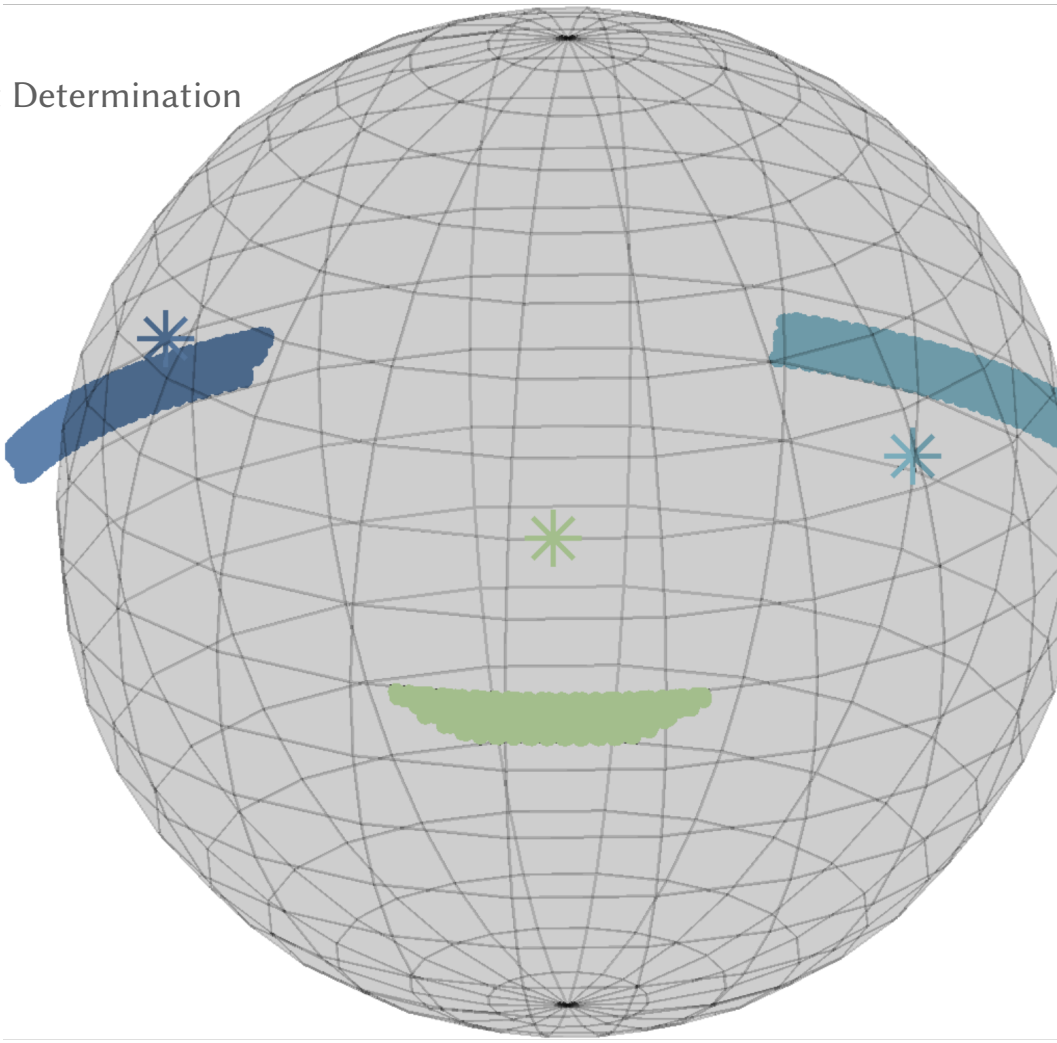
Final Report

Methods of Orbit Determination

Burton Yale

May 14, 2022



Contents

Introduction

A satellite, JahSat-1, is orbiting the Earth, while being tracked by three stations, Kwajalein, Diego Garcia, and Arecibo, in this report we will detail the dynamic models considered, how the states map to the measurements, what filter was implemented, and the overall results considering multiple cases. Given an initial *estimated* state and spacecraft parameters, each station will provide range and range-rate data to use in a Kalman Filter to determine the spacecraft's approximate state. With an initial epoch of March 23rd, 2018, 08:55:03 UTC, we will track the spacecraft using 6 days worth of data collected. After an initial filter pass, we will then extrapolate this estimated position to the first maneuver the spacecraft is expected to make, exactly 7 days later, March 30th, 2018, 08:55:03 UTC.

Dynamics Modelling

Gravity

Two gravity models were used within the filter, one for the linearized dynamics propagating the **S**tate **T**ransition **M**atrix (STM), and the other more accurate model for propagating the state. The linearized model only used the first approximation of spherical harmonics, out to J_2. The dynamics end up being:

$$\ddot{\boldsymbol{x}}_{J_2} = \begin{bmatrix} \boldsymbol{v} \\ \left(\frac{\mu}{r^3} + J_r\right)\boldsymbol{r} + J_k\hat{\boldsymbol{k}} \end{bmatrix}_{6\times 1} \tag{1}$$

Where:

$$J_r = \frac{J_0}{r^5}\left(1 - \frac{5}{r^2}\left(\boldsymbol{r}\cdot\hat{\boldsymbol{k}}\right)^2\right)$$

$$J_k = \frac{2J_0}{r^5}\left(\boldsymbol{r}\cdot\hat{\boldsymbol{k}}\right)$$

$$J_0 = \tfrac{3}{2}\mu J_2 a_e^2$$

Where values of J_2, a_e, and μ were provided as:

$$J_2 = 0.001\,082\,625\,45$$

$$a_e = 6,378.136\,3 \text{ km}$$

$$\mu = 398,600.441\,5 \ \frac{\text{km}^3}{\text{s}^2}$$

For the higher fidelity state propagation, we instead use the EGM96 spheroid Earth model out to the 20$^{\text{th}}$ degree of the C and S Stokes coefficients. The potential energy of the model can be expressed, using spherical coordinates, as the following summation*:

$$U(r,\theta,\varphi) = \frac{\mu}{r}\left(1 + \sum_{n=1}^{\infty}\left(\frac{a_e}{r}\right)^n \sum_{m=0}^{n} Y_n^m(\sin(\theta))(C_n^m\cos(m\varphi) + S_n^m\sin(m\varphi))\right) \tag{2}$$

Where:

$$Y_n^m(x) = \sqrt{\frac{(2-\delta_{0m})(2n+1)(n-m)!}{(n+m)!}}P_n^m(x)$$

Where n is the degree, m is the order, Y is the normalized Legendre polynomial, and P is the *un*normalized Legendre polynomial. The values for the Stokes coefficients were pulled from MATLAB's Aerospace Toolbox[1].

*Write-up of potential referenced from Lukas Bystricky.

The acceleration due to the potential can then be defined as $\ddot{\boldsymbol{x}} = \nabla U$. It should be noted that the output of the accelerations is in the **E**arth **C**entered **E**arth **F**ixed (ECEF), and must be rotated back into the **E**arth **C**entered **I**nertial (ECI) frame, for each propagation step.

Drag

When considering drag, the spacecraft was modeled as a cannonball, but with an attitude dependent area, depicted in the Spacecraft Attitude and Area Section. The dynamics are as follows:

$$\ddot{\boldsymbol{x}}_{drag} = \begin{bmatrix} \mathbf{0}_{3\times 1} \\ \frac{C_D A}{2m} \rho \left\| \boldsymbol{v}_{ecef} \right\|^2 \hat{\boldsymbol{v}}_{ecef} \end{bmatrix}_{6\times 1} \tag{3}$$

Where:

$$\rho = \rho_0 \exp\left(-\frac{r - r_0}{H}\right)$$

$$\boldsymbol{v}_{ecef} = \boldsymbol{v} + \begin{bmatrix} \dot{\theta} y \\ -\dot{\theta} x \\ 0 \end{bmatrix}$$

Where $\dot{\theta}$ is the rotation speed of the Earth, ρ_0 is the reference density at the reference height r_0, H is the scale height, C_D is the coefficient of drag*, and A is the perpendicular area to the travel direction. The following values were used for the calculations:

$$\dot{\theta} = 7.292\,115\,146\,706\,979 \times 10^{-5}\ \frac{\text{rads}}{\text{s}}$$

$$\rho_0 = 3.614 \times 10^{-13}\ \frac{\text{kg}}{\text{m}^3}$$

$$r_0 = 70,000 + R_{Earth}$$

$$H = 88,667\ \text{m}$$

$$C_D = 1.88$$

Spacecraft Attitude and Area

When transitioning to an attitude dependent area model, a few new quantities are needed to capture all of the characteristics of the satellite shown in Figure 1. First we must calculate our spacecraft's velocity relative to the atmosphere, which adds a small additional component in the $\hat{i}$, and $\hat{j}$ directions. As the spacecraft's opposing faces

*This is only the initial value used, this quantity is then estimated in the Filter Setup Section.

are all symmetrical creating a rectangular prism, we can consider only each of the 3 unique faces without taking into consideration all 6's orientation. The area for each face is $A_{\hat{X}} = 6$ m^2, $A_{\hat{Y}} = 8$ m^2, and $A_{\hat{Z}} = 12$ m^2. Representing the **R**adial, **I**n-Track, and **C**ross-Track (RIC) spacecraft body frame in ECI, we can find the angles between the spacecraft's relative velocity vector and the normal vector of the panel face. We can use these angles to scale the original area of the face, effectively taking its projection onto the plane normal to our relative velocity.

With the spacecraft faces sorted, all that is left is to calculate our projection of the solar panel, whose area is $A_{SP} = 15$ m^2. The solar panel is modeled to always be facing the Sun, whether it is eclipsed or not by the Earth. The source of how this vector is calculated is covered in the Third-Body Perturbations Section, but once obtained we use a similar method as before. With the solar direction vector $\boldsymbol{r}_S$ as the normal vector of the panel face, we once again calculate the angles between that and the relative velocity to get our projected area. All three body face areas, and the solar panel area are summed together to be used in Equation (3). This is recalculated at every step of propagation within the ODE call.

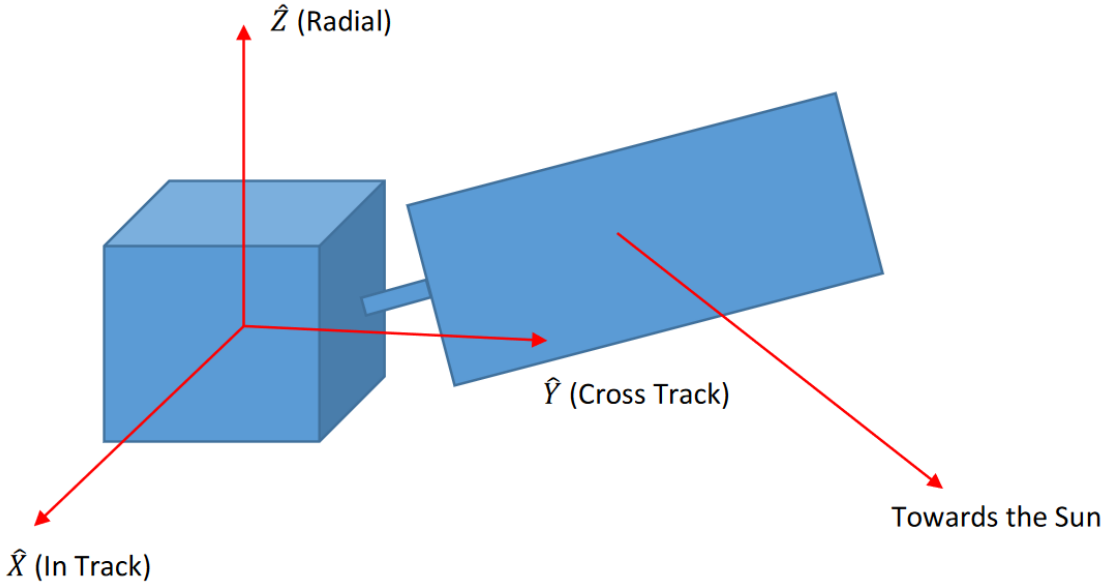


Figure 1: Model of spacecraft attitude, with $\hat{Z}$ in the direction of the of the radial direction, and $\hat{Y}$ in the normal direction.

Solar Radiation Pressure

The **S**olar **R**adiation **P**ressure (SRP) implemented is the faceted[2,3] version accounting for the specular and diffusive emissions from each face with their relative orientation to the Sun vector, $\boldsymbol{r}_S$. The equations of motion begin the following:

$$\ddot{\boldsymbol{x}}_{srp} = \begin{bmatrix} \mathbf{0}_{3\times 1} \\ -\sum_{i=1}^{N} \frac{P_{srp} A_i \cos(\phi_i)}{m} \left(2 \left(\frac{1}{3} C_{d,i} + C_{s,i} \cos(\phi_i) \right) \hat{\boldsymbol{n}} + \left(1 - C_{s,i} \right) \hat{\boldsymbol{s}} \right) \end{bmatrix}_{6\times 1} \tag{4}$$

Where $\hat{\boldsymbol{s}}$ is the vector pointing from the center of the spacecraft to the Sun, and $\hat{\boldsymbol{n}}$ is the normal vector off of the face being evaluated. Figure 2 depicts the relationship between the spacecraft orientation and the Sun. P_{srp} is the force per unit area emitted by the Sun scaled to the current distance between the spacecraft and Sun. At 1 AU, or 149, 597, 871 km, the pressure from the Sun is $P_{srp} = 4.57 \times 10^{-6}\ \mathrm{N/m^2}$. A_i is the projected area of the current face, ϕ_i is the angle between the $\hat{\boldsymbol{n}}$ and $\hat{\boldsymbol{s}}$ vectors, m is the mass of the spacecraft, $C_{d,i}$ is the diffusive coefficient of the face, and $C_{s,i}$, the specular coefficient of the face. Each face is of the spacecraft is modelled to have a different material, which is compiled in the below table:

Face	Material	Diffusive Coefficient	Specular Coefficient
$\hat{X}\pm$	MLI Kapton	0.04	0.04
$\hat{Y}\pm$	MLI Kapton	0.04	0.04
$\hat{Z}+$	White Paint	0.80	0.04
$\hat{Z}-$	Germanium Kapton	0.28	0.18
Solar Panel	Solar Cells	0.04	0.04

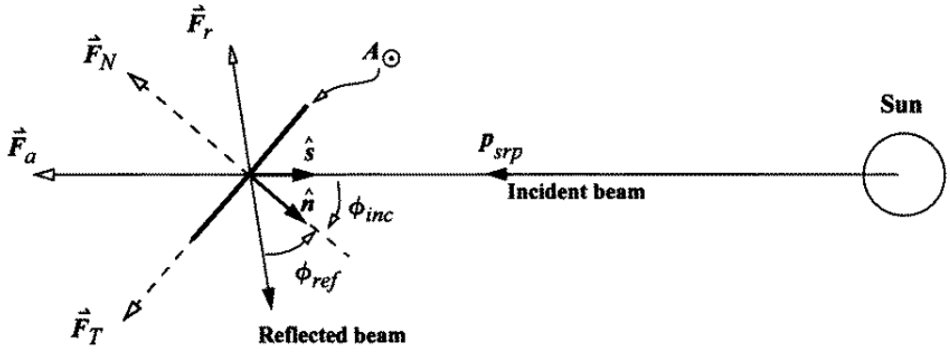


Figure 2: Diagram of the two components of the faceted SRP model??.

Third Body Perturbations

The final acceleration taken into account for determining the satellite's orbit is perturbations from non-Earth bodies, in this case the Sun and Moon are significant. For each of the bodies considered the equations of motion are:

$$\ddot{\boldsymbol{x}}_{3BP} = \begin{bmatrix} \mathbf{0}_{3\times 1} \\ -\mu_p \left(\dfrac{\boldsymbol{r}_{sc} - \boldsymbol{r}_p}{\|\boldsymbol{r}_{sc} - \boldsymbol{r}_p\|} + \dfrac{\boldsymbol{r}_p}{\|\boldsymbol{r}_p\|} \right) \end{bmatrix}_{6\times 1} \tag{5}$$

Where μ_p is the standard gravitational constant of the body, $\boldsymbol{r}_{sc}$ is the position vector from the center body to the spacecraft, and $\boldsymbol{r}_p$ is the position vector from the center body to the body. The values of μ for each body is: $\mu_{\text{Sun}} = 132,712,440,018\ {}^{\text{km}^3}/_{\text{s}^2}$ and $\mu_{\text{Moon}} = 4,902.800\,066\ {}^{\text{km}^3}/_{\text{s}^2}$.

The positions of the Sun and Moon were calculated by using NAIF Spice Kernel data[4,5], using Earth as the center body from the DE440s library. Later, it will be shown that this introduced a certain level of error in the filtering process due to the simulated data being calculated with Vallado's polynomial curves for the position[2]. SPICE's J2000 frame is its stand in for the ECI frame, while ECLIPJ2000 is the standard J2000 w.r.t. to the mean solar system inclination. These ECI vectors are multiplied by the Nutation and Precession matrices, see explanation in Measurement Modelling, to be in the same frame of reference as the rest of the spacecraft model.

Measurement Modelling

Measurements provided are in the form of range, ρ, and range-rates, $\dot{\rho}$, as well as which station the measurement comes from. The data provided is the first 6 days worth of tracking data, which is to be used to extrapolate to a delivery point at exactly 7 days elapsed from initial epoch, 23 March 2018 08:55:03 UTC. Measurements are collected from three stations, Kwajalein (λ = 8.7167° N, ϕ = 167.7333° E), Diego Garcia (λ = 7.3195° S, ϕ = 72.4229° E), and Arecibo (λ = 18.4442° N, ϕ = 66.6464° W). Additionally, each of these stations have their own measurement uncertainty: Kwajalein and Arecibo have range uncertainties of 10 m and range-rate uncertainties of 0.5 $^{\text{mm}}/_{\text{s}}$, while Diego Garcia has range uncertainties of 5 m and range-rate uncertainties of 1 $^{\text{mm}}/_{\text{s}}$. It will be found later that the range measurements may be biased, while range-rate measurements are zero-mean.

Without manipulating any data, we can pull trends to further restrict our initial conditions for filtering the resulting state. Figure 3 shows the outputs of the range and range-rate data for only the first day of measurements, for clarity. Immediately we can see that Arecibo is the station that has the smallest observation arcs, thus being the farthest from the spacecrafts orbit. This *tracks* with the station also having the largest latitude, λ. Additionally, Diego Garcia, the most southernly station, has the deepest troughs in the range data, so we can conclude the spacecraft's orbit is relatively close to the station in inclination. As well since the data is sinusoidally periodic, with roughly a repeat time of a day, we can also conclude the satellite is in a low inclination orbit. Taking into account all of these factors, we can safely say the spacecraft has an inclination less than that of Diego Garcia's latitude, and has a low eccentricity, without viewing our initial approximated station.

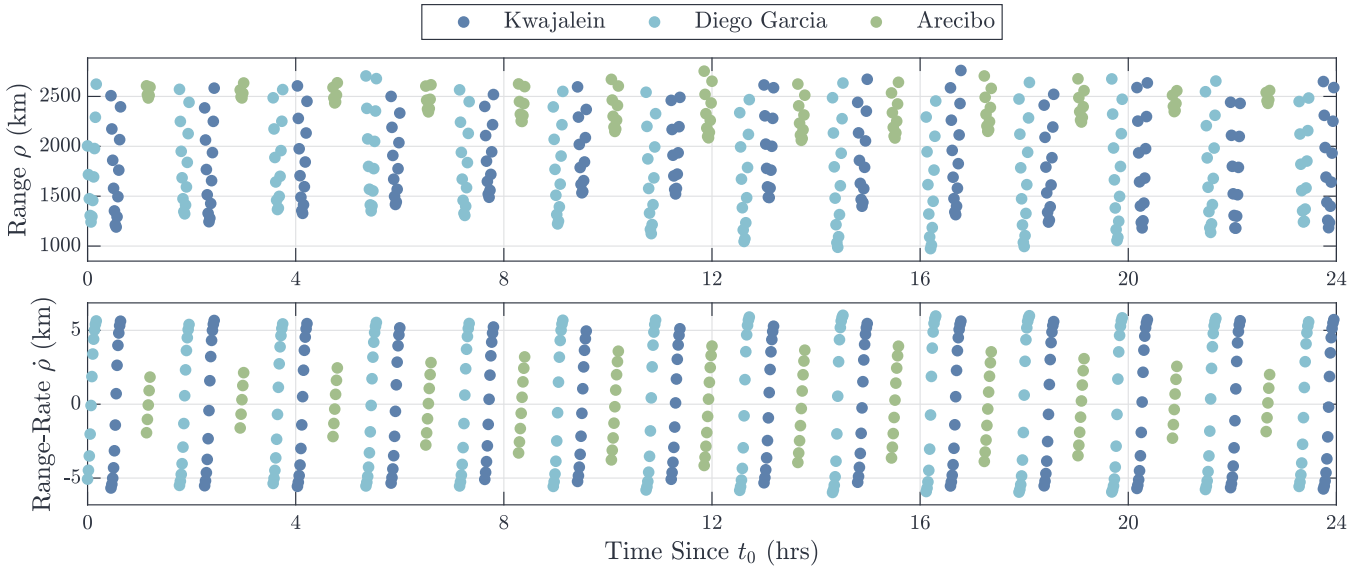


Figure 3: First day's worth of data, without any manipulation, colored by station collecting measurements.

ECEF to ECI Transformation

In order to have meter-level accuracy, we can not conclude the transformation of ECEF to ECI is a rotation about the $\hat{\boldsymbol{k}}$ axis of the ECI frame, we must also taking into account the shifting of the Earth due to its rotational dynamics. Like all rotating masses, there is a certain level of nutation and procession of the pole over time which can become significant to reach the level of accuracy of the station's sensors provided. The primary use of this rotation, in addition to the rotation of accelerations for the EGM Model, is to rotate the positions of the Stations into the ECI frame to calculate the range and range-rate measurements in the next section.

This transformation is the combination of four rotation matrices, which themselves are combinations of rotations. The values used for each rotation are based off the **I**nternational **A**stronautical **U**nion (IAU) 1980 Theory of Nutation and IAU-1976 Theory of Precession[6]:

- The first matrix is the same as in the simplified ECEF to ECI transformation, **R** which is an $\mathbf{R}_3(\theta_{GMST})$ rotation about the $\hat{\boldsymbol{k}}$ axis with respect to the vernal equinox.
- The nutation matrix, **N**, which is comprised of three rotations $\mathbf{R}_1(-\epsilon)\mathbf{R}_3(-\Delta\psi)\mathbf{R}_1(\bar{\epsilon})$, where ϵ is the true obliquity of the ecliptic, $\Delta\psi$ is the nutation of the longitude (augmented with real data), and $\bar{\epsilon}$ is the mean obliquity of the ecliptic.
- The precession matrix, **P**, similarly, is also comprised of three rotations $\mathbf{R}_3(-z)\mathbf{R}_2(\theta)\mathbf{R}_3(-\zeta)$, where z, θ, and ζ are the combined effects of the precession of the rotation axis.
- Finally, the polar motion matrix, **W**, which is based off the vector $\boldsymbol{p} = [x,\ y,\ 0]^T$. Representing the movement in the ECEF cardinal directions. As there can be no movement in the $\hat{\boldsymbol{k}}$ direction, that is set to zero. This vector is then turned into a skew-symmetric matrix, $\mathbf{W} = [\boldsymbol{p}]_x$.

These rotations are then are then combined in order of operations to build the final **D**irectional **C**osine **M**atrix (DCM) to convert a ECEF vector quantity to a ECI IAU-1980/FK5 vector quantity.

$$\mathbf{DCM}_{ECI\rightarrow ECEF} = \mathbf{WRNP} \tag{6}$$

$$\boldsymbol{r}_{ECEF} = \mathbf{DCM}_{ECI\rightarrow ECEF}\,\boldsymbol{r}_{ECI}$$

$$\mathbf{DCM}_{ECEF\rightarrow ECI} = (\mathbf{WRNP})^T \tag{7}$$

$$\boldsymbol{r}_{ECI} = \mathbf{DCM}_{ECEF\rightarrow ECI}\,\boldsymbol{r}_{ECEF}$$

Converting States to Measurements

To begin the transformation from state to measurement, we must first find the station of interest, and transform its coordinates into the ECI frame using the DCM above. From there we can find the range and range-rate with the following equations:

$$\rho = \|\boldsymbol{r}_{sc} - \boldsymbol{r}_{st}\| \tag{8}$$

$$\dot{\rho} = \frac{(\boldsymbol{r}_{sc} - \boldsymbol{r}_{st})^T (\boldsymbol{v}_{sc} - \boldsymbol{v}_{st})}{\rho} \tag{9}$$

Although, these are not the final measurements, as they do not take into account the time for the signal to bounce from the spacecraft back to the station. We can find the signal light-time using the given data range measurement and dividing it by the speed of light (ignoring any effects the atmosphere has on the speed of light). We then backpropagate the spacecraft's position from its current one by the light time to get our update state. As well we must also update the station position by the same metric to get our light-time correct state measurements.

Filter Setup

For filtering the measurements to estimate the spacecraft's state, I implemented an **E**xtended **K**alman **F**ilter (EKF), with the following state:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{v} \end{bmatrix}_{6\times 1} \tag{10}$$

Other parameters such as station range biases and spacecraft C_D were estimated separately from the Kalman filter. It should be noted that there is no post-processing/smoothing after the filter runs its course.

Extended Kalman Filter Setup

As a brief overview, the EKF propagates an estimate of the state forward using the dynamics described above. Along with the state, the linearized dynamics are used to propagate the STM, $\mathbf{F}_k$. The symbolic representation of the linearized dynamics, $\mathbf{A}$ is available in the Appendix[7].

$$\begin{aligned} \text{Extrapolation:} \quad \boldsymbol{x}_{k+1|k} &= \boldsymbol{f}\left(\boldsymbol{x}_{k|k}\right) \\ \mathbf{P}_{k+1|k} &= \mathbf{F}_k \mathbf{P}_{k|k} \mathbf{F}_k^T + \mathbf{\Gamma Q \Gamma}^T \end{aligned} \tag{11}$$

Due to the inability to perfectly model the dynamics the spacecraft is under, we must implement a certain level of process noise to "mask" the uncertainty in our propagation steps. In addition, our time between measurements isn't consistent, while being tracked by the station, the satellites range and range-rate is recorded every 60 seconds. Time between stations is variable due to the locations of stations on the rotating Earth under the spacecraft's inertial orbit. To compensate for this, we must include a $\mathbf{\Gamma}$ matrix:

$$\mathbf{\Gamma Q \Gamma}^T = \Delta t^2 \begin{bmatrix} \frac{\Delta t^2}{4}\sigma_{\ddot{R}}^2 & 0 & 0 & \frac{\Delta t}{2}\sigma_{\ddot{R}}^2 & 0 & 0 \\ 0 & \frac{\Delta t^2}{4}\sigma_{\ddot{I}}^2 & 0 & 0 & \frac{\Delta t}{2}\sigma_{\ddot{I}}^2 & 0 \\ 0 & 0 & \frac{\Delta t^2}{4}\sigma_{\ddot{C}}^2 & 0 & 0 & \frac{\Delta t}{2}\sigma_{\ddot{C}}^2 \\ \frac{\Delta t}{2}\sigma_{\ddot{R}}^2 & 0 & 0 & \sigma_{\ddot{R}}^2 & 0 & 0 \\ 0 & \frac{\Delta t}{2}\sigma_{\ddot{I}}^2 & 0 & 0 & \sigma_{\ddot{I}}^2 & 0 \\ 0 & 0 & \frac{\Delta t}{2}\sigma_{\ddot{C}}^2 & 0 & 0 & \sigma_{\ddot{C}}^2 \end{bmatrix} \tag{12}$$

Since our dynamics are predominantly affecting the spacecraft relative to the RIC frame (in-track with drag, etc.), it would be much more effective to model our state noise compensation also in that frame. When propagating forward, this matrix is then rotated back into ECI before being added to our covariances. A majority of our uncertainty

in determining the satellites orbit is in measuring its C_D, while the other terms like SRP, gravity, and third-body effects are more well understood and modelled. As such, our $\sigma_{\ddot{I}}$ will be higher as a result[8]. Unless stated otherwise the process noise for all results will have the following uncertainties:

Direction	Uncertainty
$\sigma_{\ddot{R}}$	10^{-16} km/s^2
$\sigma_{\ddot{I}}$	10^{-12} km/s^2
$\sigma_{\ddot{C}}$	10^{-16} km/s^2

This extrapolated state is then converted into a measurement and compared against the given data. This difference, $\boldsymbol{y}_{k+1|k}$, is the pre-fit residual.

$$\begin{aligned} \text{Comparison:} \quad \boldsymbol{z}_{k+1|k} &= \boldsymbol{h}\left(\boldsymbol{x}_{k+1|k}\right) \\ \boldsymbol{y}_{k+1|k} &= \boldsymbol{z}_{k+1|k} - \boldsymbol{z}^a_{k+1} \end{aligned} \tag{13}$$

Using the pre-fit residual, and the measurement sensitivity matrix, $\mathbf{H}_{k+1}$, which is the Jacobian of the measurement function, $\boldsymbol{h}(\boldsymbol{x})$, with respect to the state, we can generate the innovation covariance, $\mathbf{S}_{k+1}$, and Kalman Gain, $\mathbf{K}_{k+1}$. These are used to update our extrapolated state and covariance to become more in line with the measurements. This updated state is then passed through the measurement function again, and compared to find the post-fit residual, $\boldsymbol{y}_{k+1|k+1}$.

$$\begin{aligned} \text{Update:} \quad \mathbf{S}_{k+1} &= \mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}^T_{k+1} + \mathbf{R} \\ \mathbf{K}_{k+1} &= \mathbf{P}_{k+1|k}\mathbf{H}^T_{k+1}\mathbf{S}^{-1}_{k+1} \\ \boldsymbol{x}_{k+1|k+1} &= \boldsymbol{x}_{k+1|k} + \mathbf{K}_{k+1}\boldsymbol{y}_{k+1|k} \\ \mathbf{P}_{k+1|k+1} &= (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1})\mathbf{P}_{k+1|k} \\ \boldsymbol{z}_{k+1|k+1} &= \boldsymbol{h}\left(\boldsymbol{x}_{k+1|k+1}\right) \\ \boldsymbol{y}_{k+1|k+1} &= \boldsymbol{z}_{k+1|k+1} - \boldsymbol{z}^a_{k+1} \end{aligned} \tag{14}$$

Bias Estimation

Next was to estimate the bias of each station, to do this we first checked the range residuals assuming no bias within the system. Looking at the mean of the resulting error, we can adjust the biases until this value becomes less than that of the stations range measurement uncertainty. This is done while setting the process noise, $\mathbf{Q}$, to zero to show the true effects of the implemented dynamics, and to avoid the process noise masking the actual bias of the stations.

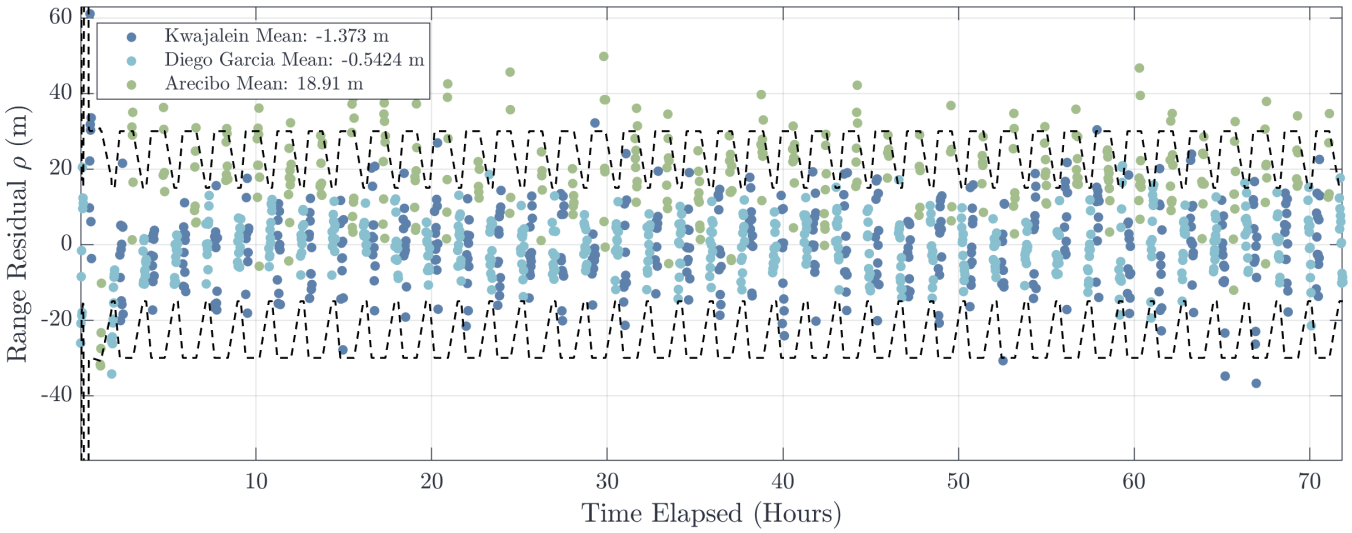


Figure 4: Bias assumed to be zero with 3 days worth of data, Arecibo shows large difference from a zero mean white noise.

After a couple of iterations, we came to the conclusion that only Arecibo was in need of bias adjustments, as both Diego Garcia and Kwajalein did not have a mean residual of larger than their measurement uncertainty. With Arecibo adjusted to a bias of +18.9 m, the post-fit residuals became the following:

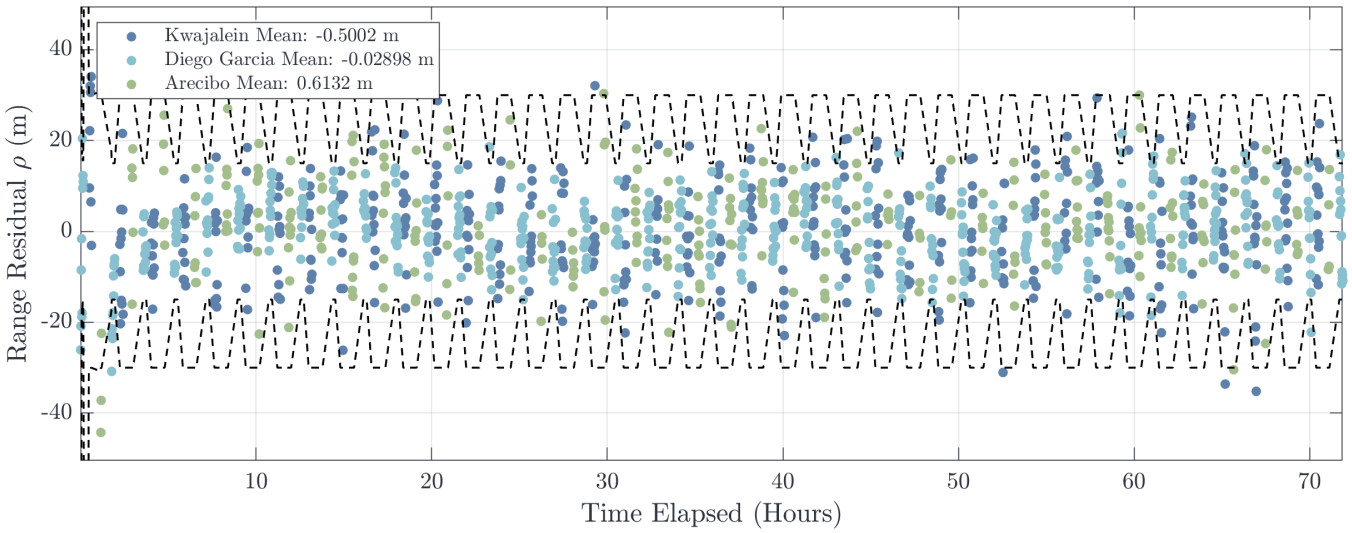


Figure 5: Biases adjusted with 3 days worth of data, Arecibo (and other stations) show reduction in their overall post-fit residuals.

Drag Estimation

The approach for C_D estimation is similar to bias estimation. Turning process noise to zero, and reviewing the resultant residuals proved to be a good method of estimating drag without needing to include it in the state. Starting with the initial $C_D = 1.88$, we saw that the range-rate residual data begin to slowly creep away from zero-mean, in a positive direction showing that we are underestimating the effects of drag on the spacecraft. This is since residuals are calculated as $\boldsymbol{y} = \boldsymbol{z}_{est} - \boldsymbol{z}_{act}$. Figure 6-Figure 8 show the effects of increasing drag from $C_D = 1.88$ through to $C_D = 2$. Moving to $C_D = 1.95$ showed a positive improvement in residuals, but still had a departure from zero mean. While $C_D = 2$ was an over-correction where now the spacecraft's range-rate residuals shifting in the negative direction, meaning the spacecraft is now moving slower than measured.

After multiple filter runs, we settled on a $C_D = 1.975$, in Figure 9. This value minimized the range and range-rate residuals mean, and not showing any advanced "creep". A note for the range-rate residuals is that all stations seem to have an oscillation in there value. With a period of 1-day, the source of this error is most certainly the ECEF to ECI transformation not exactly matching the simulated data. Primary culprit is how the EOP data is interpolated between dates, in the Appendix, we show the linear interpolation between data points.

This increase in C_D, roughly 5%, can be attributed either a bad initial estimate, or show that the spacecraft's attitude is not as what was assumed in the Drag section. The spacecraft could be tumbling or have been re-oriented, showing a different face to the atmosphere.

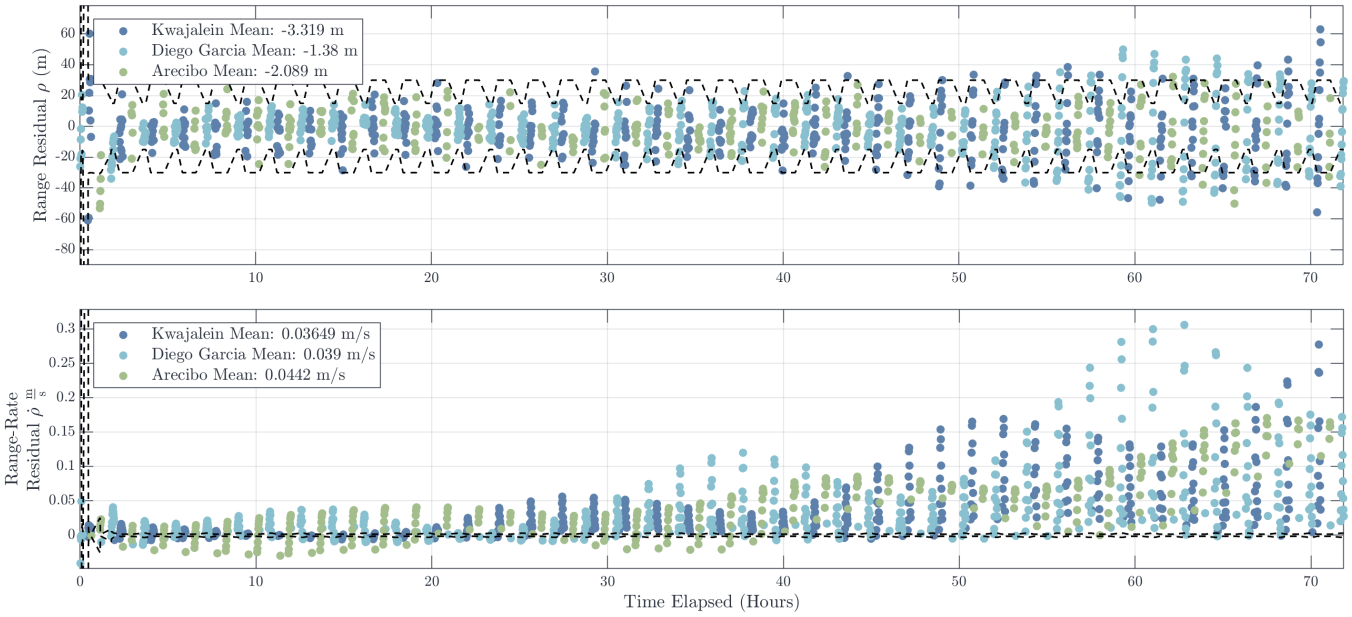


Figure 6: $C_D = 1.88$ and $\mathbf{Q} = \mathbf{0}$. The range-rate residuals show the spacecraft is moving faster than the measurements say it does.

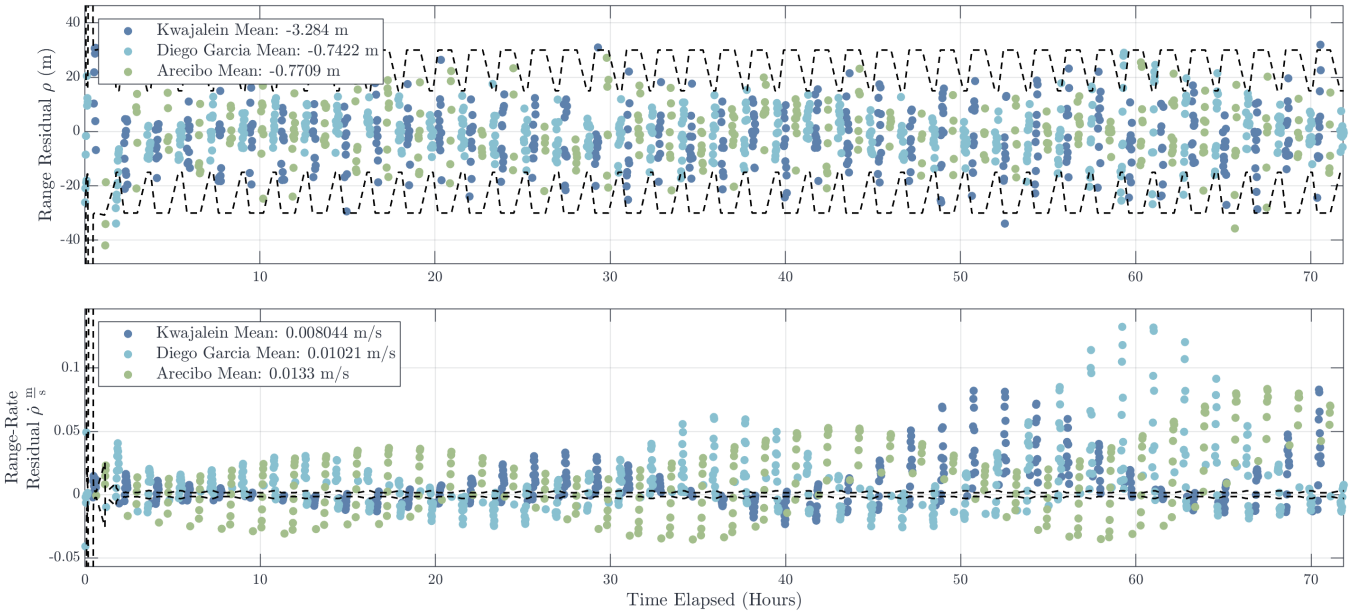


Figure 7: $C_D = 1.95$ and $\mathbf{Q} = \mathbf{0}$. Adjusting to a higher C_D, has reduced overall residuals, but still shows an offset in velocity.

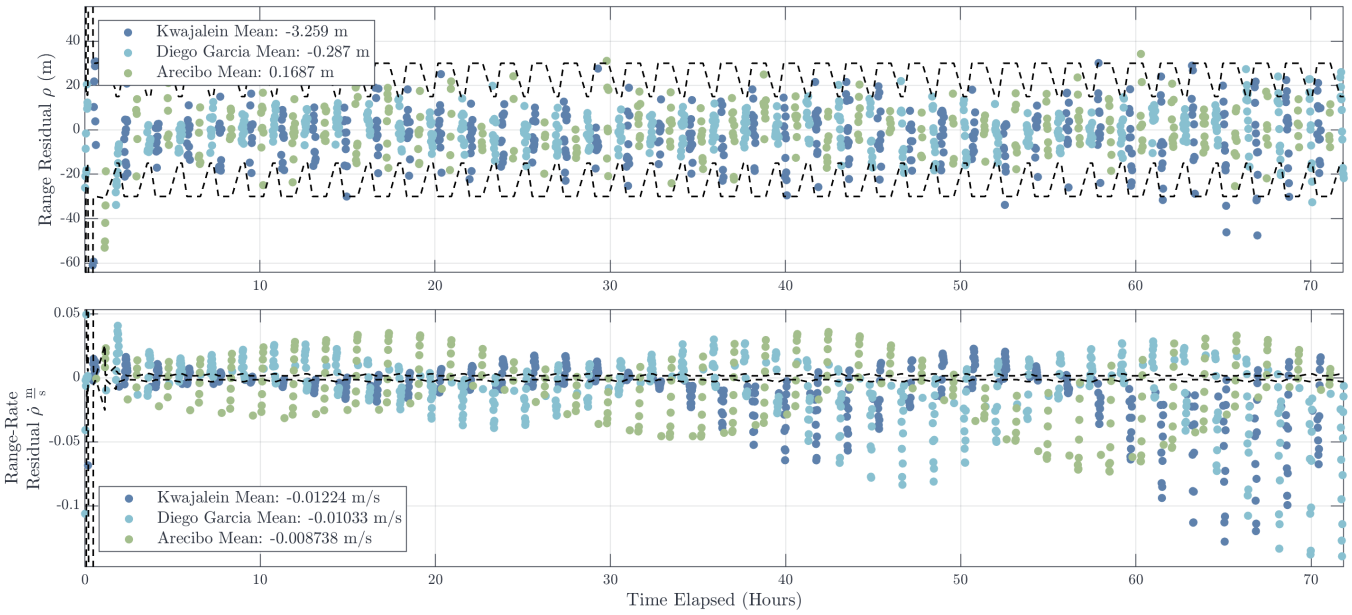


Figure 8: $C_D = 2$ and $\mathbf{Q} = \mathbf{0}$. Increasing to 2, shows an over-correction in the drag coefficient of the spacecraft.

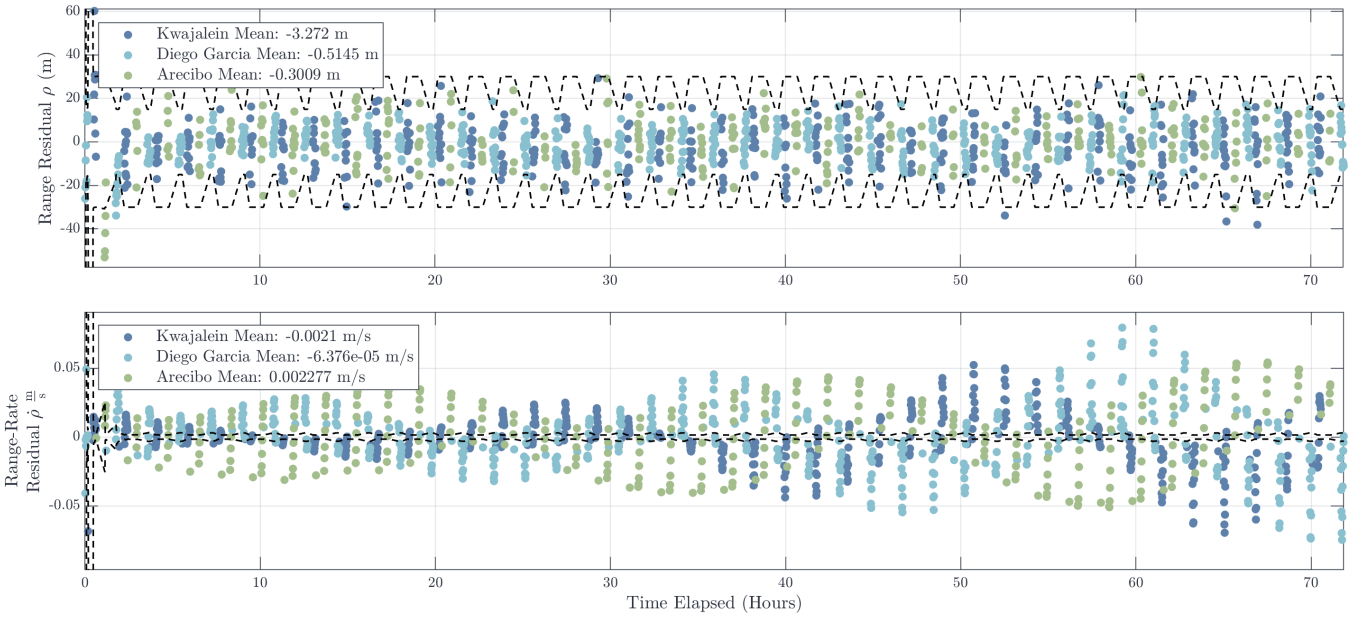


Figure 9: $C_D = 1.975$ and $\mathbf{Q} = \mathbf{0}$. While the range-rate residuals still grow overtime, they remain roughly zero mean and shows that 1.975 is a relatively good estimate of the spacecraft's C_D.

Results

Running the filter with the above models, we can compare our residuals, both pre and postfits, considering the following cases:

- **Case A**: Only range sensors are considered
- **Case B**: Only range-rate sensors are considered
- **Case C**: Only measurements from Kwajalein are considered
- **Case D**: Only measurements from Diego Garcia are considered
- **Case E**: Only measurements from Arecibo are considered
- **Case F**: All stations and sensors are available
- **Case G**: The initial state and covariance are propagated to the last days worth of measurements, then all stations and sensors are available

The sensors and stations that are not considered within each case arbitrarily have their measurement noise increased to $\sigma = 10^8$, to avoid any contribution to the filter. In the residual plots, the inflated noise will not show in the 3-sigma bounds of each plot. First we will show all plots, then all discussion and interpretation of results will be in the Discussion. Also any plot not easily visible will have links to zoom ins (located in the Appendix), in their captions.

Pre-Fit Residuals

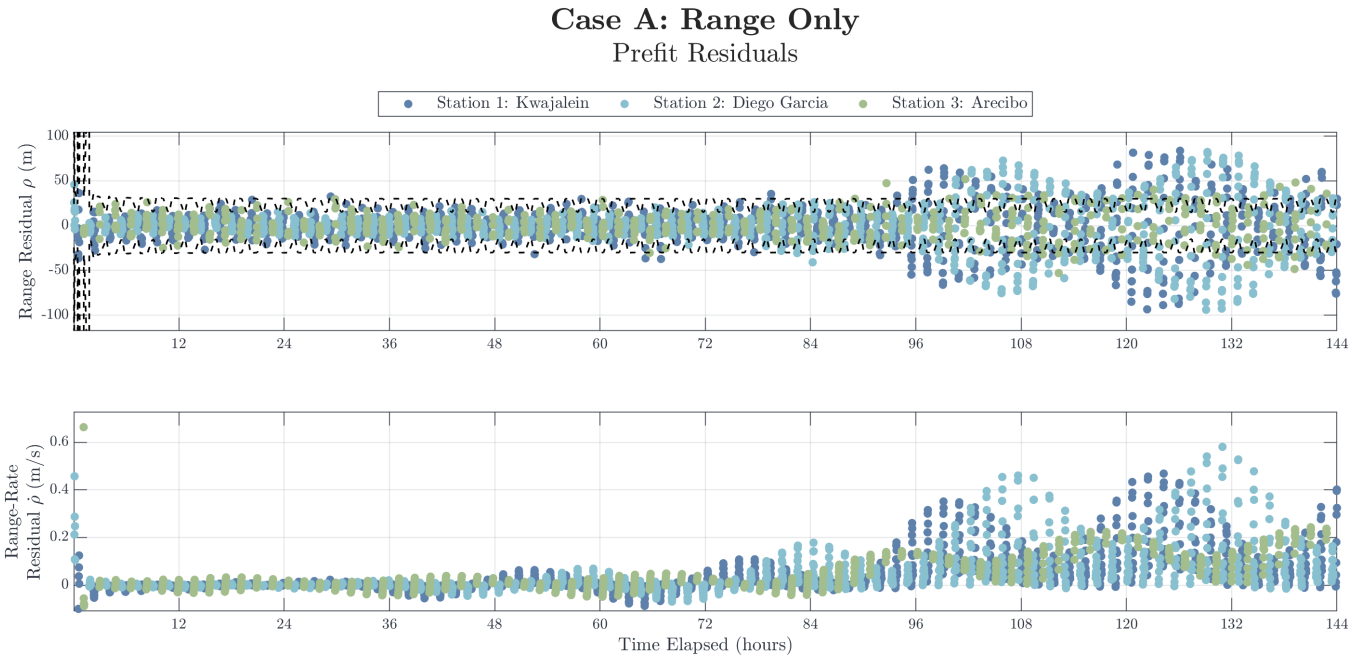


Figure 10: Prefit residuals for the range sensors only case, filter begins to diverge, mildly, after 4 days of data.

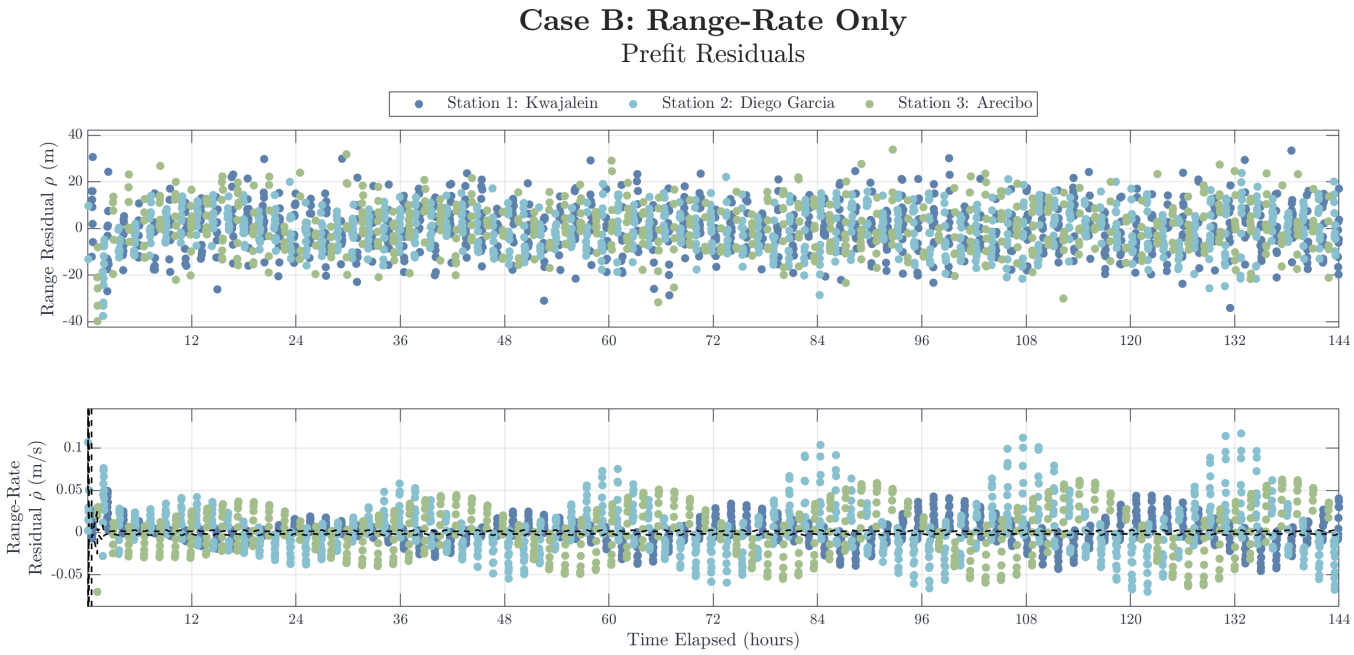


Figure 11: Prefit residuals for the range-rate sensors only case, filter performs well compared to **Case A**.

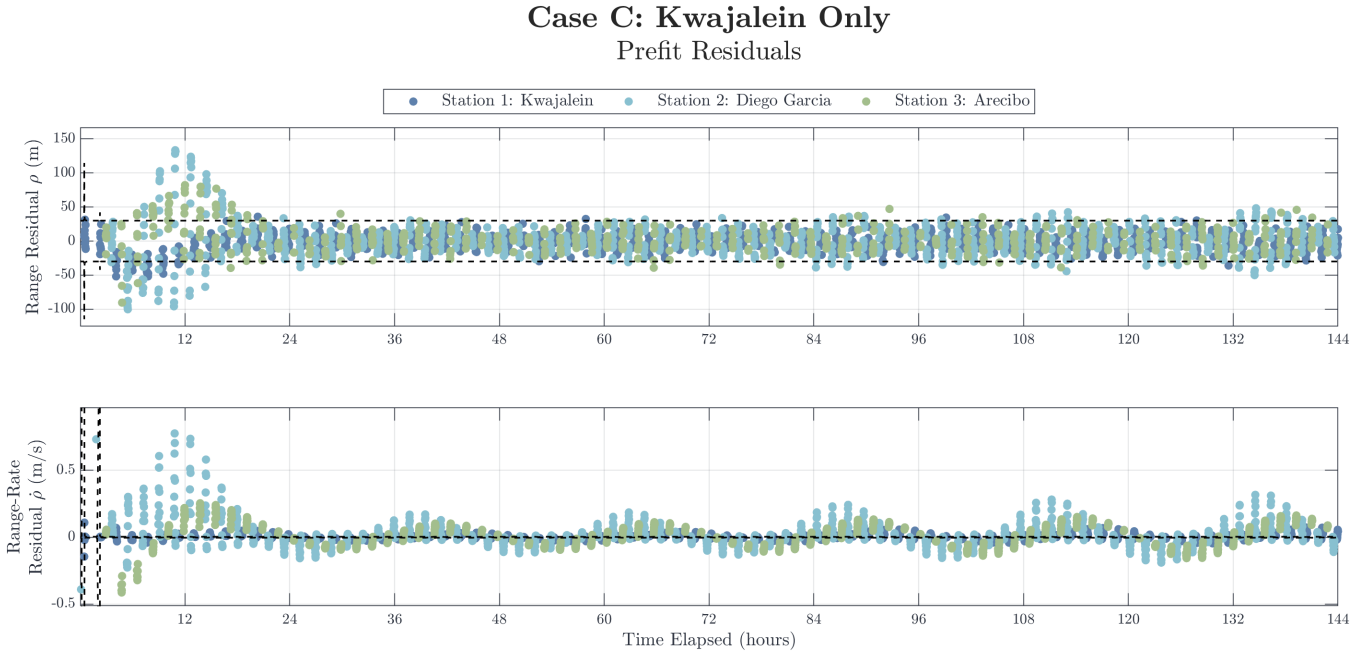


Figure 12: Prefit residuals for the Station 1 case, after an initial hiccup, the filter becomes relatively confident of its solution.

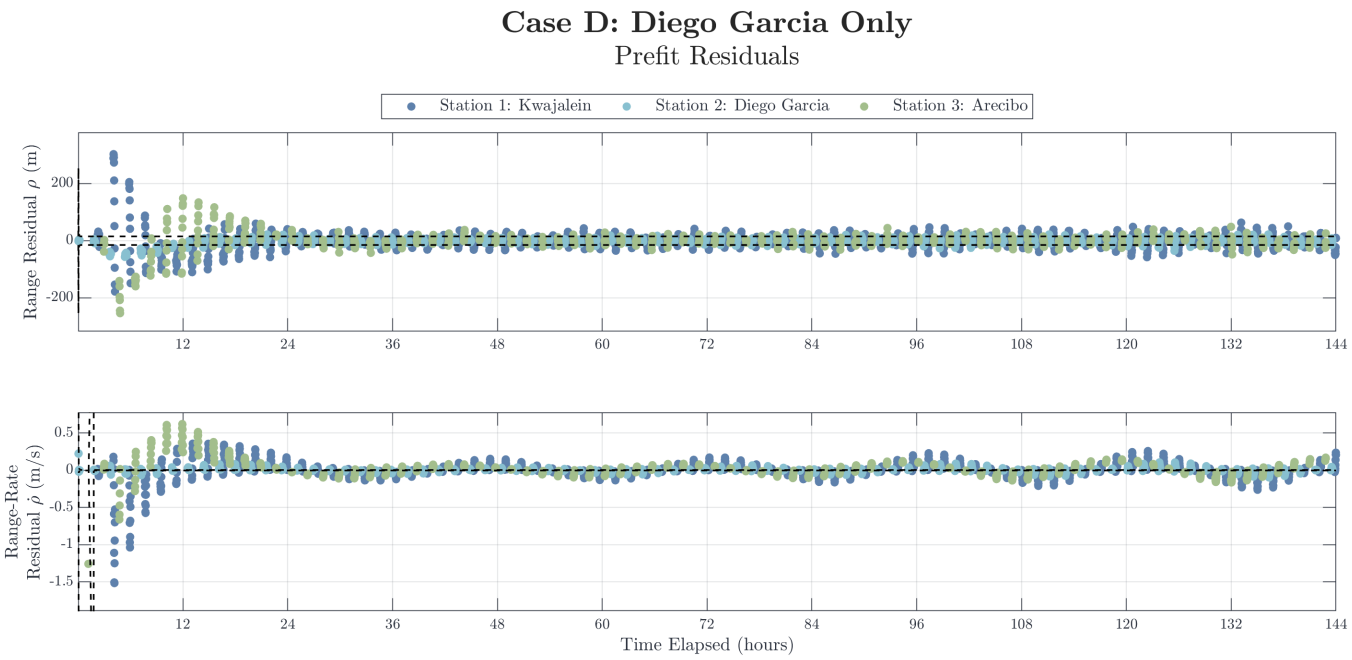


Figure 13: Prefit residuals for the Station 2 case, worst performing case relative to all others. Zoom-in in Appendix

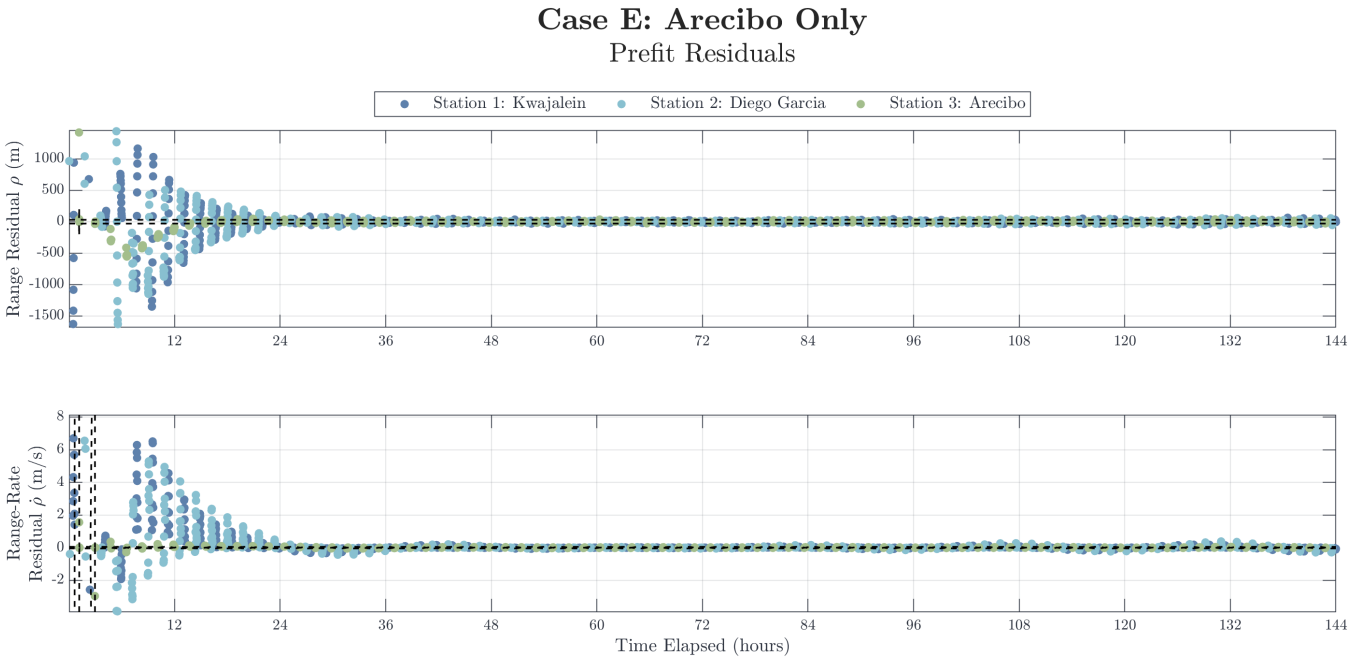


Figure 14: Prefit residuals for the Station 3 case, similar to Case C, after an initial hiccup, the filter becomes relatively confident of its solution. Due to shorter view times, this takes longer. Zoom-in in Appendix

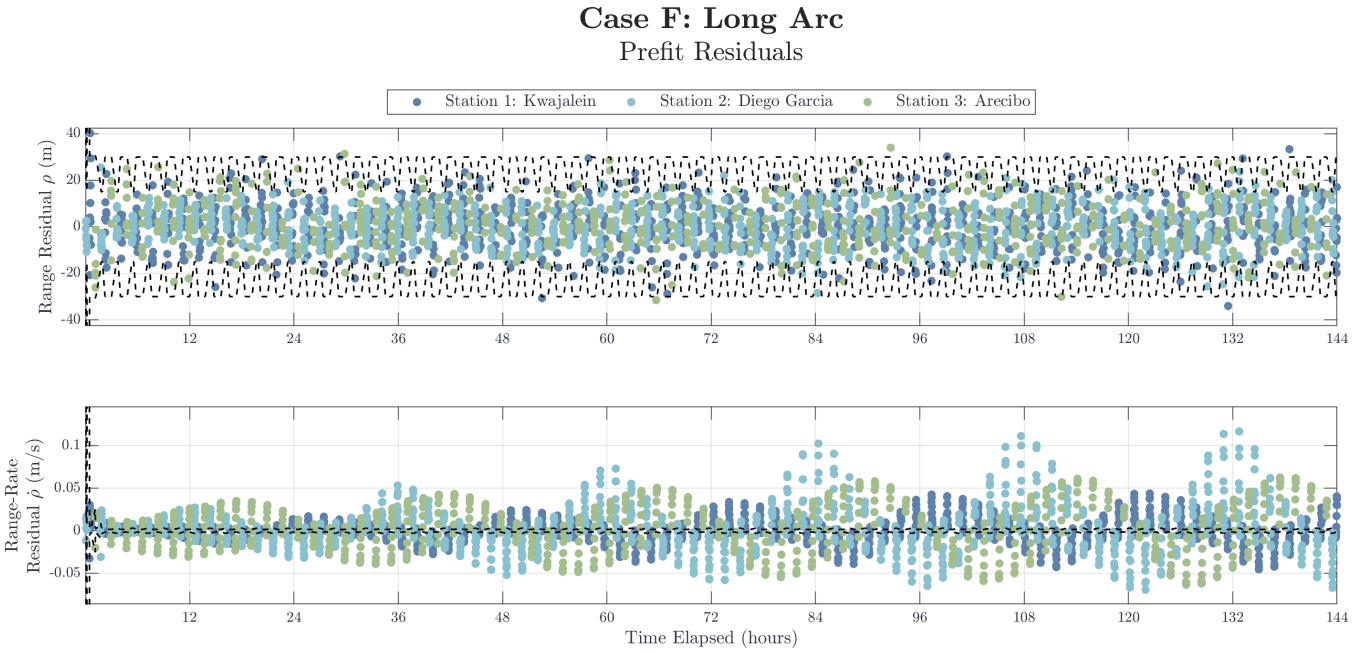


Figure 15: Prefit residuals for all data case, lines up well with Case B, with very good consistency.

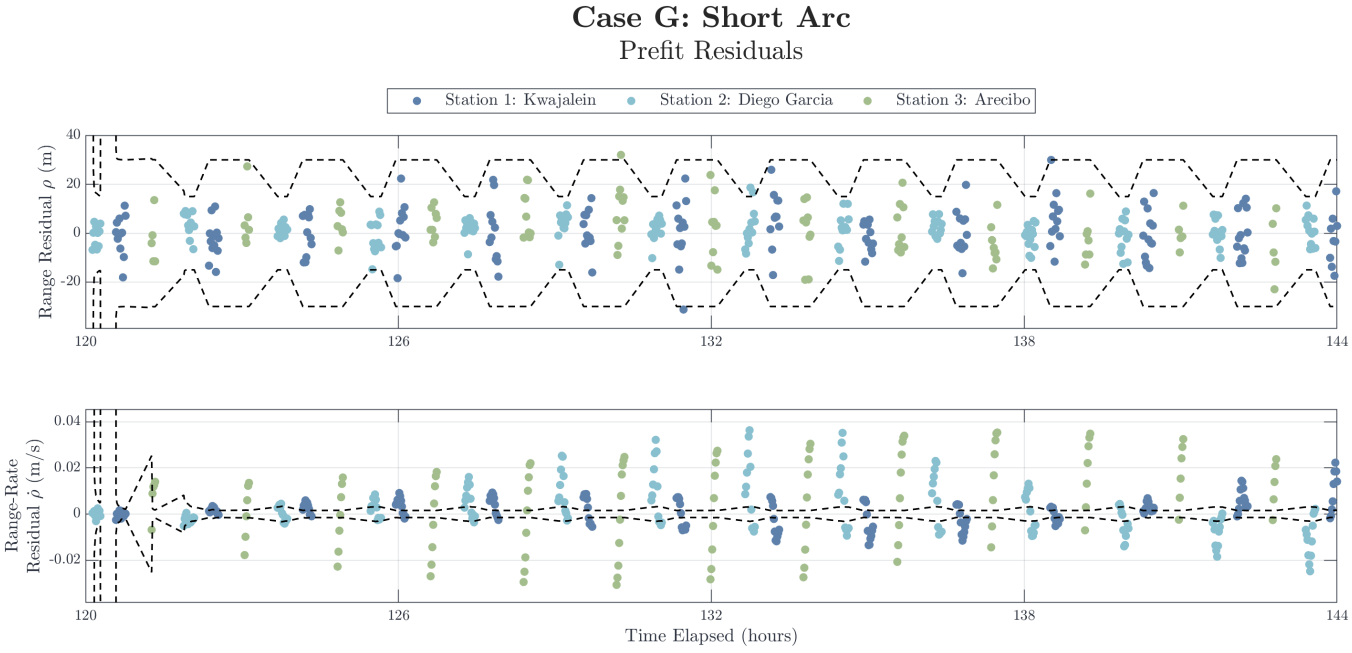


Figure 16: Prefit residuals for 1 day data case, even with propagated initial state, filter quickly figures out where to place the spacecraft.

Post-Fit Residuals

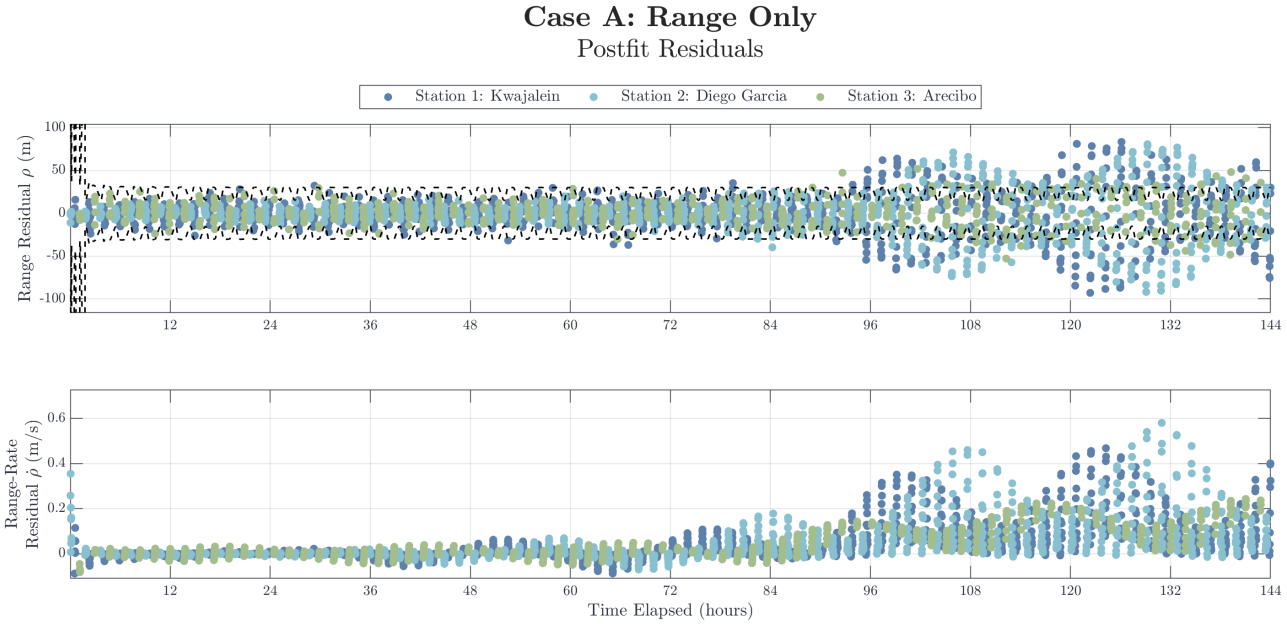


Figure 17: Postfit residuals for the range sensors only case, filter begins to diverge, mildly, after 4 days of data. With cleaner earlier residuals than prefit.

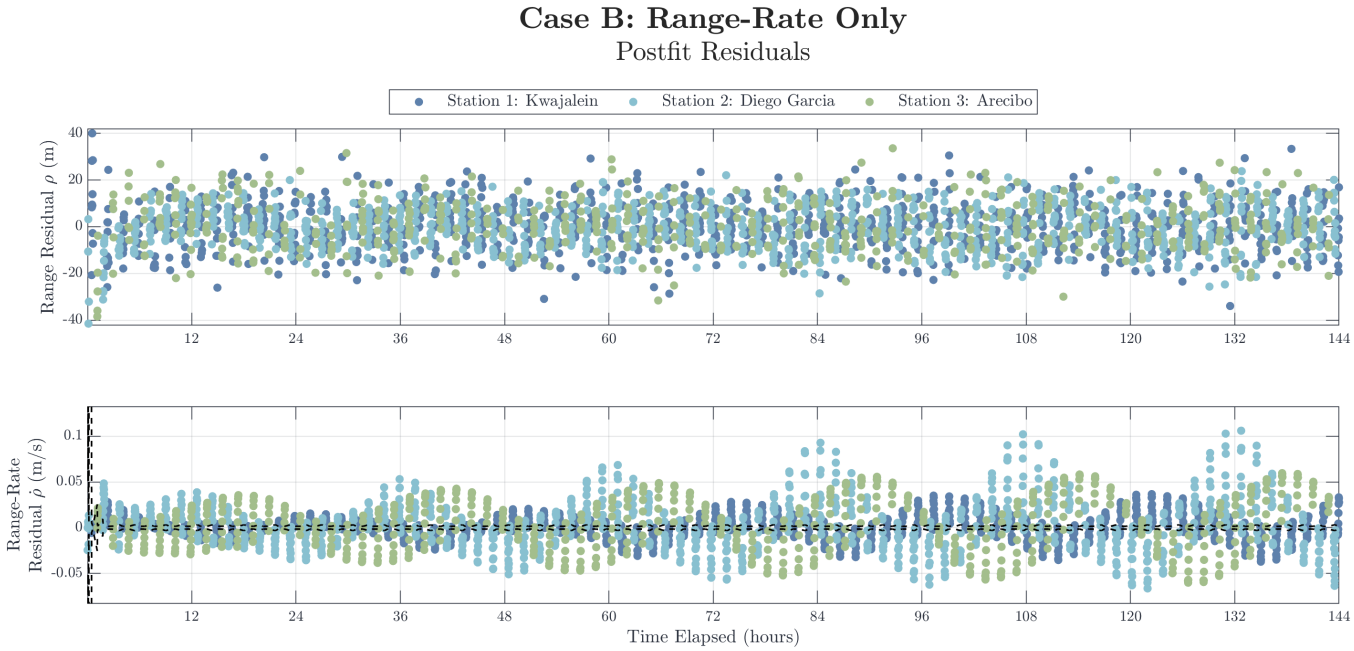


Figure 18: Postfit residuals for the range-rate sensors only case, filter performs well against all results. With cleaner earlier residuals than prefit.

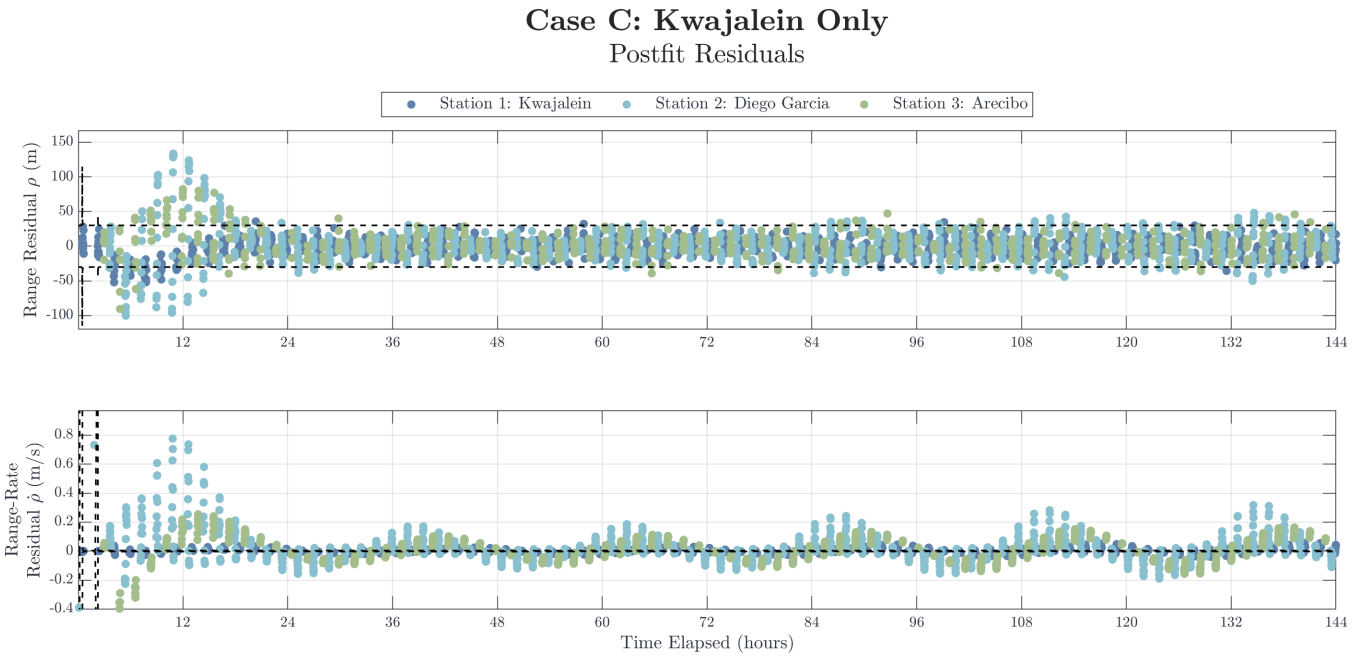


Figure 19: Postfit residuals for the Station 1 case, after an initial hiccup, the filter becomes relatively confident of its solution.

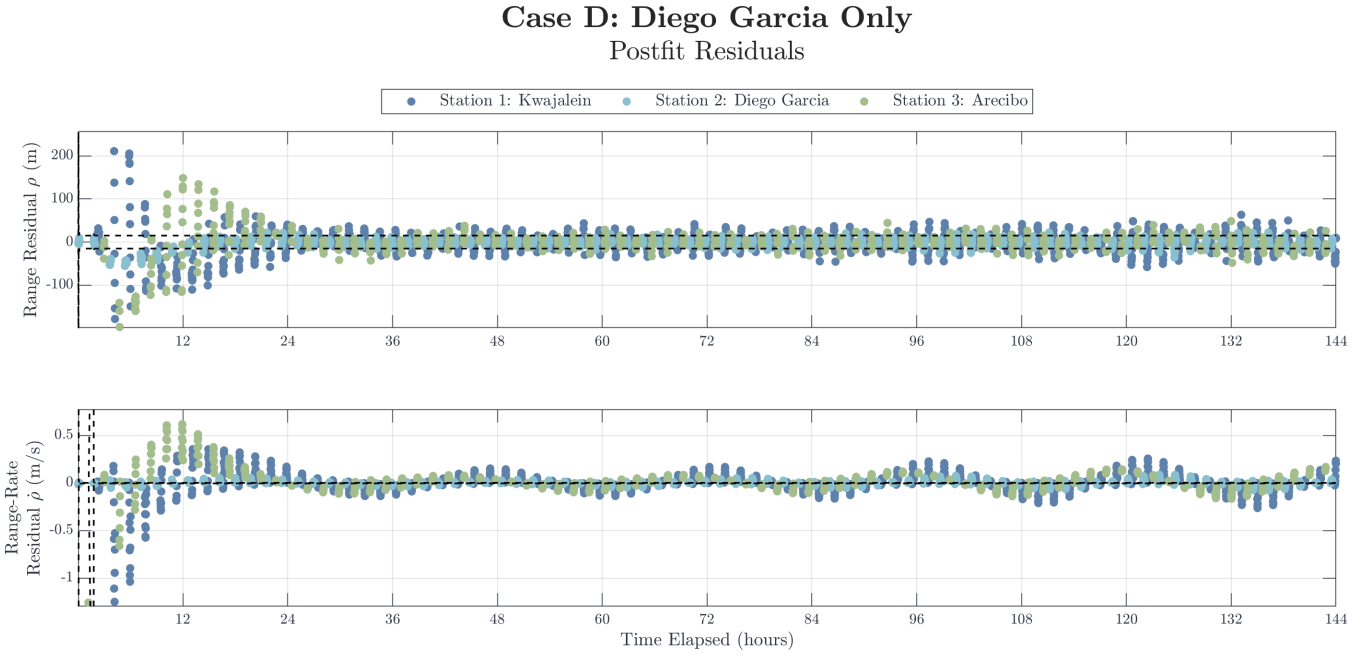


Figure 20: Postfit residuals for the Station 2 case, worst performing case relative to all others. Residuals more well behaved than prefit counterparts.

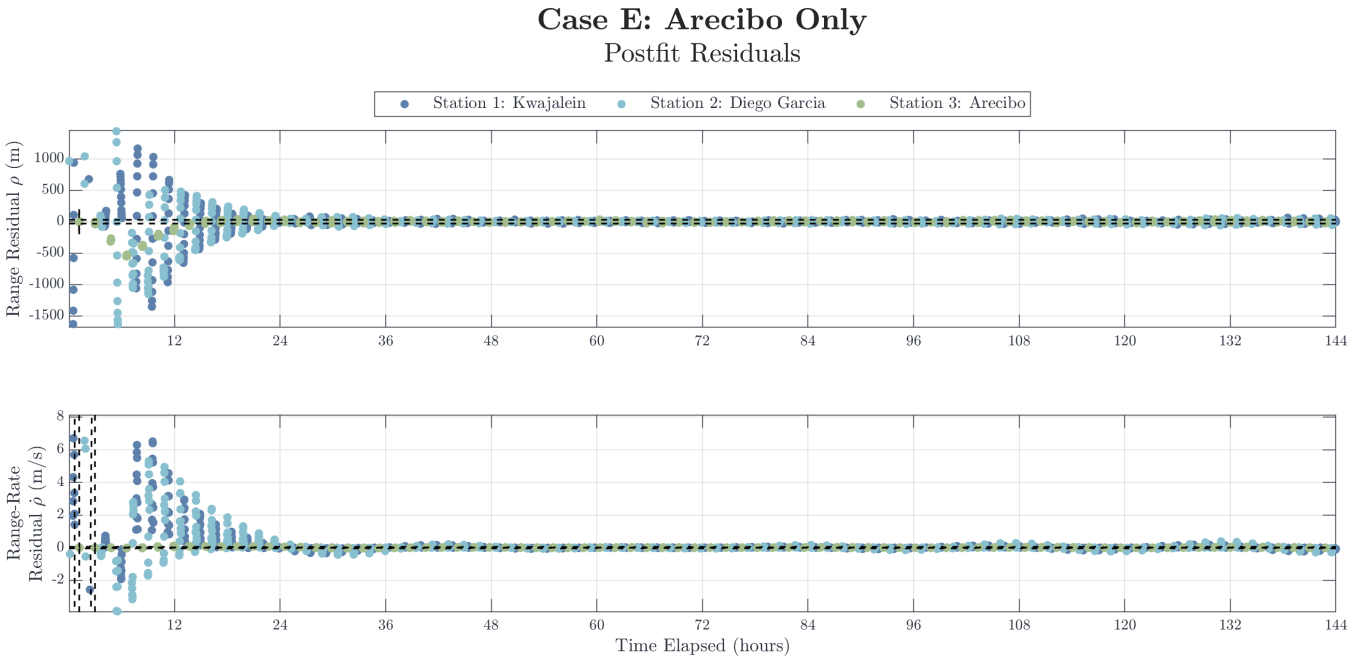


Figure 21: Postfit residuals for the Station 3 case, similar to Case C, after an initial hiccup, the filter becomes relatively confident of its solution. Due to shorter view times, this takes longer. Zoom-in in Appendix

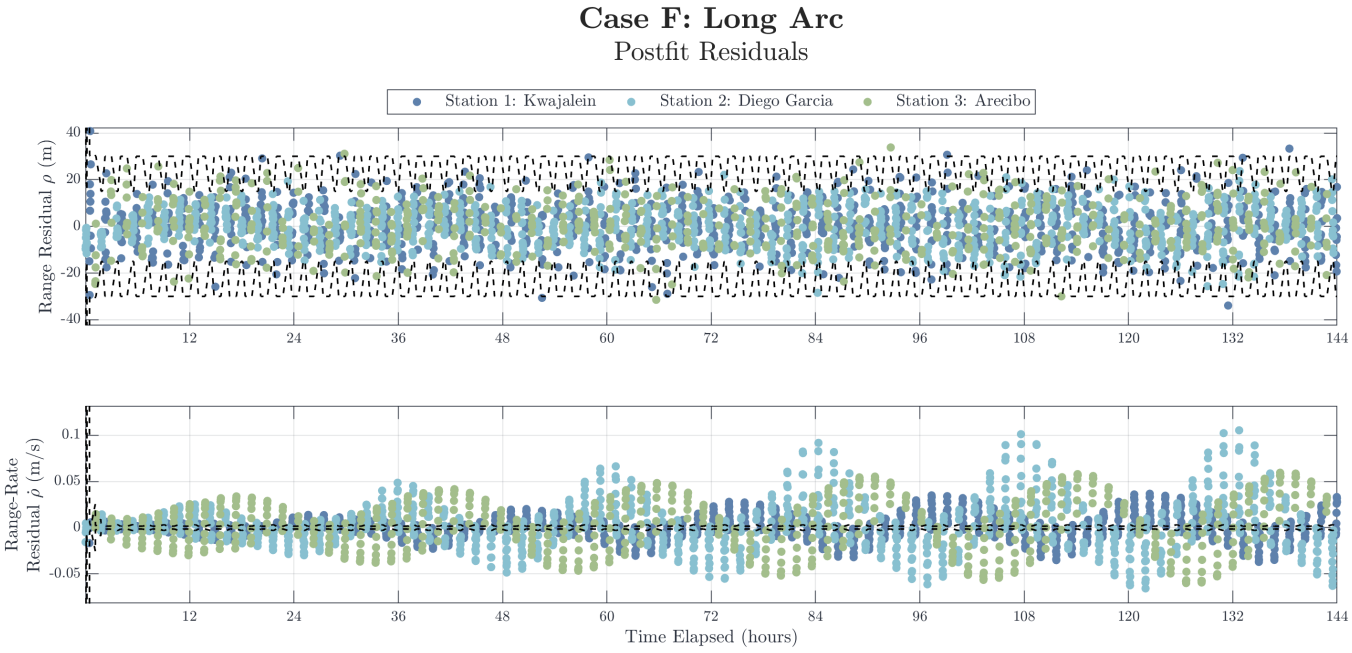


Figure 22: Postfit residuals for all data case, lines up well with Case B, with very good consistency.

Figure 23: Postfit residuals for 1 day data case, even with propagated initial state, filter quickly figures out where to place the spacecraft.

Error Ellipses

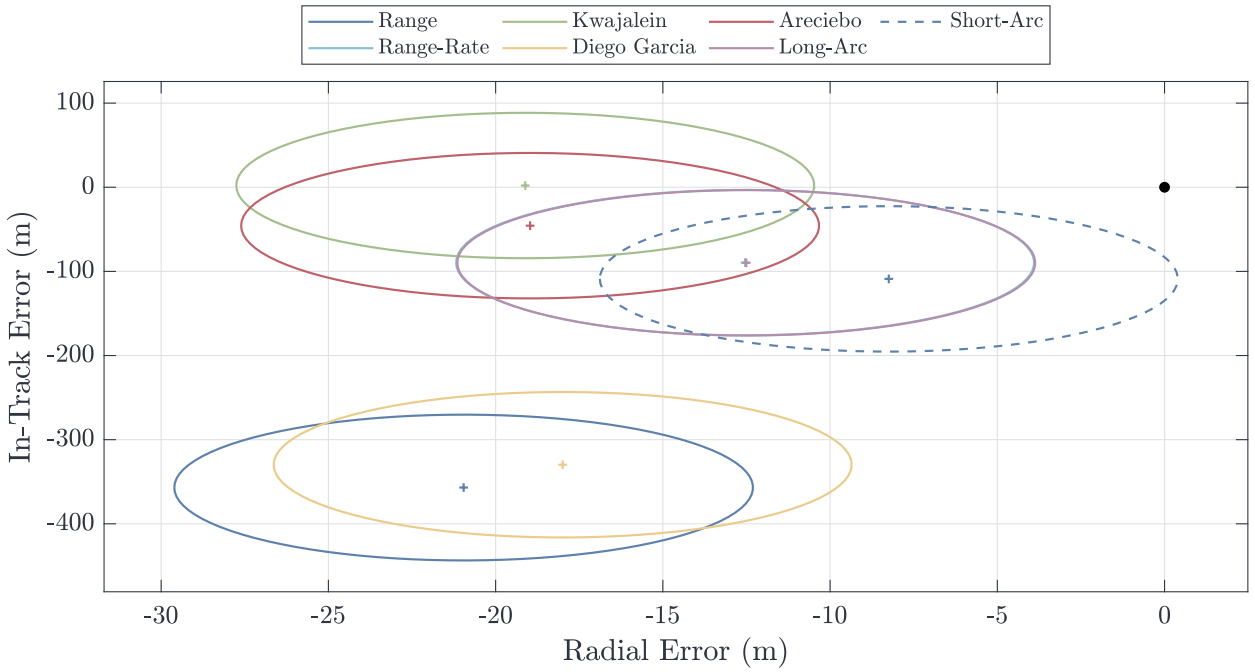


Figure 24: Radial and in-track very close to approximated truth (estimated from NAG results), apart from cases A and D. See discussion for reasoning.

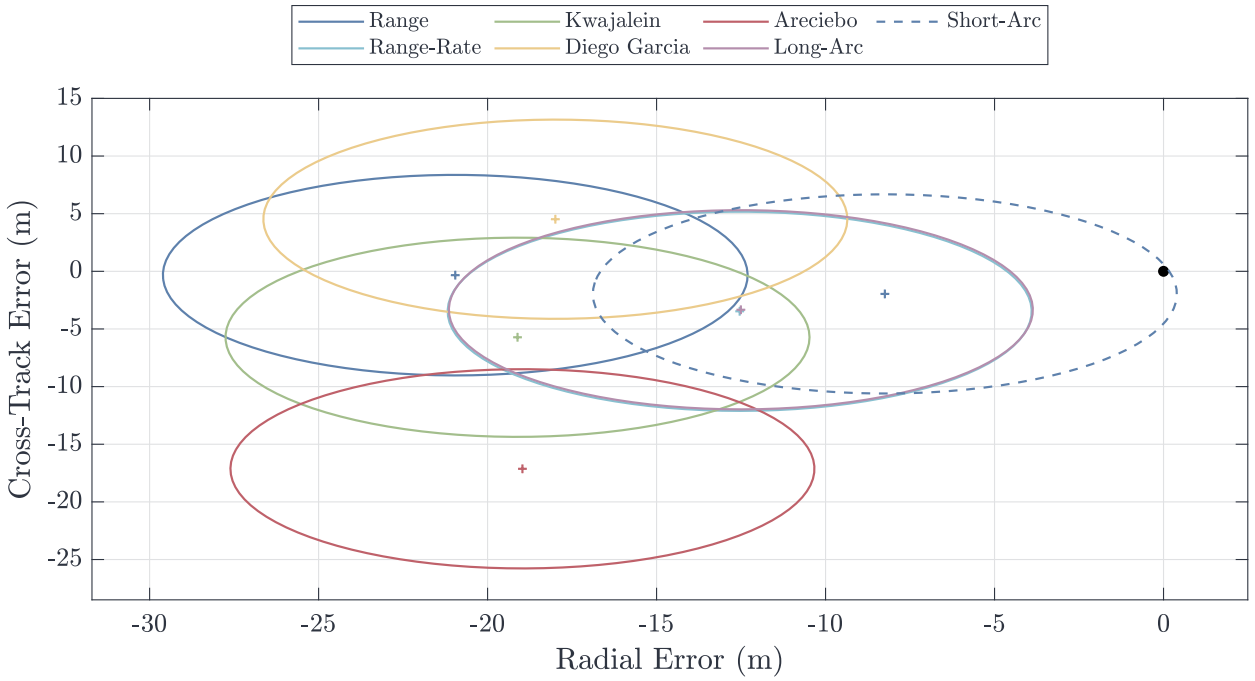


Figure 25: All results nearly within 20 m of approximated truth, pointing to correct dynamics in these directions.

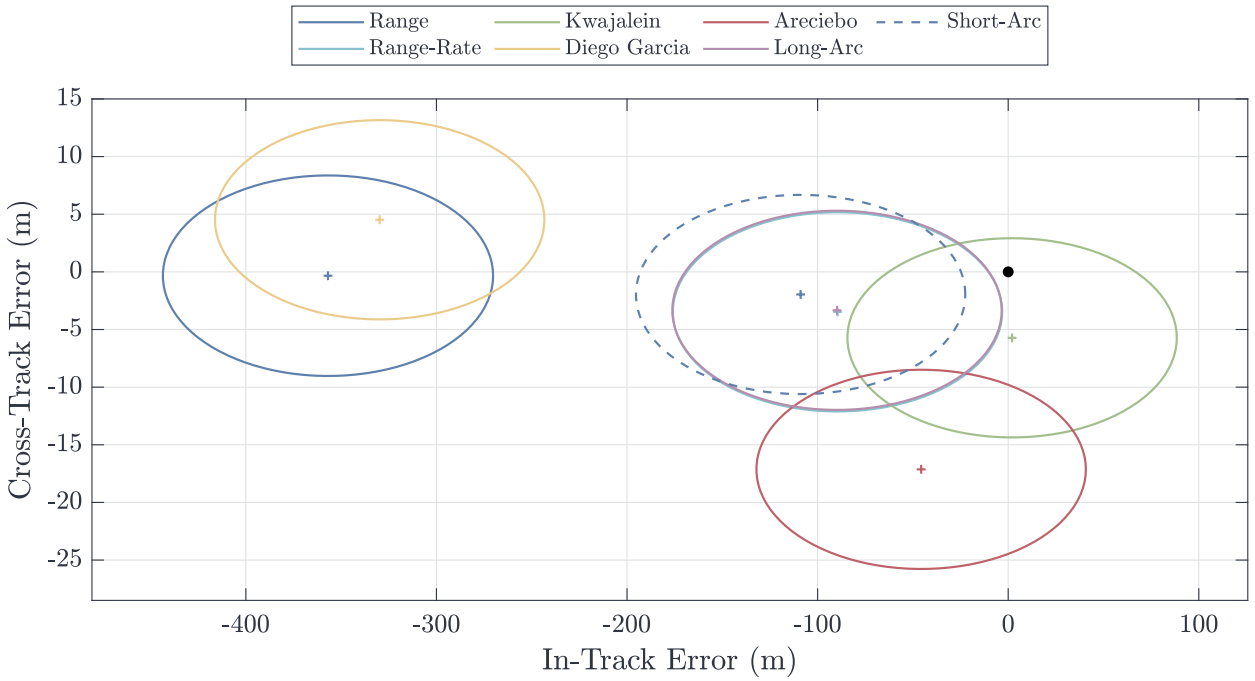


Figure 26: Once again well behaved results apart from cases A and D.

Discussion

In the filtering stage, all cases perform relatively consistent, with an average dispersion of the final position of ±30 m and a largest difference of 80 m. This difference is due to Diego Garcia and the Range only measurements, without them, the average becomes ±15 m. Even with this in mind, overall the residuals for all cases are very close to the 3-sigma measurement noise of each sensor.

Similarly residuals overall showed good performance relative to the noise floor of each sensor. Range, for most cases, was well within the 3-sigma bounds, while range-rate demonstrated an oscillation about zero. As stated previously, this is most likely due to a modelling mismatch in the ECEF to ECI rotations. For the most part this difference is on the order of millimeters per second, while still above the noise floor of the range-rate sensors, is still a good result.

Case A The range only case demonstrated one of the larger challenges in estimating the spacecraft's state. Since the range calculation does not include any velocity data from the spacecraft, the filter had low observability in half of the states, which, as we can see, lead to some divergence towards the end of the filtering run, and one of the largest delivery ellipses off. With more data this might have reconverged to a better estimate, but that's difficult to determine here. Also, due to the low measurement noise of Diego Garcia, the filter was more inclined to believe the estimate that station was giving, which is why the delivery ellipses are skewed towards the Diego Garcia only result (as to why that's wrong will be covered in Case D).

Case B In stark contrast to Case A, the range-rate only data allowed the filter to very easily find a good estimate of the spacecraft. For a similar reason as why range only was difficult, the range-rate data includes both the position and velocity state of the spacecraft in the measurement, allowing for good observability of all state elements. This also most likely benefitted from those sensors also being zero-mean, giving an unbiased measurement. As well since these sensors have very high accuracy, the Case F filter with all data was highly inclined to believe the range-rate only case, and can be seen as right on top of each other in the delivery ellipses.

Case C Surprisingly, Kwajalein proved to be one of the better stations at estimating the spacecraft's delivery position. After some investigation, this station is the last one to see the spacecraft at the end of the dataset, so there is less propagation time in total to the Δv_1 epoch. Additionally, being the second station to initially view the spacecraft caused instability during the first day of residuals, but as more data came in, the filter was handily able to identify the spacecraft's state.

Case D Easily the most troubled case, which is counter intuitive due to Diego Garcia's smaller range measurement noise, but also has a larger range-rate measurement noise. Viewing previous results, it seems that the range-rate data performs much better than the range data (due to the state observability), so it could be one possible reason as to why this estimate was the largest off. While it does not perform terribly, the filter is far too confident in its estimate even with the residuals taken into account.

Case E Apart from cross-track, this station was very well performing, and a likely source for that error is Arecibo's latitude being the highest of all stations. Even though this is not reflected in the residuals, another main concern for the propagation to delivery was that that Arecibo was the station with the last measurement farthest from the delivery epoch, so any small errors in final state were only exaggerated.

Case F Taking into account all sensors and data, results more or less match those given by the range-rate only sensors case, which makes sense, as described in that section. Simply put, range-rate is more information dense in regards to the spacecraft's state for a given measurement period, compared to range only. We see the delivery ellipse is skewed slightly towards the range only case due to this additional data nonetheless.

Case G The short-arc case proved to be the most interesting. Using the backpropagated estimate from Case F as an initial guess, the state and covariance was propagated forward 5 days, to the last day's worth of data. Of all cases, this one is the closest in terms of radial and cross-track information, but skewed farther for in-track. Looking at the range-rate residuals, we can see whenever the filter uses data from Diego Garcia, there is a large spike in the residual. Once again pointing to Deigo Garcia being the problem station of all cases. An interesting case study would be to eliminate Diego Garcia and only use Kwajalein and Arecibo, but I will leave that for a later time.

References

[1] Inc. The MathWorks, *Aerospace toolbox*. Natick, Massachusetts, United State, 2021.Available: https://www.mathworks.com/products/aerospace-toolbox.html

[2] W. D. McClain and D. A. Vallado, *Fundamentals of astrodynamics and applications*. Springer Netherlands, 2001.

[3] D. Vallado, "Errata for fundamentals of astrodynamics and applications, 4th ed." Jan. 2013.

[4] C. H. Acton, "Ancillary data services of NASA's navigation and ancillary information facility," *Planetary and Space Science*, vol. 44, no. 1, pp. 65–70, 1996, doi: https://doi.org/10.1016/0032-0633(95)00107-7.

[5] C. Acton, N. Bachman, B. Semenov, and E. Wright, "A look towards the future in the handling of space science mission geometry," *Planetary and Space Science*, vol. 150, pp. 9–12, 2018, doi: https://doi.org/10.1016/j.pss.2017.02.013.

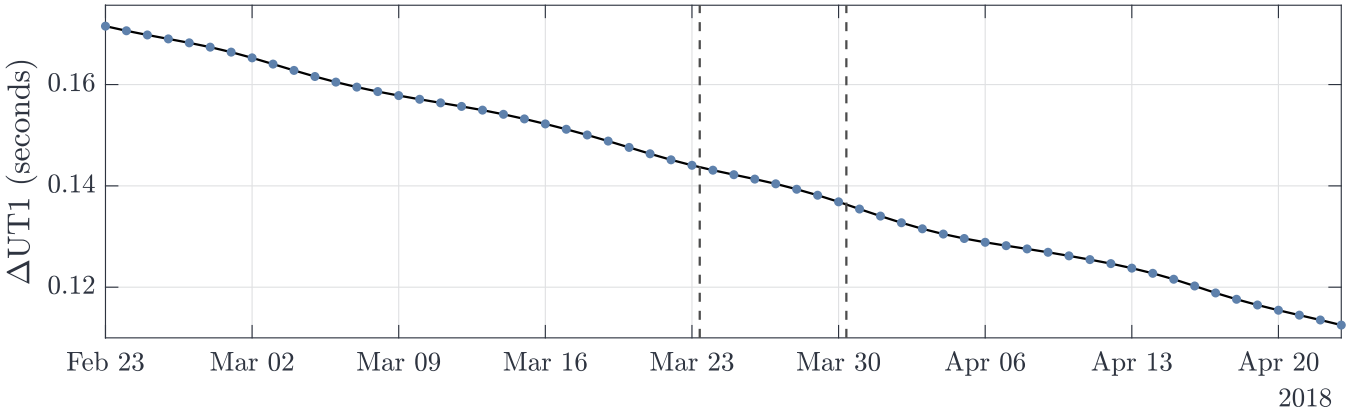
[6] P. K. Seidelmann, "1980 I.A.U. Theory of Nutation - the Final Report of the I.A.U. Working Group on Nutation," *Celestial Mechanics*, vol. 27, no. 1, pp. 79–106, May 1982, doi: 10.1007/BF01228952.

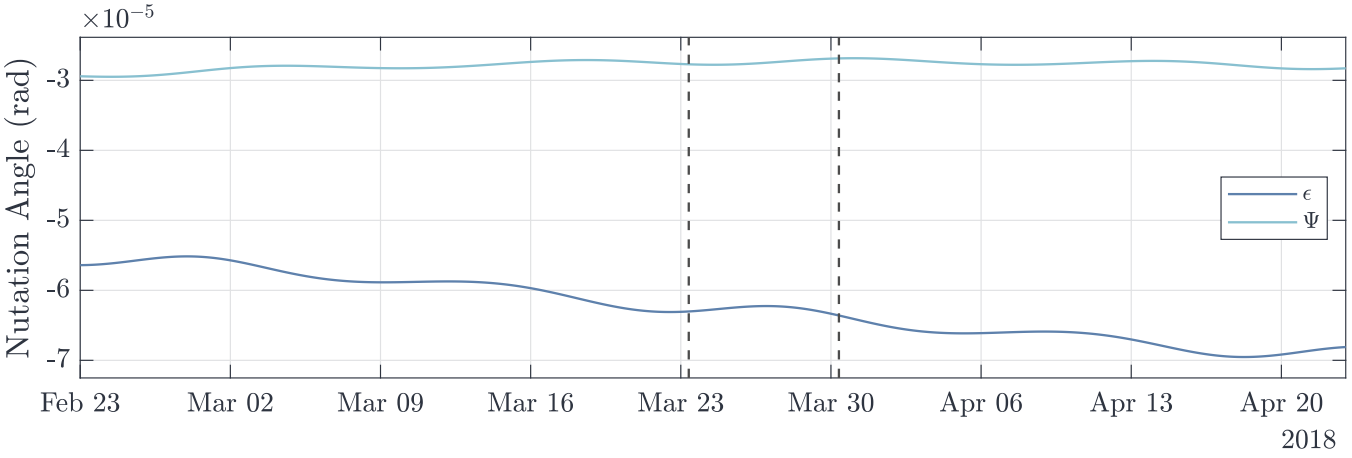
[7] Inc. The MathWorks, *Symbolic math toolbox*. Natick, Massachusetts, United State, 2021.Available: https://www.mathworks.com/help/symbolic/

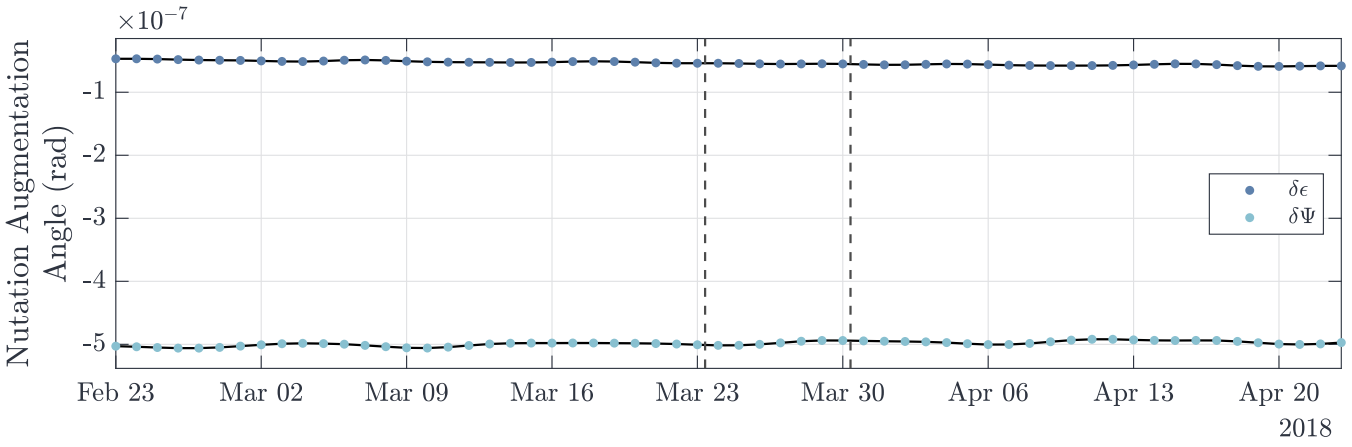
[8] B. Schutz, B. Tapley, and G. H. Born, *Statistical orbit determination*. Elsevier Science, 2004.

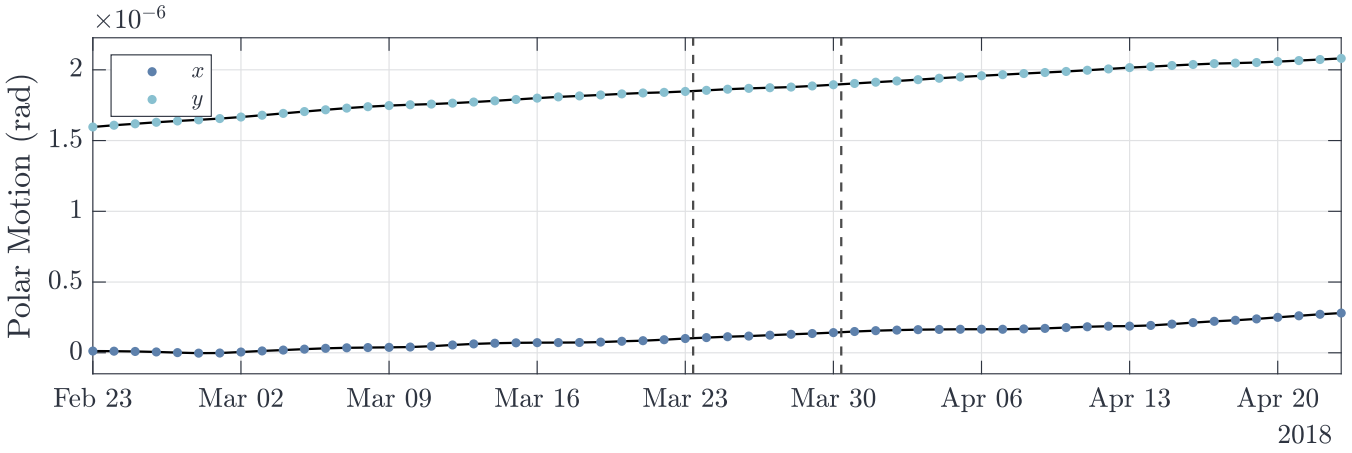
Appendix

EOP Data Visualization






×10−7
Nutation Augmentation Angle (rad)
-1
-2
-3
-4
-5
δϵ
δΨ
Feb 23
Mar 02
Mar 09
Mar 16
Mar 23
Mar 30
Apr 06
Apr 13
Apr 20
2018


×10−6
Polar Motion (rad)
2
1.5
1
0.5
0
x
y
Feb 23
Mar 02
Mar 09
Mar 16
Mar 23
Mar 30
Apr 06
Apr 13
Apr 20
2018

Results Zoom Ins

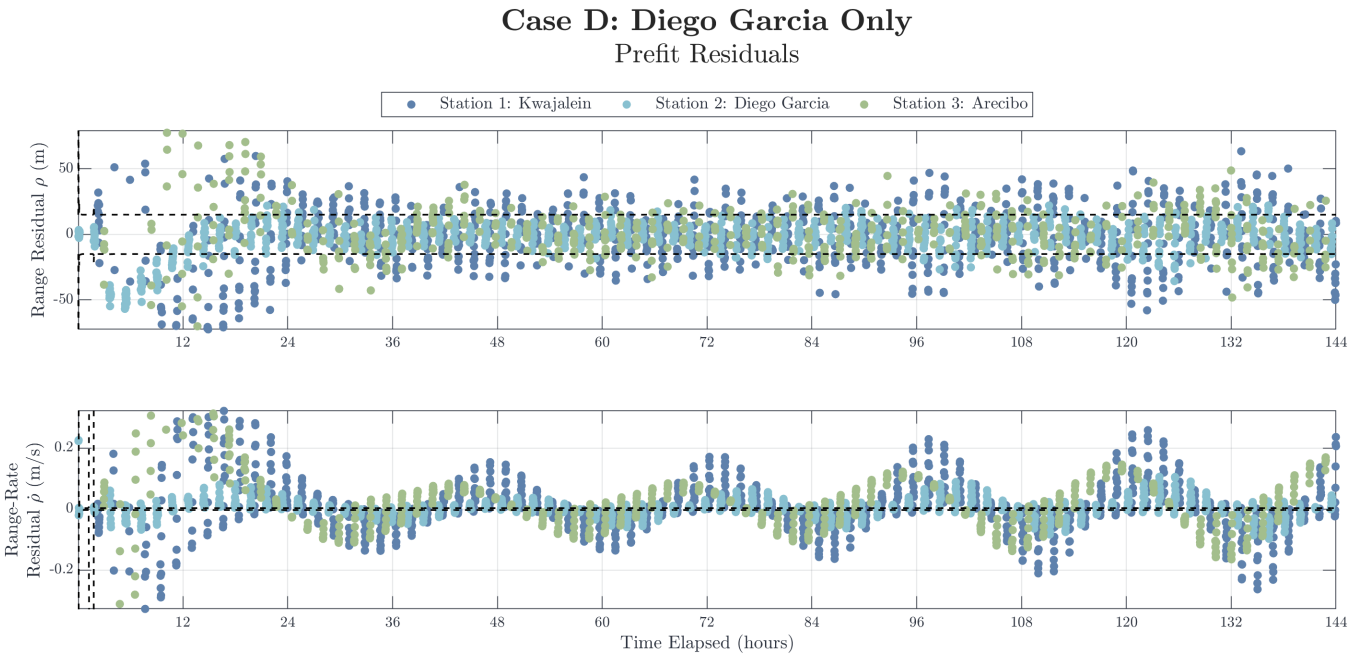


Figure 27: Prefit Case D Zoom-In

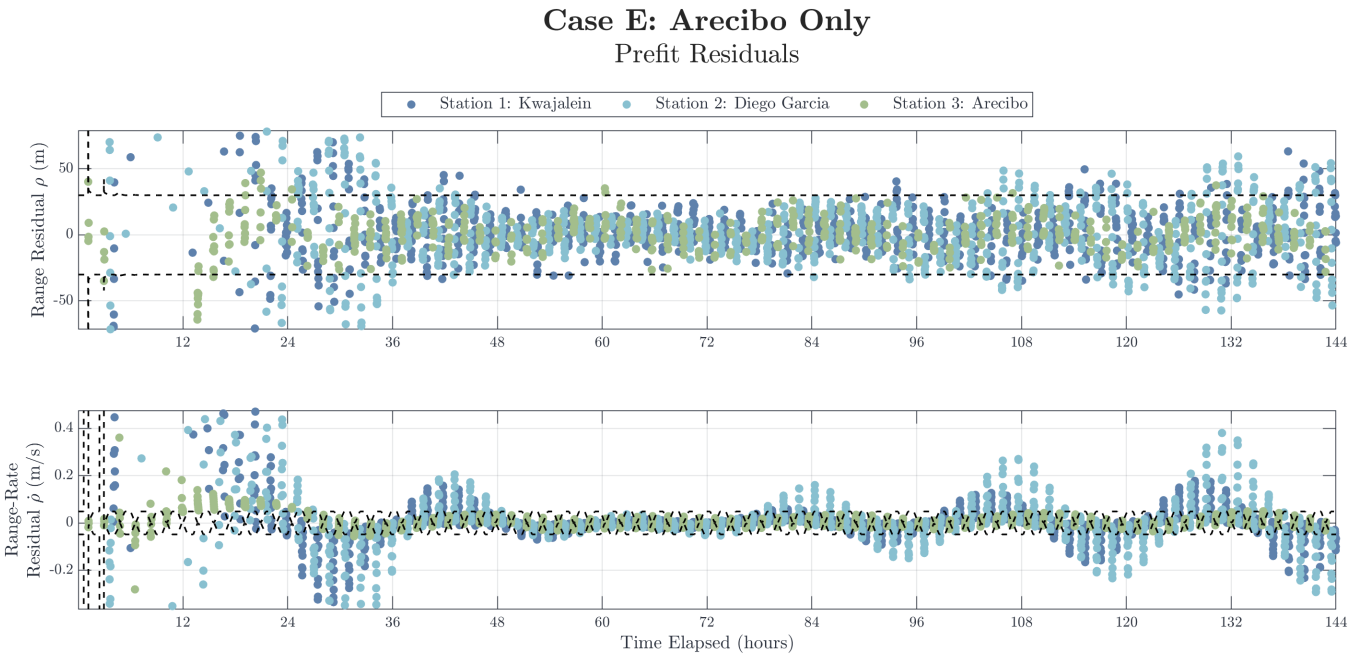


Figure 28: Prefit Case E Zoom-In

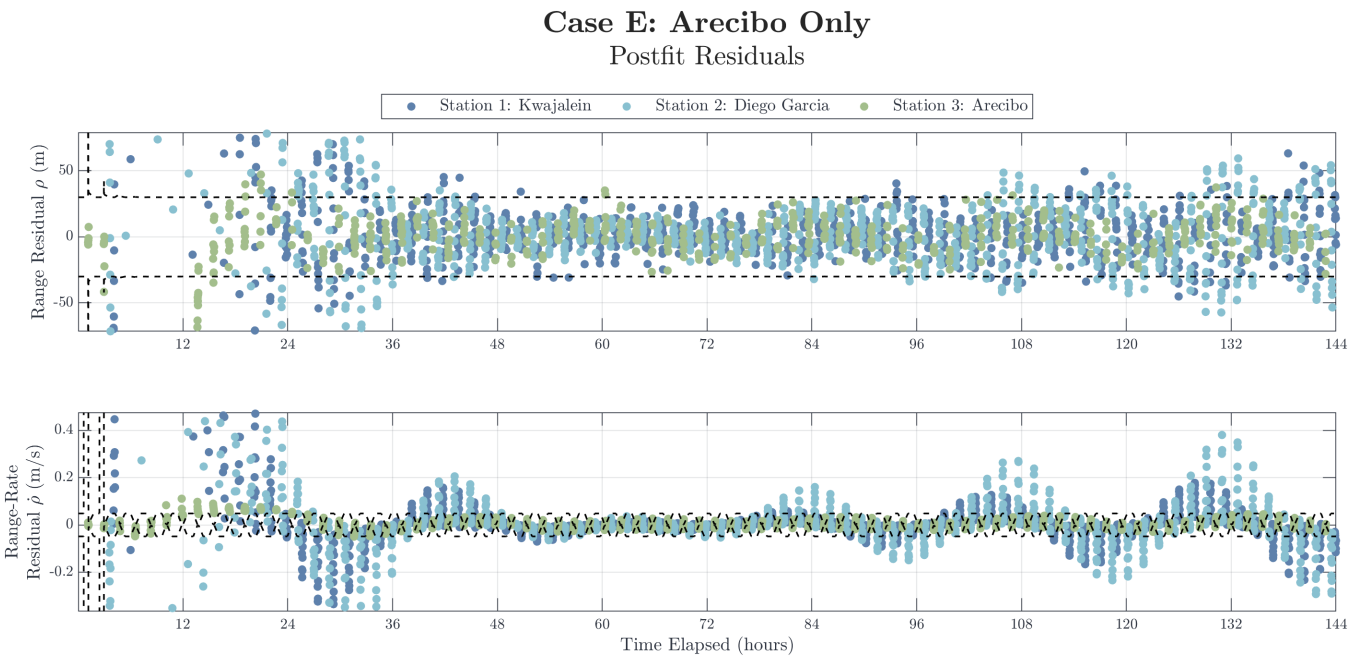
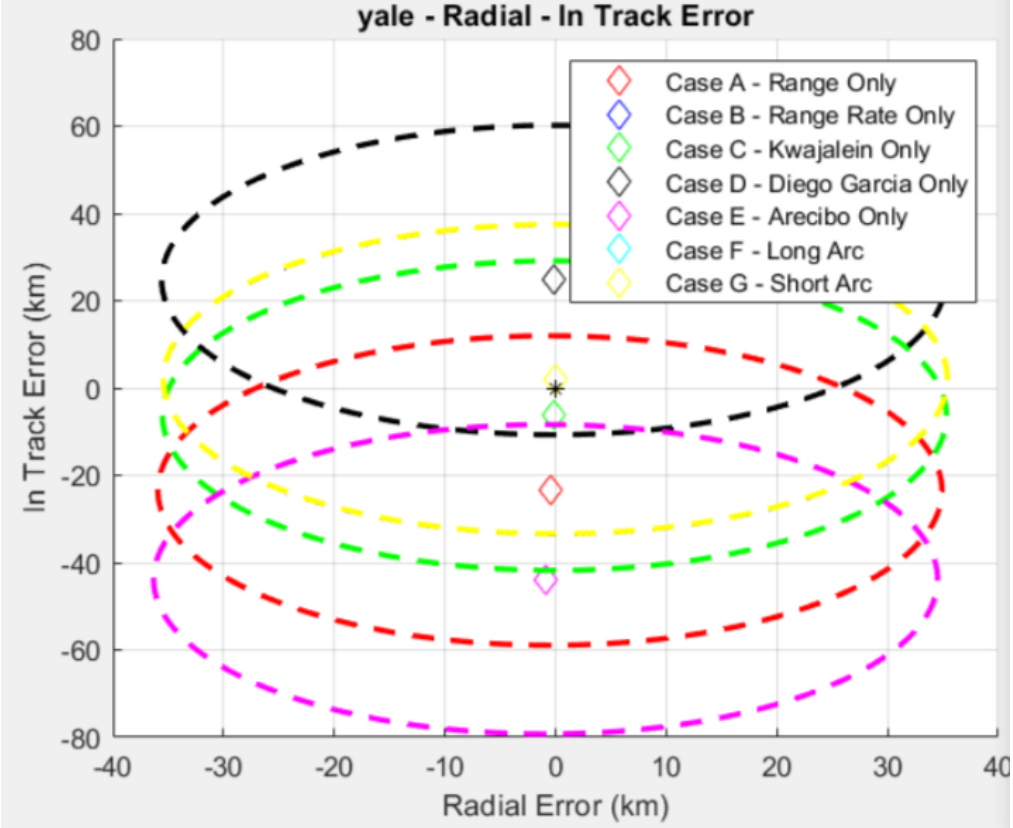
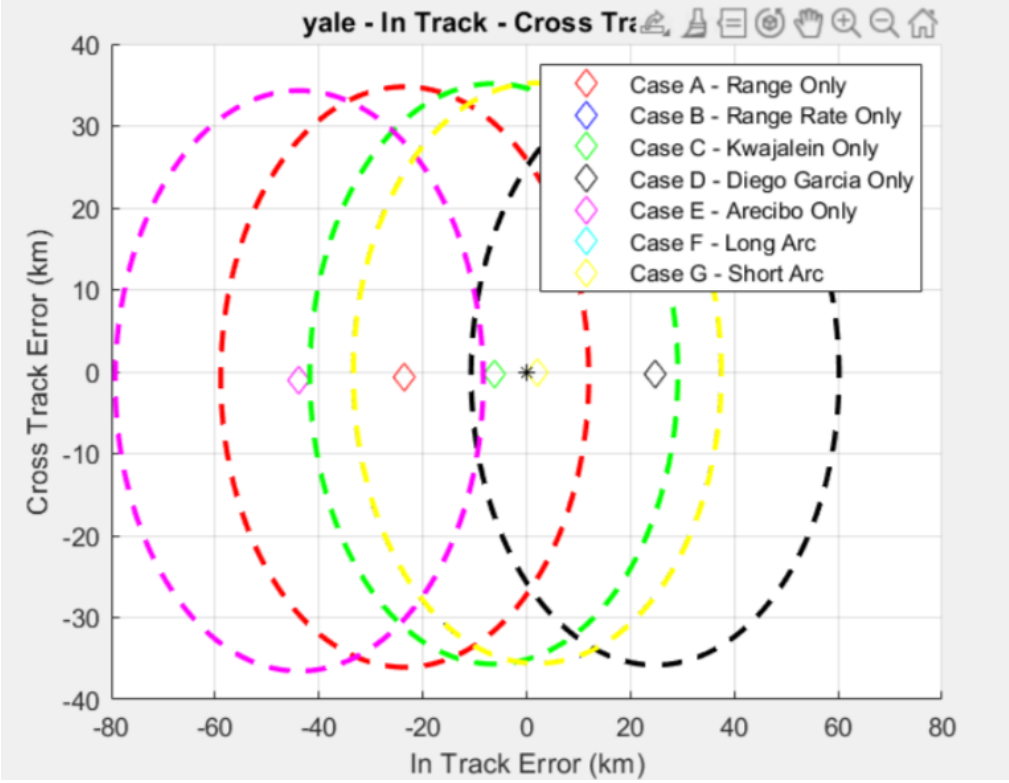
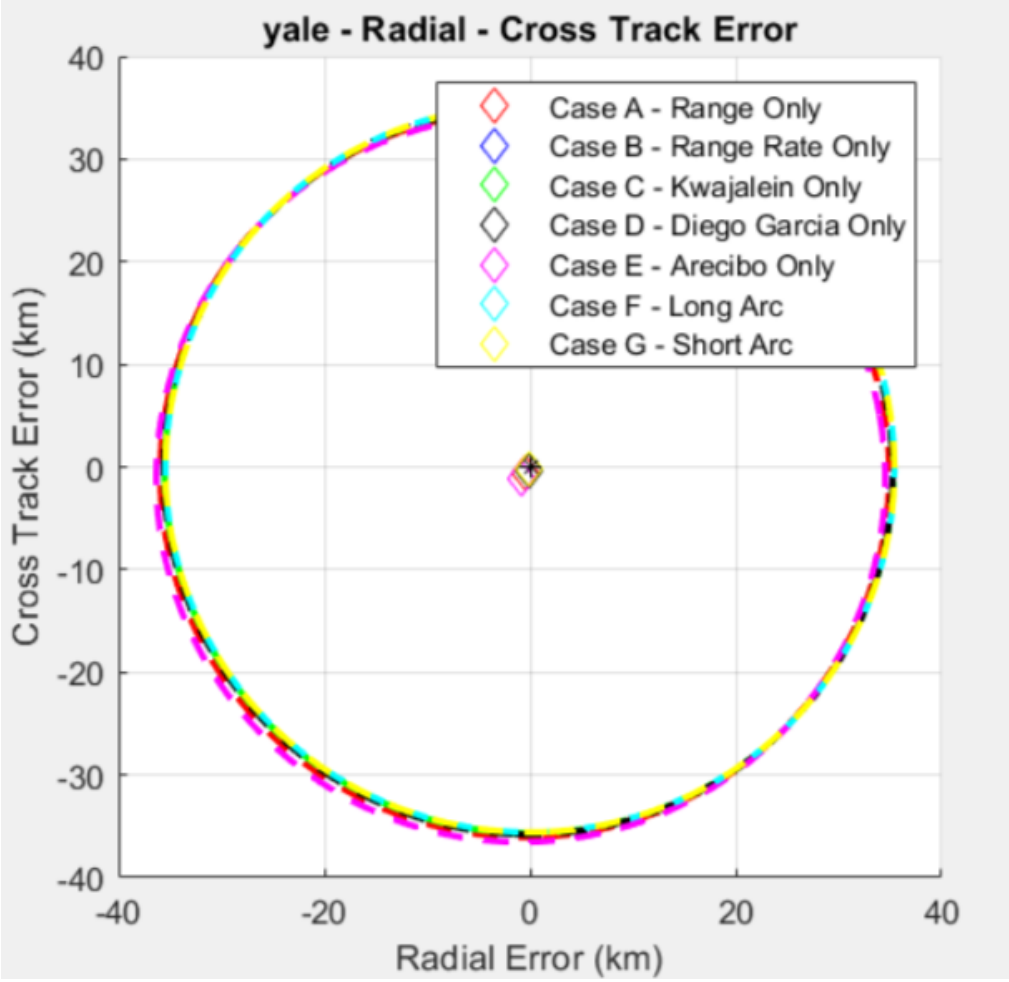


Figure 29: Postfit Case E Zoom-In

Results from Initial NAG




yale - In Track - Cross Tra
Case A - Range Only
Case B - Range Rate Only
Case C - Kwajalein Only
Case D - Diego Garcia Only
Case E - Arecibo Only
Case F - Long Arc
Case G - Short Arc
Cross Track Error (km)
In Track Error (km)
40
30
20
10
0
-10
-20
-30
-40
-80
-60
-40
-20
0
20
40
60
80


yale - Radial - Cross Track Error
Case A - Range Only
Case B - Range Rate Only
Case C - Kwajalein Only
Case D - Diego Garcia Only
Case E - Arecibo Only
Case F - Long Arc
Case G - Short Arc
Cross Track Error (km)
Radial Error (km)
40
30
20
10
0
-10
-20
-30
-40
-40
-20
0
20
40

Symbolic A Matrix

$$
\begin{pmatrix}
0 \\
0 \\
0 \\
\begin{array}{l}
\frac{3\,\mu\,x_1{}^2}{(x_1{}^2+x_2{}^2+x_3{}^2)^{5/2}} - \mu_M\left(\frac{1}{\left(|r_{\mathrm{M1}}-x_1|^2+|r_{\mathrm{M2}}-x_2|^2+|r_{\mathrm{M3}}-x_3|^2\right)^{3/2}} - \frac{3\,|r_{\mathrm{M1}}-x_1|\,\mathrm{sign}(r_{\mathrm{M1}}-x_1)\,(r_{\mathrm{M1}}-x_1)}{\left(|r_{\mathrm{M1}}-x_1|^2+|r_{\mathrm{M2}}-x_2|^2+|r_{\mathrm{M3}}-x_3|^2\right)^{5/2}}\right) - \dots \\
\quad \mu_S\left(\frac{1}{\left(|r_{\mathrm{S1}}-x_1|^2+|r_{\mathrm{S2}}-x_2|^2+|r_{\mathrm{S3}}-x_3|^2\right)^{3/2}} - \frac{3\,|r_{\mathrm{S1}}-x_1|\,\mathrm{sign}(r_{\mathrm{S1}}-x_1)\,(r_{\mathrm{S1}}-x_1)}{\left(|r_{\mathrm{S1}}-x_1|^2+|r_{\mathrm{S2}}-x_2|^2+|r_{\mathrm{S3}}-x_3|^2\right)^{5/2}}\right) - \frac{\mu}{(x_1{}^2+x_2{}^2+x_3{}^2)^{3/2}} + \frac{3\,J_2\,a_e{}^2\,\mu\left(\frac{5\,(\overline{x_3})^2}{x_1{}^2+x_2{}^2+x_3{}^2}-1\right)}{2\,(x_1{}^2+x_2{}^2+x_3{}^2)^{5/2}} - \frac{15\,J_2\,a_e{}^2\,\mu\,x_1{}^2\,(\overline{x_3})^2}{(x_1{}^2+x_2{}^2+x_3{}^2)^{9/2}} - \dots \\
\quad \frac{15\,J_2\,a_e{}^2\,\mu\,x_1{}^2\left(\frac{5\,(\overline{x_3})^2}{x_1{}^2+x_2{}^2+x_3{}^2}-1\right)}{2\,(x_1{}^2+x_2{}^2+x_3{}^2)^{7/2}} - \frac{A\,\mathrm{AU}^2\,C_R\,P_{\mathrm{srp}}}{m\left((r_{\mathrm{S1}}-x_1)^2+(r_{\mathrm{S2}}-x_2)^2+(r_{\mathrm{S3}}-x_3)^2\right)^{3/2}} + \frac{3\,A\,\mathrm{AU}^2\,C_R\,P_{\mathrm{srp}}\,(2\,r_{\mathrm{S1}}-2\,x_1)\,(r_{\mathrm{S1}}-x_1)}{2\,m\left((r_{\mathrm{S1}}-x_1)^2+(r_{\mathrm{S2}}-x_2)^2+(r_{\mathrm{S3}}-x_3)^2\right)^{5/2}} + \dots \\
\quad \frac{A\,C_D\,\rho_0\,\dot\theta\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\left(x_4+\dot\theta\,x_2\right)\left(x_5-\dot\theta\,x_1\right)}{2\,m\sqrt{\left(x_4+\dot\theta\,x_2\right)^2+\left(x_5-\dot\theta\,x_1\right)^2+x_6{}^2}} + \frac{A\,C_D\,\rho_0\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\,|x_1|\,\mathrm{sign}(x_1)\left(x_4+\dot\theta\,x_2\right)\sqrt{\left(x_4+\dot\theta\,x_2\right)^2+\left(x_5-\dot\theta\,x_1\right)^2+x_6{}^2}}{2\,H\,m\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}
\end{array} \\
\begin{array}{l}
\frac{3\,\mu\,x_1\,x_2}{(x_1{}^2+x_2{}^2+x_3{}^2)^{5/2}} + \frac{3\,\mu_M\,|r_{\mathrm{M1}}-x_1|\,\mathrm{sign}(r_{\mathrm{M1}}-x_1)\,(r_{\mathrm{M2}}-x_2)}{\left(|r_{\mathrm{M1}}-x_1|^2+|r_{\mathrm{M2}}-x_2|^2+|r_{\mathrm{M3}}-x_3|^2\right)^{5/2}} + \frac{3\,\mu_S\,|r_{\mathrm{S1}}-x_1|\,\mathrm{sign}(r_{\mathrm{S1}}-x_1)\,(r_{\mathrm{S2}}-x_2)}{\left(|r_{\mathrm{S1}}-x_1|^2+|r_{\mathrm{S2}}-x_2|^2+|r_{\mathrm{S3}}-x_3|^2\right)^{5/2}} + \frac{A\,C_D\,\rho_0\,\dot\theta\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\sqrt{\left(x_4+\dot\theta\,x_2\right)^2+\left(x_5-\dot\theta\,x_1\right)^2+x_6{}^2}}{2\,m} - \dots \\
\quad \frac{15\,J_2\,a_e{}^2\,\mu\,x_1\,x_2\,(\overline{x_3})^2}{(x_1{}^2+x_2{}^2+x_3{}^2)^{9/2}} - \frac{15\,J_2\,a_e{}^2\,\mu\,x_1\,x_2\left(\frac{5\,(\overline{x_3})^2}{x_1{}^2+x_2{}^2+x_3{}^2}-1\right)}{2\,(x_1{}^2+x_2{}^2+x_3{}^2)^{7/2}} + \frac{3\,A\,\mathrm{AU}^2\,C_R\,P_{\mathrm{srp}}\,(2\,r_{\mathrm{S1}}-2\,x_1)\,(r_{\mathrm{S2}}-x_2)}{2\,m\left((r_{\mathrm{S1}}-x_1)^2+(r_{\mathrm{S2}}-x_2)^2+(r_{\mathrm{S3}}-x_3)^2\right)^{5/2}} + \frac{A\,C_D\,\rho_0\,\dot\theta\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\left(x_5-\dot\theta\,x_1\right)^2}{2\,m\sqrt{\left(x_4+\dot\theta\,x_2\right)^2+\left(x_5-\dot\theta\,x_1\right)^2+x_6{}^2}} + \dots \\
\quad \frac{A\,C_D\,\rho_0\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\,|x_1|\,\mathrm{sign}(x_1)\left(x_5-\dot\theta\,x_1\right)\sqrt{\left(x_4+\dot\theta\,x_2\right)^2+\left(x_5-\dot\theta\,x_1\right)^2+x_6{}^2}}{2\,H\,m\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}
\end{array} \\
\begin{array}{l}
\frac{3\,\mu\,x_1\,x_3}{(x_1{}^2+x_2{}^2+x_3{}^2)^{5/2}} + \frac{3\,\mu_M\,|r_{\mathrm{M1}}-x_1|\,\mathrm{sign}(r_{\mathrm{M1}}-x_1)\,(r_{\mathrm{M3}}-x_3)}{\left(|r_{\mathrm{M1}}-x_1|^2+|r_{\mathrm{M2}}-x_2|^2+|r_{\mathrm{M3}}-x_3|^2\right)^{5/2}} + \frac{3\,\mu_S\,|r_{\mathrm{S1}}-x_1|\,\mathrm{sign}(r_{\mathrm{S1}}-x_1)\,(r_{\mathrm{S3}}-x_3)}{\left(|r_{\mathrm{S1}}-x_1|^2+|r_{\mathrm{S2}}-x_2|^2+|r_{\mathrm{S3}}-x_3|^2\right)^{5/2}} + \frac{15\,J_2\,a_e{}^2\,\mu\,x_1\,\overline{x_3}}{(x_1{}^2+x_2{}^2+x_3{}^2)^{7/2}} - \frac{15\,J_2\,a_e{}^2\,\mu\,x_1\,x_3\,(\overline{x_3})^2}{(x_1{}^2+x_2{}^2+x_3{}^2)^{9/2}} - \dots \\
\quad \frac{15\,J_2\,a_e{}^2\,\mu\,x_1\,x_3\left(\frac{5\,(\overline{x_3})^2}{x_1{}^2+x_2{}^2+x_3{}^2}-1\right)}{2\,(x_1{}^2+x_2{}^2+x_3{}^2)^{7/2}} + \frac{3\,A\,\mathrm{AU}^2\,C_R\,P_{\mathrm{srp}}\,(2\,r_{\mathrm{S1}}-2\,x_1)\,(r_{\mathrm{S3}}-x_3)}{2\,m\left((r_{\mathrm{S1}}-x_1)^2+(r_{\mathrm{S2}}-x_2)^2+(r_{\mathrm{S3}}-x_3)^2\right)^{5/2}} + \frac{A\,C_D\,\rho_0\,\dot\theta\,x_6\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\left(x_5-\dot\theta\,x_1\right)}{2\,m\sqrt{\left(x_4+\dot\theta\,x_2\right)^2+\left(x_5-\dot\theta\,x_1\right)^2+x_6{}^2}} + \dots \\
\quad \frac{A\,C_D\,\rho_0\,x_6\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\,|x_1|\,\mathrm{sign}(x_1)\sqrt{\left(x_4+\dot\theta\,x_2\right)^2+\left(x_5-\dot\theta\,x_1\right)^2+x_6{}^2}}{2\,H\,m\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}
\end{array}
\end{pmatrix}
$$

$$
\begin{pmatrix}
0 \\
0 \\
0 \\
0 \\
\frac{3\mu x_1 x_2}{(x_1^2+x_2^2+x_3^2)^{5/2}} + \frac{3\mu_M |r_{M2}-x_2|\,\mathrm{sign}(r_{M2}-x_2)\,(r_{M1}-x_1)}{(|r_{M1}-x_1|^2+|r_{M2}-x_2|^2+|r_{M3}-x_3|^2)^{5/2}} + \frac{3\mu_S |r_{S2}-x_2|\,\mathrm{sign}(r_{S2}-x_2)\,(r_{S1}-x_1)}{(|r_{S1}-x_1|^2+|r_{S2}-x_2|^2+|r_{S3}-x_3|^2)^{5/2}} - \frac{A C_D \rho_0 \dot\theta\, e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} \sqrt{(x_4+\dot\theta x_2)^2+(x_5-\dot\theta x_1)^2+x_6^2}}{2m} - \dots \\
\frac{3\mu x_2 x_3}{(x_1^2+x_2^2+x_3^2)^{5/2}} + \frac{3\mu_M |r_{M2}-x_2|\,\mathrm{sign}(r_{M2}-x_2)\,(r_{M3}-x_3)}{(|r_{M1}-x_1|^2+|r_{M2}-x_2|^2+|r_{M3}-x_3|^2)^{5/2}} + \frac{3\mu_S |r_{S2}-x_2|\,\mathrm{sign}(r_{S2}-x_2)\,(r_{S3}-x_3)}{(|r_{S1}-x_1|^2+|r_{S2}-x_2|^2+|r_{S3}-x_3|^2)^{5/2}} - \frac{A C_D \rho_0 \dot\theta\, e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} (x_4+\dot\theta x_2)}{2m\sqrt{(x_4+\dot\theta x_2)^2+(x_5-\dot\theta x_1)^2+x_6^2}} + \dots \\
\frac{3\mu x_2^2}{(x_1^2+x_2^2+x_3^2)^{5/2}} - \mu_M \left(\frac{1}{(|r_{M1}-x_1|^2+|r_{M2}-x_2|^2+|r_{M3}-x_3|^2)^{3/2}} - \frac{3\,|r_{M2}-x_2|\,\mathrm{sign}(r_{M2}-x_2)\,(r_{M2}-x_2)}{(|r_{M1}-x_1|^2+|r_{M2}-x_2|^2+|r_{M3}-x_3|^2)^{5/2}} \right) - \dots \\
\mu_S \left(\frac{1}{(|r_{S1}-x_1|^2+|r_{S2}-x_2|^2+|r_{S3}-x_3|^2)^{3/2}} - \frac{3\,|r_{S2}-x_2|\,\mathrm{sign}(r_{S2}-x_2)\,(r_{S2}-x_2)}{(|r_{S1}-x_1|^2+|r_{S2}-x_2|^2+|r_{S3}-x_3|^2)^{5/2}} \right) - \frac{\mu}{(x_1^2+x_2^2+x_3^2)^{3/2}} + \frac{3 J_2 a_e^2 \mu \left(\frac{5(\overline{x_3})^2}{x_1^2+x_2^2+x_3^2} - 1 \right)}{2(x_1^2+x_2^2+x_3^2)^{5/2}} - \frac{15 J_2 a_e^2 \mu x_2^2 (\overline{x_3})^2}{(x_1^2+x_2^2+x_3^2)^{9/2}} - \dots \\
\frac{15 J_2 a_e^2 \mu x_2^2 \left(\frac{5(\overline{x_3})^2}{x_1^2+x_2^2+x_3^2} - 1 \right)}{2(x_1^2+x_2^2+x_3^2)^{7/2}} - \frac{A A U^2 C_R P_{\mathrm{srp}}}{m\left((r_{S1}-x_1)^2+(r_{S2}-x_2)^2+(r_{S3}-x_3)^2\right)^{3/2}} + \frac{3 A A U^2 C_R P_{\mathrm{srp}} (2 r_{S2}-2 x_2)\,(r_{S2}-x_2)}{2m\left((r_{S1}-x_1)^2+(r_{S2}-x_2)^2+(r_{S3}-x_3)^2\right)^{5/2}} - \frac{A C_D \rho_0 \dot\theta\, e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} (x_4+\dot\theta x_2)(x_5-\dot\theta x_1)}{2m\sqrt{(x_4+\dot\theta x_2)^2+(x_5-\dot\theta x_1)^2+x_6^2}} + \dots \\
\frac{A C_D \rho_0\, e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}}{2Hm\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}} \left(\frac{|x_2|\,\mathrm{sign}(x_2)\,(x_4+\dot\theta x_2)}{\sqrt{(x_4+\dot\theta x_2)^2+(x_5-\dot\theta x_1)^2+x_6^2}} + \frac{|x_2|\,\mathrm{sign}(x_2)\,(x_5-\dot\theta x_1)}{\sqrt{(x_4+\dot\theta x_2)^2+(x_5-\dot\theta x_1)^2+x_6^2}} \right) \\
\frac{3\mu x_2 x_3}{(x_1^2+x_2^2+x_3^2)^{5/2}} + \frac{3\mu_M |r_{M2}-x_2|\,\mathrm{sign}(r_{M2}-x_2)\,(r_{M3}-x_3)}{(|r_{M1}-x_1|^2+|r_{M2}-x_2|^2+|r_{M3}-x_3|^2)^{5/2}} + \frac{3\mu_S |r_{S2}-x_2|\,\mathrm{sign}(r_{S2}-x_2)\,(r_{S3}-x_3)}{(|r_{S1}-x_1|^2+|r_{S2}-x_2|^2+|r_{S3}-x_3|^2)^{5/2}} + \frac{15 J_2 a_e^2 \mu x_2 x_3}{(x_1^2+x_2^2+x_3^2)^{7/2}} - \frac{15 J_2 a_e^2 \mu x_2 x_3 (\overline{x_3})^2}{(x_1^2+x_2^2+x_3^2)^{9/2}} - \frac{15 J_2 a_e^2 \mu x_2 x_3 \left(\frac{5(\overline{x_3})^2}{x_1^2+x_2^2+x_3^2} - 1 \right)}{2(x_1^2+x_2^2+x_3^2)^{7/2}} + \dots \\
\frac{3 A A U^2 C_R P_{\mathrm{srp}} (2 r_{S2}-2 x_2)}{2m\left((r_{S1}-x_1)^2+(r_{S2}-x_2)^2+(r_{S3}-x_3)^2\right)^{5/2}} - \frac{A C_D \rho_0 \dot\theta\, x_6\, e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}}{2m\sqrt{(x_4+\dot\theta x_2)^2+(x_5-\dot\theta x_1)^2+x_6^2}} + \frac{A C_D \rho_0\, x_6\, e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} |x_2|\,\mathrm{sign}(x_2) \sqrt{(x_4+\dot\theta x_2)^2+(x_5-\dot\theta x_1)^2+x_6^2}}{2Hm\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}
\end{pmatrix}
$$

$$
\begin{pmatrix}
0 \\
0 \\
0 \\
0 \\
\frac{3\,\mu\,x_1\,x_3}{\left(x_1^2+x_2^2+x_3^2\right)^{5/2}}+\frac{3\,\mu_M\,|r_{M3}-x_3|\,\mathrm{sign}(r_{M3}-x_3)\,(r_{M1}-x_1)}{\left(|r_{M1}-x_1|^2+|r_{M2}-x_2|^2+|r_{M3}-x_3|^2\right)^{5/2}}+\frac{3\,\mu_S\,|r_{S3}-x_3|\,\mathrm{sign}(r_{S3}-x_3)\,(r_{S1}-x_1)}{\left(|r_{S1}-x_1|^2+|r_{S2}-x_2|^2+|r_{S3}-x_3|^2\right)^{5/2}}+\frac{3\,J_2\,a_e{}^2\,\mu\,x_1\left(\frac{10\,\overline{x_3}}{x_1^2+x_2^2+x_3^2}-\frac{10\,x_3\,(\overline{x_3})^2}{(x_1^2+x_2^2+x_3^2)^2}\right)}{2\,(x_1^2+x_2^2+x_3^2)^{5/2}}-\frac{15\,J_2\,a_e{}^2\,\mu\,x_1\,x_3\left(\frac{5\,(\overline{x_3})^2}{x_1^2+x_2^2+x_3^2}-1\right)}{2\,(x_1^2+x_2^2+x_3^2)^{7/2}}+\ldots \\
\frac{3\,A\,AU^2\,C_R\,P_{srp}\,(2\,r_{S3}-2\,x_3)\,(r_{S1}-x_1)}{2\,m\left((r_{S1}-x_1)^2+(r_{S2}-x_2)^2+(r_{S3}-x_3)^2\right)^{5/2}}+\frac{A\,C_D\,\rho_0\,e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\,|x_3|\,\mathrm{sign}(x_3)\,\left(x_4+\dot\theta\,x_2\right)\sqrt{\left(x_4+\dot\theta\,x_2\right)^2+\left(x_5-\dot\theta\,x_1\right)^2+x_6{}^2}}{2\,H\,m\,\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}} \\
\frac{3\,\mu\,x_2\,x_3}{\left(x_1^2+x_2^2+x_3^2\right)^{5/2}}+\frac{3\,\mu_M\,|r_{M3}-x_3|\,\mathrm{sign}(r_{M3}-x_3)\,(r_{M2}-x_2)}{\left(|r_{M1}-x_1|^2+|r_{M2}-x_2|^2+|r_{M3}-x_3|^2\right)^{5/2}}+\frac{3\,\mu_S\,|r_{S3}-x_3|\,\mathrm{sign}(r_{S3}-x_3)\,(r_{S2}-x_2)}{\left(|r_{S1}-x_1|^2+|r_{S2}-x_2|^2+|r_{S3}-x_3|^2\right)^{5/2}}+\frac{3\,J_2\,a_e{}^2\,\mu\,x_2\left(\frac{10\,\overline{x_3}}{x_1^2+x_2^2+x_3^2}-\frac{10\,x_3\,(\overline{x_3})^2}{(x_1^2+x_2^2+x_3^2)^2}\right)}{2\,(x_1^2+x_2^2+x_3^2)^{5/2}}-\frac{15\,J_2\,a_e{}^2\,\mu\,x_2\,x_3\left(\frac{5\,(\overline{x_3})^2}{x_1^2+x_2^2+x_3^2}-1\right)}{2\,(x_1^2+x_2^2+x_3^2)^{7/2}}+\ldots \\
\frac{3\,A\,AU^2\,C_R\,P_{srp}\,(2\,r_{S3}-2\,x_3)\,(r_{S2}-x_2)}{2\,m\left((r_{S1}-x_1)^2+(r_{S2}-x_2)^2+(r_{S3}-x_3)^2\right)^{5/2}}+\frac{A\,C_D\,\rho_0\,e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\,|x_3|\,\mathrm{sign}(x_3)\,\left(x_5-\dot\theta\,x_1\right)\sqrt{\left(x_4+\dot\theta\,x_2\right)^2+\left(x_5-\dot\theta\,x_1\right)^2+x_6{}^2}}{2\,H\,m\,\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}} \\
\frac{3\,\mu\,x_3{}^2}{\left(x_1^2+x_2^2+x_3^2\right)^{5/2}}-\mu_M\left(\frac{1}{\left(|r_{M1}-x_1|^2+|r_{M2}-x_2|^2+|r_{M3}-x_3|^2\right)^{3/2}}-\frac{3\,|r_{M3}-x_3|\,\mathrm{sign}(r_{M3}-x_3)\,(r_{M3}-x_3)}{\left(|r_{M1}-x_1|^2+|r_{M2}-x_2|^2+|r_{M3}-x_3|^2\right)^{5/2}}\right)-\ldots \\
\mu_S\left(\frac{1}{\left(|r_{S1}-x_1|^2+|r_{S2}-x_2|^2+|r_{S3}-x_3|^2\right)^{3/2}}-\frac{3\,|r_{S3}-x_3|\,\mathrm{sign}(r_{S3}-x_3)\,(r_{S3}-x_3)}{\left(|r_{S1}-x_1|^2+|r_{S2}-x_2|^2+|r_{S3}-x_3|^2\right)^{5/2}}\right)-\frac{\mu}{\left(x_1^2+x_2^2+x_3^2\right)^{3/2}}-\frac{3\,J_2\,a_e{}^2\,\mu}{\left(x_1^2+x_2^2+x_3^2\right)^{5/2}}+\frac{3\,J_2\,a_e{}^2\,\mu\left(\frac{5\,(\overline{x_3})^2}{x_1^2+x_2^2+x_3^2}-1\right)}{2\,(x_1^2+x_2^2+x_3^2)^{5/2}}-\ldots \\
\frac{15\,J_2\,a_e{}^2\,\mu\,x_3{}^2\left(\frac{5\,(\overline{x_3})^2}{x_1^2+x_2^2+x_3^2}-1\right)}{2\,(x_1^2+x_2^2+x_3^2)^{7/2}}+\frac{15\,J_2\,a_e{}^2\,\mu\,x_3\,\overline{x_3}}{\left(x_1^2+x_2^2+x_3^2\right)^{7/2}}-\frac{A\,AU^2\,C_R\,P_{srp}}{m\left((r_{S1}-x_1)^2+(r_{S2}-x_2)^2+(r_{S3}-x_3)^2\right)^{3/2}}+\frac{3\,J_2\,a_e{}^2\,\mu\,x_3\left(\frac{10\,\overline{x_3}}{x_1^2+x_2^2+x_3^2}-\frac{10\,x_3\,(\overline{x_3})^2}{(x_1^2+x_2^2+x_3^2)^2}\right)}{2\,(x_1^2+x_2^2+x_3^2)^{5/2}}-\frac{3\,J_2\,a_e{}^2\,\mu\left(\frac{5\,(\overline{x_3})^2}{x_1^2+x_2^2+x_3^2}-1\right)}{2\,(x_1^2+x_2^2+x_3^2)^{5/2}}+\ldots \\
\frac{3\,A\,AU^2\,C_R\,P_{srp}\,(2\,r_{S3}-2\,x_3)\,(r_{S3}-x_3)}{2\,m\left((r_{S1}-x_1)^2+(r_{S2}-x_2)^2+(r_{S3}-x_3)^2\right)^{5/2}}+\frac{A\,C_D\,\rho_0\,x_6\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\,|x_3|\,\mathrm{sign}(x_3)\,\sqrt{\left(x_4+\dot\theta\,x_2\right)^2+\left(x_5-\dot\theta\,x_1\right)^2+x_6{}^2}}{2\,H\,m\,\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}
\end{pmatrix}
$$

$$
\begin{pmatrix}
1 \\
0 \\
0 \\
0 \\
-\frac{A C_D \rho_0 e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} \sqrt{\left(x_4+\dot{\theta} x_2\right)^2+\left(x_5-\dot{\theta} x_1\right)^2+x_6^2}}{2 m} \\
-\frac{A C_D \rho_0 e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} \left(x_5-\dot{\theta} x_1\right) \left(2 x_4+2 \dot{\theta} x_2\right)}{4 m \sqrt{\left(x_4+\dot{\theta} x_2\right)^2+\left(x_5-\dot{\theta} x_1\right)^2+x_6^2}} \\
-\frac{A C_D \rho_0 x_6 e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} \left(2 x_4+2 \dot{\theta} x_2\right)}{4 m \sqrt{\left(x_4+\dot{\theta} x_2\right)^2+\left(x_5-\dot{\theta} x_1\right)^2+x_6^2}}
\end{pmatrix}
$$

$$
\begin{pmatrix}
0 \\
1 \\
0 \\
0 \\
-\frac{A C_D \rho_0 e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} \left(x_4+\dot{\theta} x_2\right) \left(2 x_5-2 \dot{\theta} x_1\right)}{4 m \sqrt{\left(x_4+\dot{\theta} x_2\right)^2+\left(x_5-\dot{\theta} x_1\right)^2+x_6{}^2}} \\
-\frac{A C_D \rho_0 e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} \sqrt{\left(x_4+\dot{\theta} x_2\right)^2+\left(x_5-\dot{\theta} x_1\right)^2+x_6{}^2}}{2 m} \\
-\frac{A C_D \rho_0 x_6 e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} \left(2 x_5-2 \dot{\theta} x_1\right)}{4 m \sqrt{\left(x_4+\dot{\theta} x_2\right)^2+\left(x_5-\dot{\theta} x_1\right)^2+x_6{}^2}}
\end{pmatrix}
$$

$$
\begin{pmatrix}
0 \\
0 \\
1 \\
1 \\
-\frac{A C_D \rho_0 x_6 e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} \left(x_4+\dot{\theta} x_2\right)}{2 m \sqrt{\left(x_4+\dot{\theta} x_2\right)^2+\left(x_5-\dot{\theta} x_1\right)^2+{x_6}^2}} \\
-\frac{A C_D \rho_0 x_6 e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} \left(x_5-\dot{\theta} x_1\right)}{2 m \sqrt{\left(x_4+\dot{\theta} x_2\right)^2+\left(x_5-\dot{\theta} x_1\right)^2+{x_6}^2}} \\
-\frac{A C_D \rho_0 e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}} \sqrt{\left(x_4+\dot{\theta} x_2\right)^2+\left(x_5-\dot{\theta} x_1\right)^2+{x_6}^2}}{2 m} \\
-\frac{A C_D \rho_0 {x_6}^2 e^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}}{2 m \sqrt{\left(x_4+\dot{\theta} x_2\right)^2+\left(x_5-\dot{\theta} x_1\right)^2+{x_6}^2}}
\end{pmatrix}
$$

MATLAB Code

Driver Script

Methods of Orbit Determination Project

Burton Yale

Last Update: 2022-04-25

Setup

Pathing to Correct Folder

```
clear; close all; clc
% cd('C:\Users\Burto\repos\uta\mod\proj')
% addpath('src')
```

Loading Constants

58200.5202/0.5209 -- 00.9952/0.9959

```
load("constants.mat");
dataset = importdataset(3);
```

Case Selection

```
runcase = 4;
dynname = 'EGMDragSRP3BP';
```

Filtering

```
% INPUTS
X0 = const.X0;
sigmaR = 10;
sigmaV = 10/1000;
sigmaA = [1e-16 1e-12 1e-16].';
Qdt = 1;
const.SC.CD = 2;

% RUNNING
% profile clear
% profile on
warning('off', 'MATLAB:nearlySingularMatrix'); % Removing annoyance
filter = filter_inputs(X0, runcase, ...
    sigmaR, sigmaV, sigmaA, Qdt, ...
    dataset, const, true)
```

```
Filtering: 100%    [..........] Done
filter = struct with fields:
          X: [2570×6 double]
          P: [6×6×2570 double]
          S: [2×2×2569 double]
     prefit: [2569×2 double]
    postfit: [2569×2 double]
          K: [6×2×2569 double]
```

```
    inputs: [1×1 struct]
      time: [0 60 120 180 240 300 360 420 480 540 600 1620 1680 1740 1800 1860 1920 1980 2040 2100 2160 2220 2280 4
```

```
warning('on', 'MATLAB:nearlySingularMatrix'); % Turning back on
% profile viewer

filter.X(end, :).'
```

```
ans = 6×1
      -4906.1
         5221
       64.524
      -5.4175
      -5.1112
     -0.25386
```

```
filter.P(:, :, end)
```

```
ans = 6×6
   6.1965e-09   6.4471e-09   9.9806e-10  -5.5301e-12   6.3359e-12   1.4675e-12
   6.4471e-09   7.5058e-09   2.4791e-10   -5.787e-12   7.4437e-12   1.9676e-12
   9.9806e-10   2.4791e-10   2.6821e-09   -7.928e-13   1.2652e-13  -1.8151e-14
  -5.5301e-12   -5.787e-12   -7.928e-13   5.0655e-15  -5.8148e-15  -1.1016e-15
   6.3359e-12   7.4437e-12   1.2652e-13  -5.8148e-15   7.5196e-15   1.6483e-15
   1.4675e-12   1.9676e-12  -1.8151e-14  -1.1016e-15   1.6483e-15   3.0015e-15
```

Propagating Forward to Target Time

```
% FINDING DIFFERENCE FROM FINAL MEASUREMENT TO TARGET TIME
tTarg = 7*86400;
tf = filter.time(end);
dt = tTarg - tf;

% PROPAGATING STATE AND COVARIANCE
cspice_furnsh('data\naif0012.tls')
cspice_furnsh('data\de440.bsp')
[Xf, Fk] = propagate(filter.X(end, :).', dt, const, dataset(end).Time)
```

```
Xf = 6×1
       445.5
      -7100.2
      -183.85
        7.486
      0.46958
      0.17754
Fk = 6×6
       184.12      -192.6      -2.2951   1.8678e+05   1.7193e+05       8606.4
       13.851     -13.476     -0.12068        14481        12438       627.48
         4.45     -4.5857     -0.88608       4500.5       4111.3       725.57
    -0.010421    0.012036   0.00017769      -10.164      -10.362     -0.48017
      0.19365     -0.2048   -0.0024451       197.09       182.26       9.1419
    0.0051074   -0.005314  -0.00063523       5.2131       4.8098     -0.60415
```

```
Pf = Fk*filter.P(:, :, end)*Fk.' + Qdyn(sigmaA, dt, Xf);
cspice_kclear;
```

Extended Kalman Filter

```
function filter = filter_inputs(X0, runcase, sigmaR, sigmaV, sigmaA, Qdt, dataset, const, track
```

Handling Cases

```
% CREATING COVARIANCE
P0 = [sigmaR^2*eye(3) zeros(3);
      zeros(3)        sigmaV^2*eye(3)];

% LOADING SPICE DATA
cspice_furnsh('data\naif0012.tls')
cspice_furnsh('data\de440.bsp')

switch runcase
    case 1 % Range Only
        for i = 1:3
            const.Station(i).sigmas(2) = 1e5;
        end

    case 2 % Range-Rate Only
        for i = 1:3
            const.Station(i).sigmas(1) = 1e5;
        end

    case 3 % Station 1 Only
        for i = [2 3]
            const.Station(i).sigmas = [1e5 1e5];
        end

    case 4 % Station 2 Only
        for i = [1 3]
            const.Station(i).sigmas = [1e7 1e7];
        end
        X0 = const.X0filter;

    case 5 % Station 3 Only
        for i = [1 2]
            const.Station(i).sigmas = [1e5 1e5];
        end

    case 6 % Short-Arc
        if dataset(end).Time > 86400
            % REMOVING ALL BUT LAST DAY
            tspan = [dataset(:).Time].';
            dayidx = find(-(tspan - tspan(end)) <= 86400, 1, 'first');
            dataset = dataset(dayidx:end);

            % SETTING X0
            X0 = const.X0filter;
```

```
            % FORWARD PROPAGATING INITIAL STATE
            [X0, Fk] = propagate(X0, dataset(1).Time, const, 0);

            % UPDATING COVARIANCE
            P0 = Fk*P0*Fk.' + Qdyn(sigmaA.^2, dataset(1).Time, X0);
        end

    case 7 % Long-Arc
        % Nothing changes
end
```

Setup

```
% USEFUL CONSTANTS
datalen = length(dataset);
n = datalen-1;

% OUTPUT FORMATTING
filter = struct(...
    'X', zeros(datalen, 6), 'P', zeros(6, 6, datalen), ...
    'S', zeros(2, 2, datalen-1), ...
    'prefit', zeros(datalen-1, 2), 'postfit', zeros(datalen-1, 2),...
    'K', zeros(6, 2, datalen-1));

% INITIALIZING STATE
Xk = X0; filter.X(1, :)    = X0;

% INITIALIZING COVARIANCE
Pk = P0; filter.P(:, :, 1) = P0;

% BIASES FROM (several) Q = [0] RUN
% bias = [-0.0132720954326827 + 0.0003585901484998649 - 7.474404137772083e-05;
%          0.0985739905418150 - 0.0016617382305449250 - 0.0001341613023890341;
%          0.0237491481230446 - 0.0018490239370114080 + 7.439378411874604e-05];
% bias = [-0.0005702 0 0.01989].';
% bias = [-0.005-0.0025-0.001 0.005+0.0025+0.0025+0.005 0.015-0.0025-0.0025-0.005].';
% bias = zeros(3, 1);
bias = [0 0 0.01909];
```

Filtering

```
if trackiters; textprogressbar('Filtering: '); end
for i = 1:n%datalen-1
    if trackiters; textprogressbar(i/n * 100); end
    % CONSTANTS (for each loop)
    IDk = dataset(i).StationID;      % Station at current measurement
    IDk1 = dataset(i+1).StationID;   % Station at next measurement
    tk = dataset(i).Time;            % Time of current measurement
    tk1 = dataset(i+1).Time;         % Time of next measurement
    dt =  tk1 - tk;                  % Time difference between measurements
```

```
    Q = Qdyn(sigmaA.^2, dt/Qdt, Xk); % GammaQGamma Matrix

    % GETTING MEASURMENT
    zk1_actual = [dataset(i+1).Range;
                  dataset(i+1).RangeRate];


    % PREDICT
    [Xk1, Fk] = propagate(Xk, dt, const, tk);  % Extrapolated State and STM
    Pk1 = Fk*Pk*Fk.' + Q;                      % Extrapolated Covariance

    % VARIABLES NEEDED FOR UPDATE
    [rStation, vStation] = xStation(...   % Station State in ECI
        const.Station(IDk1), ...          % |
        const.Times.mjd0 + tk1/86400, ... % |
        const);                           % #
    Hk1 = Hgen(Xk1, rStation, vStation);  % Measurement Sensitivity Matrix
    sigSt = const.Station(IDk1).sigmas;   % Station Measurement Sigmas
    R = eye(2).*sigSt.^2;                    % Measurement Uncertainty Matrix
    zk1 = measurement(Xk1, ...            % Measurement from State
        const.Station(IDk1), ...          % |
        const.Times.mjd0 + tk1/86400, ... % |
        bias(IDk1), ...                   % | -- Station Range Bias
        const, ...                        % |
        tk1);                             % #
    prefit = zk1_actual - zk1;            % Pre-Fit Residuals

    % UPDATE
    % MALTAB Article
    Sk1 = Hk1*Pk1*Hk1.' + R;              % Innovation Covariance
    Kk1 = Pk1*Hk1.' / Sk1;                % Kalman Gain
    Xk1 = Xk1 + Kk1*prefit;               % Corrected State
    Pk1 = (eye(6) - Kk1*Hk1)*Pk1;         % Corrected Covariance
    zk1 = measurement(Xk1, ...            % Measurement from State
        const.Station(IDk1), ...          % |
        const.Times.mjd0 + tk1/86400, ... % |
        bias(IDk1), ...                   % | -- Station Range Bias
        const, ...                        % |
        tk1);                             % #
    postfit = zk1_actual - zk1;           % Post-Fit Residuals


    % SAVING
    filter.X(i+1, :) = Xk1;
    filter.P(:, :, i+1) = Pk1;
    filter.S(:, :, i) = Sk1;
    filter.K(:, :, i) = Kk1;
    filter.prefit(i, :) = prefit;
    filter.postfit(i, :) = postfit;
```

```
    % RESETTING INITIAL STATE AND COVARIANCE
    Xk = Xk1;
    Pk = Pk1;

end
if trackiters; textprogressbar(' Done'); end
cspice_kclear;
```

Formatting Output

```
% SAVING INPUTS TO FILTER
filter.inputs.X0 = X0;
filter.inputs.runcase = runcase;
filter.inputs.sigmaR = sigmaR;
filter.inputs.sigmaV = sigmaV;
filter.inputs.sigmaA = sigmaA;
filter.inputs.Qdt = Qdt;
filter.time = [dataset(:).Time];
end
```

Constants

Modelling Constants

```
const = struct;
```

Dynamical

```
% EARTH
const.Earth.Mu = 398600.4415;               % km3/s2
const.Earth.Radius = 6378.1363;             % km
const.Earth.J2 = 0.00108262545;             % n/a
const.Earth.RotVel =  7.292115146706979e-5; % rad/s
ld = load('data\EGM96.mat');
const.Earth.C = ld.C(1:21, 1:21);
const.Earth.S = ld.S(1:21, 1:21);
clear ld

% SUN
const.Sun.Mu = 132712440018;   % km3/s2
const.Sun.Dist =  149597870.7; % km (km/AU)

% MOON
const.Moon.Mu =  4902.800066; % km3/s2
const.Moon.Dist = 384472.282; % km

% DRAG VARIABLES
const.Drag.rho0 = 3.614e-13;                          % kg/m3
const.Drag.r0 = 700000.0 + const.Earth.Radius*1000; % m
const.Drag.H = 88667;                                 % m

% SRP CONSTANTS
const.SRP.P0 = 4.57e-6; % N/m2 @ 1 AU

% SPEED OF LIGHT
const.c = 299792458/1000; % km/s
```

Spacecraft

```
% SPACECRAFT FACE CONSTANTS
const.SC.Mass = 2000;         % kg
% const.SC.Faces = ["+X" "-X";
%                   "+Y" "-Y";
%                   "+Z" "-Z"];
const.SC.Area = [6  6;        % m
                 8  8;        % |
                 12 12];      % #
const.SC.Cd = [0.04 0.04;     % n/a
               0.04 0.04;     % |
               0.80 0.28];    % #
const.SC.Cs = [0.59 0.59;     % n/s
```

```
               0.59 0.59;     % |
               0.04 0.18];    % #

% SOLAR PANEL CONSTANTS
const.SolarPanel.Area = 15; % m
const.SolarPanel.Cd = 0.04; % n/a
const.SolarPanel.Cs = 0.04; % n/a
```

Assumptions

Not totally known values. Might need to be estimated like the state and CD

```
% STATION POSITIONS
% in km and km/s
const.Station(1) = struct('Name', 'Kwajalein', ...
                          'Position', [-6143584 1364250 1033743].'/1000, ...
                          'sigmas', [10/1000 0.5/1000/1000]);
const.Station(2) = struct('Name', 'Diego Garcia', ...
                          'Position', [1907295 6030810 -817119].'/1000, ...
                          'sigmas', [5/1000 1/1000/1000]);
const.Station(3) = struct('Name', 'Arecibo', ...
                          'Position', [2390310 -5564341 1994578].'/1000, ...
                          'sigmas', [10/1000 0.5/1000/1000]);

% SPACECRAFT ASSUMPTIONS
const.SC.CD = 1.88; % n/a

% INITIAL CONDITIONS
const.X0 = [6984.45711518852;   % km
            1612.2547582643;    % |
            13.0925904314402;   % #
            -1.67667852227336;  % km/s
            7.26143715396544;   % |
            0.259889857225218]; % #

% LEAST SQUARES OPTIMIZED STATE
const.X0lsq = [6978.6;
               1612.0;
                 13.098;
                 -1.6589;
                  7.263;
                  0.26872];

% IMPORTANT Times
const.Times.utc0 = datetime('2018-03-23 08:55:03');
const.Times.mjd0 = juliandate(const.Times.utc0, 'modifiedjuliandate');
const.Times.utcdv1 = datetime('2018-03-30 08:55:03');
const.Times.mjddv1 = juliandate(const.Times.utcdv1, 'modifiedjuliandate');
```

EOP Data

```
const.EOP = frame.readEOP([const.Times.utc0-calmonths(1), ...
                           const.Times.utc0+calmonths(1)]);
const.EOP = rmfield(const.EOP, 'func');
```

Nutation Data

```
n = 1000;
nut = zeros(n, 2);
mjdarray = linspace(const.EOP.data.mjd(1), const.EOP.data.mjd(end), n);
for i = 1:n
    nut(i, :) = earthNutation(2400000.5 + mjdarray(i) + 32.184/86400);
end
const.EOP.data.Nutation = nut;
```

Other

```
% tol = ;
const.etc.ODEOpts = odeset("RelTol", 1e-10, "AbsTol", 1e-10);
```

EOP Data Gatherer

Contents


```
function EOP = readEOP(utc, debug)
```

```
% READEOP collects and organizes coefficients used for ITRF->GCRF
%   transformation. Using C04 with Celestial Pole offsets (dPsi,dEps)
%   referred to IAU 1980 precession-nutation model.
%
% <strong>Function Calls:</strong>
%   - EOP = READEOP(utc)
%
% <strong>Required Inputs:</strong>
%   - utc::<strong>Vector{Datetime}</strong>: Array of dates to check. If there are
%       two elements, READEOP will output the per date data in EOP.data and
%       linearly interopolated functions in EOP.func. Otherwise READEOP
%       will output just the data at the corresponding dates.
%
% <strong>Outputs:</strong>
%   - EOP::<strong>Struct</strong>: Output struct
%       - EOP.mjd::<strong>Vector{Double}</strong>: Modified Julian Date
%       - EOP.deltaUT1::<strong>Vector{Double}</strong>: Difference between UTC and UT1
%       - EOP.pmx::<strong>Vector{Double}</strong>: Polar motion in x-direction
%       - EOP.pmy::<strong>Vector{Double}</strong>: Polar motion in y-direction
%       - EOP.dPsi::<strong>Vector{Double}</strong>: Delta psi for nutation
%       - EOP.dEps::<strong>Vector{Double}</strong>: Delta epsilon for nutation

if nargin < 2; debug = false; end

% SETUP
utc2mjd = @(utc) floor(juliandate(utc, 'modifiedjuliandate'));

% GATHERING DATA
% https://hpiers.obspm.fr/eop-pc/index.php?index=C04&lang=en
site = urlread("https://hpiers.obspm.fr/iers/eop/eopc04/eopc04.62-now");
site = strsplit(site, '\n').';
site = site(13:end-1);
tmp = str2num(char(site{1}));
data = zeros(length(site), length(tmp));
for i = 1:length(site)
    data(i, :) = str2num(char(site{i}));
end

% FINDING DATE BOUNDS
mjdbounds = [data(1, 4) data(end, 4)];
maxdatestring = sprintf('Limits are [%s, %s]', ...
    datestr(datetime(data(1, 1:3))), datestr(datetime(data(end, 1:3))) );

if debug; disp(site); end
usecase = length(utc);
```

COLLECTING VALUES

```
if usecase == 1     % Single-value output
    % CONVERTING JDATE
    mjd = utc2mjd(utc);
    if mjd < mjdbounds(1) || mjd > mjdbounds(2)
        error('Date pulled is out of bounds')
    end

    % PULLING FROM DATA
    out = collectvalues(data, mjd, debug);
    EOP = struct('mjd', mjd, 'deltaUT1', out.deltaUT1, ...
        'pmx', out.polarMotion(1), 'pmy', out.polarMotion(2), ...
        'dPsi', out.dPsi, 'dEps', out.dPsi);
elseif usecase == 2 % Range-output w/ curve-fits
    % CONVERTING TO MJD
    mjdrange = utc2mjd(utc);
    if any(mjdrange < mjdbounds(1) | mjdrange > mjdbounds(2))
        error('Date pulled is out of bounds\n%s', maxdatestring)
    end
    mjdrange = mjdrange(1):mjdrange(2);

    % PULLING DATA
    len = length(mjdrange);
    [EOP.data.mjd, EOP.data.deltaUT1, ...
     EOP.data.pmx, EOP.data.pmy, ...
     EOP.data.dPsi, EOP.data.dEps] = deal(zeros(len, 1));
    for i = 1:len
        out = collectvalues(data, mjdrange(i), debug);
        EOP.data.mjd(i) = mjdrange(i);
        EOP.data.deltaUT1(i) = out.deltaUT1;
        EOP.data.pmx(i) = out.polarMotion(1);
        EOP.data.pmy(i) = out.polarMotion(2);
        EOP.data.dPsi(i) = out.dPsi;
        EOP.data.dEps(i) = out.dEps;
    end

    % CREATING POLYNOMIALS
    fd = fields(EOP.data);
    for i = 1:length(fd)
        f = fit(EOP.data.mjd, EOP.data.(fd{i}), 'linearinterp');
        EOP.func.(fd{i}) = @(mjd) f(mjd);
    end
else                % Multi-value output
    % CONVERTING TO MJD
    mjdrange = utc2mjd(utc);
    if any(mjdrange < mjdbounds(1) | mjdrange > mjdbounds(2))
        error('Date pulled is out of bounds\n%s', maxdatestring)
    end

    % COLLECTING VALUES
    len = length(mjdrange);
    [EOP.mjd, EOP.deltaUT1, ...
     EOP.pmx, EOP.pmy, ...
     EOP.dPsi, EOP.dEps] = deal(zeros(len, 1));
    for i = 1:len
        out = collectvalues(data, mjdrange(i), debug);
        EOP.mjd(i) = mjdrange(i);
        EOP.deltaUT1(i) = out.deltaUT1;
        EOP.pmx(i) = out.polarMotion(1);
        EOP.pmy(i) = out.polarMotion(2);
        EOP.dPsi(i) = out.dPsi;
```

```
        EOP.dEps(i) = out.dEps;
    end
end
```

PULLS INFORMATION FROM MAIN DATA VARIABLE

```
function out = collectvalues(data, mjd, debug)
    % PULLING FROM CORRECT ROW
    idx = find(data(:, 4) == mjd, 1, 'first');
    data = data(idx, :);

    if debug; disp(data); end

    % COLLECTING INTO STRUCT
    out = struct;
    out.deltaUT1 = data(7);
    out.polarMotion = deg2rad(data(5:6)/3600);
    out.dPsi = deg2rad(data(9)/3600);
    out.dEps = deg2rad(data(10)/3600);
end
```

```
end
```

Published with MATLAB® R2022a

Propagator

```
function [Xf, STMf] = propagate(Xi, dt, const, t0)

% HANDLING INPUT
dt = dt(:).'; % Ensuring row vector

% DEFINING EOM
EoM = @(t, X) dynamics(t, X, const, t0);

% SETTING OPTIONS

% PROPAGATING
Xi = [Xi; reshape(eye(6), [], 1)]; % Adding STM initial conditions
[~, X] = ode113(EoM, [0 dt], Xi, const.etc.ODEOpts);

% HANDLING OUTPUTS
if nargout == 1
    % ONLY OUTPUTTING FINAL STATE
    if length(dt) == 1
        Xf = X(end, 1:6).';

    % OUTPUTTING ALL PROPAGATED STEPS
    else
        Xf = X(:, 1:6);

    end

else
    % ONLY OUTPUTTING FINAL STATE AND STM
    if length(dt) == 1
        Xf = X(end, 1:6).';
        STMf = reshape(X(end, 7:end), 6, 6);


    % OUTPUTTING ALL PROPAGATED STEPS
    else
        Xf = X(:, 1:6);
        len = size(X, 1);
        STMf = zeros(6, 6, len);
        for i = 1:len
            STMf(:, :, i) = reshape(X(i, 7:end), 6, 6);
        end

    end
end
```

Published with MATLAB® R2022a

Dynamics

Spacecraft Dynamics Function

```
function dX = dynamics(t, X, const, t0)

jd = const.Times.mjd0 + 2400000.5 + t0/86400 + t/86400;
et = cspice_str2et(sprintf('JD%0.16f', jd));
rS = mice_spkezr('SUN', et, 'J2000', 'NONE', 'EARTH').state(1:3);
```

Gravity

```
% TWO-BODY
% grav = eom.Kep(t, X, const.Earth.Mu);

% J2
% grav = eom.KepJ2(t, X, const.Earth.Mu, const.Earth.J2, const.Earth.Radius);

% EGM
[~, DCM] = ecef2eci_(X(1:3), const.Times.mjd0 + t0/86400 + t/86400, const);
units = 1;
Xecef = [DCM.'*X(1:3);
         DCM.'*X(4:6)]*units;
[xgrav, ygrav, zgrav] = EGM96_mex(Xecef(1), Xecef(2), Xecef(3), ...
    const.Earth.Mu*units^3, const.Earth.Radius*units, ...
    const.Earth.C, const.Earth.S);
grav = DCM*[xgrav; ygrav; zgrav]/units;
grav = [X(4:6); grav];
```

Drag

```
% NONE
% drag = zeros(6, 1);

% CANNONBALL
area = scArea(X, rS, const); % TODO: Function to calculate
drag = eom.Drag(t, X, const.SC.CD, area, const.SC.Mass, ...
    const.Drag.rho0, const.Drag.r0, const.Drag.H, const.Earth.RotVel);

% BOX-WING
% drag = zeros(6, 1);
```

SRP

```
% NONE
% srp = zeros(6, 1);

% CONSTANT CALCULATION
isOcculted = occultation(X(1:3), rS, const); % Checking for occulations from Earth
rSP = rS - X(1:3);                           % Pointing vector of solar panel face
area = const.SolarPanel.Area*~isOcculted;    % Zero-ing area if in occulusion
```

```
CR = 1;                                          % Combined reflectivity (0 = transparent, 2 = tota

% CANNONBALL
% srp = eom.SRP(t, X, CR, const.SolarPanel.Area, const.SC.Mass, rS);

% SPECULATIVE-DIFFUSIVE
srp = eom.SRP_SD(t, X, ...
    const.SolarPanel.Cd, const.SolarPanel.Cs, area, ...
    const.SC.Mass, rSP, rS);
```

Third Body

```
% NONE
thirdbody = zeros(6, 1);

% MOON
rM = mice_spkezr('MOON', et, 'J2000', 'NONE', 'EARTH').state(1:3);
thirdbody = thirdbody + eom.ThirdBody(t, X, const.Moon.Mu, rM);

% SUN
thirdbody = thirdbody + eom.ThirdBody(t, X, const.Sun.Mu, rS);
```

State Transition Matrix

```
% GETTING A FROM A FUNCTION;
% A = Agen(X(1:6), const.Earth.Mu); % 2-Body
% A = Agen(X(1:6), const.Earth.Mu, const.Earth.J2, const.Earth.Radius); % J2
% A = Agen(X(1:6), const.Earth.Mu, const.Earth.J2, const.Earth.Radius, ... % J2 + Drag
%     const.SC.CD, area, const.SC.Mass, const.Drag.rho0, const.Drag.r0, const.Drag.H, const.Ear
% A = Agen(X(1:6), const.Earth.Mu, const.Earth.J2, const.Earth.Radius, ... % J2 + Drag + SRP
%     const.SC.CD, area, const.SC.Mass, const.Drag.rho0, const.Drag.r0, const.Drag.H, const.Ear
%     CR, rS);
A = Agen(X(1:6), const.Earth.Mu, const.Earth.J2, const.Earth.Radius, ... % J2 + Drag + SRP + 3E
    const.SC.CD, area, const.SC.Mass, const.Drag.rho0, const.Drag.r0, const.Drag.H, const.Earth
    CR, rS, ...
    const.Sun.Mu, rM, const.Moon.Mu);
% A = eye(6);

% UPDATING STM
Phi = reshape(X(7:end), 6, 6);
Phidot = A*Phi;
```

Combining Accelerations

```
dX = [grav + drag + srp + thirdbody; reshape(Phidot, [], 1)];
end
```

J2 Equations of Motion

```
function dX = KepJ2(~, X, mu, J2, ae)
% KEPJ2 is the equations of motion to propagate the state (X) over time
%   interval (t), under the inertially-fixed, spherical harmonics
%   graviational force model.
%
% <strong>Function Calls:</strong>
%   - dX = KEPJ2(t, X, mu, J2, ae)
%
% <strong>Required Inputs:</strong>
%   - t::<strong>Double</strong>: Current time of propagation related to
%       the state
%   - X::<strong>Vector{Double}</strong>: State vector at propgation time
%   - mu::<strong>Double</strong>: Standard Gravitational Parameter
%   - J2::<strong>Double</strong>: Oblateness Factor
%   - ae::<strong>Double</strong>: Reference Distance
%
% <strong>Outputs:</strong>
%   - dX::<strong>Vector{Double}</strong>: Derivative of input vector at
%       propagation time

% PULLING VALUES FROM STATE
r = sqrt(sum(X(1:3).^2));
R = X(1:3);

% GENERATING CONSTANTS
J0 = 1.5*mu*J2*ae^2;
Jr = (J0 / r^5)*(1 - (5 / r^2)*dot(R, [0; 0; 1])^2);
Jk = 2*(J0 / r^5)*dot(R, [0; 0; 1]);
P = Jr*R + Jk*[0; 0; 1];

% DEFINING DERIVATIVE
dX = [X(4:6);
    -mu/r^3*R - P];
end
```

Published with MATLAB® R2022a

EGM96 Equations of Motion

```
function [agx_ECEF,agy_ECEF,agz_ECEF]=EGM96_2(x,y,z,mu,RE,C,S)

%  This function computes the gravitational accelerations using a
%  spherical harmonic expansion of the geopotential.  The geopotential
%  is expressed in ECEF coordinates, thus the input coordinates need to
%  be in ECEF.  The acceleration must be expressed in inertial
%  coordinates for ease of integration. The gravity coefficients are
%  assumed to already be normalized.
%
%  Reference:  Pines, Samuel, "Uniform Representation of the Gravitational
%              Potential and its Derivatives," AIAA Journal, Vol. 11,
%              No. 11, November, 1973, pp. 1508-1511.
%
%  INPUTS:
%     x,y,z     Spacecraft coordinates in ECI coordinates
%     mu        Earth's Gravitational Constant
%     RE        Earth's Mean Equitorial Radius
%     C(i,j)    A matrix containing the C coefficients,
%               which are used in the spherical harmonic
%               expansion of the geopotential
%     S(i,j)    A matrix containing the S coefficients,
%               which are similar to C coefficients.
%  OUTPUTS:
%     agx_ECEF   x-component (ECEF) of the acceleration vector
%     agy_ECEF   y-component (ECEF) of the acceleration vector
%     agz_ECEF   z-component (ECEF) of the acceleration vector
%
% Initialize a few variables and determine their size

[nmaxp1,mmaxp1] = size(C);
nmax = nmaxp1-1;
mmax = mmaxp1-1;
Anm = zeros(nmaxp1+1,1);
Anm1 = zeros(nmaxp1+1,1);
Anm2 = zeros(nmaxp1+2,1);
R = zeros(nmaxp1+1,1);
I = zeros(nmaxp1+1,1);
rb2 = x*x + y*y + z*z;
rb = sqrt(rb2);
mur2 = mu/rb2;
mur3 = mur2/rb;

% direction of spacecraft position
s = x/rb;
t = y/rb;
u = z/rb;

%       /* ********************************************************* */
%       /* Calculate contribution of only Zonals */
%       /* ********************************************************* */

Anm1(1) = 0;
Anm1(2) = sqrt(3);
Anm2(2) = 0;
Anm2(3) = sqrt(3.75);
as = 0;
at = 0;
au = 0;
ar = 0;
```

```
rat1 = 0;
rat2 = 0;
Dnm = 0;
Enm = 0;
Fnm = 0;
Apor = zeros(nmaxp1,1);
Apor(1) = 1;
Apor(2) = RE/rb;

for n = 1:nmax,
    i = n+1;
    an2 = 2*n;
    rat1 = sqrt((an2+3.0)*(((an2+1.0)/n)/(n+2.0))) ;
        rat2 = sqrt((n+1.0)*(((n-1.0)/(an2-1.0))/(an2+1.0))) ;
        Anm1(i+1) = rat1*(u*Anm1(i) - rat2*Anm1(i-1)) ;
        Apor(i) = Apor(i-1)*Apor(2) ;
    if n < mmaxp1
        rat1 = sqrt((an2+5.0)*(((an2+3.0)/n)/(n+4.0))) ;
        rat2 = sqrt((n+3.0)*(((n-1.0)/(an2+1.0))/(an2+3.0))) ;
        Anm2(i+2) = rat1*(u*Anm2(i+1) - rat2*Anm2(i)) ;
    end
    if n < nmaxp1
        rat1 = sqrt(0.5*n*(n+1.0)) ;
        au = au - Apor(i)*rat1*Anm1(i)*(-C(i,1)) ;
        rat2 = sqrt(0.5*((an2+1.0)/(an2+3.0))*(n+1.0)*(n+2.0)) ;
        ar = ar + Apor(i)*rat2*Anm1(i+1)*(-C(i,1)) ;
    end
end

%   /* ********************************************************** */
%       /* Calculate contribution of Tesserals */
%       /* ********************************************************** */

R(1) = 1;
I(1) = 0;
for m = 1:mmax,
    j = m+1;
    am2 = 2*m;
    R(j) = s*R(j-1) - t*I(j-1);
    I(j) = s*I(j-1) + t*R(j-1);
    for l = m:mmax,
        i = l+1;
        Anm(i) = Anm1(i) ;
        Anm1(i) = Anm2(i) ;
    end
    Anm1(mmaxp1) = Anm2(mmaxp1) ;
    for l = m:mmax,
        i = l+1;
        an2 = 2*l;
        if l == m
            Anm2(j+1) = 0.0 ;
            Anm2(j+2) = sqrt((am2+5.0)/(am2+4.0))*Anm1(j+1) ;
        else
            rat1 = sqrt((an2+5.0)*(((an2+3.0)/(l-m))/(l+m+4.0))) ;
            rat2 = sqrt((l+m+3.0)*(((l-m-1.0)/(an2+1.0))/(an2+3.0))) ;
            Anm2(i+2) = rat1*(u*Anm2(i+1) - rat2*Anm2(i)) ;
        end
        Dnm = C(i,j)*R(j) + S(i,j)*I(j) ;
        Enm = C(i,j)*R(j-1) + S(i,j)*I(j-1) ;
        Fnm = S(i,j)*R(j-1) - C(i,j)*I(j-1) ;
```

```
        rat1 = sqrt((l+m+1.0)*(l-m)) ;
        rat2 = sqrt(((an2+1.0)/(an2+3.0))*(l+m+1.0)*(l+m+2.0)) ;

        as = as + Apor(i)*m*Anm(i)*Enm ;
        at = at + Apor(i)*m*Anm(i)*Fnm ;
        au = au + Apor(i)*rat1*Anm1(i)*Dnm ;
        ar = ar - Apor(i)*rat2*Anm1(i+1)*Dnm ;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute the spacecraft accelerations in ECEF %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

agx_ECEF = -mur3*x + mur2*(as + ar*s);
agy_ECEF = -mur3*y + mur2*(at + ar*t);
agz_ECEF = -mur3*z + mur2*(au + ar*u);
```

Published with MATLAB® R2022a

ECEF to ECI Transform

Contents


```
function [rECI, DCM] = ecef2eci_(rECEF, mjd, const)
```

```
%ECEF2ECI Summary of this function goes here
%   Detailed explanation goes here
```

BUILDING DCM

USING EOP CURVE FITS deltaUT1 = const.EOP.func.deltaUT1(mjd); pm = [const.EOP.func.pmx(mjd), ... const.EOP.func.pmy(mjd)]; dEps = const.EOP.func.dEps(mjd); dPsi = const.EOP.func.dPsi(mjd);

```
% INTERPOLATING
deltaUT1 = interp1(const.EOP.data.mjd, const.EOP.data.deltaUT1, mjd);
pmx = interp1(const.EOP.data.mjd, const.EOP.data.pmx, mjd);
pmy = interp1(const.EOP.data.mjd, const.EOP.data.pmy, mjd);
dEps = interp1(const.EOP.data.mjd, const.EOP.data.dEps, mjd);
dPsi = interp1(const.EOP.data.mjd, const.EOP.data.dPsi, mjd);

% GETTING NUTATION DATA
mjdarray = linspace(const.EOP.data.mjd(1), const.EOP.data.mjd(end), 1000);
nut(1) = interp1(mjdarray, const.EOP.data.Nutation(:, 1), mjd);
nut(2) = interp1(mjdarray, const.EOP.data.Nutation(:, 2), mjd);
% nut_ = earthNutation(2400000.5 + mjd + 32.184/86400);

% CREATING DCM
dt = datetime(mjd, 'ConvertFrom', 'modifiedjuliandate');
DCM = dcmeci2ecef__mex(mjd, datevec(dt), deltaUT1, pmx, pmy, dEps, dPsi, nut);
% DCM = dcmeci2ecef('IAU-76/FK5', datevec(dt), 0, deltaUT1, [pmx, pmy], ...
%     'DNUTATION', [dPsi dEps]);
%
% fprintf('mjd: %0.6f -- |DCMerr| = %0.6g -- |nuterr| = %0.6g\n', mjd, norm(DCM - DCM_, 'fro'), norm(nut - nut_));

% ROTATING TO MAKE ECEF->ECI
DCM = DCM.';
```

PERFORMING ROTATION

```
rECI = DCM*rECEF;
```

```
end
```

Published with MATLAB® R2022a

ECEF to ECI Directional Cosine Matrix

Contents


```
function DCM = dcmeci2ecef_(mjd, UTC, deltaUT1, pmx, pmy, dEps, dPsi, nutationAngles)
```

```
%   DCMECI2ECEF Convert Earth-centered inertial (ECI) to Earth-centered
%   Earth-fixed (ECEF) coordinates.
%   DCM = DCMECI2ECEF( REDUCTION,UTC,DELTAAT,DELTAUT1,POLARMOTION,'ADDPARAMNAME',ADDPARAMVALUE )
%   calculates the position direction cosine matrix (ECI to ECEF), for
%   given set of time and geophysical data, UTC, DELTAAT, DELTAUT1,
%   POLARMOTION, and DNUTATION or DCIP.
%
%   Inputs arguments for DCMECI2ECEF are:
%   REDUCTION:      String indicating the reduction process through which
%                   the function calculates the direction cosine matrix.
%                   It can either be IAU-76/FK5 (which uses the IAU 1976
%                   Precession Model and the IAU 1980 Theory of Nutation,
%                   which is no longer current but some programs still use
%                   this reduction) or IAU-2000/2006 (which uses the P03
%                   precession model). The reduction method you select
%                   determines the ADDPARAMNAME parameter pair
%                   characteristics. The IAU-76/FK5 method returns a
%                   transformation matrix that does not have the
%                   characteristics of a DCM due to the polar motion
%                   approximation.
%   UTC:            Array of Universal Coordinated Time (UTC) in year,
%                   month, day, hour, minutes and seconds for which the
%                   function calculates the transformation matrix. Define
%                   the array as one of the following: Array with 1 row and
%                   6 columns or M by 6 array for M transformation
%                   matrices, one for each UTC date. Values for year,
%                   month, day, hour and minutes should be whole numbers.
%   DELTAAT:        Difference in seconds between the International Atomic
%                   Time (TAI) and UTC. It can be defined as either a
%                   scalar or a one dimensional array with M elements (if M
%                   UTC dates defined) for equal number of transformation
%                   matrices. The default value is an M by 1 null array.
%   DELTAUT1:       Difference, in seconds, between UTC and Universal Time
%                   (UT1). It can be defined as either a scalar or a one
%                   dimensional array with M elements (if M UTC dates
%                   defined) for equal number of transformation matrices.
%                   The default value is an M by 1 null array.
%   POLARMOTION:    Polar displacement due to the motion of the Earth's
%                   crust, in radians, along the x and y axis. It can be
%                   defined as either a 1 by 2 array or an M by 2 (if M UTC
%                   dates defined) for equal number of transformation
```

```
%                   matrices. The default value is an M by 2 null array.
%
%   Input parameter Name/Value pair for 'ADDPARAMNAME' and ADDPARAMVALUE
%   are:
%   'DNUTATION':    (IAU-76/FK5 reduction only) M by 2 array for the
%                   adjustment in radians to the longitude (dDeltaPsi) and
%                   obliquity (dDeltaEpsilon). The default value is an M by
%                   2 null array.
%   'DCIP':         (IAU-2000/2006 reduction only) M by 2 array for the
%                   adjustment in radians to the location of the Celestial
%                   Intermediate Pole (CIP) along the x (dDeltaX) and y
%                   (dDeltaY) axis. The default value is an M by 2 null
%                   array.
%
%   For historical values for DNUTATION and DCIP, see the International
%   Earth Rotation and Reference Systems Service (IERS) website
%   (http://www.iers.org) under the 'Earth Orientation Data' product.
%
%   Output calculated by DCMECI2ECEF is:
%   DCM:    3 by 3 by M array where M is the number of dates for which the
%           function calculates the transformation matrix.
%
%   Examples:
%   Calculate the position direction cosine matrix (ECI to ECEF), using the
%   IAU-76/FK5 reduction, for one UTC date with all parameters defined.
%
%   DCM = dcmeci2ecef('IAU-76/FK5',[2000 1 12 4 52 12.4],32,0.234,[-0.0682e-5 ...
%                     0.1616e-5],'dNutation',[-0.2530e-6 -0.0188e-6])
%
%   Calculate the position direction cosine matrix (ECI to ECEF), using the
%   IAU-2000/2006 reduction, for two UTC dates. All other parameters
%   default to null arrays.
%
%   DCM = dcmeci2ecef('IAU-2000/2006',[2000 1 12 4 52 12.4;2000 1 12 4 52 13])
%
%
%   See also LLA2ECEF, ECEF2LLA, GEOC2GEOD, GEOD2GEOC, LLA2ECI, ECI2LLA,
%   DELTAUT1, DELTACIP, POLARMOTION.



%   References:
%   [1]Petit, Gerard and Brian Luzum. IERS Conventions (2010) - IERS
%   Technical note 36. Verlag des Bundesamts fur Kartographie und Geodasie.
%   Frankfurt am Main, 2010.
%   [2]Vallado, D. A., Fundamentals of Astrodynamics and Applications,
%      McGraw-Hill, New York, 1997.
```

Common time calculations

Calculations determining the number of Julian centuries for terrestrial time (tTT) and UT1.

```
len = 1;

% DEFINING ROTATIONS
R1 = @(angle) [ 1  0           0          ;
       0  cos(angle)  sin(angle);
       0 -sin(angle)  cos(angle) ];
```

```
R2 = @(angle) [ cos(angle)  0 -sin(angle);
       0           1  0          ;
       sin(angle)  0  cos(angle) ];

R3 = @(angle) [ cos(angle)  sin(angle)  0;
      -sin(angle)  cos(angle)  0;
       0            0           1 ];

% Getting Time from MJD
year = UTC(:,1);
month = UTC(:,2);
day = UTC(:,3);
hour = UTC(:,4);
min = UTC(:,5);
sec = UTC(:,6);

% Seconds for UT1
ssUT = sec + deltaUT1;
% Seconds for UTC
% ssTT = sec + 32.184;
% Julian date for terrestrial time
jdTT = mjd + 32.184/86400;
% Number of Julian centuries since J2000 for terrestrial time.
tTT = (jdTT - 51544.5)/36525;
tTT2 = tTT.*tTT;
tTT3 = tTT2.*tTT;
```

IAU-76/FK-5 based transformation

This transformation is based on the process described by Vallado (originally McCarthy).

```
% Additional time calculations:
jdUT1 = floor(mjd);
tUT1 = (jdUT1 - 51544.5)/36525;
tUT12 = tUT1.*tUT1;
tUT13 = tUT12.*tUT1;
```

Sidereal time

Greenwich mean sidereal time at midnight

```
thGMST0h = 100.4606184 + 36000.77005361*tUT1 + 0.00038793*tUT12 - 2.6e-8*tUT13;
% Ratio of universal to sidereal time
omegaPrec = 1.002737909350795 + 5.9006e-11*tUT1 - 5.9e-15*tUT12;
% Elapsed universal time since midnight to the time of the
% observation
UT1 = hour*60*60 + min*60 + ssUT;
% Greenwich mean sidereal time at time of the observation
thGMST = mod(thGMST0h + (360/(24*60*60))*omegaPrec.*UT1,360);
```

Nutation

Mean obliquity of the ecliptic

```
epsilonBar = deg2rad(23.439291 - 0.0130042*tTT - 1.64e-7*tTT2 + 5.04e-7*tTT3);
% Nutation angles obtained using JPL data
```

```
% nutationAngles = earthNutation(2400000.5+jdTT);
dpsi = nutationAngles(:,1); %Nutation in Longitude
depsilon = nutationAngles(:,2); %Nutation in obliquity
% Adjustments to nutation angles (provided from real measurements)
dpsi = dpsi + dPsi;
depsilon = depsilon + dEps;
%The last two terms for equation of the equinoxes are only included
%if the date is later than January 1, 1997 (MJD=50449)
omegaMoon = deg2rad(125.04455501 - (5*360 + 134.1361851)*tTT ...
                            + 0.0020756*tTT2 + 2.139e-6*tTT3);
omegaMoon(jdUT1<50449) = 0;
% Equation of the equinoxes
equinoxEq = dpsi.*cos(epsilonBar) + deg2rad(0.00264/3600)*...
    sin(omegaMoon) + deg2rad(0.000063/3600)*sin(2*omegaMoon);
% Greenwhich apparent sidereal time
thGAST = thGMST*pi/180 + equinoxEq;
% Transformation matrix for earth rotation
% R = angle2dcm(thGAST,zeros(len,1),zeros(len,1));
R = R3(thGAST);
% True obliquity of ecliptic
epsilon = epsilonBar+depsilon;
% True equator to mean equinox date transformation matrix
% N = angle2dcm(epsilonBar,-dpsi,-epsilon,'XZX');
N = R1(-epsilon)*R3(-dpsi)*R1(epsilonBar);
```

Precession

Zeta, theta and z represent the combined effects of general precession.

```
zeta = deg2rad((2306.2181*tTT + 0.30188*tTT2 + 0.017998*tTT3)/3600);
theta = deg2rad((2004.3109*tTT - 0.42665*tTT2 - 0.041833*tTT3)/3600);
z = deg2rad((2306.2181*tTT + 1.09468*tTT2 + 0.018203*tTT3)/3600);
% Mean equinox to celestial reference frame
% P = angle2dcm(-zeta,theta,-z,'ZYZ');
P = R3(-z)*R2(theta)*R3(-zeta);
```

Polar motion

```
W = repmat(eye(3),1,len);
W = reshape(W,3,3,len);
W(1,3,:) = pmx;
W(3,1,:) = -pmx;
W(2,3,:) = -pmy;
W(3,2,:) = pmy;
```

Matrix calculationsc

tmp = arrayfun(@(k) ((W(:,:,k))*R(:,:,k)*N(:,:,k)*P(:,:,k)),1:len,'UniformOutput',false); DCM = reshape(cell2mat(tmp),3,3,len);

```
DCM = W*R*N*P;
```

Outputs

varargout{1} = DCM;

Drag Equations of Motion

```
function dX = Drag(~, X, CD, A, m, rho0, r0, H, thetadot)
% DRAG is the equations of motion to propagate the state (X) over time
%   interval (t), under the inertially-fixed, cannonball drag model of
%   spacecraft drag.
%
% <strong>Function Calls:</strong>
%   - dX = DRAG(t, X, CD, A, m, rho0, r0, H, thetadot)
%
% <strong>Required Inputs:</strong>
%   - t::<strong>Double</strong>: Current time of propagation related to
%       the state. [Unused]
%   - X::<strong>Vector{Double}</strong>: State vector at propgation time  (km & km/s)
%   - CD::<strong>Double</strong>: Coefficient of drag for spacecraft
%   - A::<strong>Double</strong>: Cross-sectional area normal to velocity  (m^2)
%   - m::<strong>Double</strong>: Mass of spacecraft                        (kg)
%   - rho0::<strong>Double</strong>: Atmospheric density at height of r0   (kg/m^3)
%   - r0::<strong>Double</strong>: See above                                (m)
%   - H::<strong>Double</strong>: Scale height for atmosphere model        (m)
%   - thetadot::<strong>Double</strong>: Rotational rate of planet         (1/sec)
%
% <strong>Outputs:</strong>
%   - dX::<strong>Vector{Double}</strong>: Derivative of input vector at   (km/s^2)
%       propagation time

% CONSTANT
km2m = 1000;

% VELOCITY W.R.T. ATMOSPHERE
vvec = [X(4) + thetadot*X(2);
        X(5) - thetadot*X(1);
        X(6)];
v = sqrt(sum(vvec.^2));

% CALCULATING DENSITY
rho = rho0*exp(-(km2m*norm(X(1:3)) - r0)/H); % 1000 converts from km -> m

% CALCULATING ACCELERATION
dX = [zeros(3, 1);
      -km2m*0.5*CD*(A/m)*rho*(v*vvec)]; % 1000 accounts for extra m from density
end
```

Published with MATLAB® R2022a

Spacecraft Area Calculation

Calculate Spacecraft Area -- Drag

Using poor mans calculation of area (still better than before) $A_{transform} = Acos\theta$

```
function area = scArea(X, rS, const)

    % FINDING DIRECTION W.R.T ATMOSPHERE
    vvec = [X(4) + const.Earth.RotVel*X(2);
            X(5) - const.Earth.RotVel*X(1);
            X(6)];
    vhat = vvec/norm(vvec);

    % CALCULATING RIC FRAME
    [~, ~, DCM] = cart2ric(X(1:3), X(4:6));

    % FINDING ANGLE BETWEEN VELOCITY AND SUN VECTOR
    v2s = frame.u2vAng(vhat, rS, 'd', 'inside');

    % CALCULATING PROJECT SOLAR PANEL AREA
    Asp = abs(const.SolarPanel.Area*cosd(v2s));

    % CALCULATING BODY AREAS
    % X-Face
    xhat = DCM.'*[0; 1; 0];
    v2x = frame.u2vAng(xhat, vhat, 'd', 'inside');

    % Y-Face
    yhat = DCM.'*[1; 0; 0];
    v2y = frame.u2vAng(yhat, vhat, 'd', 'inside');

    % Z-Face
    zhat = DCM.'*[0; 0; 1];
    v2z = frame.u2vAng(zhat, vhat, 'd', 'inside');

    % Adding
    Abd = abs(const.SC.Area(1, 1)*cosd(v2x)) + abs(const.SC.Area(2, 1)*cosd(v2y)) + abs(const.S

    % COMBINING AREAS
    area = Asp + Abd;
%     area = 20;

end
```

Cartesian to RIC Transform

```
function [r, v, DCM] = cart2ric(r, v)
%CART2RIC Summary of this function goes here
%   Detailed explanation goes here

% DEFINING RIC UNIT VECTORS
rhat = r / sqrt(sum(r.^2));
h = cross(r, v);
chat = h / sqrt(sum(h.^2));
ihat = cross(chat, rhat);

% MAKING ROTATION MATRIC
DCM = [rhat ihat chat].';

% APPLYING ROTATION
r = DCM*r;
v = DCM*v;

end
```

Published with MATLAB® R2022a

Vector-to-Vector Angle

```
function angle = u2vAng(u, v, units, type)

% ARGUMENT VALIDATION
arguments
    u {mustBeNumeric}
    v {mustBeNumeric}
    units {mustBeTextScalar} = 'degrees'
    type {mustBeTextScalar, mustBeMember(type, ...
        ["clockwise", "inside", "counterclockwise"])} = 'clockwise'
end

% Units
if strncmpi(units, 'degrees', 1)
    units = 180/pi;
else
    units = 1;
end

% Angle Type
if strcmpi(type, 'inside')
    func = @(angle) -angle;
else
    func = @(angle) 2*pi - angle;
end


% Handling Column Vecs
if all(size(u) == [3 1])
    u = u(:).';
    v = v(:).';
end

% CALCULATING ANLGES
angle = zeros(size(u, 1), 1);
for i = 1:size(u, 1)
    % ENFORCING COLUMN VECS
    u_ = u(i, :).';
    v_ = v(i, :).';

    % FINDING VECTOR PRODUCTS
    c = cross(u_, v_);
    d = dot(u_, v_);

    % FINDING ANGLE
    angle(i) = atan2(norm(c), d);

    % ACCOUNTING FOR QUADRENT
    if c(3) < 0
        angle(i) = func(angle(i));
    end
end

% OUTPUTTING
angle = angle*units;

end
```

Solar Radiation Pressure Equations of Motion

```
function dX = SRP_SD(~, X, Cd, Cs, A, m, rBody, rS)
% SRP_SD is the equations of motion to propagate the state (X) over time
%   interval (t), under the inertially-fixed, sollar radiation pressure.
%
% From Vallado Section 8.6.4 pg. 582 (610)
%
% <strong>Function Calls:</strong>
%   - dX = SRP_SD(t, X, Cd, Cs, A, m, rS)
%
% <strong>Required Inputs:</strong>
%   - t::<strong>Double</strong>: Current time of propagation related to
%       the state. [Unused]
%   - X::<strong>Vector{Double}</strong>: State vector at propgation time  (km & km/s)
%   - Cd::<strong>Double</strong>: Diffusive reflection coefficient        (ND)
%   - Cs::<strong>Double</strong>: Specular reflection coefficient         (ND)
%   - A::<strong>Double</strong>: Affected area                            (m^2)
%   - m::<strong>Double</strong>: Spacecraft mass                          (kg)
%   - rBody::<strong>Vector{Double}</strong>: Unit vector normal to face   (ND)
%   - rS::<strong>Vector{Double}</strong>: Radius from coordinate origin   (km)
%       to Sun
%
% <strong>Outputs:</strong>
%   - dX::<strong>Vector{Double}</strong>: Derivative of input vector at
%       propagation time

% PRESSURE AT 1 AU
AU = 149597871;
Psrp = 4.57e-6; % N/m^2 (For absorbative material, 2x for reflective)

% VECTOR FROM SATELLITE TO SUN
rsc2S = X(1:3) - rS;
Rsc2S = sqrt(sum(rsc2S.^2));
rsc2S = rsc2S / Rsc2S;          % Unit Vectorizing

% SCALING PRESSURE BY DISTANCE FROM REFERENCE
ratio = Rsc2S / AU;
scale = 1/ratio^2;
Psrp = scale*Psrp;

% GETTING UNIT VECTORS
n = rBody / norm(rBody);
s = -rsc2S;
phi = abs(frame.u2vAng(s, n, 'd', 'inside'));

% CALCULATING SPECULAR FORCE
frs = -2*Psrp*Cs*A*cosd(phi)^2*n;

% CALCULATING DIFFUSIVE FORCE
frd = -Psrp*Cd*A*cosd(phi)*((2/3)*n + s);

% ACCOUNTING FOR UNITS
m2km = 1/1000;          % Accounts for extra m in Psrp

% CONSTRUCTING ACCELERATION
dX = [zeros(3, 1);
      m2km*(frs + frd)/m];


end
```

Occultation Check

```
function isOcculted = occultation(rSC, rS, const)

angle = frame.u2vAng(rS, rSC, 'deg', "inside"); % Interior angle between Earth and Sun vectors

if angle > 90
    dist = norm(rSC)*sind(angle); % Normal distance from Sun Vector to Spacecraft Vector
    isOcculted = dist - const.Earth.Radius < 0;

else
    isOcculted = false;

end
end
```

Published with MATLAB® R2022a

Process Noise Matrix (Q) from RIC to ECI

Contents


```
function Q = Qdyn(sig, dt, X)
```

```
%Q Summary of this function goes here
%   Detailed explanation goes here

% sig = sig^2;

if length(sig) == 1; sig = sig*ones(3, 1); end
```

CALCULATING Q in RIC

INCORPORTATING TIME VARYING DYNAMICS

```
Q = dt^2 * ...
    [0.25*dt^2,  0,          0,          0.5*dt,  0,       0;
     0,          0.25*dt^2,  0,          0,       0.5*dt,  0;
     0,          0,          0.25*dt^2,  0,       0,       0.5*dt;
     0.5*dt,     0,          0,          1,       0,       0;
     0,          0.5*dt,     0,          0,       1,       0;
     0,          0,          0.5*dt,     0,       0,       1];

% ADDING UNKNOWN DYNAMICS TERM
sigvec = [sig; sig];
Q = Q.*sigvec;
```

CONVERTING TO ECI

```
[~, ~, DCM] = cart2ric(X(1:3), X(4:6));
DCM = DCM.'; % RIC to ECI
DCM = [DCM zeros(3); zeros(3) DCM];
Q = DCM*Q*DCM.';
```

```
end
```

Published with MATLAB® R2022a

State to Measurement Conversion

Contents


```
function z = measurement(X, Station, mjd, bias, const, t0)
```

LIGHT-TIME FREE RANGE

SPACECRAFT STATE

```
r = X(1:3);
v = X(4:6);

% GETTING STATION STATE IN ECI
[rStation, vStation] = xStation(Station, mjd, const);

% DISPLACEMENT VECTOR
rS2SC = r - rStation;
vS2SC = v - vStation;

% FINDING RANGE
range = sqrt(sum(rS2SC.^2)) + bias;
```

LIGHT-TIME INCLUDED RANGE

LIGHT TIME CALCULATION

```
lt = range/const.c;

% UPDATING STATE BASED ON LIGHT TIME
[Xlt, ~] = propagate(X, -lt, const, t0);
r = Xlt(1:3);
v = Xlt(4:6);

% ACCOUNTING FOR STATION ABBERATION
[rStation, vStation] = xStation(Station, mjd - lt/86400, const);

% UPDATED DISPLACEMENT VECTOR
rS2SC = r - rStation;
vS2SC = v - vStation;

% UPDATED RANGE
range = sqrt(sum(rS2SC.^2)) + bias;

% FINDING MEASUREMENT
rangerate = (rS2SC.' * vS2SC) / range;
```

OUTPUTTING

```
z = [range;
     rangerate];
```

Station Position in ECI

```
function [rStation, vStation] = xStation(Station, mjd, const)
%VSTATION Summary of this function goes here
%   Detailed explanation goes here

% STATION POSITION AND ROTATION
[rStation, DCM] = ecef2eci_(Station.Position, mjd, const);

% COMPUTING BODY FIXED VELOCITY
vlat = const.Earth.RotVel*norm(Station.Position(1:2));
theta = atan2d(Station.Position(2), Station.Position(1));
vECEF = vlat * [-sind(theta); cosd(theta); 0];

% CONVERTING TO ECI
vStation = DCM*vECEF;

end
```

Published with MATLAB® R2022a

Symbolic Jacobians Evaluation

Generating H and A Matrice Functions

Last Update: 2022-04-09

```
clear; clc
symcell2strcell = @(C) cellfun(@char, C, 'UniformOutput', false);
overlap = @(a, b) ismember(symcell2strcell(b), symcell2strcell(a));
```

A Matrix

Settings

```
model = struct;
model.grav      = "J2";
model.drag      = "Cannonball";
model.srp       = "Cannonball";
model.thirdbody = "Moon-Sun";
savetofile = false;
```

Building Function

```
% GRAVITY
syms x [6 1]
syms mu
J2 = sym('J_2');
ae = sym('a_e');
switch model.grav
    case "2-Body"
        vars = {x, mu};
        grav = eom.Kep([], x, mu);

    case "J2"
        vars = {x, mu, J2, ae};
        grav = eom.KepJ2([], x, mu, J2, ae);
end

% DRAG
syms A m H
CD = sym('C_D');
rho0 = sym('rho_0');
r0 = sym('r_0');
thetadot = sym('theta_dot');
switch model.drag
    case "None"
        drag = zeros(6, 1);

    case "Cannonball"
        vars = cat(2, vars, {CD, A, m ,rho0, r0, H, thetadot});
        drag = eom.Drag([], x, CD, A, m ,rho0, r0, H, thetadot)
end
```

drag =

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ -\frac{A\,C_D\,\rho_0\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\,(x_4+\dot{\theta}\,x_2)\,\sqrt{(x_4+\dot{\theta}\,x_2)^2+(x_5-\dot{\theta}\,x_1)^2+x_6^2}}{2\,m} \\ -\frac{A\,C_D\,\rho_0\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\,(x_5-\dot{\theta}\,x_1)\,\sqrt{(x_4+\dot{\theta}\,x_2)^2+(x_5-\dot{\theta}\,x_1)^2+x_6^2}}{2\,m} \\ -\frac{A\,C_D\,\rho_0\,x_6\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\,\sqrt{(x_4+\dot{\theta}\,x_2)^2+(x_5-\dot{\theta}\,x_1)^2+x_6^2}}{2\,m} \end{pmatrix}$$

```
% SRP
CR = sym('C_R');
rS = sym('r_S', [3, 1]);
switch model.srp
    case "None"
        srp = zeros(6, 1);

    case "Cannonball"
        newvars = {CR, A, m, rS};
        tf = ~overlap(vars, newvars);
        vars = cat(2, vars, newvars(tf));
        srp = eom.SRP([], x, CR, A, m, rS)
end
```

srp =

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{A\,\mathrm{AU}^2\,C_R\,P_{\mathrm{srp}}\,(r_{S1}-x_1)}{m\,\left((r_{S1}-x_1)^2+(r_{S2}-x_2)^2+(r_{S3}-x_3)^2\right)^{3/2}} \\ \frac{A\,\mathrm{AU}^2\,C_R\,P_{\mathrm{srp}}\,(r_{S2}-x_2)}{m\,\left((r_{S1}-x_1)^2+(r_{S2}-x_2)^2+(r_{S3}-x_3)^2\right)^{3/2}} \\ \frac{A\,\mathrm{AU}^2\,C_R\,P_{\mathrm{srp}}\,(r_{S3}-x_3)}{m\,\left((r_{S1}-x_1)^2+(r_{S2}-x_2)^2+(r_{S3}-x_3)^2\right)^{3/2}} \end{pmatrix}$$

```
% THIRD-BODY
muS = sym('mu_S');
rM = sym('r_M', [3 1]);
```

```
muM = sym('mu_M');
switch model.thirdbody
    case "None"
        trb = zeros(6, 1);

    case "Moon-Sun"
        newvars = {muS, rM, muM};
        tf = ~overlap(vars, newvars);
        vars = cat(2, vars, newvars(tf));
        trb = eom.ThirdBody([], x, muS, rS) + eom.ThirdBody([], x, muM, rM)
end
```

trb =

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ \mu_M \left(\frac{r_{M1} - x_1}{\left(|r_{M1} - x_1|^2 + |r_{M2} - x_2|^2 + |r_{M3} - x_3|^2 \right)^{3/2}} - \frac{r_{M1}}{\left(|r_{M1}|^2 + |r_{M2}|^2 + |r_{M3}|^2 \right)^{3/2}} \right) + \mu_S \left(\frac{r_S}{\left(|r_{S1} - x_1|^2 + |r_{S2} \right.} \right. \\ \mu_M \left(\frac{r_{M2} - x_2}{\left(|r_{M1} - x_1|^2 + |r_{M2} - x_2|^2 + |r_{M3} - x_3|^2 \right)^{3/2}} - \frac{r_{M2}}{\left(|r_{M1}|^2 + |r_{M2}|^2 + |r_{M3}|^2 \right)^{3/2}} \right) + \mu_S \left(\frac{r_S}{\left(|r_{S1} - x_1|^2 + |r_{S2} \right.} \right. \\ \mu_M \left(\frac{r_{M3} - x_3}{\left(|r_{M1} - x_1|^2 + |r_{M2} - x_2|^2 + |r_{M3} - x_3|^2 \right)^{3/2}} - \frac{r_{M3}}{\left(|r_{M1}|^2 + |r_{M2}|^2 + |r_{M3}|^2 \right)^{3/2}} \right) + \mu_S \left(\frac{r_S}{\left(|r_{S1} - x_1|^2 + |r_{S2} \right.} \right. \end{pmatrix}$$

```
% COMBINING
func = grav + drag + srp + trb
```

func =

$$\begin{pmatrix} \\ \\ \\ \\ \\ \\ \mu_M \left(\frac{r_{M1} - x_1}{\left(|r_{M1} - x_1|^2 + |r_{M2} - x_2|^2 + |r_{M3} - x_3|^2 \right)^{3/2}} - \frac{r_{M1}}{\left(|r_{M1}|^2 + |r_{M2}|^2 + |r_{M3}|^2 \right)^{3/2}} \right) + \mu_S \left(\frac{}{\left(|r_{S1} - x_1|^2 + \right.} \right. \\ \mu_M \left(\frac{r_{M2} - x_2}{\left(|r_{M1} - x_1|^2 + |r_{M2} - x_2|^2 + |r_{M3} - x_3|^2 \right)^{3/2}} - \frac{r_{M2}}{\left(|r_{M1}|^2 + |r_{M2}|^2 + |r_{M3}|^2 \right)^{3/2}} \right) + \mu_S \left(\frac{}{\left(|r_{S1} - x_1|^2 + \right.} \right. \\ \mu_M \left(\frac{r_{M3} - x_3}{\left(|r_{M1} - x_1|^2 + |r_{M2} - x_2|^2 + |r_{M3} - x_3|^2 \right)^{3/2}} - \frac{r_{M3}}{\left(|r_{M1}|^2 + |r_{M2}|^2 + |r_{M3}|^2 \right)^{3/2}} \right) + \mu_S \left(\frac{r_S}{\left(|r_{S1} - x_1|^2 + |r_{S2} \right.} \right. \end{pmatrix}$$

```
symvar(func)
```

```
ans = (A  AU  C_D  C_R  H  J_2  P_srp  a_e  m  μ  μ_M  μ_S  r_0  r_M1  r_M2  r_M3  r_S1  r_S2  r_S3  ρ_0  θ̇  x_1  x_2  x
```

Taking Derivative

```
% TAKING JACOBIAN
A = jacobian(func, x);
% sympref('AbbreviateOutput', false); % Stops sigma creation
% A(:, 6)
```

ans =

$$\begin{pmatrix} 0 \\ 0 \\ 1 \\ -\dfrac{A\,C_D\,\rho_0\,x_6\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\,(x_4+\dot\theta\,x_2)}{2\,m\,\sqrt{(x_4+\dot\theta\,x_2)^2+(x_5-\dot\theta\,x_1)^2+x_6^2}} \\ -\dfrac{A\,C_D\,\rho_0\,x_6\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\,(x_5-\dot\theta\,x_1)}{2\,m\,\sqrt{(x_4+\dot\theta\,x_2)^2+(x_5-\dot\theta\,x_1)^2+x_6^2}} \\ -\dfrac{A\,C_D\,\rho_0\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}\,\sqrt{(x_4+\dot\theta\,x_2)^2+(x_5-\dot\theta\,x_1)^2+x_6^2}}{2\,m} - \dfrac{A\,C_D\,\rho_0\,x_6^2\,\mathrm{e}^{\frac{r_0-\sqrt{|x_1|^2+|x_2|^2+|x_3|^2}}{H}}}{2\,m\,\sqrt{(x_4+\dot\theta\,x_2)^2+(x_5-\dot\theta\,x_1)^2+x_6}} \end{pmatrix}$$

Outputting to File

```
if savetofile
    matlabFunction(A, 'Vars', vars, 'File', 'src/Agen.m')
end
```

H Matrix

```
clear; clc
```

Settings

```
model = struct;
model.ecef2eci = "Simple";
model.lighttime = "None";
savetofile = true;
```

Building Function

```
syms r [3 1]
syms v [3 1]
rSt = sym('r_Station', [3 1]);
```

```
vSt = sym('v_Station', [3 1]);

% FINDING DIFFERENCE VECTORS
rrel = r - rSt
```

rrel =

$$\begin{pmatrix} r_1 - r_{\text{Station1}} \\ r_2 - r_{\text{Station2}} \\ r_3 - r_{\text{Station3}} \end{pmatrix}$$

```
vrel = v - vSt;

% CALCULATING RANGE
range = sqrt(sum(rrel.^2))
```

range =

$$\sqrt{(r_1 - r_{\text{Station1}})^2 + (r_2 - r_{\text{Station2}})^2 + (r_3 - r_{\text{Station3}})^2}$$

```

% RANGE RATE
rangerate = (rrel.' * vrel)/range
```

rangerate =

$$\frac{(r_1 - r_{\text{Station1}})\ (v_1 - v_{\text{Station1}}) + (r_2 - r_{\text{Station2}})\ (v_2 - v_{\text{Station2}}) + (r_3 - r_{\text{Station3}})\ (v_3 - v_{\text{Station3}})}{\sqrt{(r_1 - r_{\text{Station1}})^2 + (r_2 - r_{\text{Station2}})^2 + (r_3 - r_{\text{Station3}})^2}}$$

```

% COMBINING
z = [range; rangerate];
```

Taking Derivative

```
H = jacobian(z, [r; v])
```

H =

$$\begin{pmatrix} \dfrac{r_1 - r_{\text{Station1}}}{\sqrt{(r_1 - r_{\text{Station1}})^2 + (r_2 - r_{\text{Station2}})^2 + (r_3 - r_{\text{Station3}})^2}} & \dfrac{r_2 - r_{\text{Station2}}}{\sqrt{(r_1 - r_{\text{Station1}})^2 + (r_2 - r_{\text{Station2}})^2 + (r_3 - r_{\text{Station3}})^2}} \\ \dfrac{v_1 - v_{\text{Station1}}}{\sqrt{(r_1 - r_{\text{Station1}})^2 + (r_2 - r_{\text{Station2}})^2 + (r_3 - r_{\text{Station3}})^2}} - \dfrac{(2\,r_1 - 2\,r_{\text{Station1}})\ ((r_1 - r_{\text{Station1}})\ (v_1 - v_{\text{Station1}}) + (r_2 - \right. & \left. 2\ \left((r_1 - r_{\text{Station1}})^2 + (r_2 - \right. \end{pmatrix}$$

Outputting to File

```
if savetofile
    matlabFunction(H, 'Vars', {[r; v], rSt, vSt}, 'File', 'src/Hgen.m')
end
```

ans = *function_handle with value:*

Linearized Dynamics Matrix (A)

```
function A = Agen(in1,mu,J_2,a_e,C_D,A,m,rho_0,r_0,H,theta_dot,C_R,in13,mu_S,in15,mu_M)
%AGEN
%    A = AGEN(IN1,MU,J_2,A_E,C_D,A,M,RHO_0,R_0,H,THETA_DOT,C_R,IN13,MU_S,IN15,MU_M)

%    This function was generated by the Symbolic Math Toolbox version 8.7.
%    28-Apr-2022 00:21:38

r_M1 = in15(1,:);
r_M2 = in15(2,:);
r_M3 = in15(3,:);
r_S1 = in13(1,:);
r_S2 = in13(2,:);
r_S3 = in13(3,:);
x1 = in1(1,:);
x2 = in1(2,:);
x3 = in1(3,:);
x4 = in1(4,:);
x5 = in1(5,:);
x6 = in1(6,:);
t2 = abs(x1);
t3 = abs(x2);
t4 = abs(x3);
t5 = conj(x3);
t6 = sign(x1);
t7 = sign(x2);
t8 = sign(x3);
t9 = theta_dot.*x1;
t10 = theta_dot.*x2;
t11 = a_e.^2;
t12 = r_S1.*2.0;
t13 = r_S2.*2.0;
t14 = r_S3.*2.0;
t15 = x1.*2.0;
t16 = x2.*2.0;
t17 = x3.*2.0;
t18 = x4.*2.0;
t19 = x5.*2.0;
t20 = x1.^2;
t21 = x2.^2;
t22 = x3.^2;
t23 = x6.^2;
t30 = 1.0./H;
t31 = 1.0./m;
t32 = -x1;
t34 = -x2;
t36 = -x3;
t95 = r_S1.*8.936740963196373e-17;
t96 = r_S2.*8.936740963196373e-17;
t97 = r_S3.*8.936740963196373e-17;
t98 = x1.*8.936740963196373e-17;
t99 = x2.*8.936740963196373e-17;
t100 = x3.*8.936740963196373e-17;
t24 = t2.^2;
t25 = t3.^2;
t26 = t4.^2;
t27 = t5.^2;
t28 = t9.*2.0;
t29 = t10.*2.0;
t33 = -t15;
t35 = -t16;
t37 = -t17;
t38 = t10+x4;
t39 = -t9;
t41 = r_M1+t32;
t42 = r_M2+t34;
t43 = r_M3+t36;
t44 = r_S1+t32;
t45 = r_S2+t34;
t46 = r_S3+t36;
t75 = (t9-x5).^2;
t76 = t20+t21+t22;
t103 = -t98;
t104 = -t99;
t105 = -t100;
t40 = -t28;
t47 = abs(t41);
t48 = abs(t42);
t49 = abs(t43);
t50 = abs(t44);
t51 = abs(t45);
t52 = abs(t46);
t53 = sign(t41);
t54 = sign(t42);
t55 = sign(t43);
t56 = sign(t44);
t57 = sign(t45);
t58 = sign(t46);
t59 = t39+x5;
t60 = t38.^2;
```

```
t61 = t12+t33;
t62 = t13+t35;
t63 = t14+t37;
t64 = t44.^2;
t65 = t45.^2;
t66 = t46.^2;
t67 = t18+t29;
t77 = t24+t25+t26;
t78 = 1.0./t76;
t80 = 1.0./t76.^(3.0./2.0);
t81 = 1.0./t76.^(5.0./2.0);
t82 = 1.0./t76.^(7.0./2.0);
t128 = t95+t103;
t129 = t96+t104;
t130 = t97+t105;
t68 = t47.^2;
t69 = t48.^2;
t70 = t49.^2;
t71 = t50.^2;
t72 = t51.^2;
t73 = t52.^2;
t74 = t19+t40;
t79 = t78.^2;
t83 = t80.^3;
t84 = mu.*t80;
t85 = sqrt(t77);
t87 = t5.*t78.*1.0e+1;
t89 = t27.*t78.*5.0;
t90 = mu.*t81.*x1.*x2.*3.0;
t91 = mu.*t81.*x1.*x3.*3.0;
t92 = mu.*t81.*x2.*x3.*3.0;
t93 = t23+t60+t75;
t109 = t64+t65+t66;
t112 = t64./2.237952300773264e+16;
t113 = t65./2.237952300773264e+16;
t114 = t66./2.237952300773264e+16;
t86 = 1.0./t85;
t88 = -t84;
t94 = t27.*t79.*x3.*1.0e+1;
t101 = t89-1.0;
t106 = t85.*1.0e+3;
t108 = sqrt(t93);
t115 = t68+t69+t70;
t116 = t71+t72+t73;
t117 = J_2.*mu.*t11.*t27.*t83.*x1.*x2.*1.5e+1;
t119 = 1.0./sqrt(t109);
t142 = t112+t113+t114;
t102 = -t94;
t107 = -t106;
t111 = 1.0./t108;
t120 = t119.^3;
t122 = -t117;
t123 = 1.0./t115.^(3.0./2.0);
t124 = 1.0./t115.^(5.0./2.0);
t125 = 1.0./t116.^(3.0./2.0);
t126 = 1.0./t116.^(5.0./2.0);
t131 = J_2.*mu.*t11.*t81.*t101.*(3.0./2.0);
t132 = J_2.*mu.*t11.*t82.*t101.*x1.*x2.*(1.5e+1./2.0);
t133 = J_2.*mu.*t11.*t82.*t101.*x1.*x3.*(1.5e+1./2.0);
t134 = J_2.*mu.*t11.*t82.*t101.*x2.*x3.*(1.5e+1./2.0);
t143 = 1.0./t142;
t110 = r_0+t107;
t127 = t87+t102;
t135 = -t132;
t136 = -t133;
t137 = -t134;
t144 = t143.^2;
t145 = A.*C_R.*t31.*t119.*t143.*4.57e-9;
t118 = t30.*t110;
t146 = -t145;
t121 = exp(t118);
t138 = A.*C_D.*rho_0.*t31.*t108.*t121.*5.0e+2;
t141 = A.*C_D.*rho_0.*t31.*t38.*t111.*t121.*theta_dot.*(t9-x5).*-5.0e+2;
t139 = t138.*theta_dot;
t140 = -t138;
mt1 = [0.0,0.0,0.0,t88+t131+t141+t146-mu_M.*(t123-t41.*t47.*t53.*t124.*3.0)-mu_S.*(t125-t44.*t50.*t56.*t126.*3.0)+mu.*t20.*t81.*3.0-J_2.*mu.*t11.*t20.*t27.*t83.*1.5
mt2 = [t90+t122+t135+t139+mu_M.*t42.*t47.*t53.*t124.*3.0+mu_S.*t45.*t50.*t56.*t126.*3.0+A.*C_R.*t31.*t45.*t61.*t120.*t143.*2.285e-9+A.*C_R.*t31.*t45.*t119.*t128.*t1
mt3 = [t91+t136+mu_M.*t43.*t47.*t53.*t124.*3.0+mu_S.*t46.*t50.*t56.*t126.*3.0+J_2.*mu.*t5.*t11.*t82.*x1.*1.5e+1+A.*C_R.*t31.*t46.*t61.*t120.*t143.*2.285e-9+A.*C_R.*
mt4 = [t90+t122+t135+mu_M.*t41.*t48.*t54.*t124.*3.0+mu_S.*t44.*t51.*t57.*t126.*3.0-A.*C_D.*rho_0.*t31.*t108.*t121.*theta_dot.*5.0e+2+A.*C_R.*t31.*t44.*t62.*t120.*t1
mt5 = [t88+t131+t146-mu_M.*(t123-t42.*t48.*t54.*t124.*3.0)-mu_S.*(t125-t45.*t51.*t57.*t126.*3.0)+mu.*t21.*t81.*3.0-J_2.*mu.*t11.*t21.*t27.*t83.*1.5e+1-J_2.*mu.*t11.
mt6 = [t92+t137+mu_M.*t43.*t48.*t54.*t124.*3.0+mu_S.*t46.*t51.*t57.*t126.*3.0+J_2.*mu.*t5.*t11.*t82.*x2.*1.5e+1+A.*C_R.*t31.*t46.*t62.*t120.*t143.*2.285e-9+A.*C_R.*
mt7 = [t91+t136+mu_M.*t41.*t49.*t55.*t124.*3.0+mu_S.*t44.*t52.*t58.*t126.*3.0+J_2.*mu.*t11.*t81.*t127.*x1.*(3.0./2.0)+A.*C_R.*t31.*t44.*t63.*t120.*t143.*2.285e-9+A.
mt8 = [t92+t137+mu_M.*t42.*t49.*t55.*t124.*3.0+mu_S.*t45.*t52.*t58.*t126.*3.0+J_2.*mu.*t11.*t81.*t127.*x2.*(3.0./2.0)+A.*C_R.*t31.*t45.*t63.*t120.*t143.*2.285e-9+A.
mt9 = [t88+t131+t146-mu_M.*(t123-t43.*t49.*t55.*t124.*3.0)-mu_S.*(t125-t46.*t52.*t58.*t126.*3.0)+mu.*t22.*t81.*3.0-J_2.*mu.*t11.*t81.*3.0-J_2.*mu.*t11.*t22.*t82.*t1
mt10 = [t140-A.*C_D.*rho_0.*t31.*t38.*t67.*t111.*t121.*2.5e+2,A.*C_D.*rho_0.*t31.*t67.*t111.*t121.*(t9-x5).*2.5e+2,A.*C_D.*rho_0.*t31.*t67.*t111.*t121.*x6.*-2.5e+2,
A = reshape([mt1,mt2,mt3,mt4,mt5,mt6,mt7,mt8,mt9,mt10],6,6);
```

Published with MATLAB® R2022a

Measurement Sensitivity Matrix (H)

```
function H = Hgen(in1,in2,in3)
%HGEN
%    H = HGEN(IN1,IN2,IN3)

%    This function was generated by the Symbolic Math Toolbox version 8.7.
%    20-Apr-2022 15:50:31

r1 = in1(1,:);
r2 = in1(2,:);
r3 = in1(3,:);
r_Station1 = in2(1,:);
r_Station2 = in2(2,:);
r_Station3 = in2(3,:);
v1 = in1(4,:);
v2 = in1(5,:);
v3 = in1(6,:);
v_Station1 = in3(1,:);
v_Station2 = in3(2,:);
v_Station3 = in3(3,:);
t2 = r1.*2.0;
t3 = r2.*2.0;
t4 = r3.*2.0;
t5 = r_Station1.*2.0;
t6 = r_Station2.*2.0;
t7 = r_Station3.*2.0;
t8 = -r_Station1;
t10 = -r_Station2;
t12 = -r_Station3;
t14 = -v_Station1;
t15 = -v_Station2;
t16 = -v_Station3;
t9 = -t5;
t11 = -t6;
t13 = -t7;
t17 = r1+t8;
t18 = r2+t10;
t19 = r3+t12;
t20 = t14+v1;
t21 = t15+v2;
t22 = t16+v3;
t23 = t2+t9;
t24 = t3+t11;
t25 = t4+t13;
t26 = t17.^2;
t27 = t18.^2;
t28 = t19.^2;
t29 = t17.*t20;
t30 = t18.*t21;
t31 = t19.*t22;
t32 = t26+t27+t28;
t35 = t29+t30+t31;
t33 = 1.0./sqrt(t32);
t34 = t33.^3;
H = reshape([(t23.*t33)./2.0,t20.*t33-(t23.*t34.*t35)./2.0,(t24.*t33)./2.0,t21.*t33-(t24.*t34.*t35)./2.0,(t25.*t33)./2.0,t22.*t33-(t25.*t34.*t35)./2.0,0.0,t17.*t33,
```

Published with MATLAB® R2022a