoch of this scenario in JD is 2456296.25. Finally, we modeled the rotation of the earth as a sphere with a period of ~24 hours around its north pole (the z-axis), with an initial rotation in the ECI frame of 0° at the initial epoch. Finally, we were given the actual trajectory over the first 50 days for comparison in order to verify that our dynamics were correct. Implementing all of this resulted in Figures 2 and 3.

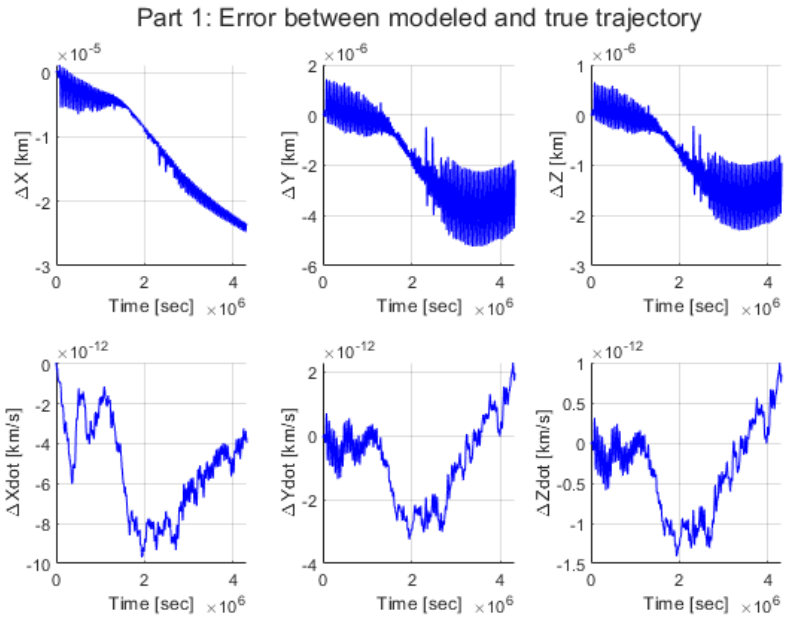

Fig. 2 Error between truth trajectory and propagated dynamics

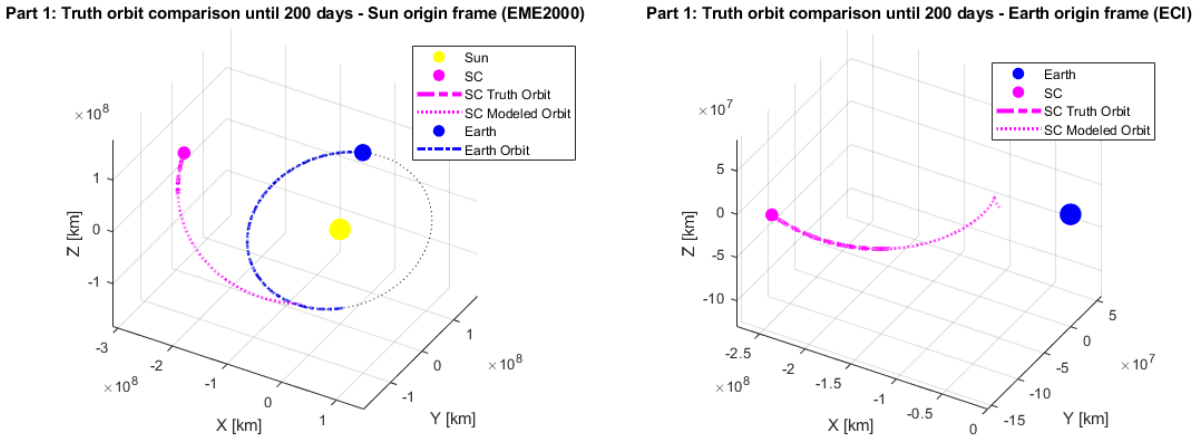

Fig. 3 Dynamics in EME2000 and ECI frames

Clearly, the propagated dynamics and truth trajectory match to the tolerance of my integrator (ode45 with AbsTol and RelTol of 1e-12), and the dynamics look physically correct in the EME2000 and ECI frames. Thus, I'm confident that my dynamics implementation is correct!

B. Measurement Model Verification

The next thing to do was to verify that the measurement model was coded correctly. The provided measurement log was formatted differently from the data log provided in Project 1, so I had to modify my data log ingestion scheme. Once that was done, I read in the data log for part 2 and plotted it to see what we were working with, the results of which can be seen in Figure 4.

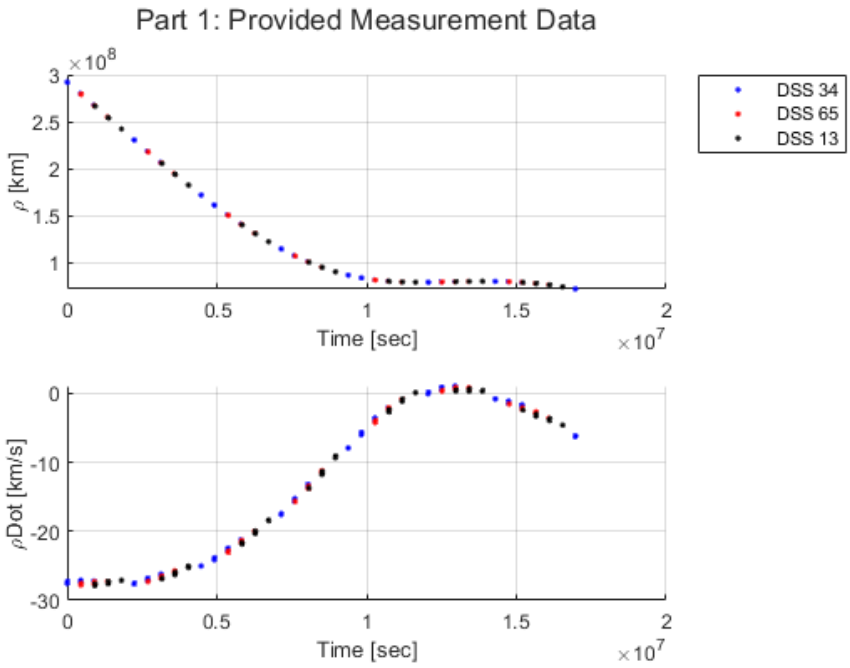

Fig. 4 Processed measurements provided for Part 2

Then, I attempted to recreate the measurements given my verified truth dynamics by leveraging the initial coordinates of each ground station and implementing the measurement model described in Section I. Doing so resulted in Figure 5.

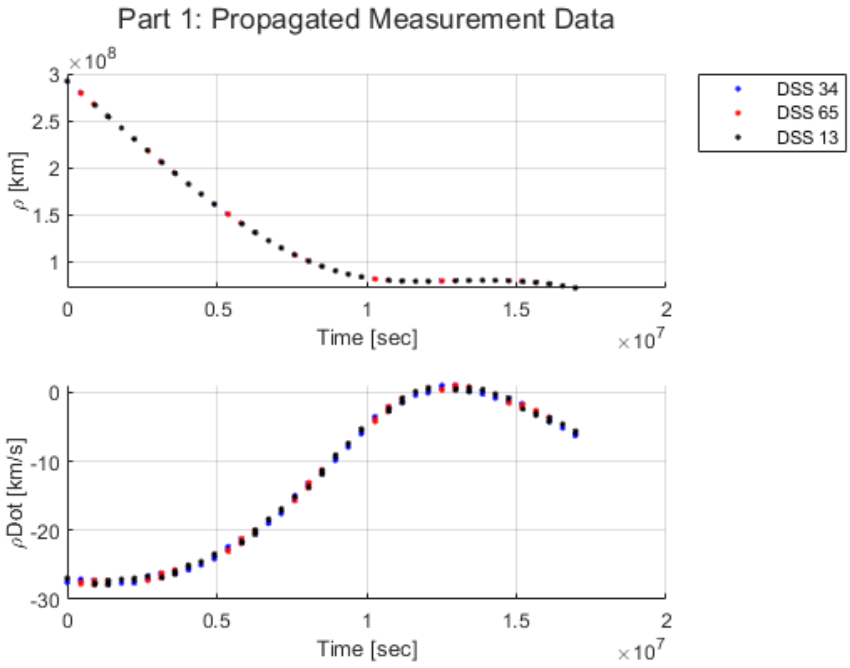

Fig. 5 Propagated measurements based on provided truth trajectory for Part 2

Clearly, Figures 4 and 5 match almost identically in shape. The only difference between the two is that the provided measurements have been cherry-picked to provide measurements in 60 second intervals at specific times, and manipulated so that only one station provides a measurement at a time (which was very nice of Professor McMahon!). Thus, I'm confident that my measurement model implementation is correct!

C. B-plane Calculation Verification

Finally, I needed to verify that I could correctly calculate the B-plane of a hyperbolic trajectory, as the entire point of this project is to take in a set of measurements and predict where a spacecraft will fly by Earth. Thus, in order to do so I needed to implement an algorithm to execute the following steps:

1) Propagate the final filtered state estimate forward in time to 3x the Earth's sphere of influence, which was taken to be $R_{\text{SOI}} = 925000$ km for this project.
2) Calculate the resulting B vector based on the incoming hyperbolic velocity and radius: $\vec{B} = \vec{r}_\infty - (\vec{r}_\infty \cdot \vec{v}_\infty)\vec{v}_\infty$
3) Calculate the linearized time of flight to approximate when the spacecraft will pass the B-plane.
4) Propagate the covariance at $3R_{\text{SOI}}$ to the B plane crossing.
5) Calculate the STR frame and subsequent DCM from the STR frame to ECI frame.

After implementing the above algorithm and attaching a dummy covariance to the end of the truth trajectory, I got Figure 6.

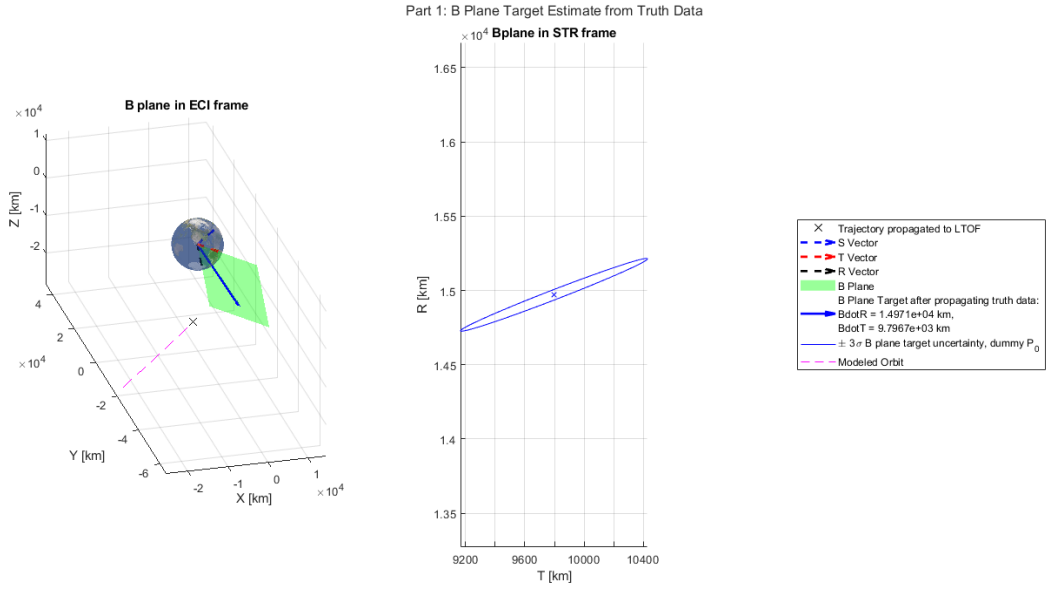

Fig. 6 Propagated B-plane target based on truth trajectory

From Figure 6, I got a propagated B-plane target of $B \cdot \hat{R} = 14971$ km and $B \cdot \hat{T} = 9796.7$ km. We were told that the truth trajectory provided had a true B-plane target of $B \cdot \hat{R} = 14970.824$ km and $B \cdot \hat{T} = 9796.737$ km. Thus, my propagated B plane clearly matches the truth, and I'm confident that my B-plane calculation algorithm is correct and accurate!

III. Estimating Spacecraft State with a Known Target and Models

The next step of the project was to apply a filter to the flyby in order to estimate the state of the spacecraft with known models and a known B-plane target. This is to verify that our filters are updated with the dynamics and measurement models from Section II, and to test the B-plane algorithm with an actual covariance estimate.

A. DSN Measurement Processing

Using my verified measurement ingestion scheme from Section II, I was able to back out the provided measurements for this part. One thing I had to be aware of is that the Madrid station had it's longitude as opposite the correct value, which required a small bit of extra care. All of that being said, the provided measurements are shown in Figure 7, which match the measurements presented in Section II.

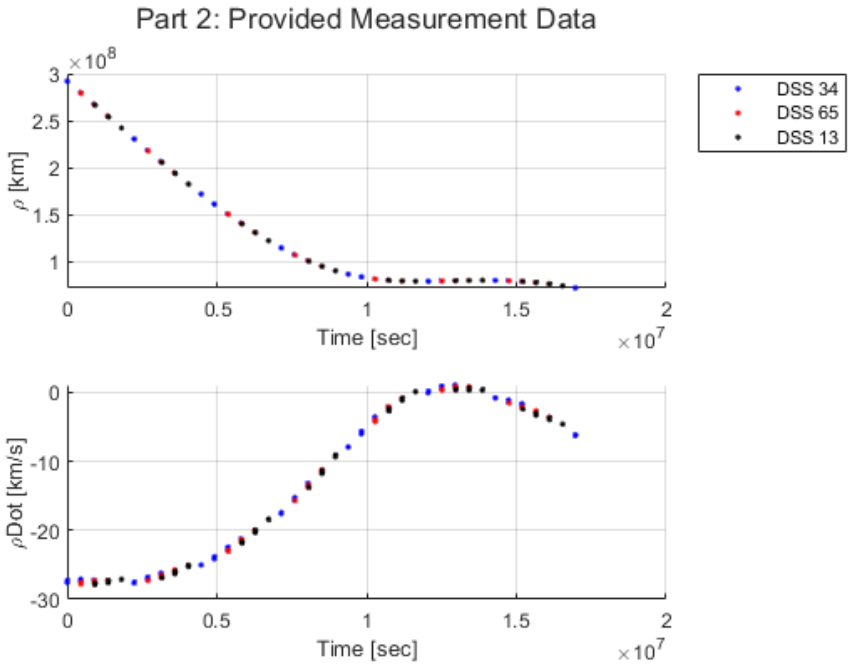

Fig. 7 Provided measurements for Part 2

These measurements look like what we'd expect from a hyperbolic orbit, so I decided to move on to filtering the observations with my verified dynamics and measurement models.

B. DSN Measurement Filtering

To filter the DSN measurements, we were given a specified a-priori state as follows:

$$
\begin{aligned}
X &= -274096790.0 \text{ km} \\
Y &= -92859240.0 \text{ km} \\
Z &= -40199490.0 \text{ km} \\
\dot{X} &= 32.67 \text{ km/s} \\
\dot{Y} &= -8.94 \text{ km/s} \\
\dot{Z} &= -3.88 \text{ km/s} \\
C_R &= 1.2
\end{aligned}
$$

Further, we were given a specified a-priori covariance of $\texttt{diag}\left(\left[\sigma_r^2, \sigma_v^2, \sigma_{C_R}^2\right]\right)$, with 1-$\sigma$ diagonal entries as follows:

$$\sigma_r = 100 \text{ km}$$
$$\sigma_v = 0.1 \text{ km/s}$$
$$\sigma_{C_R} = 0.1$$

To filter this part of the project, I found that all I needed for decent performance was a simple LKF! Since our dynamics are known, I was confident that deviations from the reference trajectory that the LKF estimates would always be within the linear regime, and if I was smart with how I initialized each set of measurements, I could get the LKF to behave well. I was mostly successful in this endeavor!

My basic strategy with the LKF was to break up the measurements into 50 day arcs, where I would reset the initial covariance at the start of each new arc to effectively make each set of measurements independent. To do this, I propagated the final covariance of the last best estimate forward to the start of the next measurement gap. Then, I inflate the resulting covariance by a factor of 10000. This accomplishes 2 things: First, I can avoid the issue of the LKF becoming saturated after processing too many measurements. Second, by seeding the inflated covariance with a covariance that is propagated using the same dynamics, I achieve dynamic filter consistency between arcs. I could have just reinitialized each measurement arc with just the initial covariance, but this could lead to infeasible orbits and unreasonably large Kalman gain updates in the filter. The data log had 200 days of measurements, which resulted in 4 filtering arcs and 4 successively more confident B-plane targets, most of which included the correct truth B-plane target that was provided.

1. Filtering from 0 to 50 days

With the first measurement arc, I encountered a strange anomaly with the range prefit residuals. Despite having my dynamics match the truth data to integrator precision, I saw a slow divergence of the prefit residuals, which were mostly noise otherwise. This is most likely because my LKF algorithm didn't iterate enough to converge, but no matter how I changed my convergence criteria I couldn't get my LKF to iterate more than once. This, and the fact that the next sections of data all look as expected, made me decide to move onwards with the project. Further, all of the state errors are within the 3σ bounds. However, at the time of writing I still haven't resolved this issue. Figures 8 and 9 show the residuals and state errors for this filtering arc:

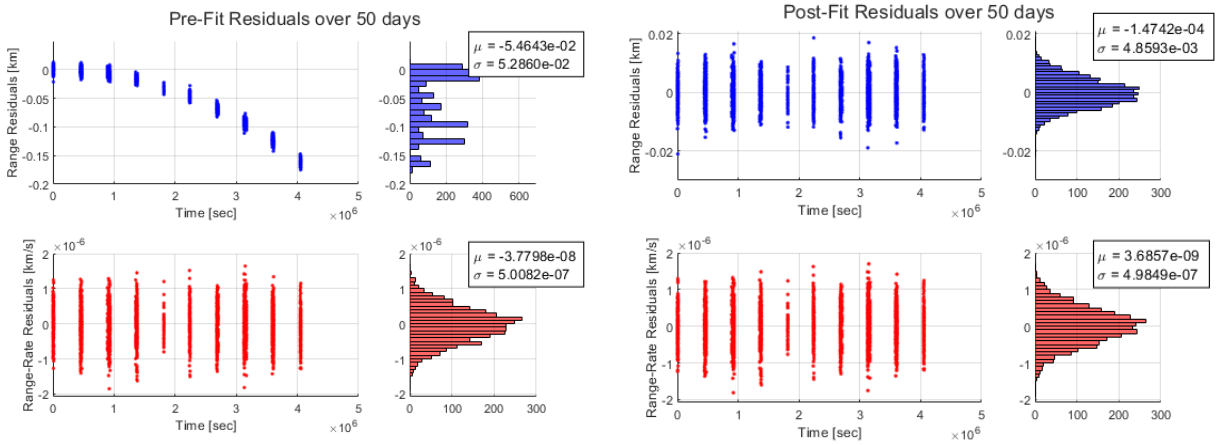

Fig. 8 Part 2 LKF Residuals from 0 to 50 days

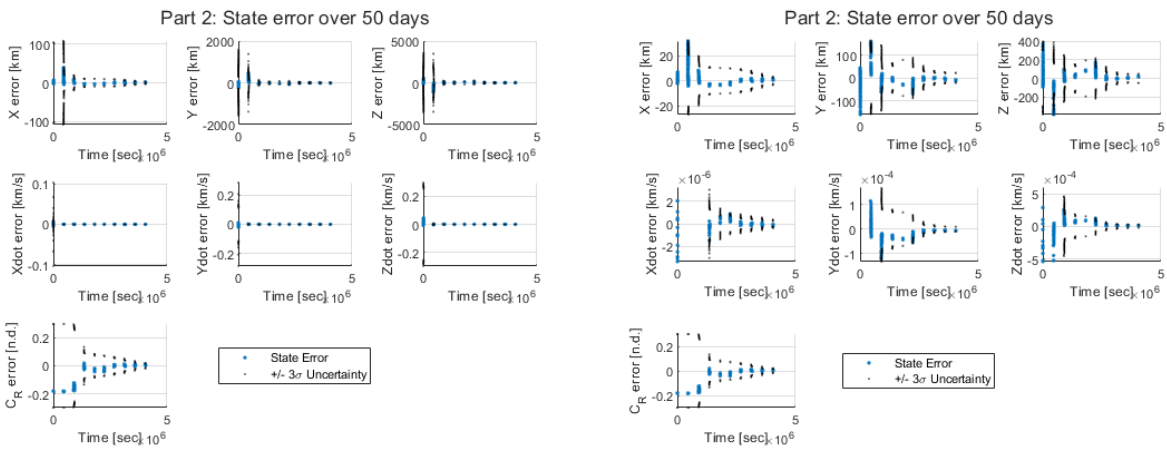

Fig. 9 Part 2 LKF State Errors (original scale and zoomed in) from 0 to 50 days

2. Filtering from 50 to 100 days

In contrast to the first filtering arc, the arc from 50 to 100 days worked out beautifully, with no residual anomalies and all state errors within the 3σ bounds. Figures 10 and 11 show the residuals and state errors from the initial time through the end of this filtering arc:

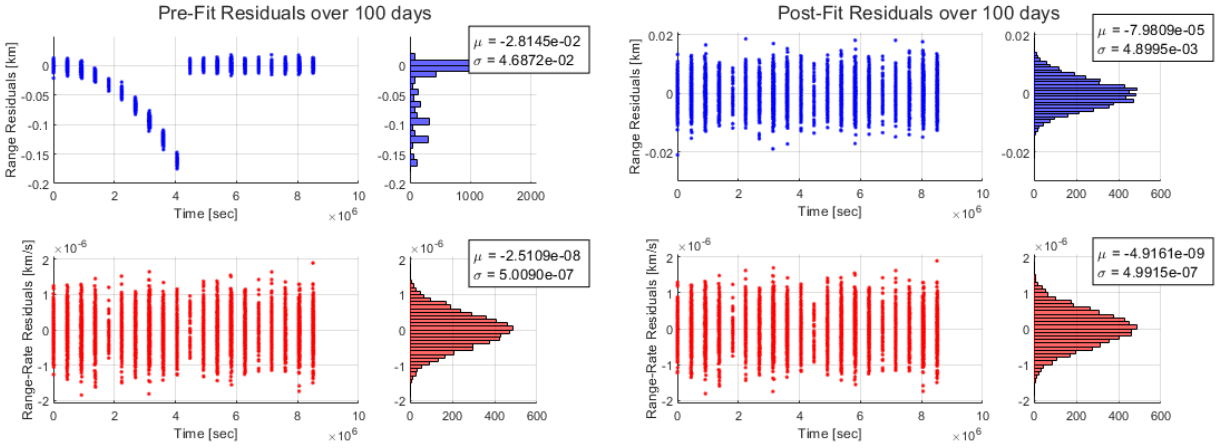

Fig. 10 Part 2 LKF Residuals from 0 to 100 days

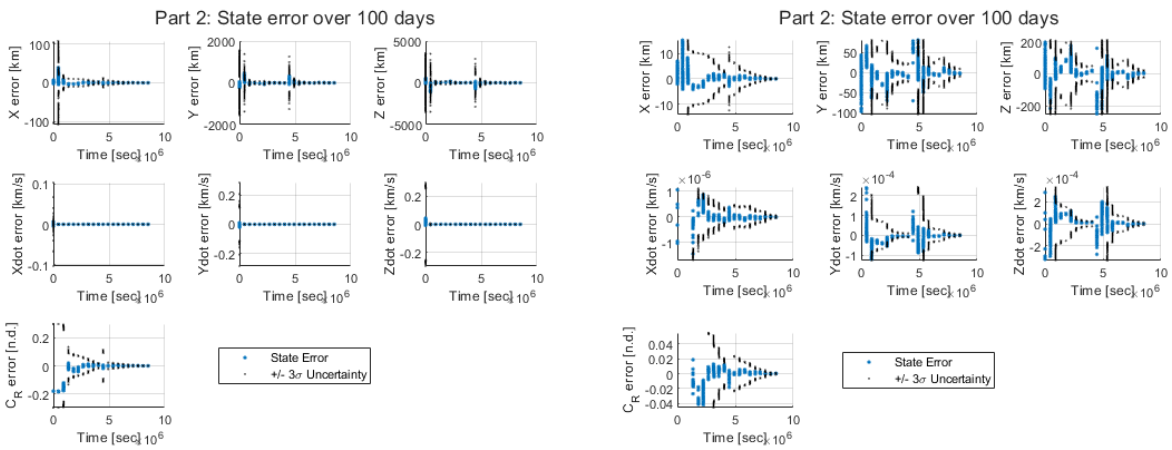

Fig. 11 Part 2 LKF State Errors (original scale and zoomed in) from 0 to 100 days

Of particular note is the covariance spike at the start of the filtering arc, which worked exactly as planned! Without that spike, the filter likely would have saturated and converged to a biased estimate.

3. Filtering from 100 to 150 days

The third filtering arc started off great, then started to become saturated in the $\dot{X}$ estimate. This can be seen by how close the estimate gets to the 3σ bound, implying that it may not converge to an unbiased estimate. However, given that Part 2 is just a verification step and all the other estimates looked good, I decided to leave it as is and move on. Figures 12 and 13 show the residuals and state errors from the initial time through the end of this filtering arc:

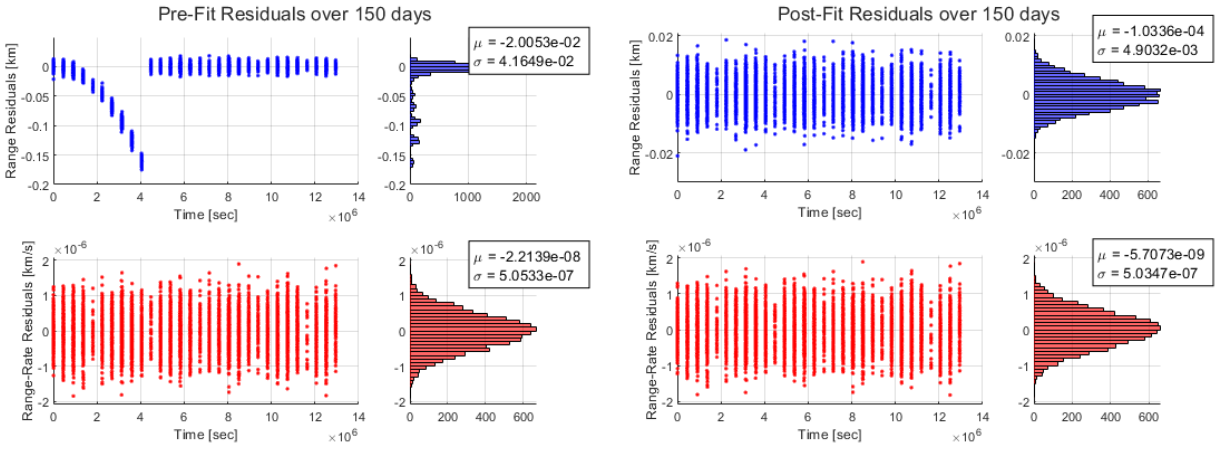

Fig. 12 Part 2 LKF Residuals from 0 to 150 days

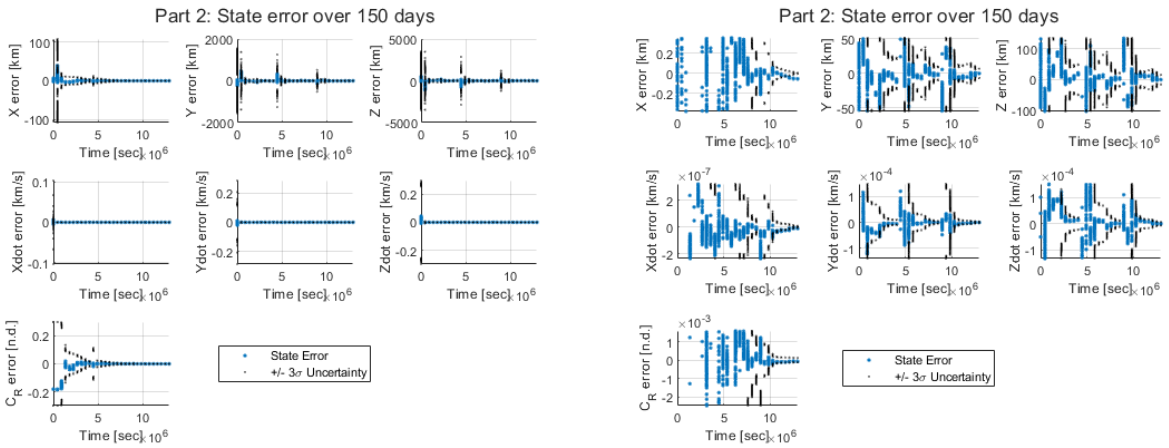

Fig. 13 Part 2 LKF State Errors (original scale and zoomed in) from 0 to 150 days

4. Filtering from 150 to 200 days

The last filtering arc really started to show signs of saturation, most likely because of the fact that I didn't fix the saturation from the last arc and the fact that my method of propagating and inflating the covariance still has its limits. In particular, if the propagated covariance gets smaller and smaller from arc to arc, eventually inflating it by some constant scale won't be enough to ensure the filter doesn't saturate. This resulted in a biased final X and C_R estimate, with the other estimates threatening to saturate as well. However, the residuals still converged to noise in both the prefits and postfits, and the final state estimate resulted in a B-plane estimate and covariance with the truth target incredibly close to the bounds, so I didn't put too much effort into fixing it in the interest of time and getting Part 3, arguably the more important part, to work. Figures 14 and 15 show the residuals and state errors from the initial time through the end of this filtering arc, and hence over the entire Part 2 scenario:

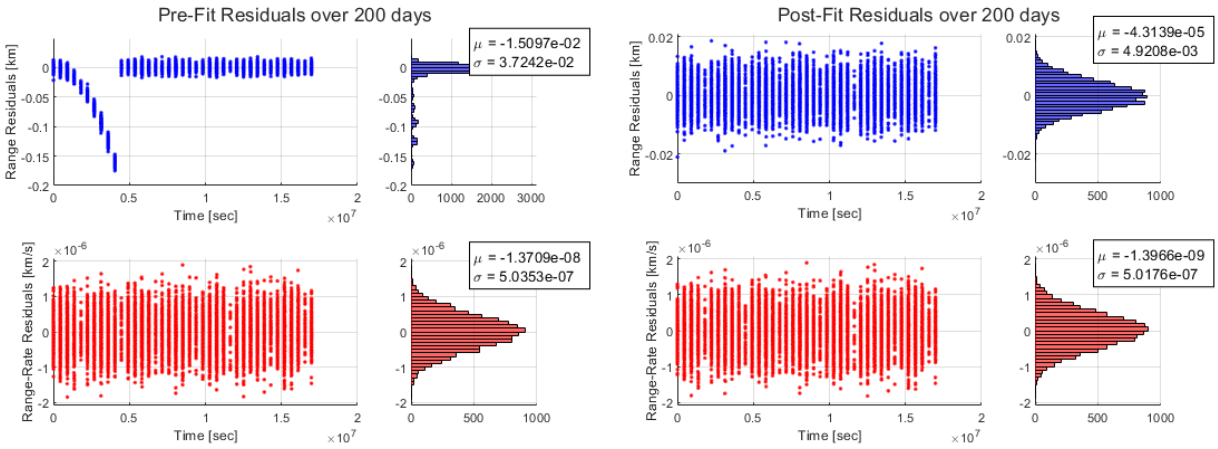

Fig. 14 Part 2 LKF Residuals from 0 to 200 days

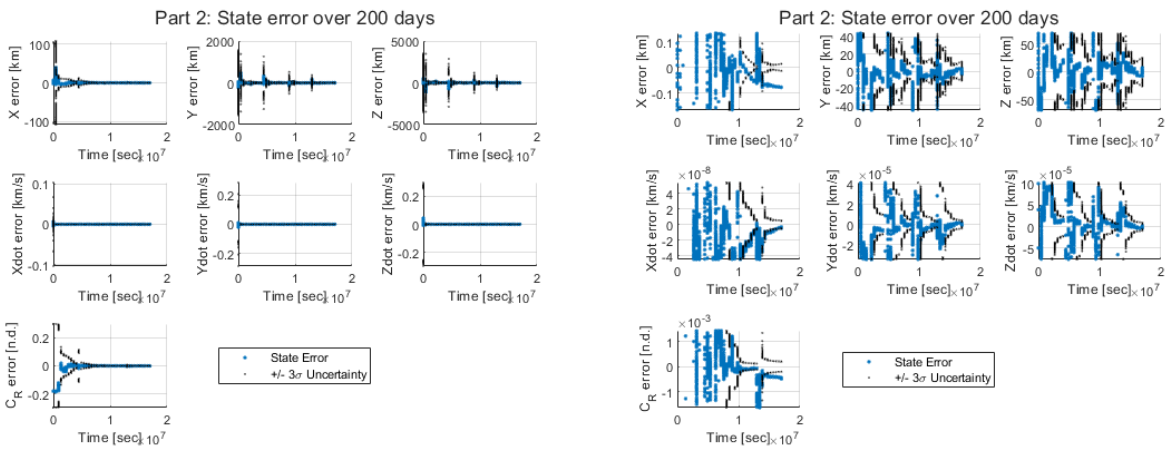

Fig. 15 Part 2 LKF State Errors (original scale and zoomed in) from 0 to 200 days

5. B-plane targets from each filtering arc

As expected, as I processed more data, my covariance bounds on the B plane target got smaller and smaller, and for the most part each updated covariance and B-plane target lies within all previous sets of covariance bounds. The covariance of the third arc doesn't lie completely within the covariance of the second arc due to the near saturation of the filter near the end, and the estimate of the last filter doesn't lie within the covariance of the second arc due to saturation, but the overall trend is there and all of the covariance bounds include (or very nearly include, for the last arc) the truth target. Thus, I called Part 2 mostly good and moved onto Part 3. Figure 16 shows the evolution of the B-plane targets as more data was processed, and Figure 17 shows the same B-plane but zoomed into the final 3 estimates and the truth.

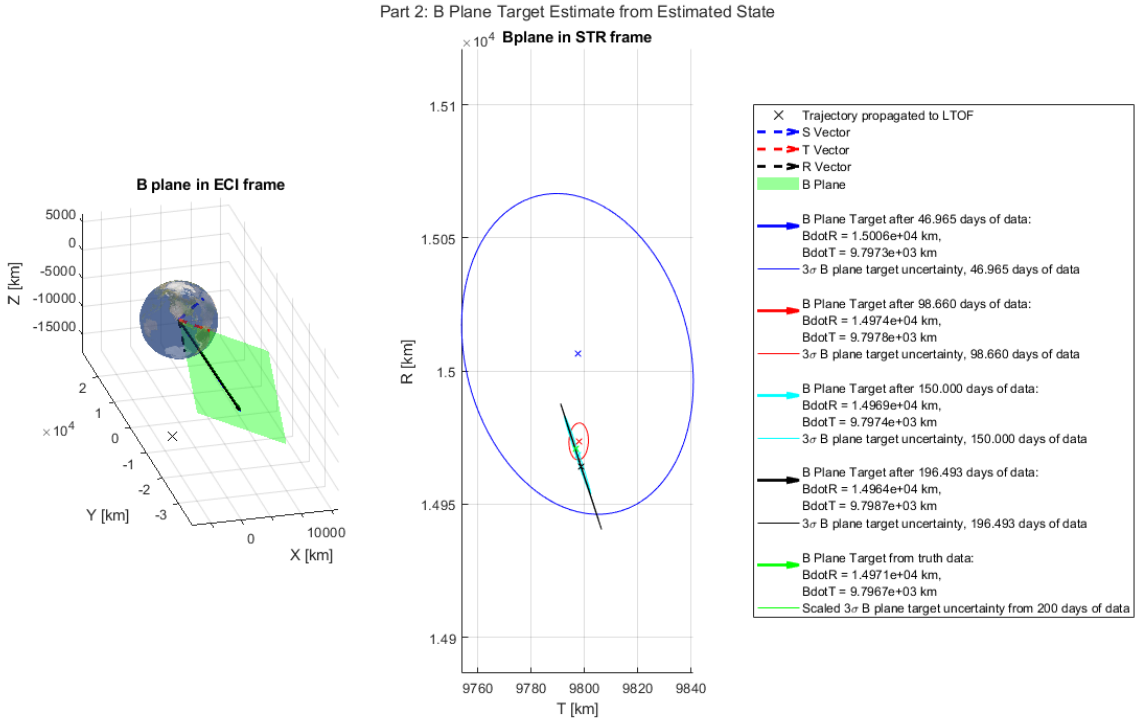

Fig. 16 B-plane Evolution for Part 2

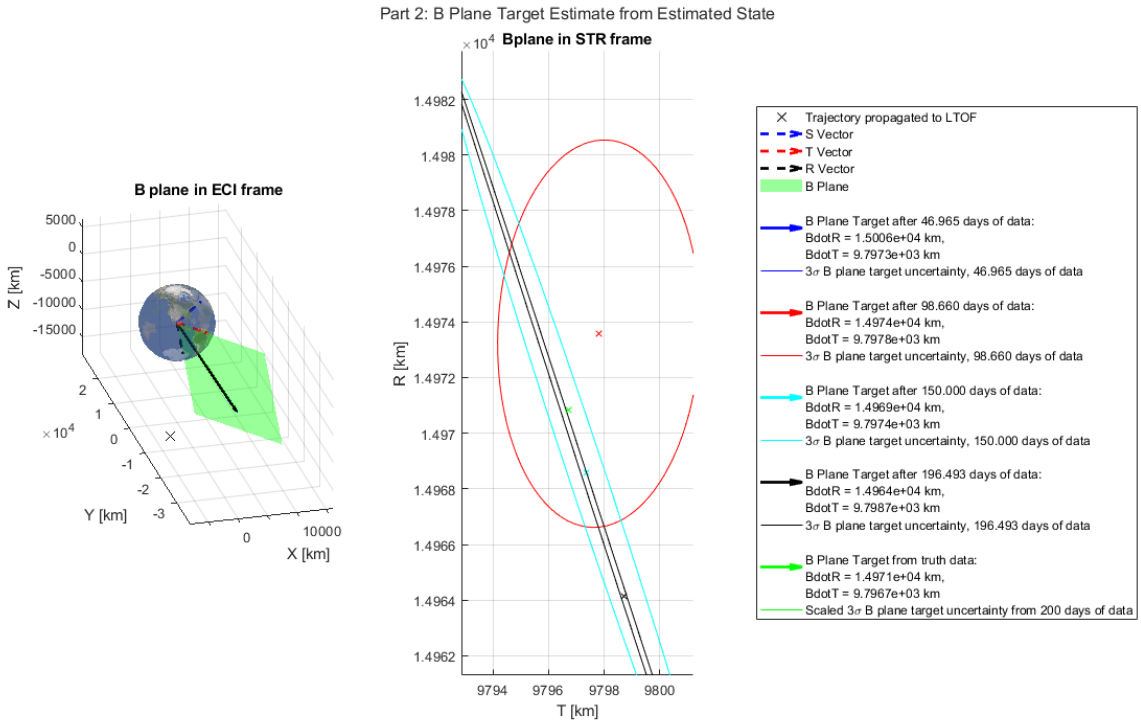

Fig. 17 B-plane Evolution for Part 2 (zoomed in)

IV. Estimating Spacecraft State with Unknown Dynamics and Data Issues

Next was the fun part of this project, where we got to play as real GNC engineers and fit a set of observations only knowing an initial state and some assumptions about the dynamics of the problem! For this part of the project, we were told that there were definitely issues with the data, as well as potentially unknown dynamics at play. This section details my journey as I went through Part 3, culminating in a final solution that I am quite happy with!

For this part, the only guidance we were provided was an a-priori state to use, as well as an initial covariance. The initial covariance is identical to Part 2, but the a-priori state was as follows:

$$
\begin{aligned}
X &= -274096770.76544 \text{ km} \\
Y &= -92859266.4499061 \text{ km} \\
Z &= -40199493.6677441 \text{ km} \\
\dot{X} &= 32.6704564599943 \text{ km/s} \\
\dot{Y} &= -8.93838913761049 \text{ km/s} \\
\dot{Z} &= -3.878819140503316 \text{ km/s} \\
C_R &= 1.0
\end{aligned}
$$

The biggest difference between the a-priori state for Part 3 vs. Part 2 is that C_R is now 1.0 instead of 1.2, and the initial state is much more precise than Part 2. However, by and large it appears to be the same overall initial state, which clued me into thinking that the overall situation for Part 3 is similar to Part 2.

A. DSN Measurement Processing

As with part 2, the very first thing I did was to process the provided measurements and see if I could spot any obvious differences between them and what I would expect. After processing the measurements, I was presented with Figure 18.

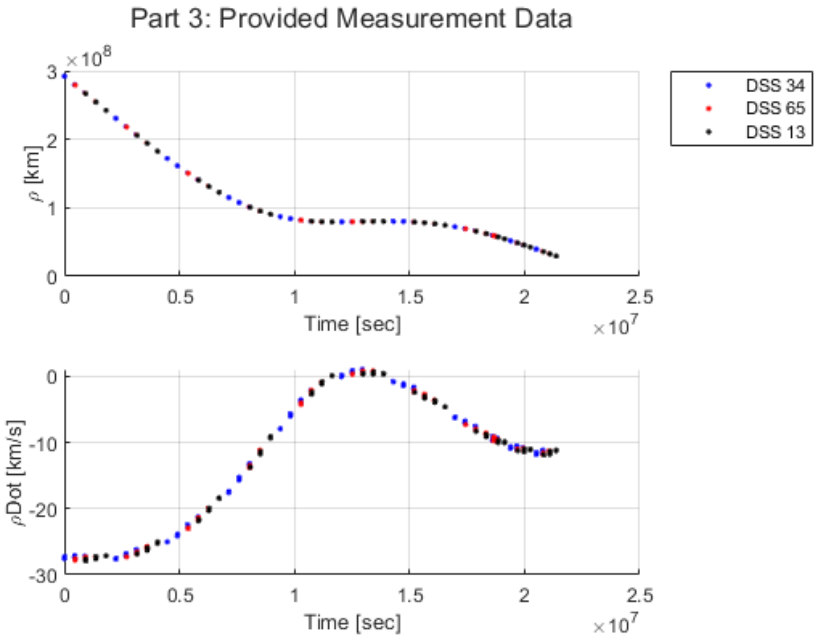

Fig. 18 Provided Measurements for Part 3

Then, I propagated the given initial state forward in time and calculated measurements at the same timesteps as the given data to see if there were any obvious differences. To do so, I kept in mind that the Madrid's longitude was now noted to be positive rather than negative, as in Part 2. Propagating the initial state forward in time resulted in the following expected measurements in Figure 19.

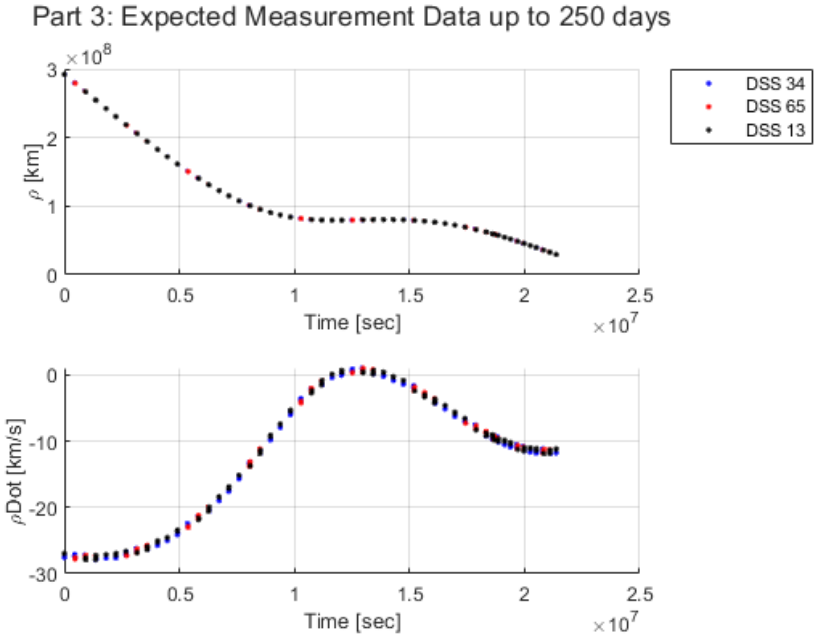

Fig. 19 Expected Measurements for Part 3

At first glance, there isn't anything clearly different between the two, other than the same station cherry picking as discussed in Part 2. This was my first indication that this problem was *not* going to be as easy to solve as I initially thought. Suddenly, two weeks didn't feel like a lot of time at all!

B. Initial Filtering Attempts

After looking at the measurements, it was time to try filtering them. Unfortunately, plots are sparse in this section as I tore down and rewrote code in a frantic effort to get something to work. Further, my intent with this section was to provide a story of my process, and including every debug plot I generated would balloon this report to unreasonable sizes for little payoff. However, where things survived into the final code, I have provided plots!

1. Vanilla LKF

The first filter that I tried (and expected to fail) was the LKF from Part 2. As expected, it did not do a good job. In fact, I'd categorize the job it did as catastrophic. After the first measurement arc, the filter diverged. It then proceeded to diverge even further across the second measurement arc, in which the pre- and postfit residuals jointly decided to take a hiatus to Alpha Centauri. In other words, they got exponentially worse. They got so bad that their magnitudes reached levels rivaling the distance to our nearest and dearest stellar neighbor. This pointed to something happening between the first and second measurement arcs, as well as over the second measurement arc. This clued me into the idea that I might need to use something more robust. In hindsight, this was one of the biggest understatements uttered in recent human history.

2. LKF with DMC

The next thing I tried was to add DMC to the LKF. This fared better during the first measurement arc, but still handled the gap between the first and second measurement arcs poorly. I attempted to tune the DMC parameters until I could get it past the measurement gap, but everything I tried didn't seem to work. The best parameters I got to work were σ_u = 2.5e-12 km/s^2 and τ = 1 hour in each axis. These parameters got my residuals in the first measurement gap to approach noise, and the state estimates didn't explode too badly across the measurement gap. However, as the filter progressed across the second arc, it appeared that the state deviations got so large that they likely weren't in the linear regime of the reference trajectory. This made me think that there must be a maneuver in between the first and second measurement arcs, and that I needed something that could dynamically update the reference as it inevitably changed between arcs. Enter: the EKF!

3. EKF initialized with 100 DMC LKF measurements

The next filter I tried in order to test my shifting reference trajectory hypothesis was an EKF initialized with 100 measurements of an LKF using DMC. The first measurement arc was very well-behaved and the LKF had no issues with processing it, so I thought this was a good way to initialize my EKF. The resulting state errors from the LKF over 100 measurements are shown in Figure 20.

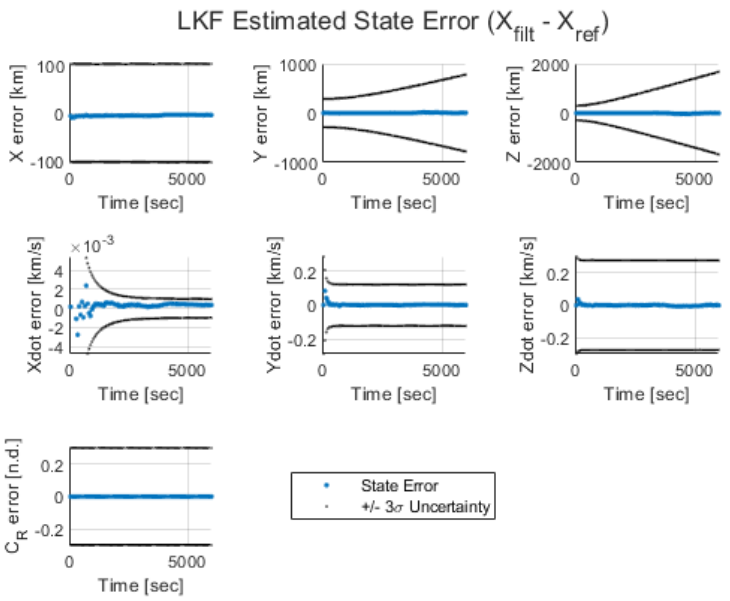

Fig. 20 State Errors for LKF over the first 100 measurements of Part 3

Clearly, the state errors are well within the 3σ bounds of the reference trajectory. Now we're getting somewhere! The first thing I immediately noticed was the rapid reduction in $\dot{X}$ uncertainty compared to every other state. I believe this is because the spacecraft is approaching Earth from mostly the X direction of the ECI frame, meaning that any measurements of the spacecraft are much more likely to affect state estimates along this direction. Indeed, we can see that the uncertainty in the X state stays relatively constant, while the uncertainty in Y and Z appears to grow due to this phenomenon. Further, the prefit residuals over the first measurement arc all went to noise, which is indicative that we have correctly modeled the trajectory. These residuals can be seen in Figure 21.

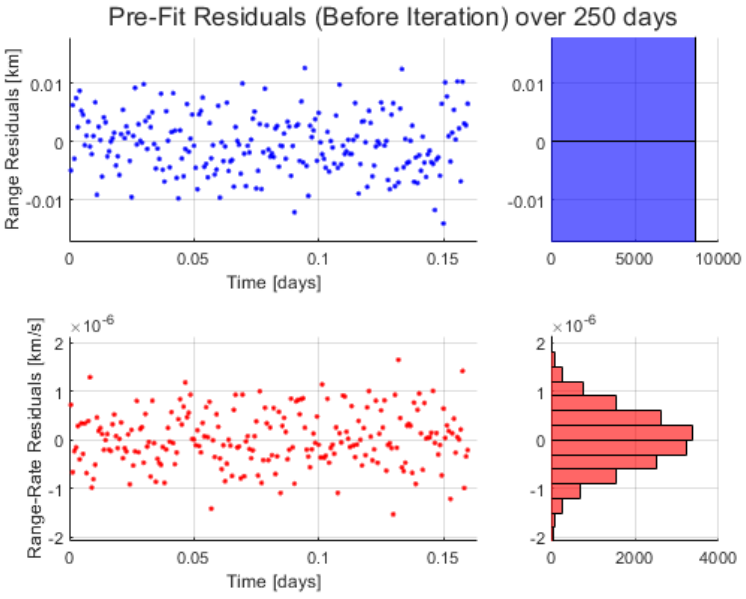

Fig. 21 Part 3 First Measurement Arc Prefit Residuals

My excitement at seeing the LKF with DMC perform well over the first 100 measurements and subsequently seeing the EKF finish out the first measurement arc was quickly dashed when the EKF attempted to deal with the first measurement gap, and promptly exploded its state estimates. Undeterred, I attempted to add DMC to the EKF, only to see very slightly better results. Clearly, the EKF's ability to change the reference trajectory wasn't nearly agile enough to handle whatever was happening between the first two measurement arcs. Feeling slightly disheartened, I brainstormed potential ideas to try. I liked the idea of having a filter that could attempt to handle anything thrown at it and estimate what was giving it trouble, so I knew that I wanted to keep DMC to detect deterministic modeling errors. I also knew that I wanted something that could update the reference trajectory in response to maneuvers, as I was tunnel visioned into getting a linear filter to work and all linear filters either require or manipulate a reference trajectory.

Then, I remembered Professor McMahon's lecture on the Iterated EKF, in which the basic idea is to repeat the measurement update, and hence update the reference trajectory, a number of times until the postfit residuals approach noise. I hadn't seriously considered the IEKF until now, as I didn't think the changes to the underlying reference trajectory would be so bad as to need to repeatedly iterate on it. However, I was now sufficiently desperate enough to give it a try. Spoiler alert: It worked magnificently!

C. The Solution and In-Depth Analysis/Discussion: Iterated EKF with DMC

My final and most successful attempt at trying to filter Part 3's data was to implement an iterated EKF. After getting confused by the notation and making some embarrassing coding mistakes, I got the filter to reproduce my Part 2 results, which gave me confidence that it was working. Then, I applied it to Part 3. As in my last attempt, I initialized the filter with 100 DMC LKF measurements, then I let the IEKF run for the rest of the data window. The idea is that the filter would alternate between a normal EKF when the residuals look acceptable, then when an egregious change in the reference trajectory occurred, iterate on it until the residuals become acceptable. I capped the number of iterations to 9999, and set an acceptable residual threshold as 2σ. For the DMC parameters, I chose σ_u = 2.5e-13, with τ = 0.825 hours in each axis.

1. Residual Analysis

Thanks to the nature of the IEKF, we can glean a lot of information just by looking at the filter pre- and postfits. The pre- and postfits of my IEKF over the entire scenario are shown in Figure 22. Here, I've defined prefits as the residuals before iterating the reference trajectory, and postfits as the residuals after iterating the trajectory. Thus, if the postfits approach noise, then we can say we've done a good job iterating the reference trajectory to fit the data, and the next set of prefits will be propagated from the correct reference.

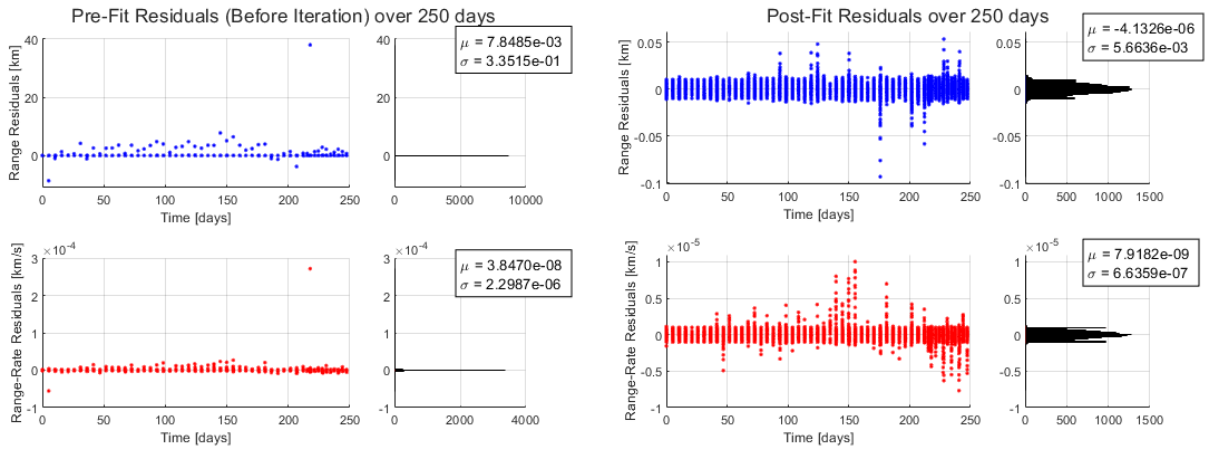

Fig. 22 Part 3 IEKF Residuals from 0 to 250 days

Let's start by analyzing the prefit residuals. We can see in the prefits of Figure 22 that there are 2 particularly bad outliers, one at roughly 5 days and one at roughly 217 days, which implies that something significant happened to the

reference trajectory at those times. My leading theory is that these are maneuvers, especially given their scale relative to the other residuals. We can highlight the offending residuals in Figure 23.

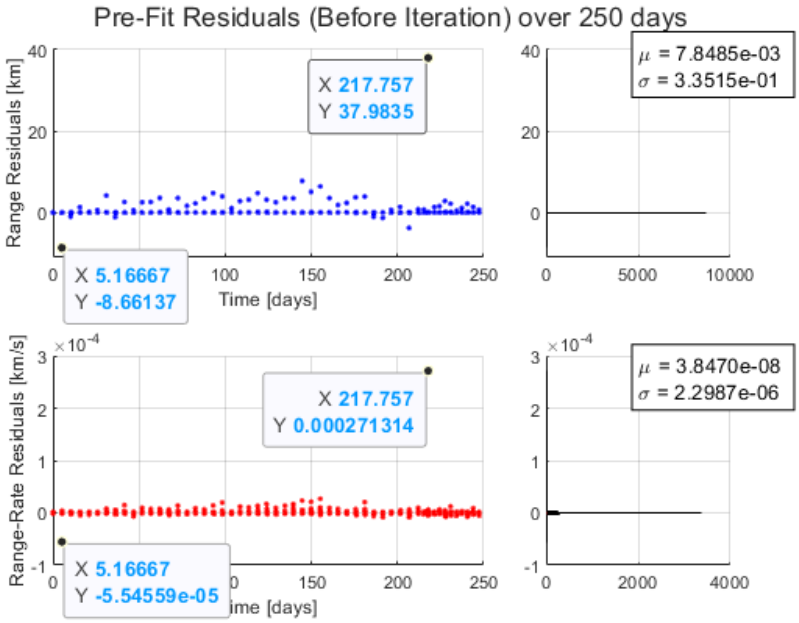

Fig. 23 Part 3 Potential Maneuver Prefit Residuals

We can see that the first range rate outlier at 5 days has a value of -5.5e-5 km/s, or about -0.055 m/s. This is a plausible maneuver for a low thrust spacecraft, and the corresponding negative range residual of -8.66 km could be explained by a maneuver that slowed the spacecraft compared to what the measurement model would predict. The second range rate outlier at 217 days has a value of 2.7e-4 km/s, or about 0.27 m/s. This is also a plausible maneuver for a low thrust spacecraft, and the correspondingly larger range residual of 37.98 km could be explained by a maneuver that sped up the spacecraft compared to what the measurement model would predict. Zooming into the residual plots to ignore these maneuvers, we can also pick out a few other patterns in Figure 24.

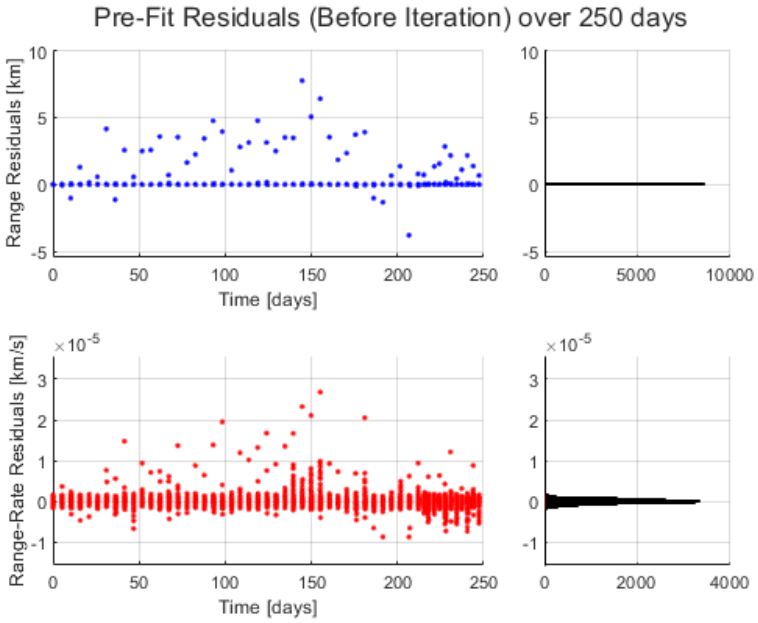

Fig. 24 Part 3 Prefit Residuals, zoomed in

Here, we can see a clear pattern in the range residuals, namely, that there is a spike at the start of nearly every measurement cluster. I believe this is due to some ground station modeling error, i.e. some mismatch in station coordinates or deliberate doctoring to get a pattern that doesn't match the dynamics. The outlier range residuals at the start of each arc, excluding the potential maneuvers, vary from around 1 to 7.7 km. On the scale of our problem, this could very easily be caused by a mismatch of the ground station coordinates on the order of a few degrees in longitude or latitude. Indeed, looking up the true coordinates of Madrid gives coordinates of 40.4167° N, 3.7033° W (or 356.2967° E), as compared to our given values of 40.427222° N and 355.749444° E. There are similar mismatches for both Canberra and Goldstone. Thus, we could have been given the wrong coordinates, but the data was generated with the correct ones. This would also explain the corresponding spikes in the velocity residuals at the start of each measurement arc, as the measurement model expects the stations to have a different velocity vector than the data would suggest.

Finally, if we zoom even further into the prefit residuals as in Figure 25, we can see that the rest of the residuals are nearly all noise. Thus, this suggests that the previous iterated reference trajectory accurately fits the data and is likely an accurate estimate. It isn't perfect, but anything resembling noise is a success in my book!

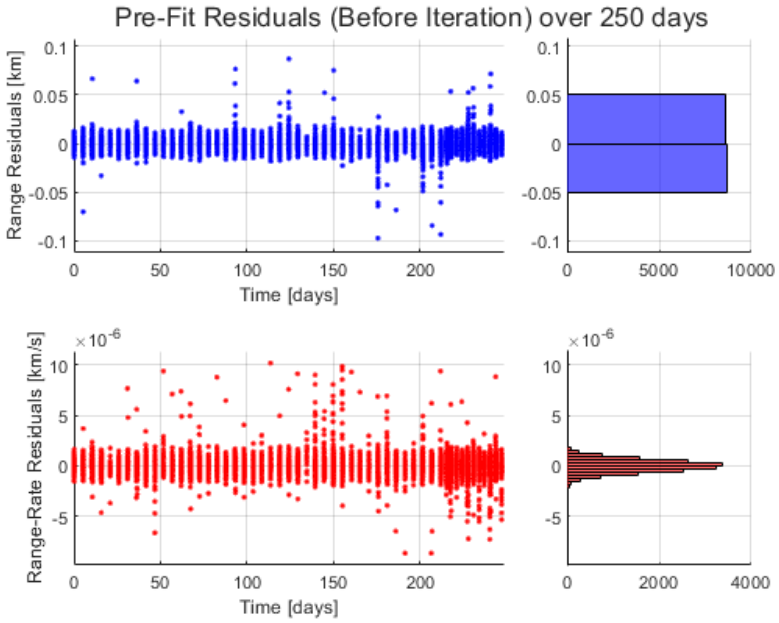

Fig. 25 Part 3 Prefit Residuals, zoomed in to noise

Now, let's talk about the postfit residuals. Across the board, we can see that the postfits in Figure 22 are largely noise. This is good, as it implies that the IEKF properly iterated the reference trajectory to the point that it couldn't fit the data any more than noise. There are a few residual spikes near the end of the data, but they aren't enough to overcome the predominantly noise residuals in the rest of the scenario. However, there are still some issues based in how I coded the filter. If we zoom into the postfit residuals in Figure 26, we can see two distinct spikes at the 2σ level of both the range and range rate histograms, which I've lovingly referred to as "IEKF Horns".

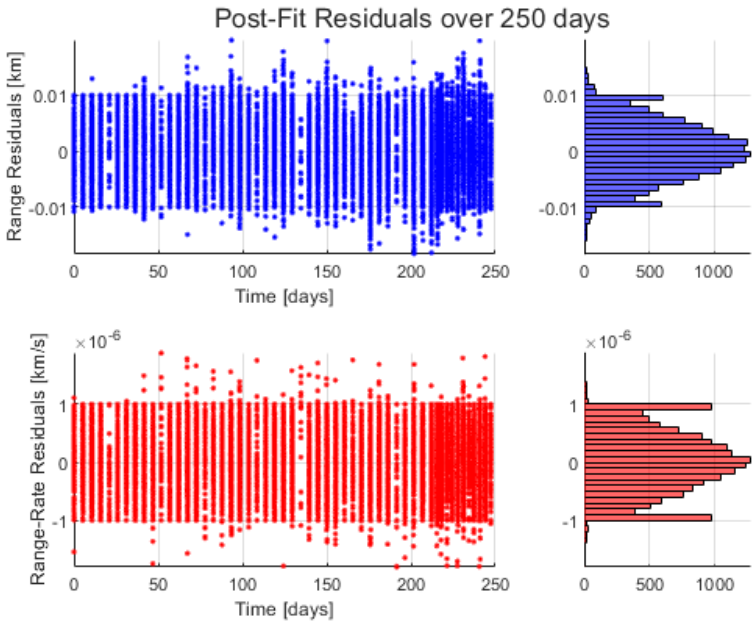

Fig. 26 Part 3 Postfit Residuals, zoomed in to see IEKF Horns

These horns are a direct artifact of how I coded the reference iteration convergence criteria of the IEKF. In essence, I iterate the reference trajectory until the range and range rate residuals approach an acceptable level of noise, which I chose to be 2σ, or 9999 iterations are reached (or 10000 total measurement updates). Thus, some iterations of the reference trajectory get the noise to be 2σ, then don't go any further even if they could. I settled on 2σ because it balanced the runtime of my code and accuracy of the final result, which currently has a postfit RMS of 1.2353 (very good!). If I went any lower on the bound, my code slowed down considerably as more sections of the data tried to iterate to get the perfect residual. The code currently takes 7 minutes to run part 3, which is already pushing it. If I went any higher on the bound, then my final residuals became unreasonable and the entire filter fell apart in terms of accuracy. I'm not sure if this is a coding mistake on my part or if the IEKF is just that sensitive, but an RMS of near 1 is very good for this very uncertain dataset, so I took it and ran. If I had more time and bandwidth, I would dig deep into the weeds to find a better solution, but I'm happy with the result I have even if the final residuals look a bit strange.

2. State Difference Analysis

Next, let's look at what the IEKF actually did to the reference trajectory. Figure 27 shows the difference between the IEKF's final state estimate compared to the initial state propagated with the dynamics I verified in Section II to 250 days.

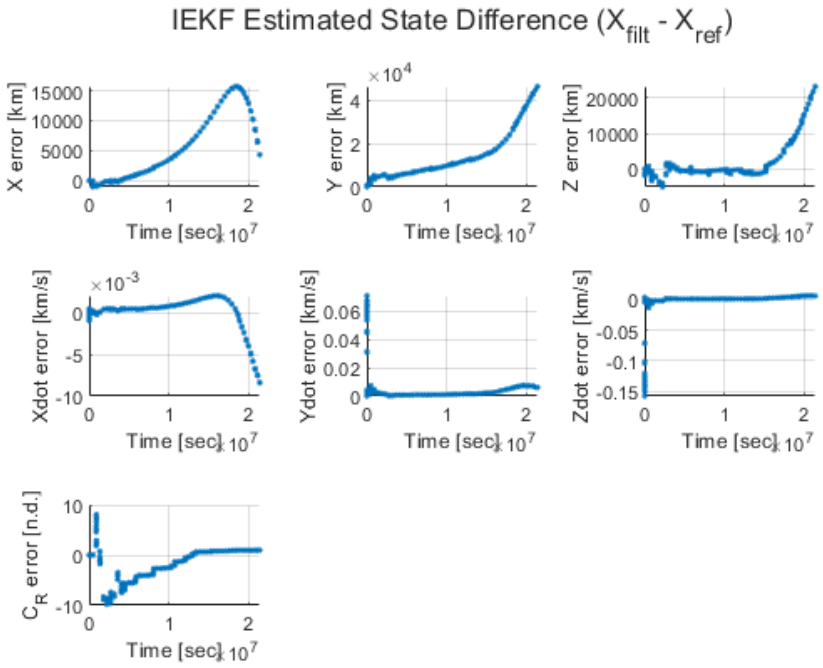

Fig. 27 Part 3 IEKF State Difference

We can clearly see that the IEKF has been busy throughout this scenario! In particular, it adjusts the reference trajectory nearly continuously from timestep to timestep. This implies a few points:

1) There are definitely mismodeled dynamics, as mismodeled dynamics would continuously skew the reference trajectory. This is because the reference trajectory assumes a specific form of the dynamics at each time step. If there are additional dynamics at play than the reference trajectory suggests, the true state and "reference trajectory" will inexorably grow apart with time. This point will be expanded upon below.
2) The scale of adjustment that the IEKF performs strongly suggests that conventional linear filters will struggle with this problem. Of particular note are the position estimates, which differ from the original reference trajectory by magnitudes on the order of 1e4 to 4e4 km. This is way too far outside of the linearized regime for our standard linear filter assumptions to hold, and as we saw in my initial attempts of part 3, this breakdown of assumptions makes those filters break down,

3) The two maneuvers are made even more apparent here, as there is a distinct change in shape at both 5 days (432000 sec) and 217 days (1.87488e7 sec). In particular, the 5 day maneuver can be seen in the spike in velocity and C_R estimates near the start of the scenario, and the 217 day maneuver can be seen in the abrupt change in slope of the position and X velocity estimates.

In regards to point 1 above, We can look at the estimated value of C_R to make some inferences. Nominally, C_R is a positive value, and acts like a "coefficient of drag" term to account for solar wind. Thus, when C_R is positive, it implies a force directed away from the sun. However, when C_R is negative, the acceleration from SRP points towards the sun. This could imply that there is an error in modeling the Sun's gravity, such as a mismodeled μ parameter, that could underestimate the Sun's gravity. Thus, this makes the IEKF change C_R to compensate with an additional force towards the sun. This is further supported by the fact that this effect tapers off as the spacecraft gets closer to Earth, where Earth's gravity becomes increasingly more significant. Another explanation for the changing C_R is that attitude dynamics could be involved, and hence a more accurate SRP model than the cannonball model could be at play. With real spacecraft, the orientation of the spacecraft relative to the sun can change the value of C_R, either increasing or decreasing it. We can see this at play a bit before the maneuver on day 5, where C_R is estimated to be larger than where the a-priori guess suggests.

Another fun thing to look at in regards to the IEKF estimated state is the number of times it iterates the reference trajectory. Figure 28 shows how many times the IEKF iterated the reference trajectory over the entire 250 day scenario, where 0 iterations has been buffered to 0.1 in order to show up on a log plot.

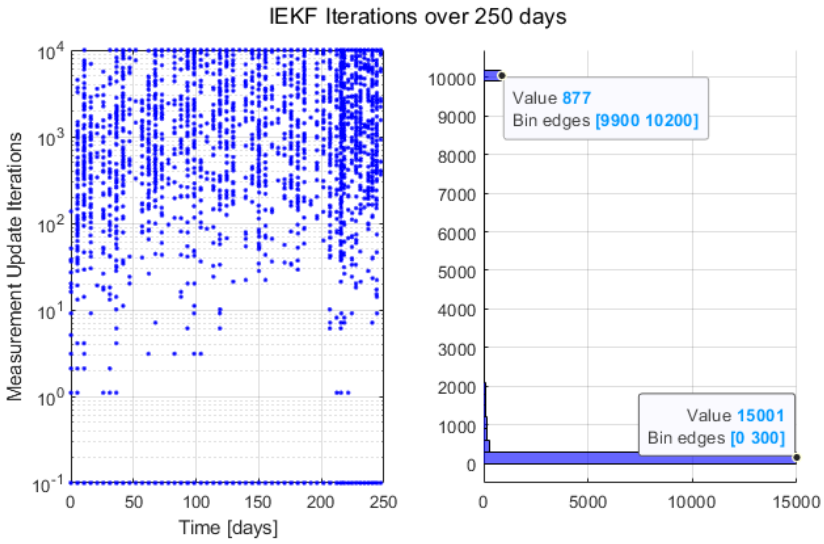

Fig. 28 Part 3 IEKF Reference Trajectory Iterations

We can clearly see that the IEKF prefers to iterate as little as possible, with most of the measurements iterating 300 or less times. However, when the IEKF iterates more than 300 times, it tends to iterate a *large* number of times, with the second most common number of iterations being between 9900 and 9999, the maximum number allowed. These larger iterations explain why my code takes 7 minutes to run, despite most of the measurements taking less than 300 iterations to converge.

3. DMC Analysis

While trajectory iteration is the IEKF's main tool, I also included DMC as a way to ensure that the filter doesn't get too smug. When I was tuning DMC for the IEKF, I discovered that it was incredibly sensitive to the parameters I chose - to the point where a change of 0.05 hours in τ increased my final RMS by 10's. Thus, I believe that DMC was necessary in order for this filter to work as well as it does. Figure 29 shows the estimated unmodeled accelerations from DMC, i.e. w_x, w_y, and w_z.

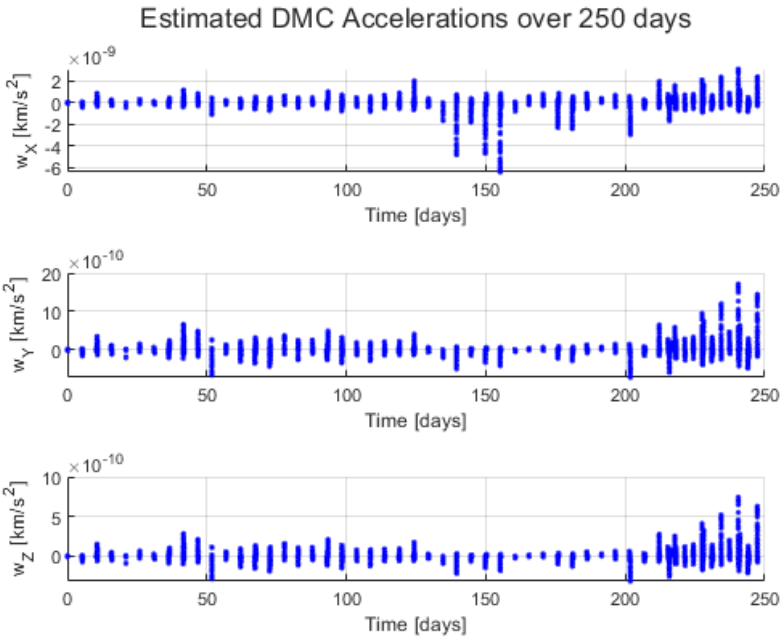

Fig. 29 Part 3 DMC Unmodeled Accelerations

We can see that DMC does quite a bit of heavy lifting throughout this scenario. In particular, we can see spikes at the start of each measurement window, similar to what we saw with the range prefit residuals. This further supports the hypothesis that there are errors in the station dynamics, as DMC induces a pretty clear acceleration to account for whatever is going on. Zooming into one of the measurement arcs in Figure 30, this becomes even more apparent.

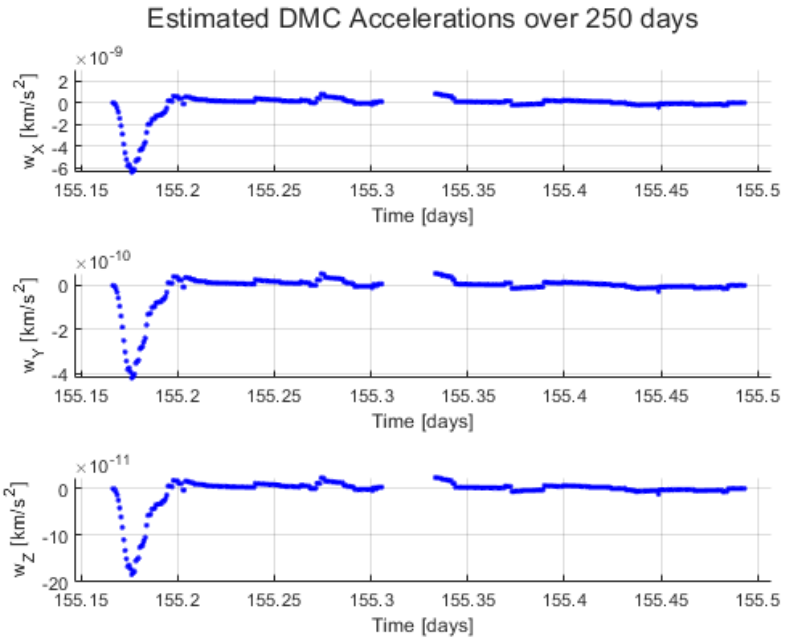

Fig. 30 Part 3 DMC Unmodeled Accelerations - zoomed into start of a representative measurement arc

Similarly, we can be reasonably confident that the first measurement arc aligns pretty well with the reference trajectory, as DMC does virtually nothing over the entire arc in Figure 31, with magnitudes around 10x smaller than the representative arc in Figure 30.

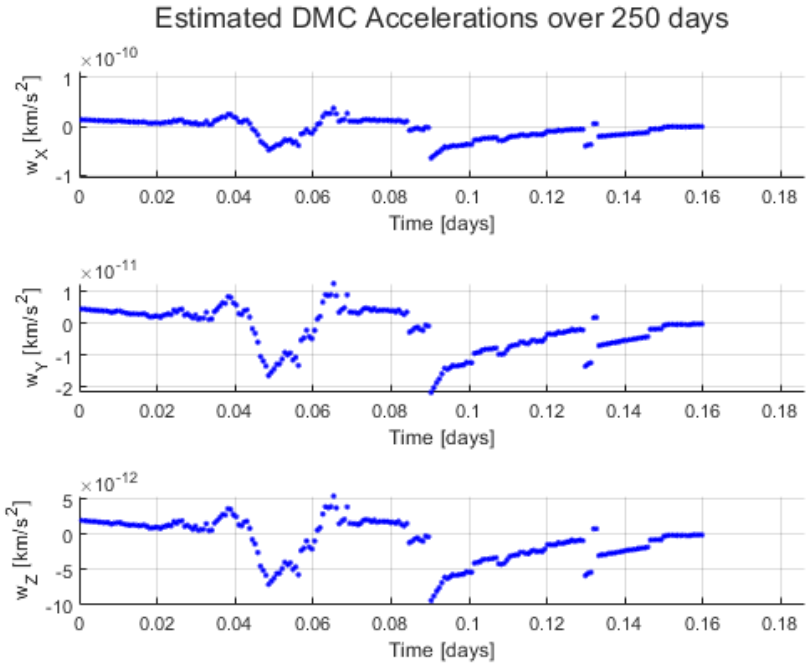

Fig. 31 Part 3 DMC Unmodeled Accelerations - zoomed into first measurement arc

D. Final B-plane Estimate and B-plane Evolution

Given everything I've said about the IEKF, I feel very confident that my filter provides a nearly complete picture of the scenario proposed in this project. Once I was convinced that my filter was working and I fully understood what it was telling me, I moved on to estimating the B-plane target after 50 day segments of data. Table 1 summarizes my findings, and Figure 32 shows how the B-plane vector evolved as the scenario went on.

Table 1 Comparison of B-plane targets after 50 day segments of data

Data cutoff (DCO)	$\vec{B} \cdot \hat{R}$ [km]	$\vec{B} \cdot \hat{T}$ [km]	σ_R [km]	σ_T [km]	σ_{RT} [km$_{\hat{R}}$km$_{\hat{T}}$]
50 days	8019.8	-10935	2.965	8.366	15.15
100 days	6930.1	-12904	1.773	5.270	9.247
150 days	10729	-5108.9	1.513	4.272	5.874
200 days	14480	9879.7	0.971	0.953	-0.1453
250 days	8076.913	8961.030	0.686	0.578	-0.06694

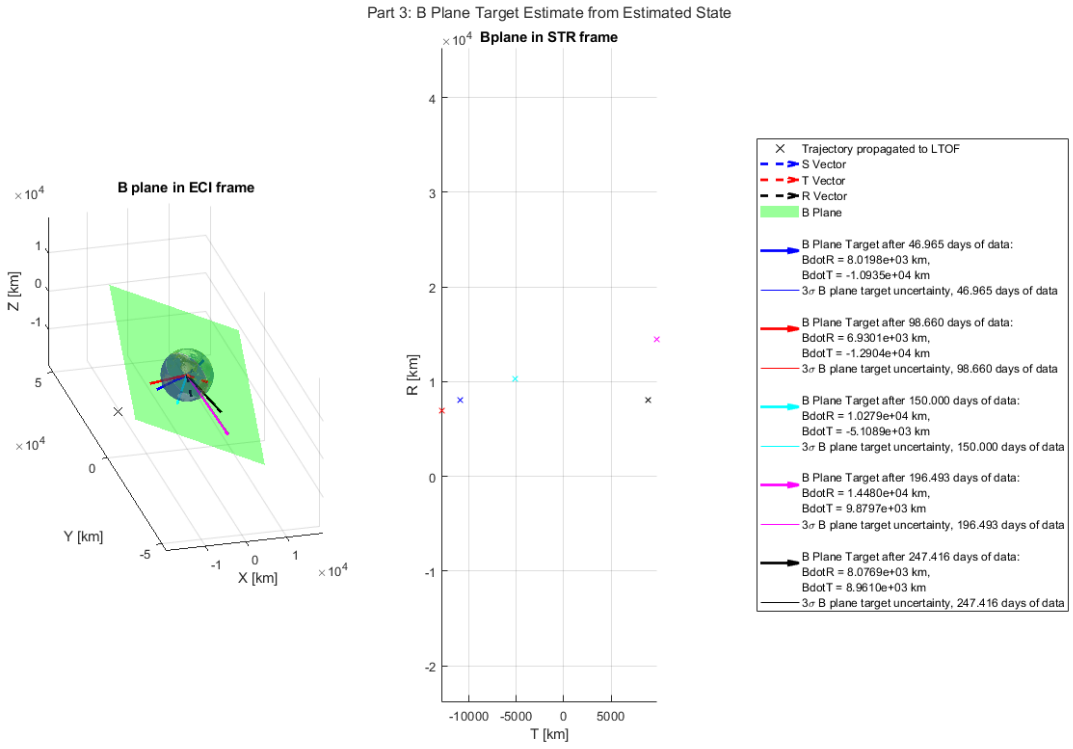

Fig. 32 Part 3 B-plane Target Evolution

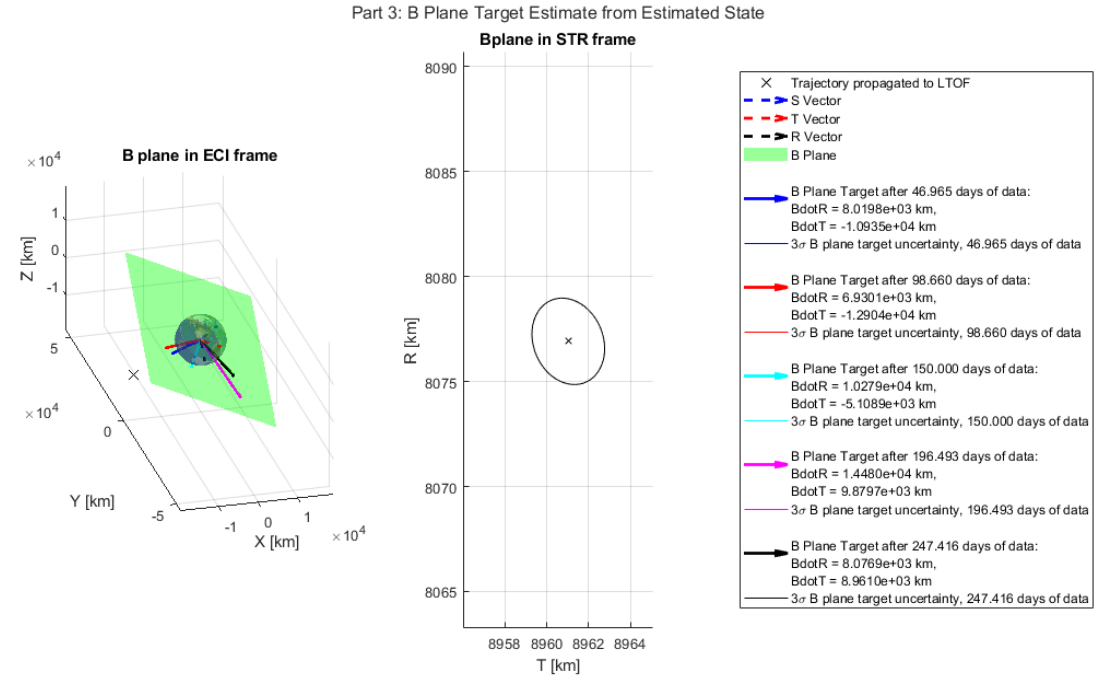

Fig. 33 Part 3 B-plane Final Target

We can clearly see the B-plane vector moving as the scenario goes on due to mismodeled dynamics, in particular maneuvers. However, the vector doesn't move at the times I'd expect if the maneuvers occur when I think they do. If there truly was a maneuver at 217 days, I wouldn't expect to see a change in the B vector until after that time, i.e. when processing from 200 to 250 days. As it stands, we see the B vector move between 100 and 200 days, as evidenced by the cyan B vector estimate in Figures 32 and 33 that is nearly exactly in the middle of the two extremes. The abrupt changes we see could be from the possibility that the dynamics are badly mismodeled, or it could be an artifact from the sign of C_R. As I mentioned before, the sign of C_R changes substantially throughout this scenario, which could be interpreted as an additive acceleration that changes where the B vector is pointing. In particular, it could show up as a constantly shifting low thrust input, which wouldn't do much from timestep to timestep like an impulsive maneuver, but would have a massive impact over the timespan of 250 days.

One other point of concern I have with my results is how small the final covariances are. By nature, the IEKF will have small covariances due to the number of times it iterates the measurement update, which shrinks the covariance each time it is iterated. Thus, while Part 2 started with a 3σ uncertainty of ~ 60 km in $\hat{R}$ and ~ 40 m in $\hat{T}$ after 50 days, the IEKF starts with ~ 9 km and ~ 25 km respectively. This probably isn't a big deal, as iterating on measurements will necessarily provide more information to the filter, but it's too large of a difference from Part 2 for me to feel 100% confident about the uncertainty figures. However, based on the residuals and state differences discussed earlier, I feel confident that the filter is working and can live with the seemingly low covariance estimates.

E. Initial State Estimate at the Epoch and Comparison with Batch Filter

Since the IEKF is a sequential filter, I didn't get an explicit estimate for the initial state at the epoch. However, we know that the first measurement arc of data is very well behaved, and a batch filter gives exactly the state deviation at the initial epoch. So, to get the initial epoch state estimate, I fed the first measurement arc of data to the batch filter and got the following initial state:

$$\vec{X}_0 = \begin{bmatrix} -274096773.52078 \\ -92859278.3240894 \\ -40199493.9684749 \\ 32.6708738436275 \\ -8.94670021583699 \\ -3.858377451509278 \\ 1.000009040959014 \end{bmatrix}$$

After getting this updated initial state, I decided to propagate it with our original dynamics to see what the state error looked like for the entire scenario. Interestingly, it looked very similar to my IEKF's estimated state difference, with the exception of the Y, Z, and C_R estimates. Figure 34 shows the state error for the Batch Filter based on the new initial state, and Figure 35 shows the pre- and postfit residuals for the Batch over the first measurement arc.

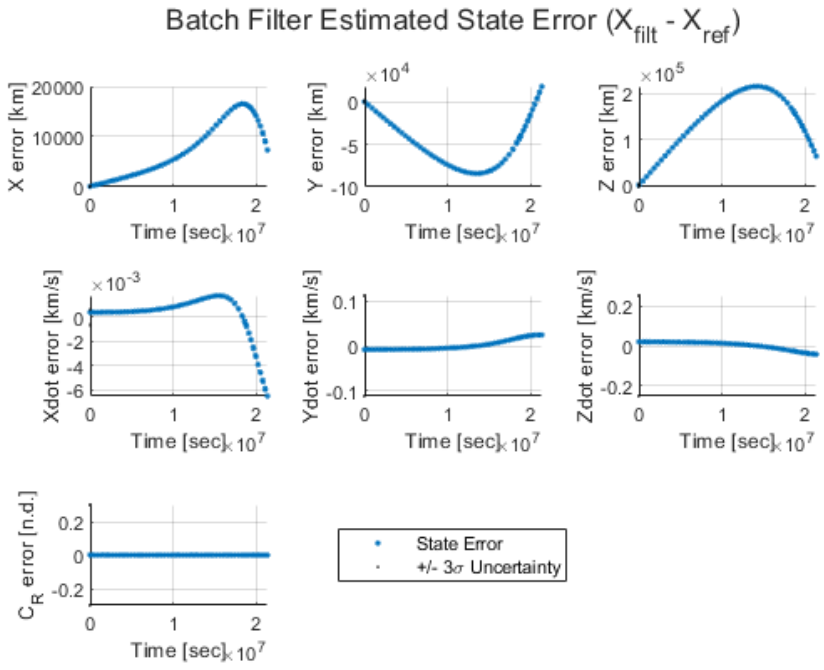

Fig. 34 Part 3 Batch Filter State Error, based on updated initial epoch state

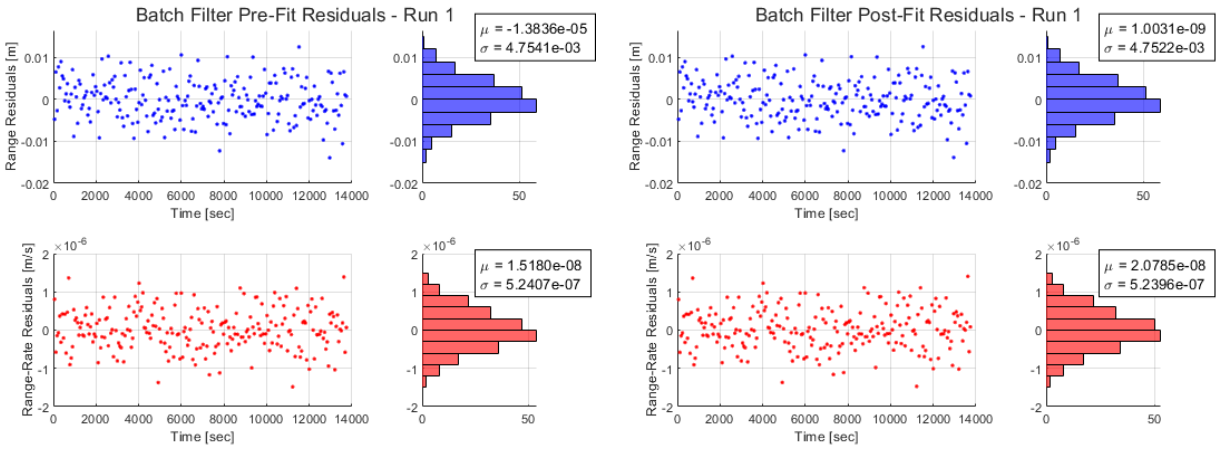

Fig. 35 Part 3 Batch Filter Residuals over first measurement arc

Clearly, the Batch does great over the first measurement arc, with residuals that all approach the correct noise level. Interestingly, just propagating the Batch's initial state solution provides the same shape of state error that I attributed to maneuvers, in particular the sudden slope change of the position and X velocity estimates. However, the shape of the Y, Z, and $\dot{Z}$ estimates is still different between the two, so the maneuver hypothesis is still valid. Part of me wishes that I had tried a batch filter right away to guide my efforts, but at this point it's a bit too late to go back. Plus, it is very encouraging to see that the IEKF recovers some of the same state deviation estimates as the propagated batch solution, which makes me think that the IEKF has done a *very* good job overall! It turns out that iterating the measurement update until the filter gives you what you want *is* a valid way to approach the problem, which is cool to see in action.

V. Conclusion

Over the course of this project, I truly felt like a GNC engineer. Puzzling through each data set to figure out what was going on was a fun challenge, and when I finally got things working in the end, there was a satisfaction attached to it that made me feel great! I was able to take an estimation problem from nearly first principles and come out the other side with a solution that's not only viable, but also pretty good based on what I've reasoned through. I can see myself applying what I've done here to other problems in my career and in other classes, which is simply priceless. Overall, the hands-on nature of this course has been incredibly refreshing, as while the assignments were no doubt tough, it was fun to actually build a working software solution rather than just turn in written assignments and exams. 10/10 would recommend!

This project was incredibly fun and engaging, and I feel as though I've thoroughly exercised every tool in my toolkit that I built up this semester. If I had one suggestion to improve the project, it would be to have some more clear instructions/guidance on the B-plane modeling. There was conflicting guidance on how to actually calculate the B-plane between what was in the provided PDF handout vs. what was shared in class and on Canvas, which burned some time trying to figure out the discrepancies. Overall though, this was a great project and I'm kinda sad that it's over! Have a great summer!

VI. Acknowledgments

I'd like to acknowledge the following people for all of their fantastic help throughout this project:

1) Steven Liu and Anirudh Etagi for bouncing ideas back and forth and mutual debugging woes and successes!
2) Jacob Mesley, Emmett Peters, Justin Lynch, and Alex Moody for comparing answers and discussing talking points that ended up in this writeup!
3) Michael Sola for helping me get Part 2 working and providing feedback on my Part 3 process!
4) Sean Vestecka for comparing Part 1 dynamics and Part 2 B-plane answers!
5) Jay McMahon for being an engaging and fun professor who kept this project interesting!

References

[1] Kazemi, S., Azad, N. L., Scott, K. A., Oqab, H. B., and Dietrich, G. B., "Orbit determination for space situational awareness: A survey," *Acta Astronautica*, Vol. 222, 2024, pp. 272–295. https://doi.org/https://doi.org/10.1016/j.actaastro.2024.06.015, URL https://www.sciencedirect.com/science/article/pii/S0094576524003308.

[2] Byron D. Tapley, G. H. B., Bob E. Schutz, *Statistical Orbit Determination*, 1st ed., Elsevier Academic Press, 2004.

A. Appendix: Code Instructions

My code is very straightforward to run, just read the menus and it'll walk you through! However, I've put together an example code flow below just to be clear. I've also included estimated run times for each part in the menu, timed on my machine, to minimize wasting your time - although these estimates could vary based on your computer's CPU and memory (mostly CPU). For context, my computer runs a 13th Gen Intel(R) Core(TM) i7-13700HX, 2.10 GHz CPU (16 cores, 24 logical processors) and has 16 GB of RAM.

Goal: Run Part 1 to see my modeling verification results.

Steps:

1) Open Project2Main.m
2) Run Project2Main.m
3) Type the number 1 into the command window and press 'Enter'.
4) Inspect plots as desired.
5) You can exit the program flow at any time by clicking the command window and pressing 'Ctrl-C'.

Have fun!