University of Colorado - Boulder

ASEN 6080
Statistical Orbit Determination | Spring 2025

ASEN 6080 Statistical Orbit Determination: Project 1

Author:
Ian Faber[a]

[a]SID: 108577813

Instructor:
Jay McMahon

Tuesday, February 18, 2025

College of Engineering & Applied Science
UNIVERSITY OF COLORADO BOULDER

Writeup of Project 1 for ASEN 6080: Statistical Orbit Determination.

I. Problem Overview and Filter Methodology

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.

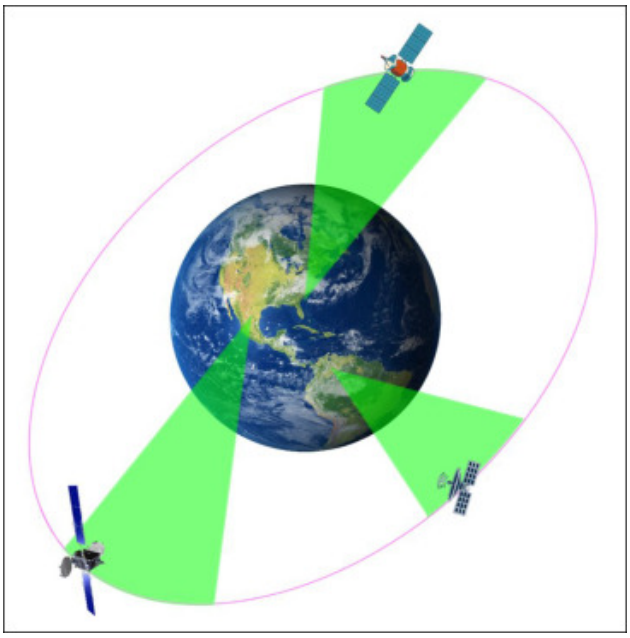

Fig. 1 Cartoon of a typical statistical orbit determination problem [1]

In this project, we were tasked with determining the orbit of a satellite using a log of observations at specific times from 3 different ground stations: Station 101 (a ship in the Pacific Ocean), Station 337 (Pirinclik, Turkey), and Station 394 (Thule, Greenland). Each station provides a range (ρ) and range rate ($\dot{\rho}$) measurement of the satellite, which are modeled as follows:

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

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

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. Figure 2 shows the measurements in the data log for this problem plotted vs. time.

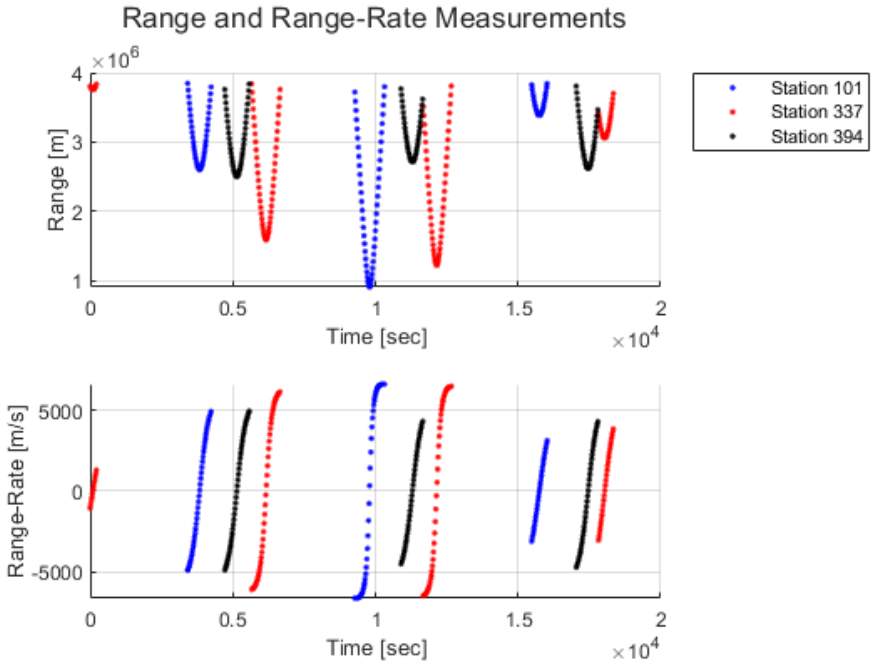

Fig. 2 Provided measurements for Project 1

We were told that the satellite orbit was modeled after the Quikscat satellite, which has a roughly circular 98.6° inclination orbit at an altitude of 800 km. Further, we were given exact coordinates of station 101 and told to estimate the others. Finally, we were told to model the effects of point mass gravity, J2 perturbations, and drag on the satellite trajectory, and to estimate the respective parameters. Thus, our final state to be estimated for this problem was

$$\vec{X} = \left[X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}, \mu, J_2, C_D, \vec{R}_{s,101}, \vec{R}_{s,337}, \vec{R}_{s,394}\right]^\intercal \tag{3}$$

Where $\vec{R}_{s,i} = \left[X_{s,i}, Y_{s,i}, Z_{s,i}\right]^\intercal$. Thus, our total state is an 18x1 column vector.

To estimate this state, we investigated two types of filters: a Batch Processor (referred to as a Batch or Batch Filter from here onward) and a Linearized Kalman Filter (referred to as an LKF from here onward). In particular, we investigated the state estimates from each filter, as well as each filter's advantages, shortcomings, and applications. We also investigated the relative strength of range data vs. range rate data, as well as the effect of fixing different stations on the final estimate from both filters. Finally, we also investigated the Potter Algorithm, which is a square root alternative to the LKF that aims to combat some of the LKF's numerical problems.

A. Batch Processor

The first type of filter we investigated was a Batch Processor. This filter generates an estimate of the initial state deviation, $\hat{x}_0$, from some reference trajectory X^* at some initial time t_0. The Batch Filter effectively implements a linear least squares procedure by solving a set of normal equations, $\Lambda\hat{x}_0 = N$. This involves accumulating measurement covariance information into an information matrix, Λ, and measurement residuals into a normal vector, N. To do this, the Batch Processor takes in an entire set of ground station observations at once and loops through each measurement in the set to extract its covariance and measurement residuals, then incorporates them into the information matrix and normal vector respectively. This process is then repeated until the postfit residuals, i.e. the measurement residuals after processing, approach the expected measurement noise characteristics. The full spacecraft state is then extracted by adding the estimated initial state deviation to the provided initial reference state, then integrating the reconstructed initial state forward in time until the last measurement timestep. A flowchart for the Batch Processing Algorithm is provided in Figure 3.

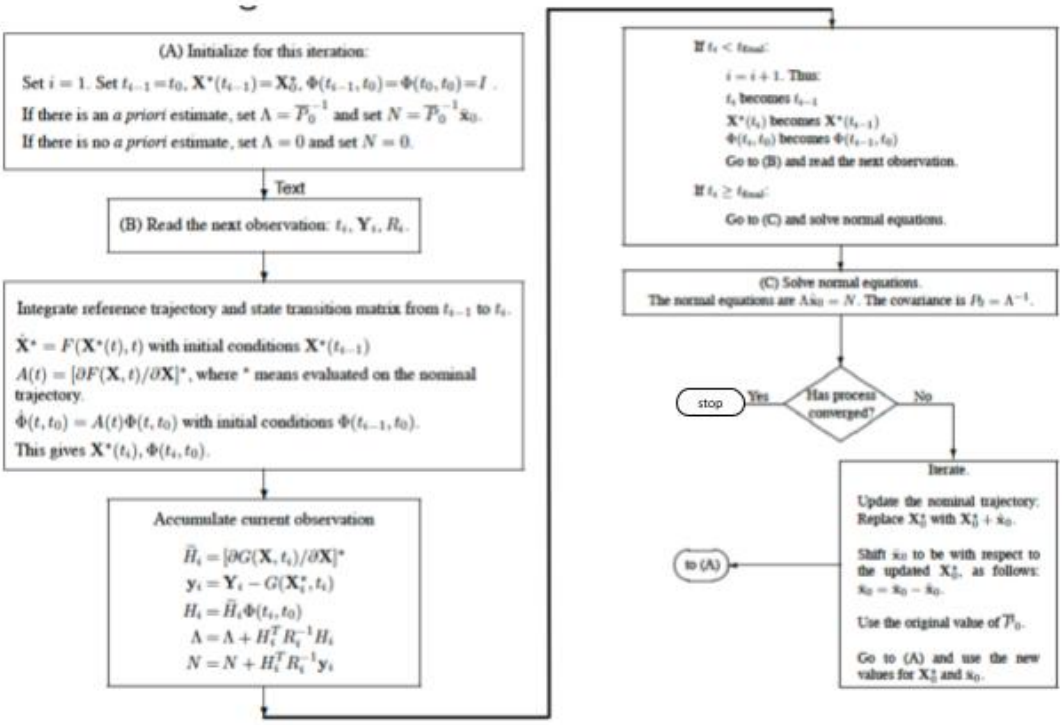

Fig. 3 Batch Processor Flowchart [2]

B. Linearized Kalman Filter

The second type of filter we investigated was a Linearized Kalman Filter. Unlike the Batch Processor, the LKF generates a state deviation estimate $\hat{x}(t)$ from the reference trajectory at each measurement timestep, not just the initial deviation $\hat{x}_0$. While the Batch processes all measurements at once, the LKF processes one measurement at a time, gradually updating its state estimate by applying a Kalman Gain to the difference in the actual vs. predicted measurement residuals. If the filter receives a particularly uncertain measurement, the Kalman Gain is smaller and trusts the dynamics model more. If the measurement is particularly good, then the Kalman gain increases to allow that measurement to correct the state estimate. The full spacecraft state is then extracted by adding the estimated deviation to the reference trajectory at every timestep. Like the Batch, the LKF can be iterated until the pre- and postfit residuals match the expected noise characteristics. Regardless of the method, the LKF solves fundamentally the same least squares problem as the Batch, and the final state estimates of both filters end up matching very closely. A flowchart of the LKF is provided in Figure 4.

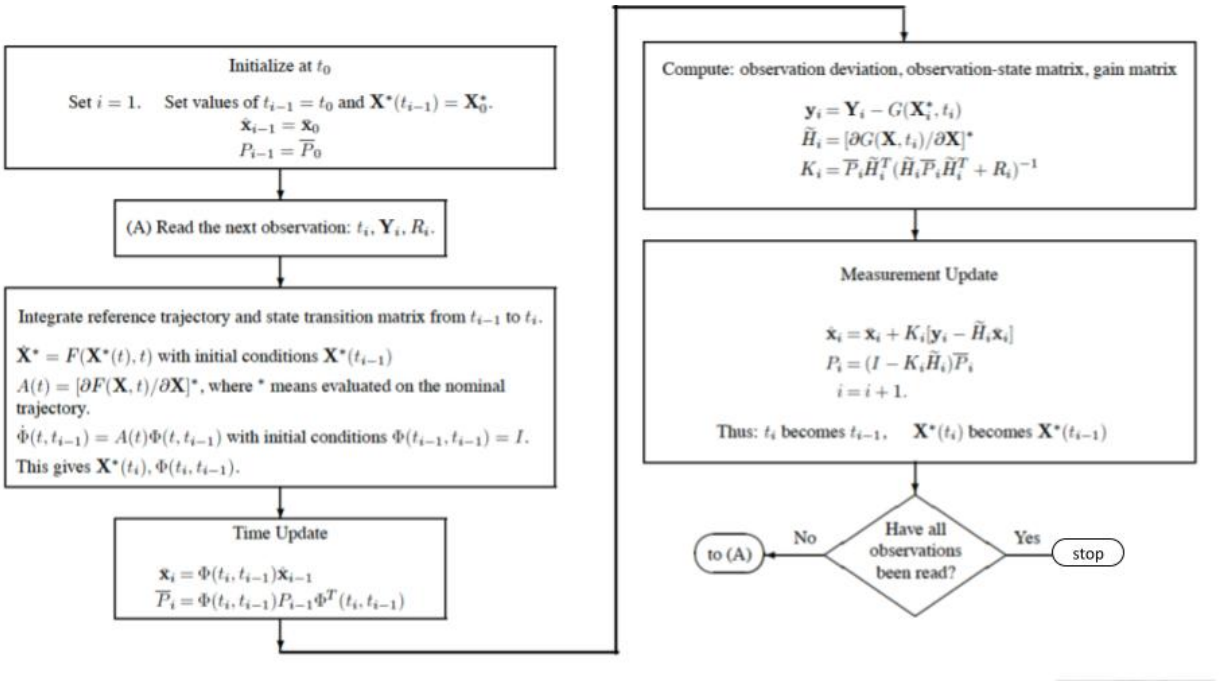

Fig. 4 LKF Flowchart [2]

II. Base Problem Results and Discussion

The first task of the project was to implement both a Batch and LKF for the provided data, then run the filters to get estimates for the satellite orbit trajectory. Both filters were initialized with the following initial state, state deviation, and covariance, with units of m for position, m/s for velocity, and m^3/s^2 for the gravitational parameter μ:

$$X_0 = \begin{bmatrix} 757700.0 \\ 5222607.0 \\ 4851500.0 \\ 2213.21 \\ 4678.34 \\ -5371.30 \\ 3.986004415 \times 10^{14} \\ 1.082626925638815 \times 10^{-3} \\ 2 \\ -5127510.0 \\ -3794160.0 \\ 0.0 \\ 3860910.0 \\ 3238490.0 \\ 3898094.0 \\ 549505.0 \\ -1380872.0 \\ 6182197.0 \end{bmatrix}, \bar{x}_0 = \text{zeros}(18,1) \tag{4}$$

$$\bar{P}_0 = \text{diag}([1\times10^6, 1\times10^6, 1\times10^6, 1\times10^6, 1\times10^6, 1\times10^6, 1\times10^{20}, 1\times10^6, 1\times10^6,$$
$$1\times10^{-10}, 1\times10^{-10}, 1\times10^{-10}, 1\times10^6, 1\times10^6, 1\times10^6, 1\times10^6, 1\times10^6, 1\times10^6])$$

A. Batch Filter Results

The Batch Filter performed rather well for the base problem. The Batch converged after 4 runs, resulting in postfit residuals that approached the expected noise of the measurements as seen in Figure 5 and summarized in Table 1.

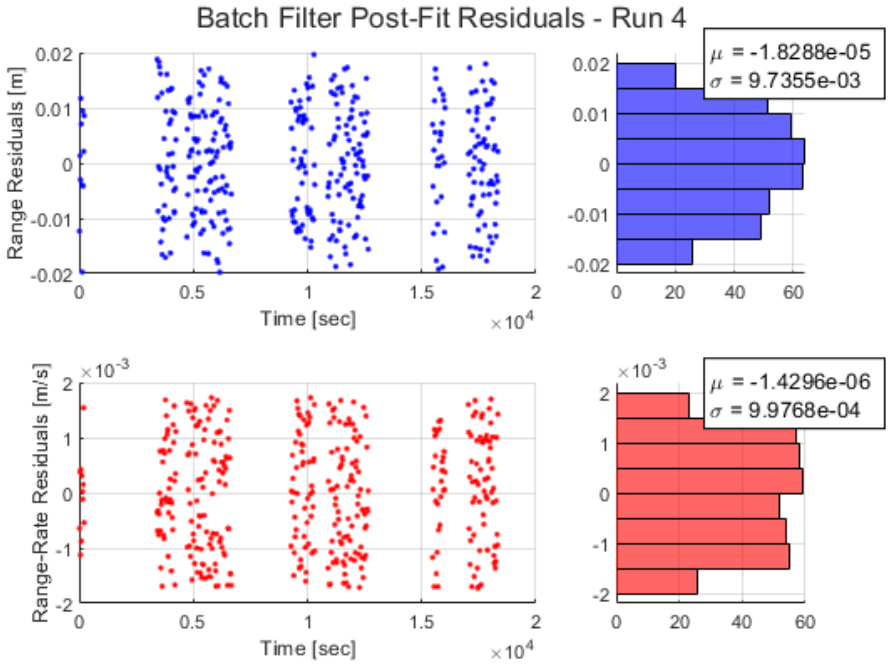

Fig. 5 Batch Filter Postfit Residuals - Base Problem

Table 1 Batch Filter Postfit Residuals Summary - Base Problem

Estimated σ_ρ [m]	Expected σ_ρ [m]	Estimated $\sigma_{\dot{\rho}}$ [m/s]	Expected $\sigma_{\dot{\rho}}$ [m/s]	Normalized RMS
9.7355×10^{-3}	1×10^{-2}	9.9768×10^{-4}	1×10^{-3}	0.9844

Only postfit residuals are presented here, as prefit residuals don't have much meaning for the Batch Filter - in fact, they end up being the same as the postfits. This is because the Batch Filter processes all measurements at once, meaning there is only one "fit" that is performed. On the other hand, the LKF is an on-line algorithm, meaning that it fits a set of prefit residuals at each timestep - effectively carrying out multiple "fits" in one run. Further, the Batch Filter calculates postfit residuals based only on the initial state deviation estimate, whereas the LKF calculates postfit residuals based on the current state deviation estimate.

As for state estimates, the Batch Filter computed an estimated initial state deviation and final state of

$$\hat{x}_0 = \begin{bmatrix} 0.302919450332411 \\ -0.424698594957590 \\ -0.265111992135644 \\ 0.0406065310335180 \\ 0.0327130539899372 \\ -0.0144196371238650 \\ -42756114.4375000 \\ -0.00216462645448106 \\ 0.236361244822065 \\ -9.31322574615479 \times 10^{-10} \\ -9.31322574615479 \times 10^{-10} \\ 5.67580883361778 \times 10^{-9} \\ -9.99974721996114 \\ 10.0007732464001 \\ 5.96673000976443 \\ -4.99856995430309 \\ 2.01956869964488 \\ 2.96526484657079 \end{bmatrix}, X_f = \begin{bmatrix} 1128588.65459834 \\ 5990056.57716476 \\ 3775422.64419787 \\ 2009.20870012732 \\ 3562.98238334955 \\ -6237.58258490610 \\ 398600398743886 \\ -0.00108199952884224 \\ 2.23636124482207 \\ 2505228.42142475 \\ -5866075.30314653 \\ 5.67580883361778 \times 10^{-9} \\ -2257615.10440640 \\ 4505286.35131179 \\ 3898099.96673001 \\ 1470525.39920200 \\ 215191.088871645 \\ 6182199.96526485 \end{bmatrix} \tag{5}$$

Using these estimates resulted in the following relative state in Figure 6:

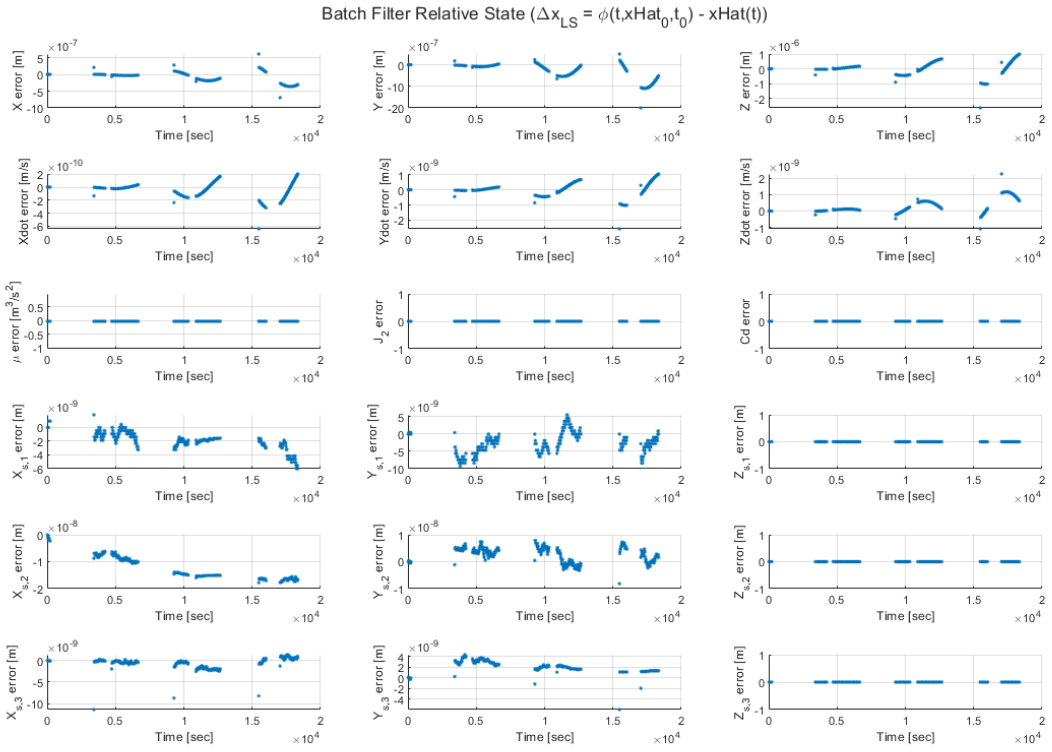

Fig. 6 Batch Filter Relative State, Base Problem

As for position and velocity covariance, the filter exhibited a cyclical, low covariance matrix trace as shown in Figure 7, with the final position and velocity covariance portrayed in the ellipsoids of Figure 8 and final position and velocity uncertainties shown in Table 2.

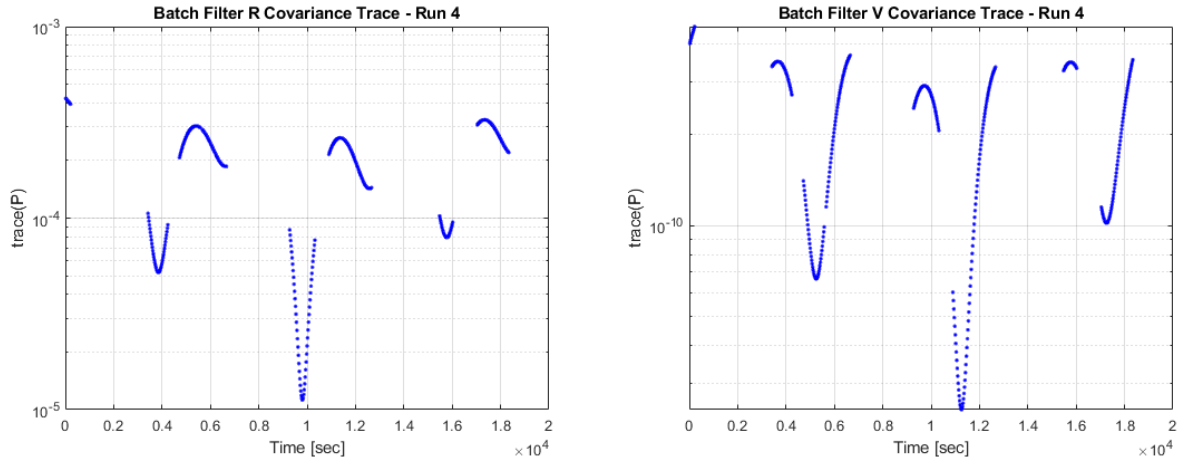

Fig. 7 Batch Filter Covariance Matrix Position and Velocity Trace vs. time

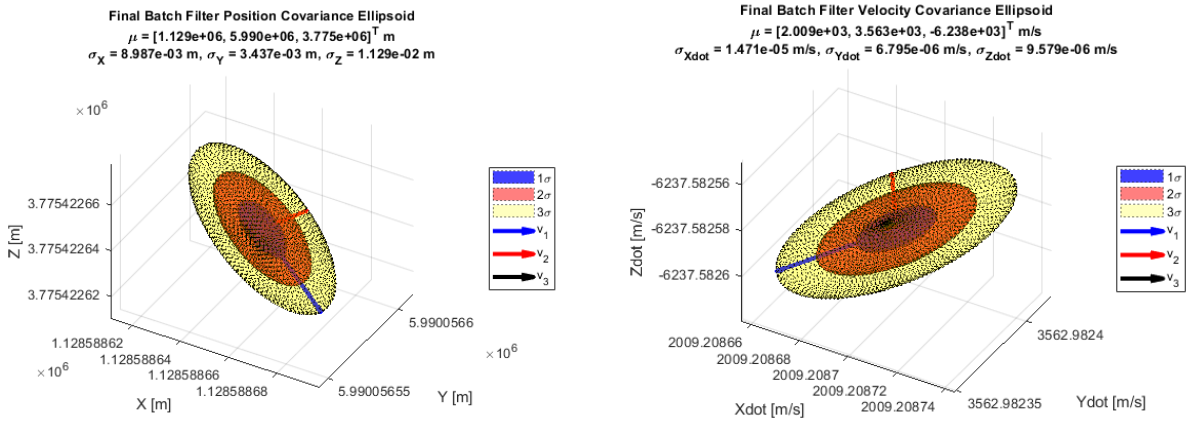

Fig. 8 Batch Filter Position and Velocity Covariance Ellipsoids

Table 2 Batch Filter Estimated Final Position and Velocity Uncertainty - Base Problem

σ_X [m]	σ_Y [m]	σ_Z [m]	$\sigma_{\dot{X}}$ [m/s]	$\sigma_{\dot{Y}}$ [m/s]	$\sigma_{\dot{Z}}$ [m/s]
8.987×10^{-3}	3.437×10^{-3}	1.129×10^{-2}	1.471×10^{-5}	6.795×10^{-6}	9.579×10^{-6}

The time varying trace of the covariance matrix in Figure 7 makes sense for the Batch, as propagating the initial estimated covariance involves pre- and post multiplying by the STM at each timestep like so: $P_i = \Phi(t_i, t_0) P_0 \Phi(t_i, t_0)^\intercal$. Since we only get one covariance estimate from the Batch, the best we can do is propagate it forward, unlike the LKF which provides a covariance estimate at each timestep. Thus, given the nature of the orbit determination problem, the uncertainty of the state will vary along the orbit, eventually repeating. This repetition isn't perfect due to the existence of drag, which is a nonconservative force. We can see this repetition by plotting the propagated uncertainty as error bars as black dashed lines around the relative state in Figure 9.

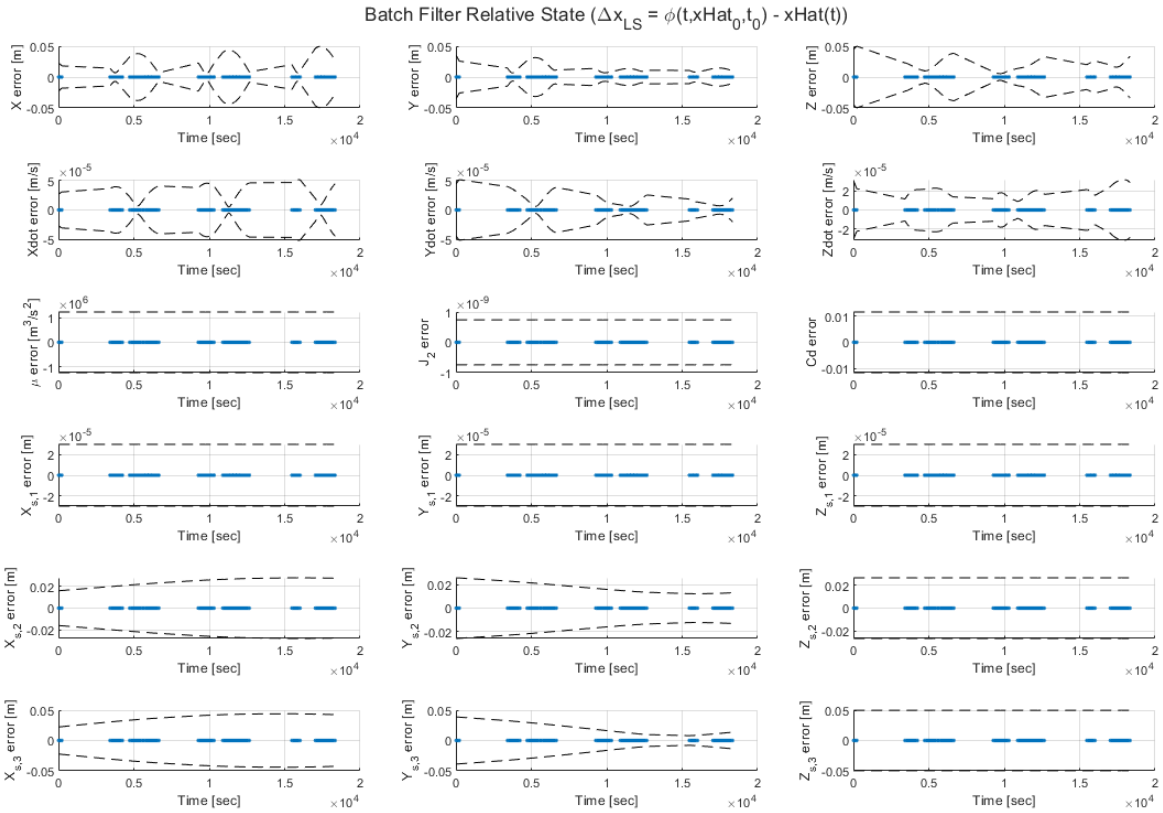

Fig. 9 Batch Filter Relative State with Uncertainty, Base Problem

B. LKF Results

Unlike the Batch Filter, the LKF didn't quite perform as well for the base problem. While the RMS of the postfit residuals did decrease across multiple iterations, the LKF didn't converge within a reasonable amount of runs. Thus, the LKF was capped at 5 runs, which resulted in the lowest postfit RMS. The residuals for the LKF got close to the expected noise characteristics, but still exhibited some patterns that suggest the LKF couldn't pull out all of the information from the measurements. The pre- and postfit residuals are shown in Figure 10 and summarized in Table 3.

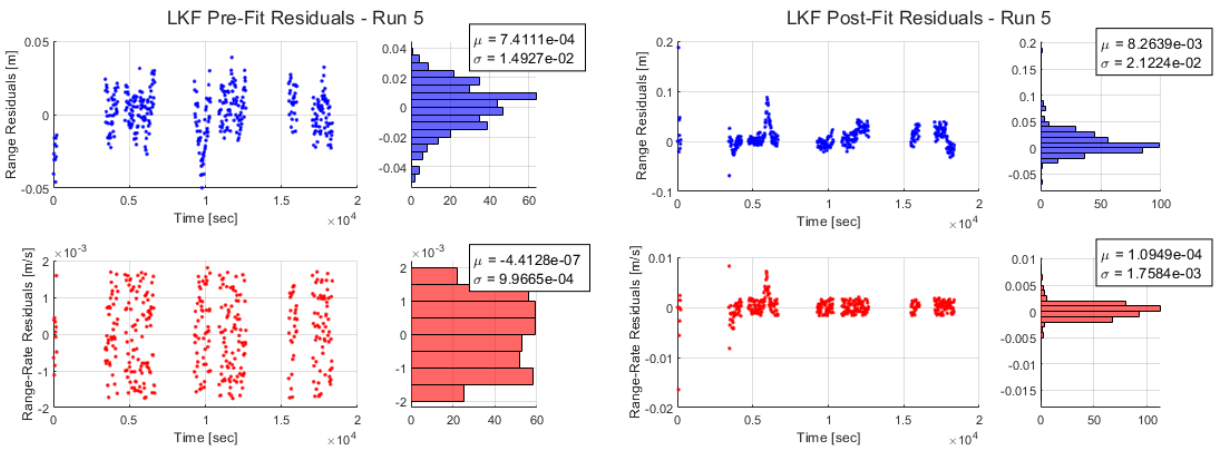

Fig. 10 LKF Pre- and Postfit Residuals - Base Problem

Table 3 Batch Filter Postfit Residuals Summary - Base Problem

Prefit σ_ρ [m]	Prefit $\sigma_{\dot{\rho}}$ [m/s]	Prefit Normalized RMS	Postfit σ_ρ [m]	Postfit $\sigma_{\dot{\rho}}$ [m/s]	Postfit Normalized RMS
1.4927×10^{-2}	9.9665×10^{-4}	1.2686	2.1224×10^{-2}	1.7564×10^{-3}	2.0337

Notably, only the prefit range rate residuals approach noise for the LKF implementation. This is because the LKF exhibits some numerical issues related to machine precision and finite-word floating point math. Since our state varies wildly with magnitude, the chances of accumulating roundoff error are very high. This is because when we multiply a very large number by a very small number using finite precision math, the computer can't store all the necessary digits for an accurate result. This issue repeats over multiple fits, resulting in a result that isn't perfect. With that being said, the largest residuals for this problem are on the order of 0.2 m and 0.01 m/s, which is acceptable for this assignment. This would be an issue if extreme precision was required, i.e. modeling the height of waves to the centimeter level, but this assignment is about exploring the advantages and limitations of each filter, so we can let it slide. The underlying reason that the filter itself breaks can be investigated by looking at the plot of the covariance matrix trace vs. time for the LKF in Figure 11.

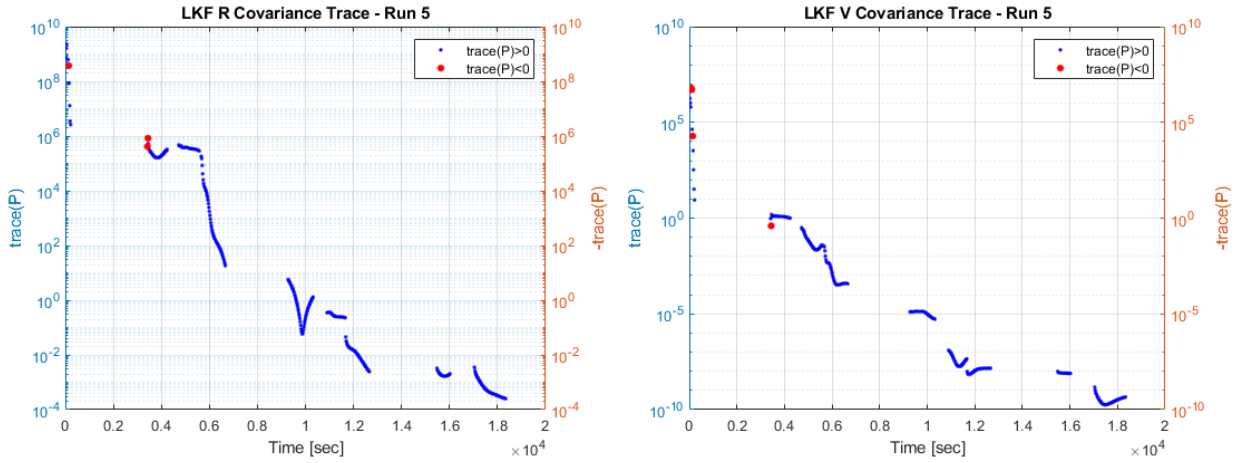

Fig. 11 LKF Covariance Matrix Position and Velocity Trace vs. time

One of the crucial assumptions of the LKF is that the covariance matrix, P, must be positive definite. However, looking at Figure 11, we can see that the trace of P goes negative at some timesteps - implying that the matrix itself is not positive definite. While the LKF implementation utilizes the Joseph formulation for P to guarantee that it is symmetric ($\hat{P}_i = (I - K_i\tilde{H}_i)\bar{P}_i(I - K_i\tilde{H}_i)^\intercal + K_i R_i K_i^\intercal$), it does not guarantee a positive definite result. Thus, there are situations where a particularly bad residual or large measurement time gap could cause the Joseph formulation to return a non-positive definite covariance estimate.

The final estimated position and velocity covariance is portrayed in the ellipsoids of Figure 12 and summarized in Table 4.

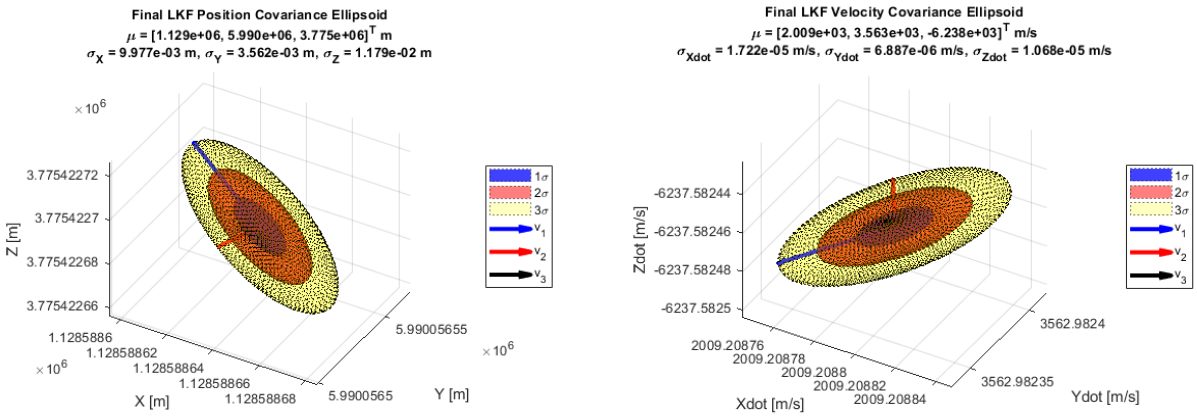

Fig. 12 LKF Position and Velocity Covariance Ellipsoids

Table 4 LKF Estimated Final Position and Velocity Uncertainty - Base Problem

σ_X [m]	σ_Y [m]	σ_Z [m]	$\sigma_{\dot{X}}$ [m/s]	$\sigma_{\dot{Y}}$ [m/s]	$\sigma_{\dot{Z}}$ [m/s]
9.977×10^{-3}	3.562×10^{-3}	1.179×10^{-2}	1.722×10^{-5}	6.887×10^{-6}	1.068×10^{-5}

Looking at estimated state, the LKF provides an initial state deviation and final state of

$$\hat{x}_0 = \begin{bmatrix} 0.0484077084748774 \\ 0.0104096010598050 \\ -0.0334812263758215 \\ 0.00297943852706638 \\ 0.00457917711366761 \\ -0.00392823627066054 \\ -2446181.32985027 \\ -3.53887176270214 \times 10^{-5} \\ 0.0335014486338858 \\ 0.0216862399591193 \\ 0.0160959808541736 \\ -0.00725113129278171 \\ 0.00426767635161078 \\ -0.00697099801045578 \\ 0.0303024955125654 \\ 0.0185777159475499 \\ -0.0150627733927852 \\ 0.0296021707788542 \end{bmatrix}, X_f = \begin{bmatrix} 1128588.64034270 \\ 5990056.53362793 \\ 3775422.69139971 \\ 2009.20879636467 \\ 3562.98237453386 \\ -6237.58246335096 \\ 398600390629916 \\ -0.00108199479366898 \\ 2.22156837649799 \\ 2505228.36827980 \\ -5866075.17870591 \\ -0.0361687625114817 \\ -2257615.13706314 \\ 4505286.32096004 \\ 3898099.95226025 \\ 1470525.33783310 \\ 215191.079252716 \\ 6182199.96962159 \end{bmatrix} \tag{6}$$

Using these estimates resulted in the following relative state in Figure 13.

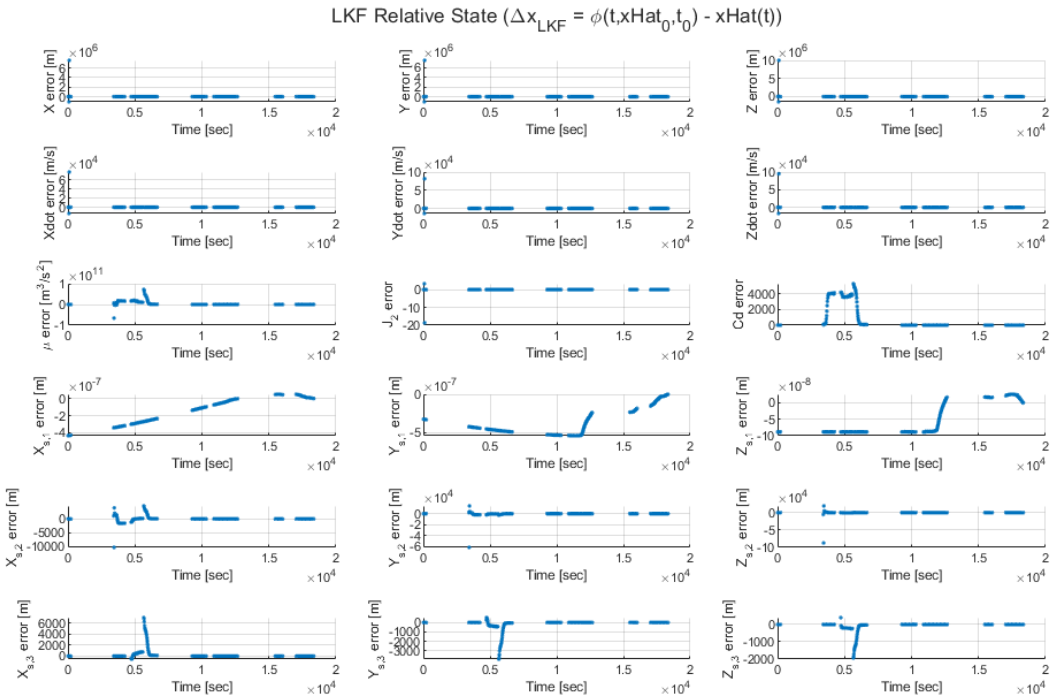

Fig. 13 LKF Relative State, Base Problem

Some of these errors are quite bad due to the numerical problems discussed previously. However, the final estimated state turns out to be acceptable, so we can mostly ignore the numerical issues. To avoid them entirely, we could investigate

square root methods of propagating covariance, such as the Potter Algorithm which will be discussed in Section V. Square root methods effectively reduce the severity of finite-word floating point math errors, resulting in more accurate estimates and nearly always positive definite covariance matrix estimates.

Superimposing uncertainty bars onto the relative state produces Figure 14. Due to the numerical issues discussed previously, many of the relative state estimates lie outside of the predicted uncertainty bounds. Further, the LKF has a smugness problem - at some point it is so confident in its state estimate that it ignores additional data and effectively saturates. This can be avoided by introducing process noise, which will be investigated in a future assignment.

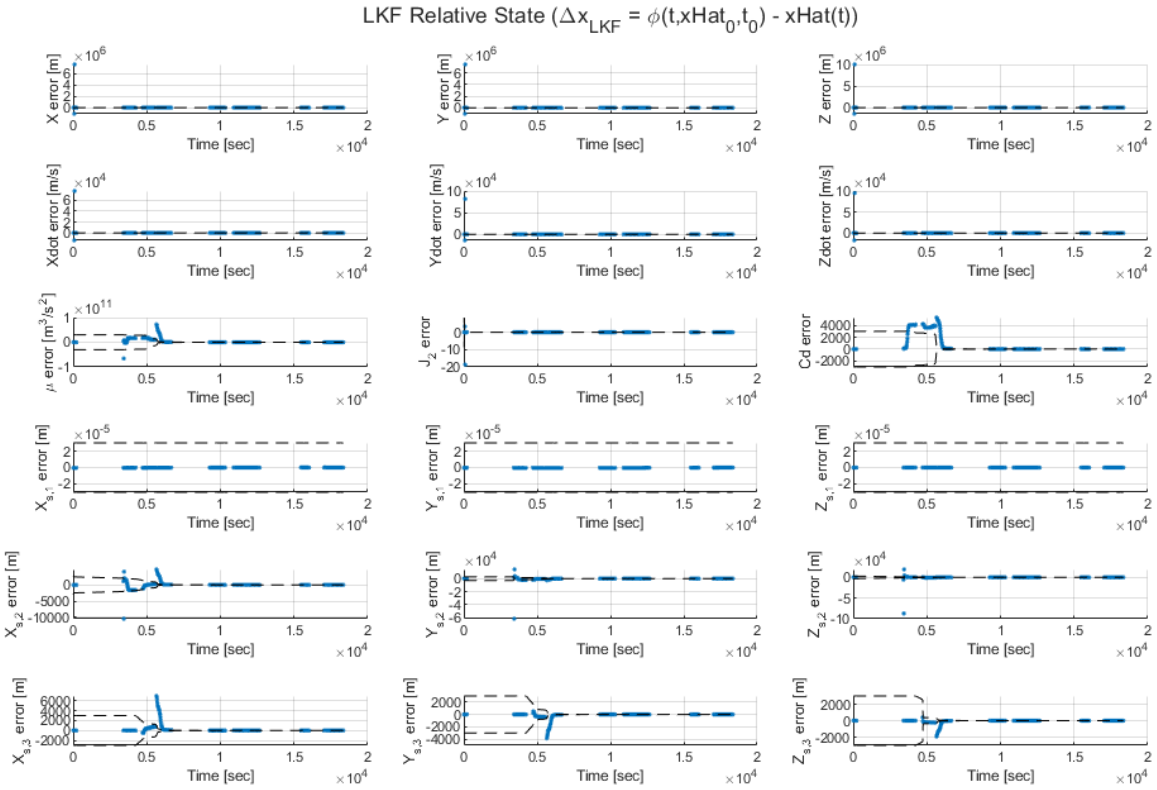

Fig. 14 LKF Relative State with Uncertainty, Base Problem

C. Comparison between the Batch and LKF

At a high level, it seems that the Batch Filter performs better than the LKF for this problem. It converges with residuals that closely resemble the expected noise profile, runs faster than the LKF, and results in a less uncertain final state estimate. Further, its relative state always lies within its predicted error bounds, and the relative state is quite small. However, the final state estimates for both the Batch and LKF are nearly identical, which makes sense since they run fundamentally the same math under the hood. Also, while the Batch is more certain about its final estimate, the LKF uncertainty isn't too far off of the Batch's. In fact, saying that the LKF is always worse than the Batch would be doing the LKF a disservice. We saw in HW 2 that when we process less measurements with the Batch, its state estimates get worse while the LKF isn't very affected. Further, the LKF can be run as an on-line algorithm, whereas the Batch requires a full dataset in order to be useful. In addition, as far as we are currently aware at this point in the class, Batch doesn't have an effective way of incorporating process noise into its estimate - implying that it is susceptible to unmodeled dynamics and will produce a biased result in their presence.

III. Relative Data Strengths

The next thing we wanted to investigate in the context of our filters was the relative strength of solely range data vs. solely range rate data. To do so, we fed a dataset consisting of each measurement into the Batch Filter to see how it performed. To initialize the filters, we used the same initial conditions as the base problem in Section II.

A. Range Data Only

The first case tested was a dataset consisting of just range data. After feeding the data to the Batch Filter, it converged after 4 runs and the range residuals matched the expected noise profile as seen in Figure 15 with a postfit normalized RMS of 0.9723.

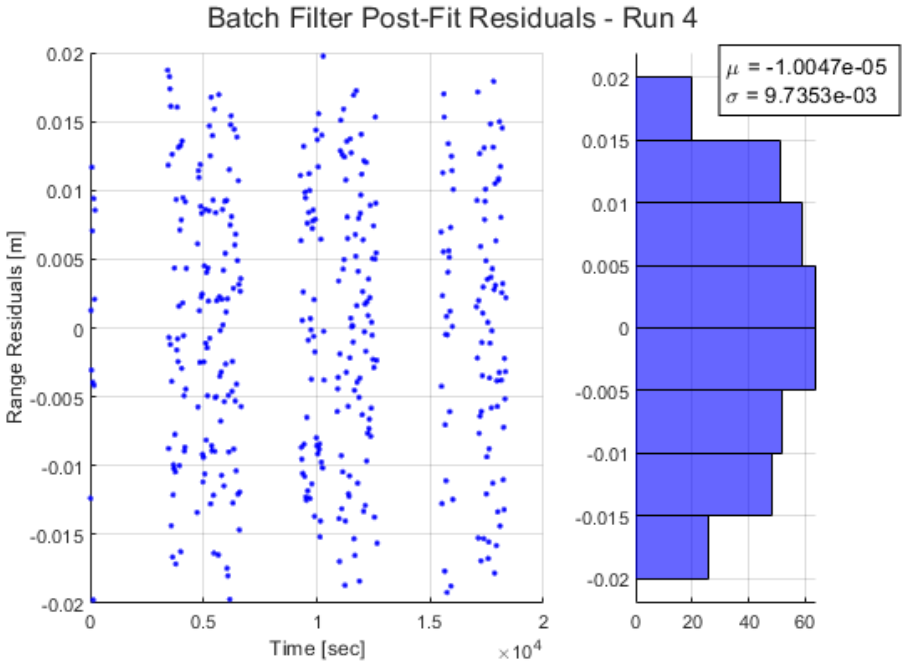

Fig. 15 Batch Filter Postfit Residuals, Range Dataset

As shown by the histogram, using just range data results in an estimated range uncertainty of 9.7353×10^{-3} m, which is close to the expected value of 1×10^{-2} m. Looking at the trace of the covariance matrix vs. time in Figure 16:

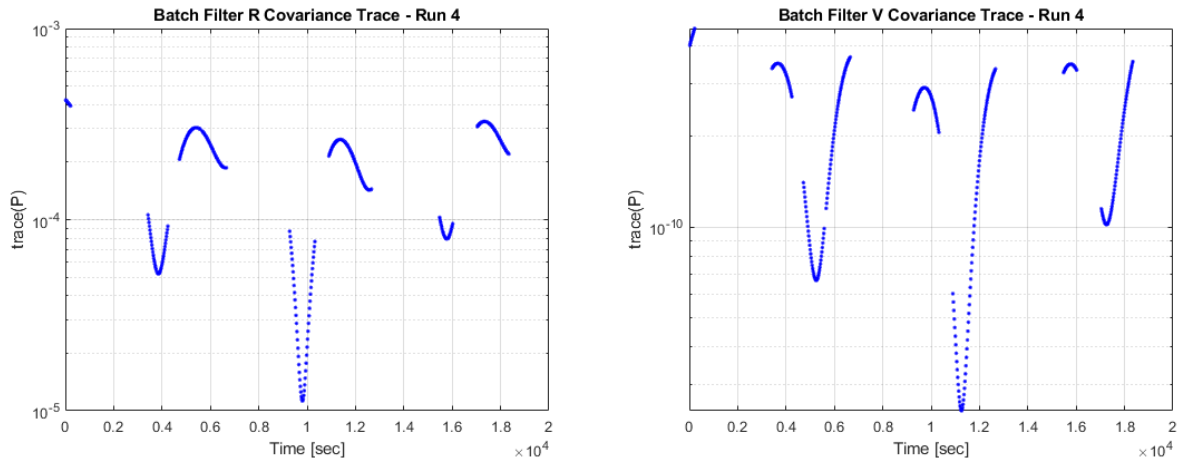

Fig. 16 Batch Covariance Matrix Position and Velocity Trace vs. time, Range Data only

We can see a similar pattern and magnitude of the trace as when processing all the data in Section II. Looking at the covariance ellipsoids in Figure 17:

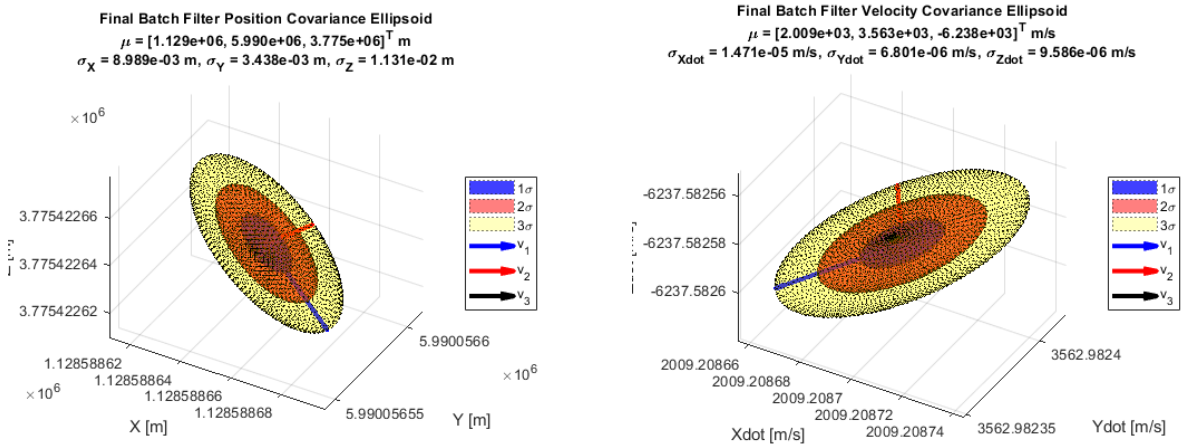

Fig. 17 Batch Position and Velocity Covariance Ellipsoids, Range Data only

We once again see similar magnitudes of uncertainty as discussed in Section II. Further, the final state estimate is almost identical, with the relative state looking incredibly similar in Figure 18.

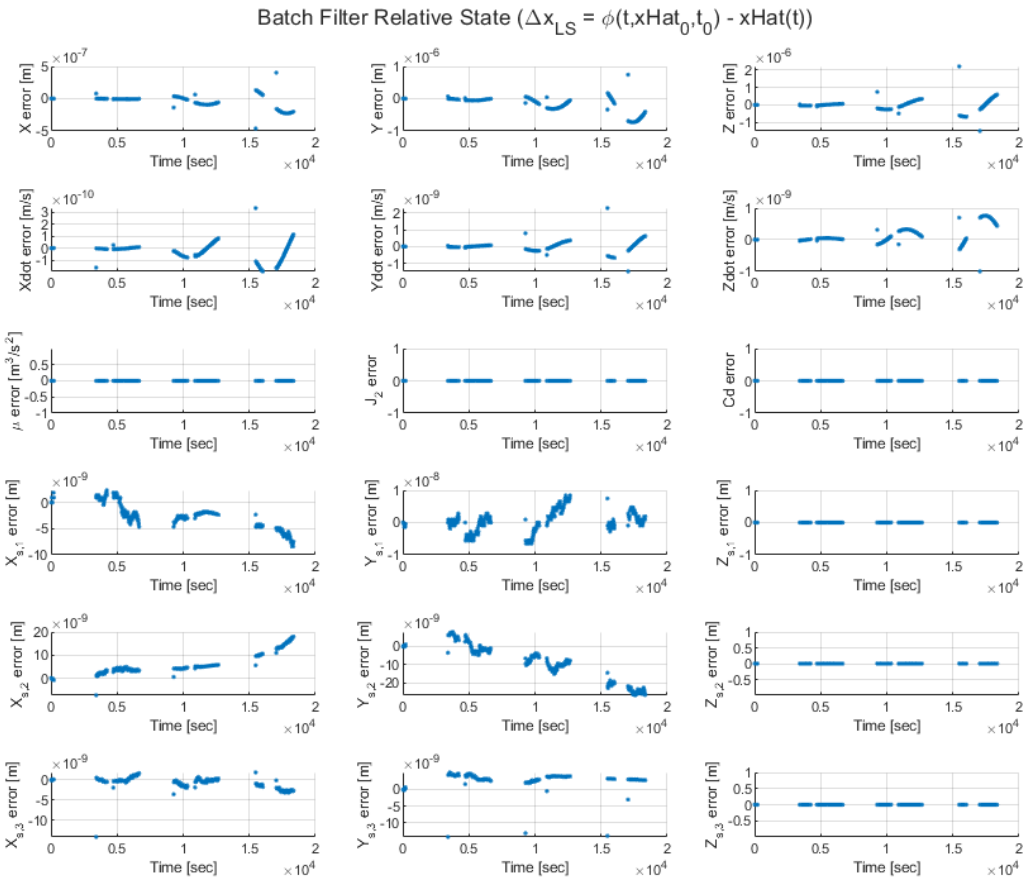

Fig. 18 Batch Relative State, Range Data only

B. Range Rate Data Only

The next test case was a dataset consisting of just range rate data. After feeding the data to the Batch Filter, it converged after 5 runs and the range rate residuals matched the expected noise profile as seen in Figure 19 with a postfit normalized RMS of 0.9762.

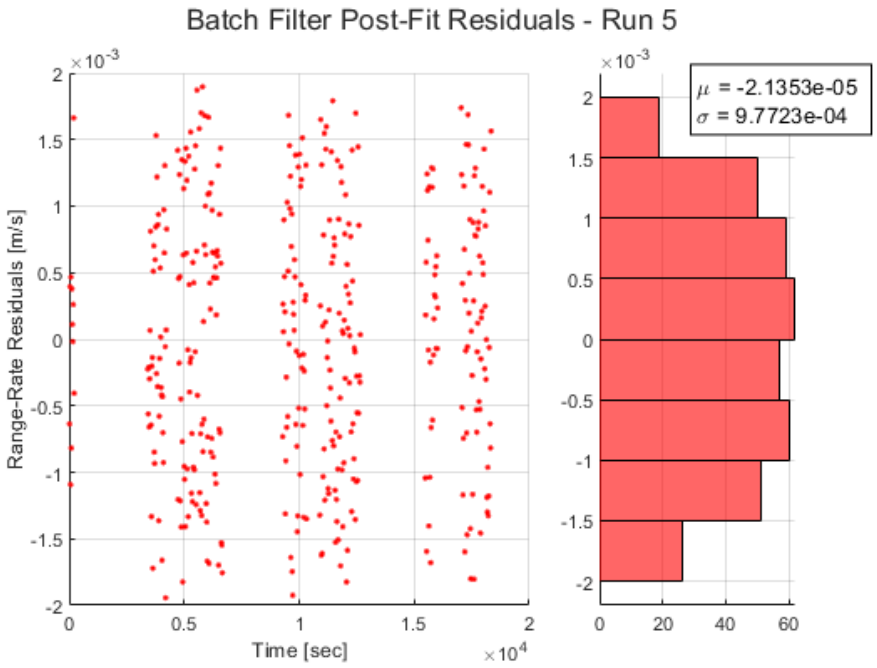

Fig. 19 Batch Filter Postfit Residuals, Range Rate Dataset

As shown by the histogram, using just range rate data results in an estimate range rate uncertainty of 9.7723×10^{-4} m/s, which is close to the expected value of 1×10^{-3} m/s. Looking at the trace of the covariance matrix vs. time in Figure 20:

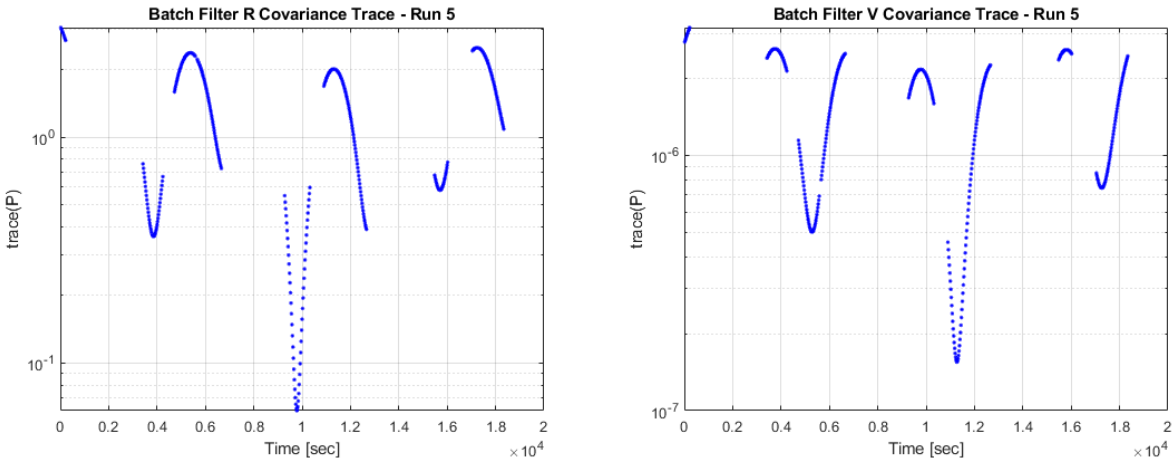

Fig. 20 Batch Covariance Matrix Position and Velocity Trace vs. time, Range Rate Data only

We can see that the covariance trace is generally higher than it was in Section II, and also higher than the solely range results. Looking at the covariance ellipsoids in Figure 21:

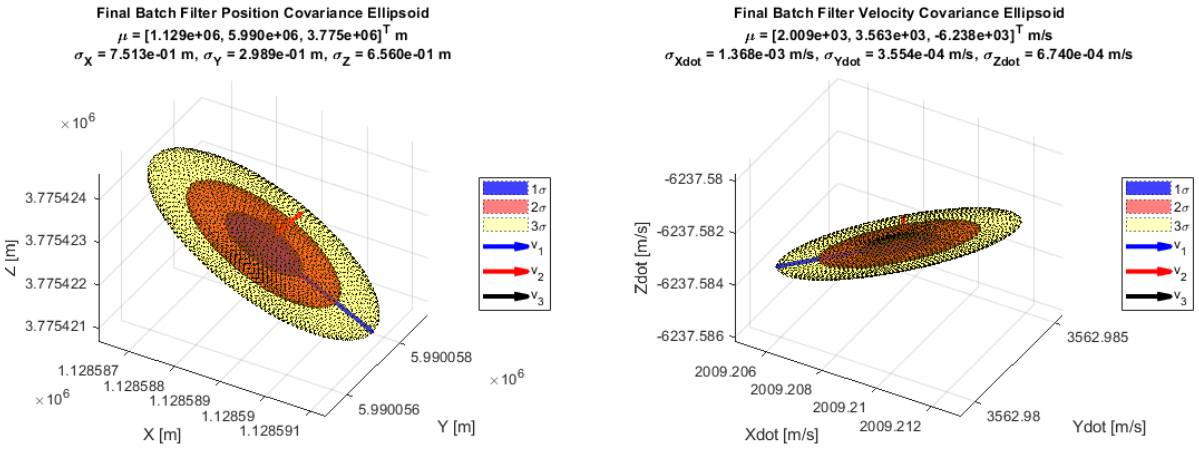

Fig. 21 Batch Position and Velocity Covariance Ellipsoids, Range Rate Data only

We can see that the uncertainty magnitudes are much higher than those in Section II and from the range data by a factor of 10-100! However, the final estimated state is the same as that in Section II and the range data, implying that just the state uncertainty is affected. This can also be seen in the relative state based on just range rate data in Figure 22.

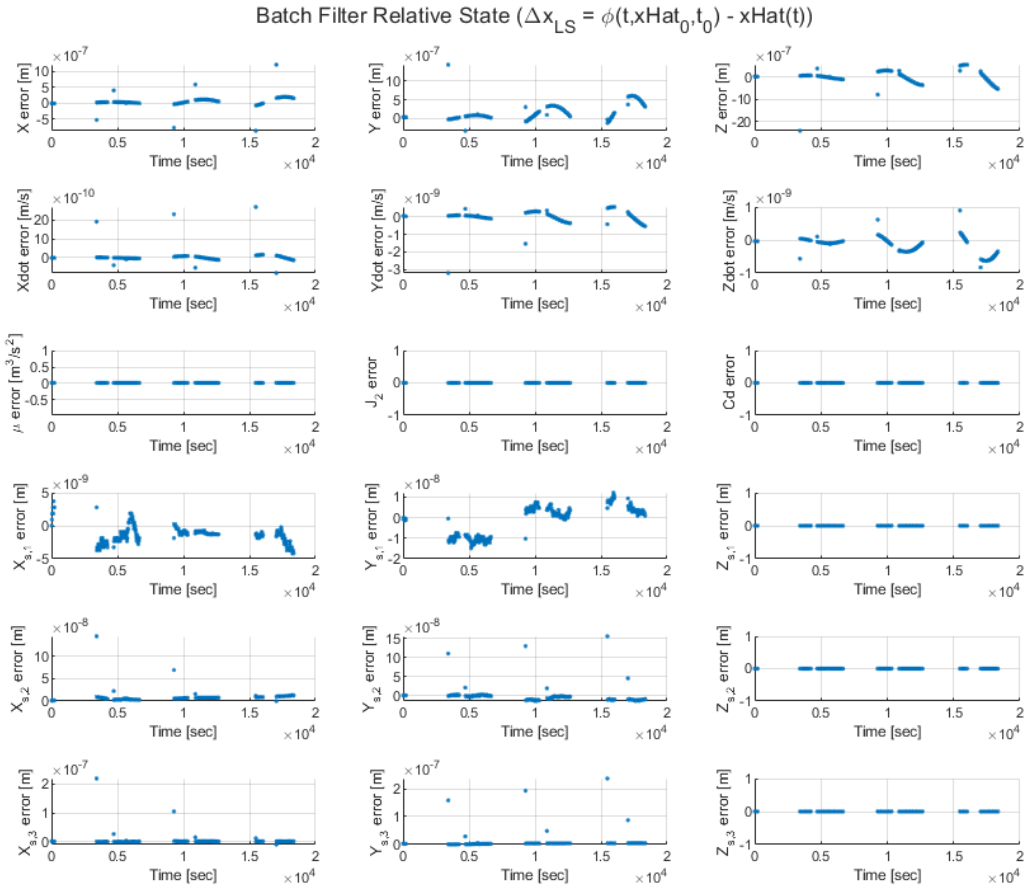

Fig. 22 Batch Relative State, Range Rate Data only

C. Discussion

A summary of the results from the data strength experiment can be seen in Table 5:

Table 5 Data Strength Comparison

Case	σ_X [m]	σ_Y [m]	σ_Z [m]	$\sigma_{\dot{X}}$ [m/s]	$\sigma_{\dot{Y}}$ [m/s]	$\sigma_{\dot{Z}}$ [m/s]	Postfit RMS
Base Problem	8.987×10^{-3}	3.437×10^{-3}	1.129×10^{-2}	1.471×10^{-5}	6.795×10^{-6}	9.579×10^{-6}	0.9844
Range Only	8.989×10^{-3}	3.438×10^{-3}	1.131×10^{-2}	1.471×10^{-5}	6.801×10^{-6}	9.586×10^{-6}	0.9723
Range Rate Only	7.513×10^{-1}	2.989×10^{-1}	6.560×10^{-1}	1.368×10^{-3}	3.554×10^{-4}	6.740×10^{-4}	0.9762

It can be clearly seen that the range rate data provides a much more uncertain estimate of the final state than using both measurements or just range data. In fact, using just range data seems strong enough to rely on almost exclusively, being nearly as certain as using both measurements.

This makes intuitive sense. Range rate is analogous to a velocity measurement, which by definition requires two position measurements to create a rate of change. In other words, a velocity measurement only provides where a spacecraft is going, but the spacecraft can still be anywhere in 3D space. There isn't a reference point to compare the velocity measurement to, so in effect all the velocity measurement provides is a direction the spacecraft is moving and how fast it is doing so. On the other hand, a range measurement is analogous to a position measurement, which is exact in 3D space and can be extrapolated to create velocity estimates. This is tied to the question of observability, in which a system that only provides velocity measurements needs more measurements to be observable than a system that only provides position measurements.

IV. Effect of Fixed Stations

The next thing we wanted to investigate in the context of our filters was the effect of fixing ground stations with respect to the Earth. To do so, we modified $\bar{P}_0$ to change the initial variance of the station positions.

A. Results of Fixing No Stations

The first test case was to investigate what happens when none of the stations are fixed in the ECEF frame. To test this case, we changed $\bar{P}_0$ to be

$$\bar{P}_0 = \text{diag}([1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^{20}, 1 \times 10^6, 1 \times 10^6, \\ 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6])$$

Which effectively makes the initial ECI position of Station 101 vary as much as the others. Running the Batch and LKF with this initial condition resulted in the following results:

Table 6 Effect of Fixing No Stations on Final State Uncertainty

Case	σ_X [m]	σ_Y [m]	σ_Z [m]	$\sigma_{\dot{X}}$ [m/s]	$\sigma_{\dot{Y}}$ [m/s]	$\sigma_{\dot{Z}}$ [m/s]	Runs
Base Problem, Batch	8.987×10^{-3}	3.437×10^{-3}	1.129×10^{-2}	1.471×10^{-5}	6.795×10^{-6}	9.579×10^{-6}	4
No Fixed Stations, Batch	6.179×10^{2}	1.164×10^{2}	1.453×10^{-2}	3.676×10^{-1}	2.073×10^{-1}	1.090×10^{-5}	6
Base Problem, LKF	9.977×10^{-3}	3.562×10^{-3}	1.179×10^{-2}	1.722×10^{-5}	6.887×10^{-6}	1.068×10^{-5}	5
No Fixed Stations, LKF	6.112×10^{2}	2.238×10^{2}	1.1711×10^{-2}	3.638×10^{-1}	2.041×10^{-1}	1.129×10^{-5}	6

We can clearly see a massive increase in the state uncertainty for X, $\dot{X}$, Y, and $\dot{Y}$ of the spacecraft, changing from on the order of 1 cm to hundreds of meters. This is a large enough uncertainty to be problematic, for example, in autonomous rendezvous and docking. If the chaser spacecraft has a position that can vary by hundreds of meters and is trying to rendezvous with a docking port that is on the order of a few meters in diameter, docking is unlikely to occur. In fact, collision becomes a major concern and likely outcome! Interestingly, the uncertainty in Z and $\dot{Z}$ doesn't change very much. This is because the inertial Z coordinate of each station doesn't change in the dynamics, so the filters effectively have a strong reference point in the Z direction from each ground station. The final state estimate changes a bit as well from the Base Problem to this test case, as shown in Table 7.

Table 7 Effect of Fixing No Stations on Final State Estimate

Case	X [m]	Y [m]	Z [m]	$\dot{X}$ [m/s]	$\dot{Y}$ [m/s]	$\dot{Z}$ [m/s]
Base Problem, Batch	1.129×10^6	5.990×10^6	3.775×10^6	2.009×10^3	3.563×10^3	-6.238×10^3
No Fixed Stations, Batch	1.129×10^6	5.990×10^6	3.775×10^6	2.009×10^3	3.563×10^3	-6.238×10^3
Base Problem, LKF	1.129×10^6	5.990×10^6	3.775×10^6	2.009×10^3	3.563×10^3	-6.238×10^3
No Fixed Stations, LKF	1.113×10^6	5.993×10^6	3.775×10^6	2.000×10^3	3.568×10^3	-6.238×10^3

From the table, we can see that the final position estimate for the LKF shifts by a few km in position and a few m/s in velocity. While slight, this small shift could be all that's needed for two satellites to collide and result in dangerous space debris. Thus, knowing how valid the estimate of the ground station position is seems crucial for ensuring a correct satellite state estimate.

B. Results of Fixing Station 337 or 394

The second test case was to fix Station 337 or 394 instead of Station 101. To do this, we just swapped the corresponding initial variance of Station 101 with either Station 337 or 394 like so:

$$\bar{P}_{0,337} = \text{diag}([1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^{20}, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^{-10}, 1 \times 10^{-10}, 1 \times 10^{-10}, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6])$$

$$\bar{P}_{0,394} = \text{diag}([1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^{20}, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^6, 1 \times 10^{-10}, 1 \times 10^{-10}, 1 \times 10^{-10}])$$

The results of this experiment can be seen in Table 8 and Table 9, where fixing Station 101 represents the Base Problem.

Table 8 Effect of Fixing Individual Stations on Final State Uncertainty

Case	σ_X [m]	σ_Y [m]	σ_Z [m]	$\sigma_{\dot{X}}$ [m/s]	$\sigma_{\dot{Y}}$ [m/s]	$\sigma_{\dot{Z}}$ [m/s]
Station 101 Fixed, Batch	8.987×10^{-3}	3.437×10^{-3}	1.129×10^{-2}	1.471×10^{-5}	6.795×10^{-6}	9.579×10^{-6}
Station 337 Fixed, Batch	3.381×10^{-3}	5.508×10^{-3}	9.680×10^{-3}	2.081×10^{-5}	4.777×10^{-6}	9.898×10^{-6}
Station 394 Fixed, Batch	1.147×10^{-2}	2.984×10^{-3}	9.751×10^{-3}	5.825×10^{-6}	7.756×10^{-6}	6.720×10^{-6}
Station 101 Fixed, LKF	9.977×10^{-3}	3.562×10^{-3}	1.179×10^{-2}	1.722×10^{-5}	6.887×10^{-6}	1.068×10^{-5}
Station 337 Fixed, LKF	3.337×10^{-3}	5.564×10^{-3}	9.714×10^{-3}	2.070×10^{-5}	4.822×10^{-6}	9.536×10^{-6}
Station 394 Fixed, LKF	1.163×10^{-2}	2.979×10^{-3}	9.776×10^{-3}	6.119×10^{-6}	7.861×10^{-6}	6.857×10^{-6}

Table 9 Effect of Fixing Individual Stations on Final State Estimate

Case	X [m]	Y [m]	Z [m]	$\dot{X}$ [m/s]	$\dot{Y}$ [m/s]	$\dot{Z}$ [m/s]
Station 101 Fixed, Batch	1.129×10^6	5.990×10^6	3.775×10^6	2.009×10^3	3.563×10^3	-6.238×10^3
Station 337 Fixed, Batch	1.129×10^6	5.990×10^6	3.775×10^6	2.009×10^3	3.563×10^3	-6.238×10^3
Station 394 Fixed, Batch	1.129×10^6	5.990×10^6	3.775×10^6	2.009×10^3	3.563×10^3	-6.238×10^3
Station 101 Fixed, LKF	1.129×10^6	5.990×10^6	3.775×10^6	2.009×10^3	3.563×10^3	-6.238×10^3
Station 337 Fixed, LKF	1.129×10^6	5.990×10^6	3.775×10^6	2.009×10^3	3.563×10^3	-6.238×10^3
Station 394 Fixed, LKF	1.129×10^6	5.990×10^6	3.775×10^6	2.009×10^3	3.563×10^3	-6.238×10^3

From the tables, we can see that changing which station is fixed does affect the final state estimate uncertainty. However, the final state estimate itself is completely unchanged. While there doesn't seem to be a clear "best" station to fix, it is evident that fixing any station is vastly superior to fixing none at all.

This makes sense intuitively. If we want a valid state estimate, we need a reference point that we can locate with near 100% certainty. Without a known reference point, there are too many complicated variables at play that getting a solid estimate becomes a herculean task, requiring an absurd number of measurements to pare down the massively increased uncertainty.

On the other hand, knowing too many variables with too much certainty could result in an overconstrained problem with very little room for the filter to adjust the state estimate. This is counterintuitive, but makes sense when taken in context with filter initialization. If we make a bad guess in relation to the known variables, it is very likely that our guess is physically impossible to resolve with a state estimate, and the result will suffer. Thus, the perfect middle ground seems to be knowing one reference point very well, while allowing everything else to vary for filter wiggle room.

V. Potter Algorithm Implementation

As an extra point of exploration for the project, we decided to implement the Potter Algorithm and compare it to the LKF. We discovered that the LKF has a number of numerical issues, chief among them the possibility of generating non-positive definite covariance matrices. The Potter Algorithm addresses this by replacing the measurement update of the LKF with a square root measurement update approach.

A. Potter Algorithm Overview

A square root solution approach attempts to replace the covariance matrix P with its matrix square root W such that $P = WW^\intercal$. If this is true, then there is a matrix $\tilde{A}$ such that

$$\tilde{A}\tilde{A}^\intercal = I - \tilde{F}\alpha\tilde{F}^\intercal \tag{7}$$

Where $\tilde{F} = \bar{W}^\intercal H^\intercal$ and $\alpha = (\tilde{F}^\intercal\tilde{F} + R)^{-1}$. Then, the square root measurement update can be carried out as follows [2]:

$$\begin{aligned}
\tilde{F} &= \bar{W}^\intercal H^\intercal \\
\alpha &= (R + \tilde{F}^\intercal\tilde{F})^{-1} \\
K &= \bar{W}\tilde{F}\alpha \\
W &= \bar{W}\tilde{A} \\
\hat{x} &= \bar{x} + K(y - H\bar{x})
\end{aligned}$$

There are a number of ways to solve for $\tilde{A}$, and the Potter Algorithm is one of them. The Potter Algorithm assumes that the measurements to process are all scalars, which makes α a scalar. Then, there is a matrix $\tilde{A}$ such that:

$$\begin{aligned}\tilde{A}\tilde{A}^\intercal &= I - \alpha\tilde{F}\tilde{F}^\intercal \\ &= \left[I - \gamma\alpha\tilde{F}\tilde{F}^\intercal\right]\left[I - \gamma\alpha\tilde{F}\tilde{F}^\intercal\right]^\intercal\end{aligned}$$

Then, using algebra we can solve for $\gamma = \frac{1}{1+\sqrt{R\alpha}}$. The measurement update then becomes [2]:

$$\begin{aligned}
&1) && \tilde{F} = \bar{W}^\intercal H^\intercal \\
&2) && \alpha = (\tilde{F}^\intercal\tilde{F} + R)^{-1} \\
&3) && \gamma = \frac{1}{1+\sqrt{R\alpha}} \\
&4) && K = \alpha\bar{W}\tilde{F} \\
&5) && \hat{x} = \bar{x} + K(y - H\bar{x}) \\
&6) && W = \bar{W} - \gamma K\tilde{F}^\intercal
\end{aligned}$$

We can then propagate W instead of P and avoid the floating point math errors that occur in the LKF, keeping in mind that if we want P at any given timestep k we just perform the operation $P_k = W_k W_k^\intercal$. To propagate W_k to the next time step, we simply use $W_{k+1} = \Phi t_{k+1}, t_k W_k$.

B. Implementation

To implement the Potter Algorithm for this project, we need to amend the measurement update to account for vector measurements. Luckily, this is very easy to do. Before performing the time update, we simply repeat steps 1-6 for each measurement in our measurement vector, where R is the uncertainty of each individual measurement in the vector and y is the measurement itself. Doing so resulted in the following residuals for the Base Problem:

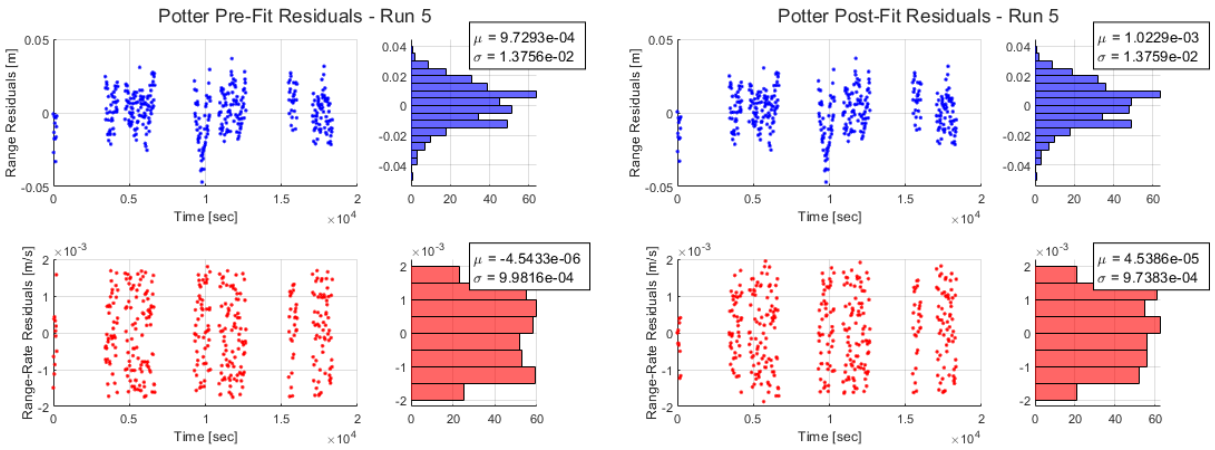

Fig. 23 Potter Pre- and Postfit Residuals - Base Problem

We can see from Figure 23 that the residuals do a much better job accross the board at matching the expected noise profile of our measurements. Whereas the LKF only had noise matching in the range rate prefit residuals, the Potter results are very nearly noise in each measurement. While the residuals aren't exactly noise due to the filter smugness problem described in Section II, they are much better than the LKF. Looking at the trace of the covariance matrices:

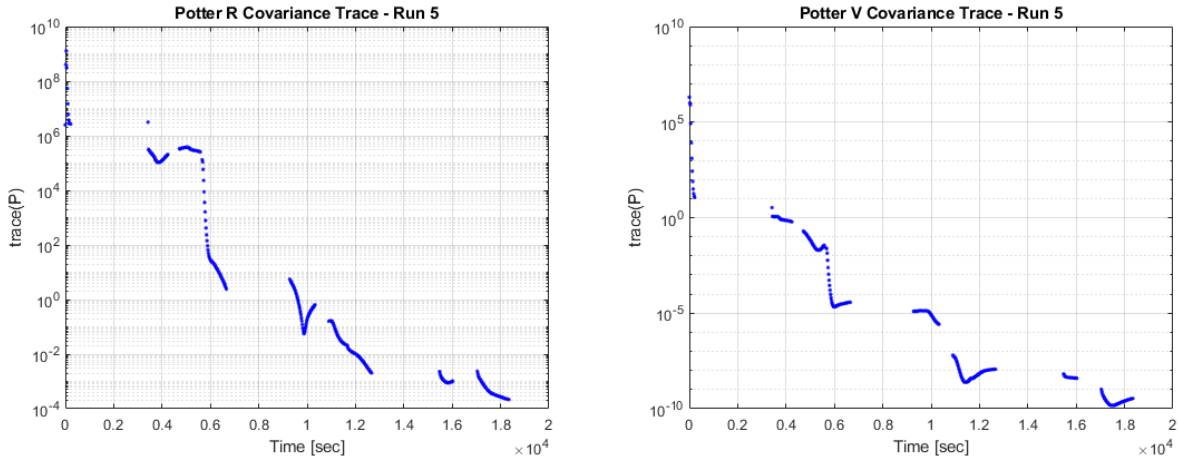

Fig. 24 Potter Covariance Matrix Position and Velocity Trace vs. time

Thankfully, we can see that while the LKF produced covariance matrices with negative traces, the Potter Algorithm shows no such nonsense. This means that the Potter Algorithm did its job, and we have guaranteed positive definite matrices in the context of this problem! Further looking at the covariance ellipsoids:

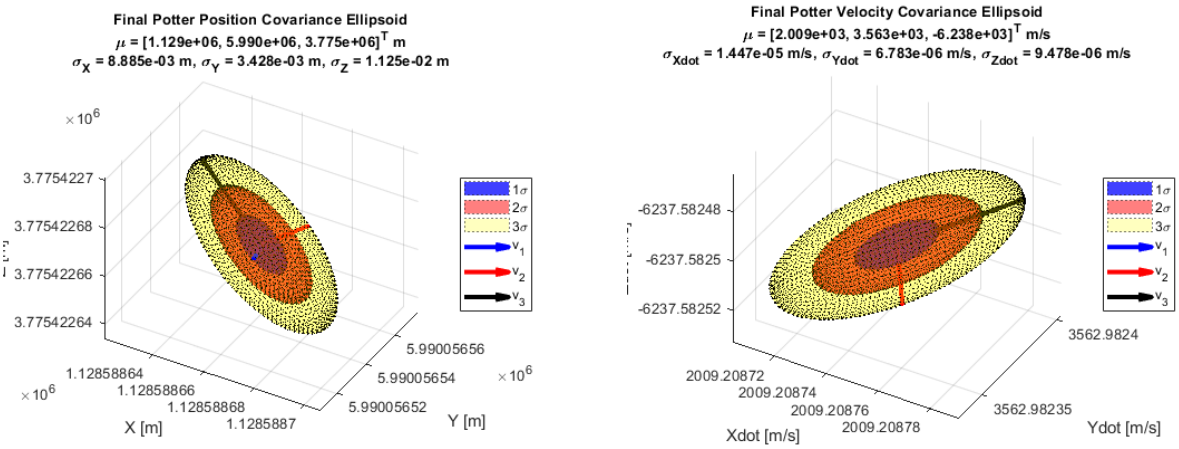

Fig. 25 Potter Position and Velocity Covariance Ellipsoids

From Figure 25, we can see that the Potter arrives at the same state estimate as the Batch and LKF for the Base Problem. Tabulating the uncertainty differences in Table 10:

Table 10 Comparison of Final State Uncertainty Across Filters

Filter	σ_X [m]	σ_Y [m]	σ_Z [m]	$\sigma_{\dot{X}}$ [m/s]	$\sigma_{\dot{Y}}$ [m/s]	$\sigma_{\dot{Z}}$ [m/s]
Batch	8.987×10^{-3}	3.437×10^{-3}	1.129×10^{-2}	1.471×10^{-5}	6.795×10^{-6}	9.579×10^{-6}
LKF	9.977×10^{-3}	3.562×10^{-3}	1.179×10^{-2}	1.722×10^{-5}	6.887×10^{-6}	1.068×10^{-5}
Potter	8.885×10^{-3}	3.428×10^{-3}	1.125×10^{-2}	1.447×10^{-5}	6.783×10^{-6}	9.478×10^{-6}

From Table 10, we can see that the Potter Algorithm outperforms both the Batch and LKF in terms of final state

uncertainty, narrowly beating the Batch while beating the LKF by a larger margin. Finally, looking at the relative state in Figure 26:

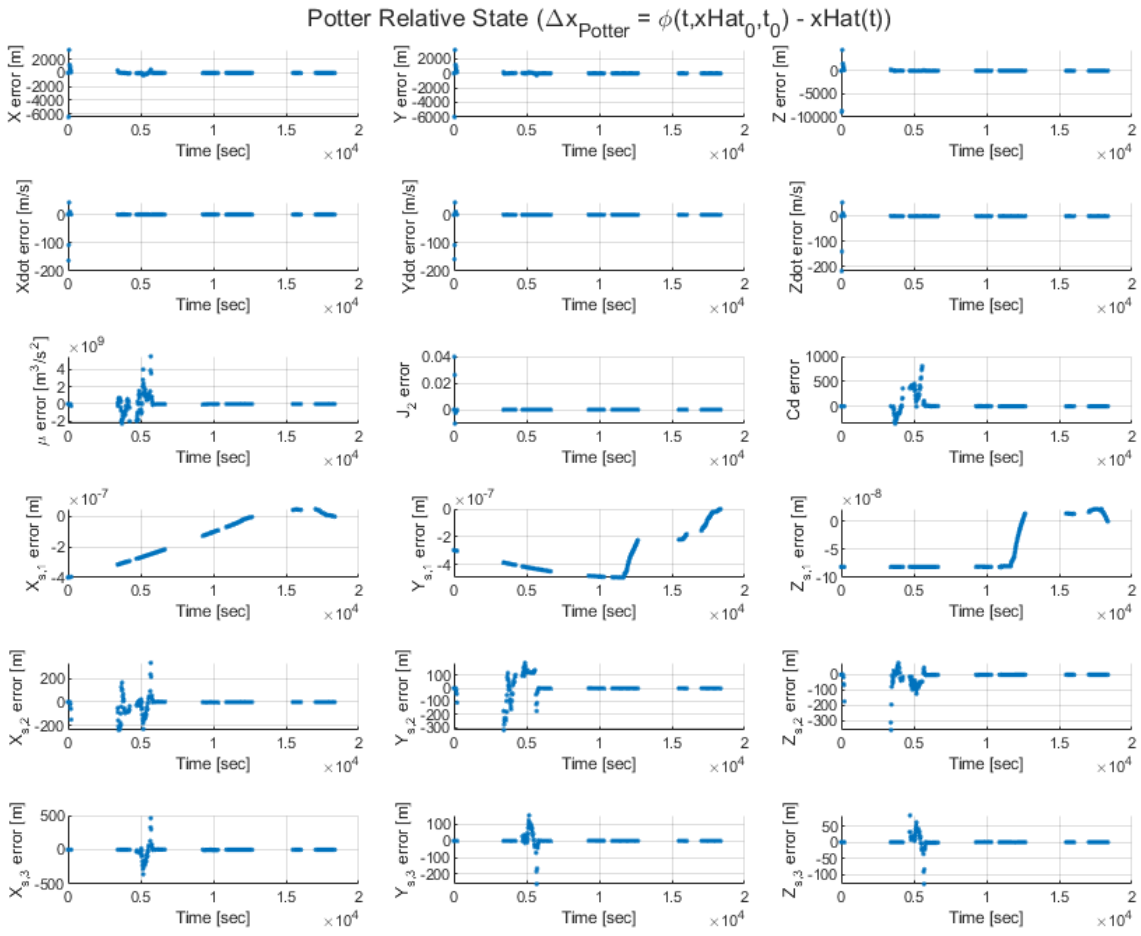

Fig. 26 Potter Relative State

We can see that the relative state is generally of a lower magnitude than the LKF's, with fewer timesteps where the relative state diverges. There are still numerical artifacts on account of filter smugness once again, but by and large it is an improvement over the LKF. Furthermore, looking at the relative state with error bars in Figure 27, we can see that the relative state is now fully encompassed within the uncertainty bounds, which the LKF didn't have at all.

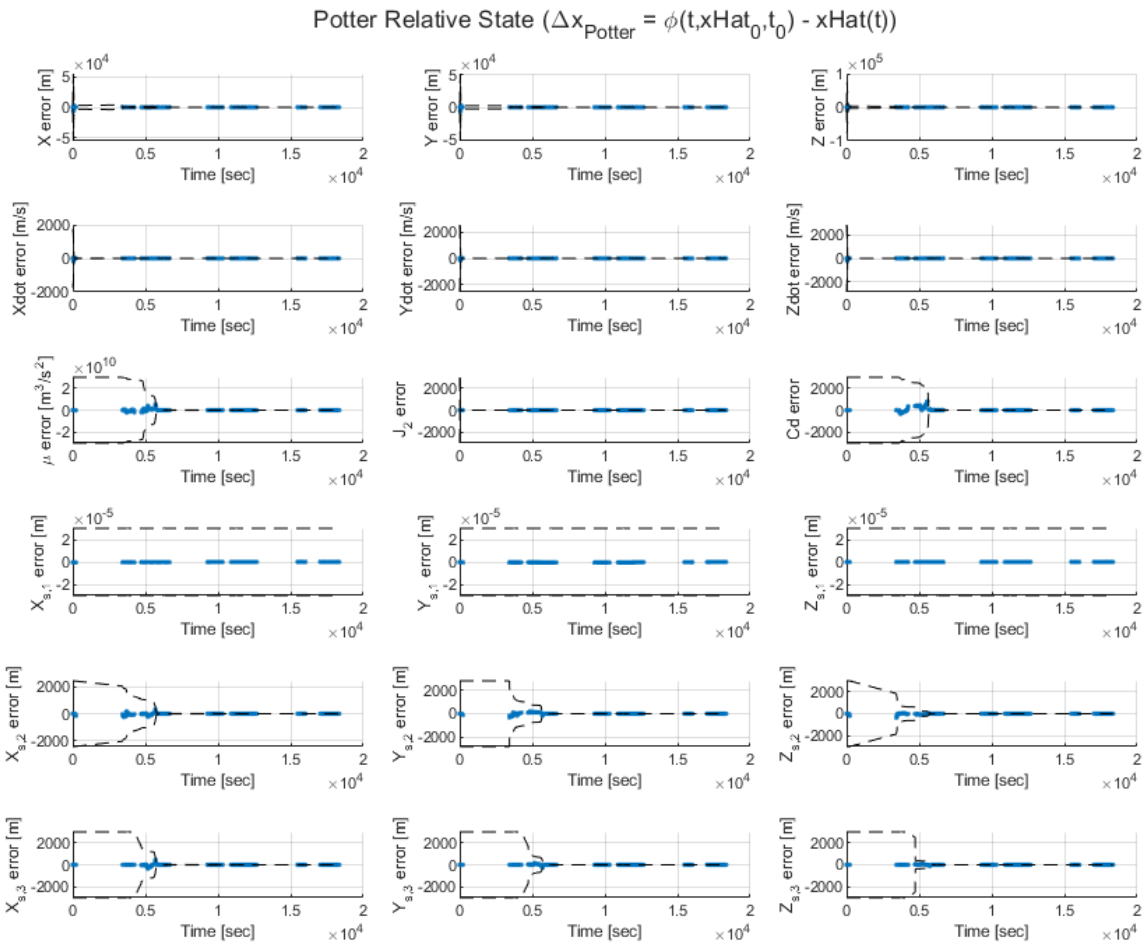

Fig. 27 Potter Relative State, with Uncertainty

VI. Conclusions, Lessons Learned, and Suggestions

In completing this project, I learned about the advantages, drawbacks, and subtleties involved in estimating spacecraft orbits with linearized filters. I learned how to analyze the results of my filters to determine where patterns/bugs/other phenomena are coming from, as well as how to critically debug my code. Further, I learned that Filters can get way more precise than I thought possible! Seeing uncertainties on the order of 1 mm on the scale of LEO was mind bending for me, and inspired me to investigate future projects in orbit determination as well as picturing all the different applications of statistical orbit determination. My favorite application of these filters so far is parameter estimation, as I found it really (really) cool that we can estimate physical parameters of vehicles (ex. C_D) simply by observing their dynamics in context of our state. It got me thinking about estimating masses of aircraft, areas of other satellites, and more practical applications of parameter estimation from orbit!

If I had one suggestion to improve the project, it would be to provide a small bit more guidance on the Potter Algorithm for those who are interested in pursuing it as an extra facet of the project. I unfortunately burned 5 hours of time trying to get a matrix form of γ to work, as I didn't realize we could just repeat the measurement step for each measurement in our measurement vector. If I had known that from the beginning, I would have completed this writeup sooner than an hour before the deadline ;). Overall, however, the project was great and I can't wait to learn more!!

VII. Acknowledgments

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

1) Steven Liu for bouncing ideas back and forth and mutual debugging woes and successes!
2) Jacob Mesley, Emmett Peters, and Justin Lynch for comparing answers and discussing talking points that ended up in this writeup!
3) Alex Moody for helping me figure out that Potter was much less complicated than I was making it out to be...
4) Giovanni Fereoli for providing some excellent sanity checks
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! I've put together an example code flow here, just in case:

Goal: Run the Base Problem to get my baseline Batch and LKF results.

Steps:

1) Open Project1Main.m
2) Run Project1Main.m
3) Look at measurements if desired
4) Type the number 1 into the console and press 'Enter'
5) Wait for Batch to run
6) Look at Batch results if desired
7) Click the command window and press 'Enter'
8) Wait for LKF to run
9) Look at LKF results if desired
10) To animate the problem: Click the command window and type the letter y, then press 'Enter' and wait for animations to run. Pressing 'Ctrl-C' will exit the animation at any time. If your monitor/screen isn't resolution 1920x1080, you can edit line 45 of animateProblem.m with your correct resolution! If you want to save a video of the animation, change the saveMovie boolean from false to true for the problem you are running - there is one flag per animation! You can see pre-generated animations in the "Videos" folder of my submission.
11) To skip animating the problem: Click the command window and type the letter n, then press 'Enter'
12) You can exit the program flow at any time by clicking the command window and pressing 'Ctrl-C'

Have fun!