ASTE 583 Term Project

Gene Luevano

December 6, 2023

1 Historical Background of Orbit Determination (Part 5a)

The methods of basic orbit determination can be traced back to the 17th and 18th century. One of the first major milestones in the orbit determination problem was Johannes Kepler publishing his 3 laws of planetary motion. Kepler's work paved the way for Sir Isaac Newton 60 years later to expand and to publish the laws of motion as well as some of the first mathematical methods capable of solving the orbit determination problem. A century later in 1801, after much intrigue about the motion of planets and other objects flying through our solar system, one of the next great breakthroughs for the orbit determination problem was the development of the method of least squares by Carl Friedrich Gauss. Gauss used this mathematical method to predict the future location of the asteroid Ceres after its solar conjunction without using the complex, nonlinear equations of planetary as derived by Kepler. In fact, the method of least squares still has a significant importance 200 years later due to its effectiveness of data fitting linear and nonlinear equations. In the upcoming solution to the orbital determination problem, a key component of the algorithm used to improve the initial best estimate of the satellite uses a linear least squares model to determine the orbital information that best fits the provided observation data.

2 MATLAB Script Description (Part 5b)

The following sections provide an overview of the coded batch processor that performed the statistical orbit determination of the sample satellite provided. Below is a breakdown of the inputs and outputs, equations used, a flow diagram of how the batch processor operates, and a copy of the code used to generate the results found in this report.

2.1 Analysis Inputs (Part 5b)

Provided by the University of Southern California's Dr. Gerald Hintz, this analysis used the following inputs to complete the orbit determination of the given satellite.

Earth Orientation:

$\dot{\theta}$ = 7.2921158543 x 10^{-5} radians/sec

Earth Gravity Model:

Gm_{Earth} = 3.986004415 x 10^{14} m^3/s^2

J_2 = 1.082626925638815 x 10^{-3}

R_{Earth} = 6378136.3 m

Atmospheric Drag Model:

ρ_0 = 3.617 x 10^{-13} kg/m^3

r_0 = (700000.0 + R_{Earth}) m

H = 88667.0 m

A_{CS} = 3.0 m^2

$m_{satellite}$ = 970 kg

$C_D = 2.0$

Ground Station Positions in Body-Fixed Coordinate System:

Station No.	X_S (m)	Y_S (m)	Z_S (m)
101	-5127510.0	-3794160.0	0.0
337	3860910.0	3238490.0	3898094.0
394	549505.0	-1380872.0	6182197.0

Satellite State in the Inertial Coordinate System:

	Position (m)	Velocity (m/s)
I	757700.0	2213.21
J	5222607.0	4678.34
K	4851500.0	-5371.30

Weighting Matrix:

$\sigma_\rho^2 = 0.01$ m

$\sigma_{\dot\rho}^2 = 0.001$ m/s

$$W = \begin{bmatrix} \frac{1}{\sigma_\rho^2} & 0 \\ 0 & \frac{1}{\sigma_{\dot\rho}^2} \end{bmatrix}$$

A Priori State Deviation:

$$\bar{\bar{x}}_0 = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}_{18x1}$$

A Priori Covariance:

$$\bar{P}_0 = \begin{bmatrix}
10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 10^{20} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^{-10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^{-10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^{-10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^6 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^6 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^6 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^6
\end{bmatrix}$$

Observation Data:

Also provided from Dr. Hintz was the range and range-rate data from the three ground tracking stations over a course of about 5 hours.

2.2 Equations Used (Part 5b)

Dynamics Equations:

$$\ddot{\mathbf{r}} = -\mu_{Earth}\left(\frac{1}{r^3} + \frac{3J_2R_{Earth}^2}{2r^5}\Big(\begin{bmatrix}1&0&0\\0&1&0\\0&0&3\end{bmatrix} - 5(\frac{z}{r})^2\Big)\right)\mathbf{r} \;-\; \frac{\rho_0}{2m}e^{\frac{-(\mathbf{r}-r_0)}{H}}A_{CS}C_D|V_A|\mathbf{V_A} \tag{1}$$

State Variables:

$$\mathbf{X} = \begin{bmatrix}\mathbf{r} & \dot{\mathbf{r}} & \mu_{Earth} & J_2 & C_D & \mathbf{X_{S1}} & \mathbf{X_{S2}} & \mathbf{X_{S3}}\end{bmatrix}^T \tag{2}$$

$$\dot{\mathbf{X}} = \begin{bmatrix}\dot{\mathbf{r}} & \ddot{\mathbf{r}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^T \tag{3}$$

$$\mathbf{r} = \begin{bmatrix}x & y & z\end{bmatrix},\ \dot{\mathbf{r}} = \begin{bmatrix}\dot{x} & \dot{y} & \dot{z}\end{bmatrix},\ \mathbf{X_{Si}} = \begin{bmatrix}x_{Si} & y_{Si} & z_{Si}\end{bmatrix}$$

Atmospheric Effects:

$$\rho_A = \rho_0 e^{\frac{-(r-r_0)}{H}} \;\; kg/m^3 \tag{4}$$

$$\mathbf{V_A} = \dot{\mathbf{r}} - \dot{\theta}\hat{K}\times\mathbf{r} \;\; m/sec \tag{5}$$

Earth's Orientation:

$$\theta = \dot{\theta}t \;\; radians \tag{6}$$

Range and Range-rate:

$$\rho_{range} = \Big(x^2+y^2+z^2+x_{Si}^2+y_{Si}^2+z_{Si}^2 - 2(xx_{Si}+yy_{Si})cos(\theta) + 2(xy_{Si}-yx_{Si})sin(\theta) - 2zz_{Si}\Big)^{1/2} \tag{7}$$

$$\begin{aligned}\dot{\rho}_{range} = \frac{1}{\rho_{range}}\Bigg(&x\dot{x}+y\dot{y}+z\dot{z} - (\dot{x}x_{Si}+\dot{y}y_{Si})cos(\theta) + \dot{\theta}(xx_{Si}+yy_{Si})sin(\theta) + (\dot{x}y_{Si}-\dot{y}x_{Si})sin(\theta)\\ &+ \dot{\theta}(xy_{Si}-yx_{Si})cos(\theta) - \dot{z}z_{Si}\Bigg)\end{aligned} \tag{8}$$

A-matrix:

$$A = \frac{\partial F(X^*,t)}{\partial \mathbf{X}} = \frac{\partial \dot{\mathbf{X}}}{\partial \mathbf{X}} = \begin{bmatrix}\frac{\partial F_1}{\partial X_1} & \frac{\partial F_1}{\partial X_2} & \cdots & \cdots & \frac{\partial F_1}{\partial X_n}\\ \frac{\partial F_2}{\partial X_1} & \ddots & & & \vdots\\ \vdots & & \ddots & & \vdots\\ \vdots & & & \ddots & \vdots\\ \frac{\partial F_n}{\partial X_1} & \cdots & \cdots & \cdots & \frac{\partial F_n}{\partial X_n}\end{bmatrix} \tag{9}$$

$A_{1,4} = \frac{\partial\dot{x}}{\partial\dot{x}} = 1$

$A_{2,5} = \frac{\partial\dot{y}}{\partial\dot{y}} = 1$

$A_{3,6} = \frac{\partial\dot{z}}{\partial\dot{z}} = 1$

$$A_{4,1} = \frac{\partial \ddot{x}}{\partial x} = -\frac{\mu_{Earth}}{r^3}\left[1-\frac{3J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2\left(5(\frac{z}{r})^2-1\right)\right]+\frac{3\mu_{Earth}x^2}{r^5}\left[1-\frac{5J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2\left(7(\frac{z}{r})^2-1\right)\right]+\frac{\rho_A C_D A_{CS} V_A}{2m}\frac{x(\dot{x}+\dot{\theta}y)}{rH}$$
$$-\frac{\rho_A C_D A_{CS}}{2m}\frac{(-\dot{\theta}\dot{y}+\dot{\theta}^2 x)(\dot{x}+\dot{\theta}y)}{V_A}$$

$$A_{4,2} = \frac{\partial \ddot{x}}{\partial y} = \frac{3\mu_{Earth}xy}{r^5}\left[1-\frac{5J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2\left(7(\frac{z}{r})^2-1\right)\right)] + \frac{\rho_A C_D A_{CS} V_A}{2m}\frac{y(\dot{x}+\dot{\theta}y)}{rH} - \frac{\rho_A C_D A_{CS}}{2m}\frac{(\dot{\theta}\dot{y}+\dot{\theta}^2 x)(\dot{x}+\dot{\theta}y)}{V_A} -$$
$$\frac{\rho_A C_D A_{CS} V_A \dot{\theta}}{2m}$$

$$A_{4,3} = \frac{\partial \ddot{x}}{\partial z} = \frac{3\mu_{Earth}xz}{r^5}\left[1-\frac{5J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2\left(7(\frac{z}{r})^2-3\right)\right)] + \frac{\rho_A C_D A_{CS} V_A}{2m}\frac{z(\dot{x}+\dot{\theta}y)}{rH}$$

$$A_{4,4} = \frac{\partial \ddot{x}}{\partial \dot{x}} = -\frac{\rho_A C_D A_{CS}}{2m}\frac{(\dot{x}+\dot{\theta}y)^2}{V_A} - \frac{\rho_A C_D A_{CS} V_A}{2m}$$

$$A_{4,5} = \frac{\partial \ddot{x}}{\partial \dot{y}} = -\frac{\rho_A C_D A_{CS}}{2m}\frac{(\dot{y}-\dot{\theta}x)(\dot{x}+\dot{\theta}y)}{V_A}$$

$$A_{4,6} = \frac{\partial \ddot{x}}{\partial \dot{z}} = -\frac{\rho_A C_D A_{CS}}{2m}\frac{\dot{z}(\dot{x}+\dot{\theta}y)}{V_A}$$

$$A_{4,7} = \frac{\partial \ddot{x}}{\partial \mu_{Earth}} = -\frac{x}{r^3}\left[1-\frac{3J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2\left(5(\frac{z}{r})^2-1\right)\right]$$

$$A_{4,8} = \frac{\partial \ddot{x}}{\partial \mu_{Earth}} = -\frac{3\mu_{Earth}x}{r^3}\left(\frac{R_{Earth}}{r}\right)^2\left(5\left(\frac{z}{r}\right)^2-1\right)$$

$$A_{4,9} = \frac{\ddot{x}}{\partial C_D} = -\frac{\rho_A A_{CS} V_A(\dot{x}+\dot{\theta}y)}{2m}$$

$$A_{5,1} = \frac{\partial \ddot{y}}{\partial x} = \frac{3\mu_{Earth}xy}{r^5}\left[1-\frac{5J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2\left(7\left(\frac{z}{r}\right)^2-1\right)\right] + \frac{\rho_A C_D A_{CS} V_A}{2m}\frac{(\dot{y}-\dot{\theta}x)x}{rH} - \frac{\rho_A C_D A_{CS}}{2m}\frac{(\dot{\theta}^2 x-\dot{\theta}\dot{y})(\dot{y}-\dot{\theta}x)}{V_A}$$
$$+\frac{\rho_A C_D A_{CS} V_A \dot{\theta}}{2m}$$

$$A_{5,2} = \frac{\partial \ddot{y}}{\partial y} = -\frac{\mu_{Earth}}{r^3}\left[1-\frac{3J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2\left(5\left(\frac{z}{r}\right)^2-1\right)\right]$$
$$+\frac{3\mu_{Earth}y^2}{r^5}\left[1-\frac{5J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2\left(7\left(\frac{z}{r}\right)^2-1\right)\right] + \frac{\rho_A C_D A_{CS} V_A}{2m}\frac{y(\dot{y}-\dot{\theta}x}{rH} - \frac{\rho_A C_D A_{CS}}{2m}\frac{(\dot{\theta}\dot{x}+\dot{\theta}^2 y)(\dot{y}-\dot{\theta}x)}{V_A}$$

$$A_{5,3} = \frac{\partial \ddot{y}}{\partial z} = \frac{3\mu_{Earth}yz}{r^5}\left[1-\frac{5J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2\left(7\left(\frac{z}{r}\right)^2-3\right)\right] + \frac{\rho_A C_D A_{CS} V_A}{2m}\frac{z(\dot{y}-\dot{\theta}x)}{rH}$$

$$A_{5,4} = \frac{\partial \ddot{y}}{\partial \dot{x}} = -\frac{\rho_A C_D A_{CS}}{2m}\frac{(\dot{y}-\dot{\theta}x)(\dot{x}+\dot{\theta}y)}{V_A}$$

$$A_{5,5} = \frac{\partial \ddot{y}}{\partial \dot{y}} = -\frac{\rho_A C_D A_{CS}}{2m} \frac{(\dot{y} - \dot{\theta} x)^2}{V_A} - \frac{rho_A C_D A_{CS} V_A}{2m}$$

$$A_{5,6} = \frac{\partial \ddot{y}}{\partial \dot{z}} = -\frac{\rho_A C_D A_{CS}}{2m} \frac{\dot{z}(\dot{y} - \dot{\theta} x)}{V_A}$$

$$A_{5,7} = \frac{\partial \ddot{y}}{\partial \mu_{Earth}} = -\frac{y}{r^3}\left[1 - \frac{3J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2 \left(5\left(\frac{z}{r}\right)^2 - 1\right)\right]$$

$$A_{5,8} = \frac{\partial \ddot{y}}{\partial J_2} = \frac{3\mu_{Earth} y}{2r^3}\left(\frac{R_{Earth}}{r}\right)^2 \left(5\left(\frac{z}{r}\right)^2 - 1\right)$$

$$A_{5,9} = \frac{\partial \ddot{y}}{\partial C_D} = -\frac{\rho_A A_{CS} V_A}{2m}(\dot{y} - \dot{\theta} x)$$

$$A_{6,1} = \frac{\partial \ddot{z}}{\partial x} = \frac{3\mu_{Earth} xz}{r^5}\left[1 - \frac{5J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2 \left(7\left(\frac{z}{r}\right)^2 - 3\right)\right] + \frac{\rho_A C_D A_{CS} V_A}{2m}\frac{\dot{z}x}{rH} - \frac{\rho_A C_d A_{CS}}{2m}\frac{\dot{z}(\dot{\theta}^2 x - \dot{\theta}\dot{y})}{V_A}$$

$$A_{6,2} = \frac{\partial \ddot{z}}{\partial y} = \frac{3\mu_{Earth} yz}{r^5}\left[1 - \frac{5J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2 \left(7\left(\frac{z}{r}\right)^2 - 3\right)\right] + \frac{\rho_A C_D A_{CS} V_A}{2m}\frac{\dot{z}y}{rH} - \frac{\rho_A C_D A_{CS}}{2m}\frac{\dot{z}(\dot{\theta}\dot{x} + \dot{\theta}^2 y)}{V_A}$$

$$A_{6,3} = \frac{\partial \ddot{z}}{\partial z} = -\frac{\mu_{Earth}}{r^3}\left[1 - \frac{3J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2 \left(5\left(\frac{z}{r}\right)^2 - 3\right)\right]$$
$$+ \frac{3\mu_{Earth} z^2}{r^5}\left[1 - \frac{5J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2 \left(7\left(\frac{z}{r}\right)^2 - 5\right)\right] + \frac{\rho_A C_D A_{CS} V_A}{2m}\frac{z\dot{z}}{rH}$$

$$A_{6,4} = \frac{\partial \ddot{z}}{\partial \dot{x}} = -\frac{\rho_A C_D A_{CS}}{2m} \frac{\dot{z}(\dot{x} + \dot{\theta} y)}{V_A}$$

$$A_{6,5} = \frac{\partial \ddot{z}}{\partial \dot{y}} = -\frac{\rho_A C_D A_{CS}}{2m} \frac{\dot{z}(\dot{y} - \dot{\theta} x)}{V_A}$$

$$A_{6,6} = \frac{\partial \ddot{z}}{\partial \dot{z}} = -\frac{\rho_A C_D A_{CS}}{2m} \frac{\dot{z}^2}{V_A} - \frac{\rho_A C_D A_{CS} V_A}{2m}$$

$$A_{6,7} = \frac{\partial \ddot{z}}{\partial \mu_{Earth}} = -\frac{z}{r^3}\left[1 - \frac{3J_2}{2}\left(\frac{R_{Earth}}{r}\right)^2 \left(5\left(\frac{z}{r}\right)^2 - 3\right)\right]$$

$$A_{6,8} = \frac{\partial \ddot{z}}{\partial J_2} = \frac{3\mu_{Earth} z}{r^3}\left(\frac{R_{Earth}}{r}\right)^2 \left(5\left(\frac{z}{r}\right)^2 - e\right)$$

$$A_{6,9} = \frac{\partial \ddot{z}}{\partial C_D} = -\frac{\rho_A A_{CS} V_A \dot{z}}{2m}$$

All other A-matrix terms are 0.

$\underline{\tilde{H}\text{-matrix:}}$

$$G(X) = \begin{bmatrix} \rho_{range} \\ \dot{\rho}_{range} \end{bmatrix} \tag{10}$$

$$\tilde{H}(t) = \frac{\partial G(X)}{\partial X} = \begin{bmatrix} \frac{\partial \rho_{range}}{\partial X} \\ \frac{\partial \dot{\rho}_{range}}{\partial X} \end{bmatrix} \tag{11}$$

$\frac{\partial \rho_{range}}{\partial x} = \frac{x - x_{Si} cos(\theta) + y_{Si} sin(\theta}{\rho_r ange}$

$\frac{\partial \rho_{range}}{\partial y} = \frac{y - y_{Si} cos(\theta) - x_{Si} sin(\theta}{\rho_{range}}$

$\frac{\partial \rho_{range}}{\partial z} = \frac{z - z_{Si}}{\rho_{range}}$

$\frac{\partial \rho_{range}}{\partial \dot{x}} = \frac{\partial \rho_{range}}{\partial \dot{y}} = \frac{\partial \rho_{range}}{\partial \dot{z}} = \frac{\partial \rho_{range}}{\partial \mu_{Earth}} = \frac{\partial \rho_{range}}{\partial J_2} = \frac{\partial \rho_{range}}{\partial C_D} = 0$

$\frac{\partial \rho_{range}}{\partial x_{Si}} = \frac{x_{Si} - x cos(\theta) - y sin(\theta)}{\rho_{range}}$

$\frac{\partial \rho_{range}}{\partial y_{Si}} = \frac{y_{Si} - y cos(\theta) + x sin(\theta)}{\rho_{range}}$

$\frac{\partial \rho_{range}}{\partial z_{Si}} = \frac{z_{Si} - z}{\rho_{range}}$

$\frac{\partial \dot{\rho}_{range}}{\partial x} = \frac{\dot{x} + \dot{\theta} x_{Si} sin(\theta) + \dot{\theta} y_{Si} cos(\theta}{\rho_{range}} - \frac{\dot{\rho}_{range}(x - x_{Si} cos(\theta) + y_{Si} sin(\theta)}{\rho_{range}^2}$

$\frac{\partial \dot{\rho}_{range}}{\partial y} = \frac{\dot{y} + \dot{\theta} y_{Si} sin(\theta) - \dot{\theta} x_{Si} cos(\theta)}{\rho_{range}} - \frac{\dot{\rho}_{range}(y - y_{Si} cos(\theta) - x_{Si} sin(\theta))}{\rho_{range}^2}$

$\frac{\partial \dot{\rho}_{range}}{\partial z} = \frac{\dot{z}}{\rho_{range}} - \frac{\dot{\rho}_{range}(z - z_{Si})}{\rho_{range}^2}$

$\frac{\partial \dot{\rho}_{range}}{\partial \dot{x}} = \frac{x - x_{Si} cos(\theta) + y_{Si} sin(\theta)}{\rho_{range}}$

$\frac{\partial \dot{\rho}_{range}}{\partial \dot{y}} = \frac{y - y_{Si} cos(\theta) - x_{Si} sin(\theta)}{\rho_{range}}$

$\frac{\partial \dot{\rho}_{range}}{\partial \dot{z}} = \frac{z - z_{Si}}{\rho_{range}}$

$\frac{\partial \dot{\rho}_{range}}{\partial \mu_{Earth}} = \frac{\partial \dot{\rho}_{range}}{\partial J_2} = \frac{\partial \dot{\rho}_{range}}{\partial C_D} = 0$

$\frac{\partial \dot{\rho}_{range}}{\partial \dot{x_{Si}}} = \frac{-\dot{x} cos(\theta) + \dot{\theta} x sin(\theta) - \dot{y} sin(\theta) - \dot{\theta} y cos(\theta)}{\rho_{range}} - \frac{\dot{\rho}_{range}(x_{Si} - x cos(\theta) - y sin(\theta)))}{\rho_{range}^2}$

$\frac{\partial \dot{\rho}_{range}}{\partial \dot{y_{Si}}} = \frac{-\dot{y} cos(\theta) + \dot{\theta} y sin(\theta) + \dot{x} sin(\theta) + \dot{\theta} x cos(\theta)}{\rho_{range}} - \frac{\dot{\rho}_{range}(x_{Si} - x cos(\theta) - y sin(\theta))}{\rho_{range}^2}$

$\frac{\partial \dot\rho_{range}}{\partial z_{\dot{S}i}} = \frac{-\dot z}{\rho_{range}} - \frac{\dot\rho_{range}(z_{Si}-z)}{\rho_{range}^2}$

State Transition Matrix:

$$\phi(t_0, t_0) = I \tag{12}$$

$$\dot\phi(t_k, t_0) = A(t_k)\phi(t_k, t_0) \tag{13}$$

H-matrix:

$$H_k = \tilde{H_k}\phi(t_k, t_0) \tag{14}$$

Observation Residuals:

$$\mathbf{y_i} = \mathbf{Y_i} - G(\mathbf{X}^*, t_i) \tag{15}$$

A Priori Estimation:

$$M = \bar P_0 + \sum_{i=1}^{k} H_k^T W_k H_k \tag{16}$$

$$N = \bar P_0 \bar x_0 + \sum_{i=1}^{k} H_k^T W_k \mathbf{y_k} \tag{17}$$

Cholesky Decomposition:
For i is 1, 2, ..., n.

$$r_{ii} = \Big(M_{ii} - \textstyle\sum_{k=1}^{i-1} r_{ki}^2\Big)^{1/2}$$

$$r_{ij} = \Big(M_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj}\Big)/r_{ii}; \qquad j = i+1,\ ...,\ n \tag{18}$$

$$z_i = \Big(N_i - \sum_{j=1}^{i-1} r_{ji} z_j\Big)/r_{ii} \tag{19}$$

For i is n, n-1, ..., 1.

$$\hat x_i = \Big(z_i - \sum_{j=i+1}^{n} r_{ij}\hat x_j\Big)/r_{ii} \tag{20}$$

Root Mean Squares:

$$\rho_{RMS} = \sqrt{\frac{\sum_{1=1}^{k} \mathbf{y}_{\rho_i}^T \mathbf{y}_{\rho_i}}{k}} \tag{21}$$

$$\dot\rho_{RMS} = \sqrt{\frac{\sum_{1=1}^{k} \mathbf{y}_{\dot\rho_i}^T \mathbf{y}_{\dot\rho_i}}{k}} \tag{22}$$

Covariance Matrix:

$$P = M^{-1} \tag{23}$$

$$P = \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \dots & \dots & \rho_{1n}\sigma_1\sigma_n \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \rho_{1n}\sigma_1\sigma_n & \dots & \dots & \dots & \sigma_n^2 \end{bmatrix} \tag{24}$$

Improvement of Initial State:

$$\mathbf{X}_0 = \mathbf{X}_0 + \hat{x}_0 \tag{25}$$

$$\bar{x}_0 = \bar{x}_0 - \hat{x}_0 \tag{26}$$

2.3 Use of Equations (Part 5b)

In order to solve for the best estimate of the initial state vector using a batch processor, all of the equations listed in the section above, 2.2 Equations Used. Beginning with Eq. (1), these equations of motion, or EOMs, will define how the satellite will orbit the Earth while being effected by the oblateness of the Earth as well as the effects of the atmosphere. Integrating this differential equation over the observation interval will produce the position and velocity vectors for the spacecraft at each time instant that it is being observed by one of the three ground sites. Along with the EOMs, the state transition matrix, or ϕ, must also be integrated using Eq. (13) over the observation interval. However, before the integration, the product of the matrix multiplication between the A-matrix and ϕ for the given time must be solved first. The A-matrix, Eq. (9), is the result of the partial derivative of $\dot{\mathbf{X}}$ with respect to the partial derivative of $\mathbf{X}$. This matrix will update each time step because the matrix entries are dependent on the state of the satellite and the revolving positions of the tracking stations around the Earth.

After solving for ϕ, the next step is to produce the H-matrix, Eq. (14), for each observation. First, $\tilde{H}$ matrix must be calculated using Eq. (11). Like the A-matrix, the $\tilde{H}$ matrix updates with each time step because its elements are dependent on the state of the satellite and the actively tracking ground station. However, depending on which ground station is tracking the satellite, different entry positions in $\tilde{H}$ will be filled. For example, if Station 101 is tracking the satellite then columns 10-12 have non-zero values while columns 13-18 are equal to 0. The non-zero columns change to 13-15 if the tracking station is Station 337 or columns 16-18 if it is Station 394. After computing $\tilde{H}$, it is multiplied by the ϕ matrix to produce the corresponding H-matrix. This process is repeated until there is a H-matrix for each time step.

The next calculation is to determine the difference the between the expected range and range-rate data and what was computed using the satellite's estimated state. To calculate the estimated range, the estimated satellite position vectors and the location of actively tracking ground station are used to solve Eq. (7). To calculate the estimated range-rate, the satellite's estimated position and velocity vectors as well as the location of tracking station are used to solve Eq. (8). Then, the estimated range and range-rate values are subtracted from the collected observation data to produce the residual error (Eq. (10) and Eq. (15).

With the H-matrices and observation residuals computed for each time step, the next step in the batch processor is to improve the initial estimate. There are two matrices that need to be calculated, the information matrix, M, and the N-matrix. Using Eq. (16) to calculate M, add the a priori covariance matrix of the state vector elements, $\bar{P}_0$ to the sum of the matrix multiplication of H^TWH for each observation. Then, the N-matrix can be calculated by using Eq. (17) and adding the product of $\bar{P}_0$ and the state a priori values, $\bar{x}_0$, to the sum of the H^TWy for each observation. With smaller data sets, to solve for the state deviation vector, $\hat{x}_0$, invert the M-matrix and multiply it by the N-matrix. However, the M-matrix is supporting a large data set and there is a loss information by simply inverting M. Instead, a Cholesky Decomposition will be performed to increase the accuracy of $\hat{x}_0$. To start, create an upper-triangular matrix, R, such that $R^TR = M$ by using Eq. (18). Then with the newly created R matrix, solve for z using Eq. (19). Finally with both z and R calculated, solve Eq. (20) to produce $\bar{x}_0$ without the inversion of the M matrix.

If this was the first iteration through the batch processor add $\hat{x}_0$ to the initial state estimation, subtract $\hat{x}_0$ from $\bar{x}_0$, and then rerun the processor a second time. If this was the second or later iteration of the batch processor, check to see if the root mean squares, RMS, of the range and range-rate residuals have converged. To do this, start by using Eqs. (23) and (24) to calculate the RMS values for this loop through the processor and compare them with the values from the previous iteration. If the error tolerance between these two values are very small, then it can be said that the RMS values have converged. Otherwise, improve the initial state estimate and run through the processor again.

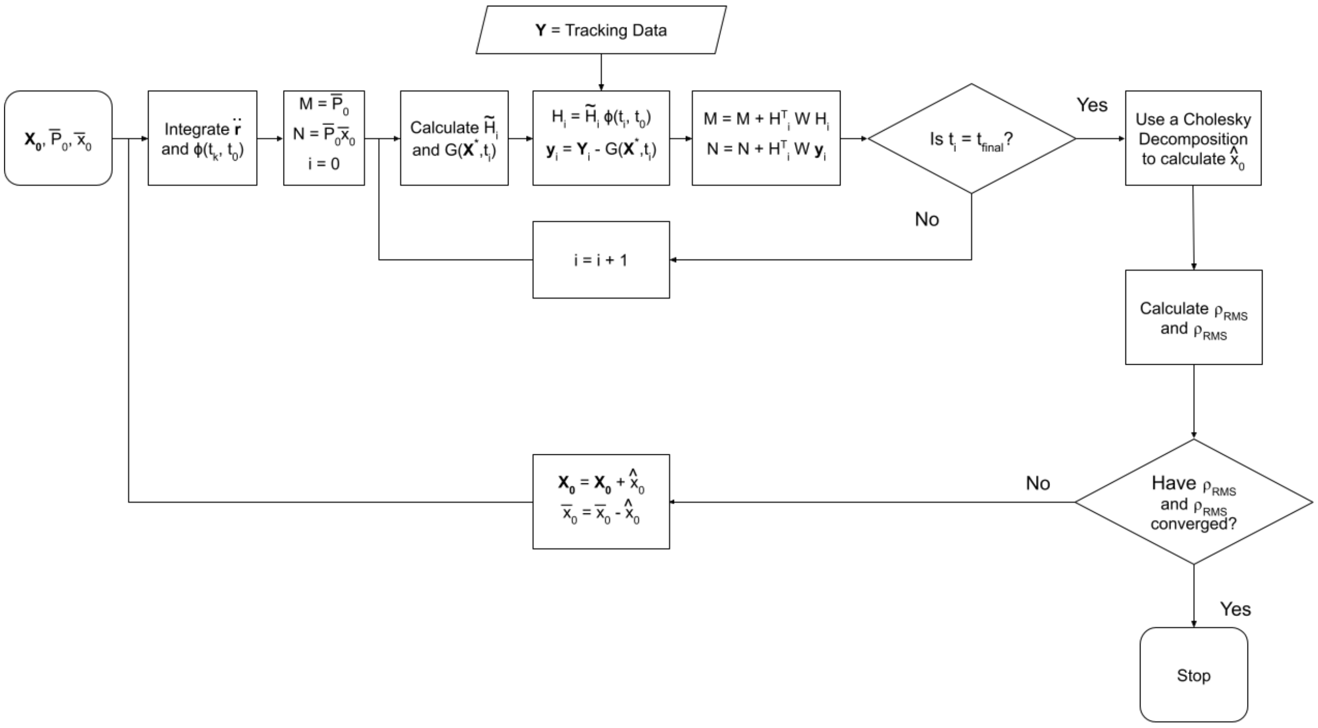

Figure 1: Flow Diagram of the Batch Processor

2.4 Analysis Outputs (Part 5b)

Using the input information above, the batch processor was able to produce the following outputs.

- Improved state estimate vector with the following units:
  - $\mathrm{r}_{Sat} = \begin{bmatrix} x & y & z \end{bmatrix}$ meters
  - $\dot{r}_{Sat} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}$ meters/second
  - μ_{Earth} = meters^3/second^2
  - J_2 = unitless
  - C_D = unitless
  - $\mathrm{r}_{S1} = \begin{bmatrix} x_{S1} & y_{S1} & z_{S1} \end{bmatrix}$ meters
  - $\mathrm{r}_{S2} = \begin{bmatrix} x_{S2} & y_{S2} & z_{S2} \end{bmatrix}$ meters
  - $\mathrm{r}_{S3} = \begin{bmatrix} x_{S3} & y_{S3} & z_{S3} \end{bmatrix}$ meters
- RMS of the observed range (ρ_{range}) residuals (meters)
- RMS of the observed range-rate ($\dot{\rho}_{range}$) residuals (meters/second)
- Covariance matrix of the state vector elements (unitless)
- Standard deviations (STDs) of the state vector elements
  - STD of $\mathrm{r}_{Sat} = \begin{bmatrix} \sigma_x & \sigma_y & \sigma_z \end{bmatrix}$ meters

- STD of $\dot{r}_{Sat}$ = $\begin{bmatrix}\sigma_{\dot{x}} & \sigma_{\dot{y}} & \sigma_{\dot{z}}\end{bmatrix}$ meters/second
- $\sigma_{\mu_{Earth}}$ = meters^3/second^2
- σ_{J_2} = unitless
- σ_{C_D} = unitless
- STD of r_{S1} = $\begin{bmatrix}\sigma_{x_{S1}} & \sigma_{y_{S1}} & \sigma_{z_{S1}}\end{bmatrix}$ meters
- STD of r_{S2} = $\begin{bmatrix}\sigma_{x_{S2}} & \sigma_{y_{S2}} & \sigma_{z_{S2}}\end{bmatrix}$ meters
- STD of r_{S3} = $\begin{bmatrix}\sigma_{x_{S3}} & \sigma_{y_{S3}} & \sigma_{z_{S3}}\end{bmatrix}$ meters

2.5 Analysis Results (Part 5c and 5d)

To produce the following results, the MATLAB integrator ode45 was used. This integrator was used to numerically integrate the EOMs and the STM, Eqs. (1) and (12), and produce a state vector and STM matrix for each time step of the observation data. The function ode45 takes three inputs: the function that needs to be integrated, the time to integrate over, and the inititial conditions. However, the initial conditions must be a column vector. After reshaping the STM matrix into a column matrix, the initial conditions would $\mathbf{X}_{0,input} = \begin{bmatrix}\mathbf{X}_0 \; \phi_{reshape}(t_0,\; t_0)\end{bmatrix}^T$, with a size of 342 by 1. The initial conditions are that size because when the STM is reshaped into a column it becomes a column vector 18^2 entries, 324, and then add the 18 from the initial state vector.

To prevent the batch processor from running indefinitely, there must be a defined max number of loops or a test of convergence. For this simulation, this processor would stop after 20 iterations or if the relative error of the ρ_{range} and $\dot{\rho}_{range}$ was less than 1e-6. For the given initial conditions, the batch processor reached convergence after the third iteration.

Iteration	r_x (m)	r_y (m)	r_z (m)	v_x (m/s)	v_y (m/s)	v_z (m/s)
1	757700	5222607	4851500	2213.21	4678.34	-5371.3
2	757699.9637	5222606.7254	4851499.8201	2213.2509	4678.3727	-5371.3148
3	757700.2904	5222606.5773	4851499.7391	2213.2506	4678.3727	-5371.3144

Table 1: Improving Initial State Estimate of $r_{Satellite}$ and $v_{Satellite}$

Iteration	μ_{Earth} (m^3/s^2)	J_2 (1)	C_D (1)	$r_{S1,x}$ (m)	$r_{S1,y}$ (m)	$r_{S1,z}$ (m)
1	3.986004415e14	1.0826e-3	2	-5127510	-3794160	0
2	3.986004320407484e14	1.082e-3	2.1475	-5127510	-3794160	-2.5413e-07
3	3.986003987346084e14	1.082e-3	2.1887	-5127510	-3794160	2.7587e-10

Table 2: Improving Initial State Estimate of μ_{Earth}, J_2, C_D, and r_{S1}

Iteration	$r_{S2,x}$ (m)	$r_{S2,y}$ (m)	$r_{S2,z}$ (m)	$r_{S3,x}$ (m)	$r_{S3,y}$ (m)	$r_{S3,z}$ (m)
1	3860910	3238490	3898094	549505	-1380872	6182197
2	3860899.4365	3238499.9832	3898099.795	549499.2182	-1380869.6559	6182198.5122
3	3860899.9917	3238500.0033	3898099.9771	549499.9914	-1380869.979	6182199.9758

Table 3: Improving Initial State Estimate of r_{S2} and r_{S3}

Iteration	r_x (m)	r_y (m)	r_z (m)	v_x (m/s)	v_y (m/s)	v_z (m/s)
1	0.0075263	0.011794	0.014842	8.6395e-06	1.4444e-05	1.0235e-05
2	0.007525	0.011794	0.014842	8.639e-06	1.4443e-05	1.0235e-05
3	0.007525	0.011794	0.014842	8.639e-06	1.4443e-05	1.0235e-05

Table 4: Standard Deviations of $r_{Satellite}$ and $v_{Satellite}$

Iteration	μ_{Earth} (m^3/s^2)	J_2 (1)	C_D (1)	$r_{S1,x}$ (m)	$r_{S1,y}$ (m)	$r_{S1,z}$ (m)
1	415742.1532	2.4455e-10	0.0038037	1e-05	1e-05	1e-05
2	415708.4163	2.4456e-10	0.0038069	1e-05	1e-05	1e-05
3	415708.4126	2.4456e-10	0.0038069	1e-05	1e-05	1e-05

Table 5: Standard Deviations of μ_{Earth}, J_2, C_D, and r_{S1}

Iteration	$r_{S2,x}$ (m)	$r_{S2,y}$ (m)	$r_{S2,z}$ (m)	$r_{S3,x}$ (m)	$r_{S3,y}$ (m)	$r_{S3,z}$ (m)
1	0.0052721	0.0084484	0.0087617	0.0073393	0.012748	0.016562
2	0.0052712	0.0084479	0.008761	0.0073389	0.012748	0.016558
3	0.0052712	0.0084479	0.008761	0.0073389	0.012748	0.016558

Table 6: Standard Deviations of r_{S2} and r_{S3}

Iteration	RMS of ρ_{range} (m)	RMS of $\dot{\rho}_{range}$ (m/s)
1	732.74831	2.9001651
2	0.31957068	0.0011997142
3	0.0097454189	0.00099792873

Table 7: Improving RMS of the Range and Range-rate

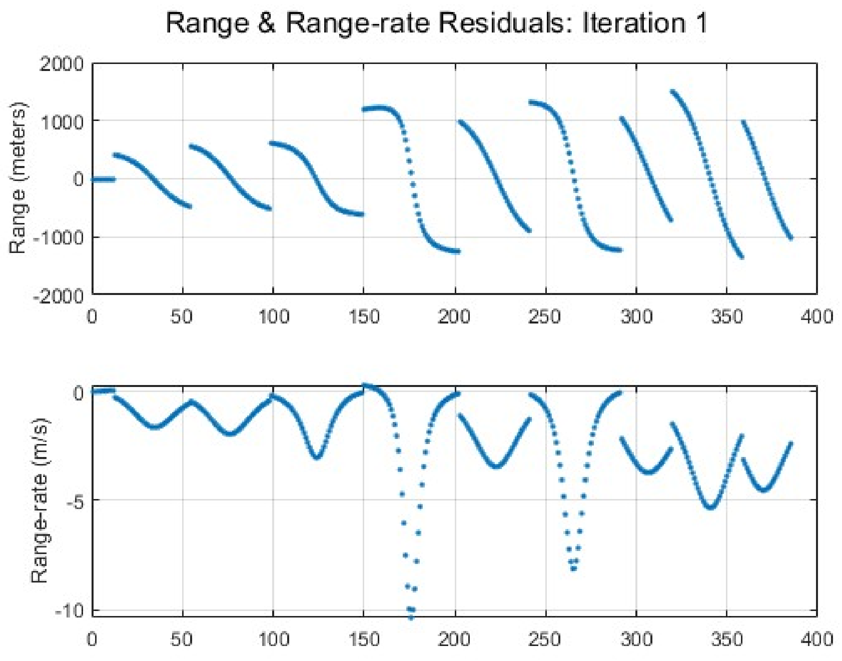

Figure 2: Range and Range-rate Observation Residuals: Iteration 1

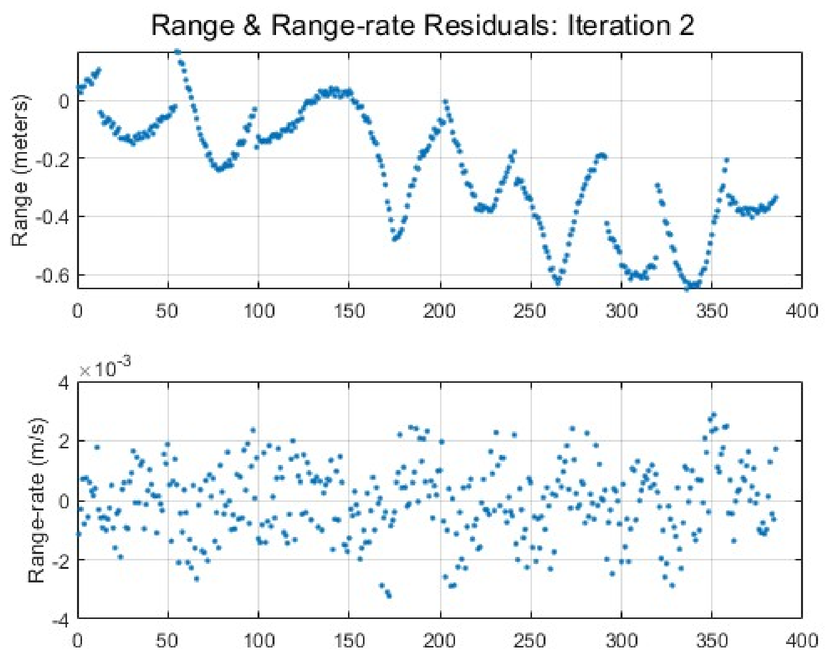

Figure 3: Range and Range-rate Observation Residuals: Iteration 2

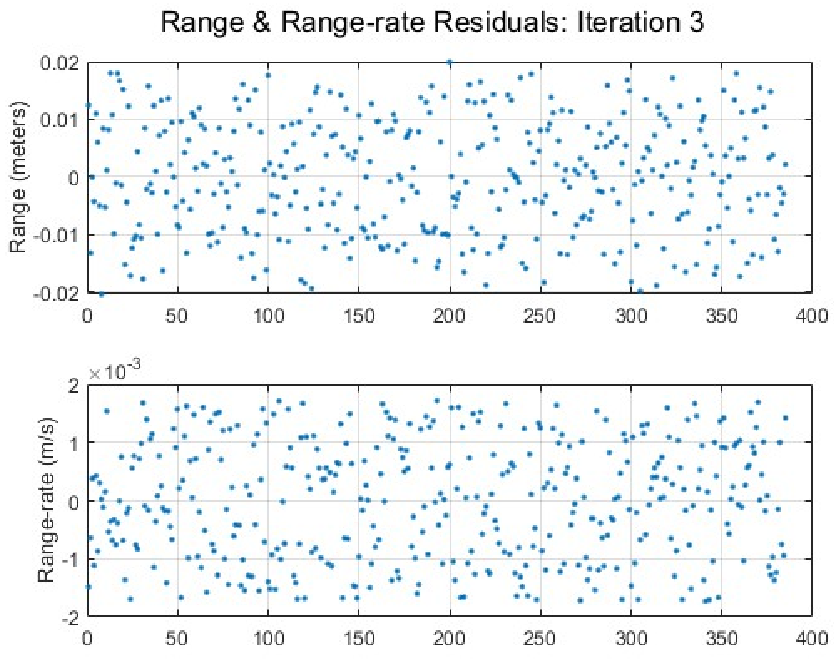

Figure 4: Range and Range-rate Observation Residuals: Iteration 3

2.6 Strength of The Observation Data (Part 4b)

It is important to know if the data being used in the analysis if of high quality. Input data sets with just a handful of bad observations can ruin most, if not all, of the output data. To learn about the quality of the observation data used in this analysis, this section will look at the effect on the RMS values and the covariance matrix when using only the range or range-rate data.

First, several lines of code from the previous analysis need to be changed to account for only the range or range-rate data.

1. As seen in Eq. (11), the $\tilde{H}$ matrix contains the partial derivative of the range and range-rate equations with respect to the elements of the state vector. Thus only one row of the $\tilde{H}$ matrix will used to calculate the H-matrix.
2. Only the residuals errors in the range or range-rate will be considered
3. The respective entry in the weighting matrix, W, will be used in Eqs. (16) and (17) to calculate the M and N matrices
4. The residual RMS value of either the range or range-rate will be used to consider convergence

Results Using Only the ρ_{range} Data:

Iteration	RMS of ρ_{range} (m)	Expected RMS of ρ_{range} (m)
1	732.7483	732.74831
2	0.31946475	0.31957068
3	0.00974522	0.0097454189

Table 8: Converging RMS Values of ρ_{range} (m)

	σ_{range} With Range Data	σ_{range} With All Data
r_x (m)	0.0075272478	0.0075250296
r_y (m)	0.011797439	0.0117944842
r_z (m)	0.014850619	0.0148420162
v_x (m/s)	8.6414579e-06	8.63897044e-06
v_y (m/s)	1.4447512e-05	1.44433210e-05
v_z (m/s)	1.0244092e-05	1.02353221e-05
μ_{Earth} (m^3/s^2)	415875.15	415708.41
J_2 (1)	2.4463800e-10	2.44559717e-10
C_D (1)	0.0038120938	0.00380688216
$r_{S1,x}$ (m)	9.9999856e-06	9.99998562e-06
$r_{S1,y}$ (m)	9.9999921-06	9.99999212e-06
$r_{S1,z}$ (m)	9.9999971-06	9.99999713e-06
$r_{S2,x}$ (m)	0.0052735374	0.0052711976
$r_{S2,y}$ (m)	0.0084507903	0.0084479311
$r_{S2,z}$ (m)	0.0087687539	0.0087610126
$r_{S3,x}$ (m)	0.0073402162	0.0073388680
$r_{S3,y}$ (m)	0.012751779	0.012748114
$r_{S3,z}$ (m)	0.016607498	0.016557969

Table 9: Comparison of the σ_{range} values

Results Using Only the $\dot{\rho}_{range}$ Data:

Iteration	RMS of $\dot{\rho}_{range}$ (m/s)	Expected RMS of $\dot{\rho}_{range}$ (m/s)
1	2.9001650	2.9001651
2	0.0016723595	0.0011997142
3	0.00097775308	0.00099792873

Table 10: Converging RMS Values of $\dot{\rho}_{range}$ (m/s)

	σ_{range} With Range Data	σ_{range} With All Data
r_x (m)	0.67358365	0.0075250296
r_y (m)	1.1075941	0.0117944842
r_z (m)	1.1839273	0.0148420162
v_x (m/s)	0.00073084011	8.63897044e-06
v_y (m/s)	0.0012628903	1.44433210e-05
v_z (m/s)	0.00078385712	1.02353221e-05
μ_{Earth} (m^3/s^2)	36253072	415708.41
J_2 (1)	2.8158668e-08	2.44559717e-10
C_D (1)	0.090696107	0.00380688216
$r_{S1,x}$ (m)	9.9999999e-06	9.99998562e-06
$r_{S1,y}$ (m)	9.9999999e-06	9.99999212e-06
$r_{S1,z}$ (m)	9.9999999e-06	9.99999713e-06
$r_{S2,x}$ (m)	0.37570188	0.0052711976
$r_{S2,y}$ (m)	0.69832768	0.0084479311
$r_{S2,z}$ (m)	0.50751587	0.0087610126
$r_{S3,x}$ (m)	0.67081381	0.0073388680
$r_{S3,y}$ (m)	1.1641558	0.012748114
$r_{S3,z}$ (m)	0.45003207	0.016557969

Table 11: Comparison of the $\sigma_{range-rate}$ values

While the respective RMS values are similar to what was found in the previous analysis, the final covariances using only the range or range-rate information differ quite significantly. Looking at Table 9 and 11, it can been seen that the STDs using only the range data provides values similar to the expected values while using only the range-rate data returns much higher STDs. This shows that the quality of the observed range data is much higher than the quality of the observed range-rate data. The justification for this difference lies within the noise added to the data. The range data has an average magnitude is 2.8 million meters with an added noise that has a STD of 1cm, a noise-to-range ratio of 4e-9. The range-rate data, on the other hand, has an average of 3500 m/sec with an added noise that has a STD of 1mm/sec, a noise-to-range-rate ratio of 3e-7. The results of the STD from the range only data is about 100 times smaller than the results of range-rate only data, closely resembling the difference in magnitude of the two noise-mean-value ratios.

3 Additional Analysis (Part 6)

3.1 Ground Station Tracking Custody

Using the improved estimate of the satellite's initial conditions, the following analysis will propagating the satellite's orbit for 1 week after the last known track to determine any future observation intervals and determine which of the three ground sites contributes the greatest amount of satellite observation time. To constitute an active observation, the satellite must appear above the ground site's local horizon for more than 20 seconds. Like the analysis above, the ground sites are being modeled to have a measurement rate of 0.05 Hz.

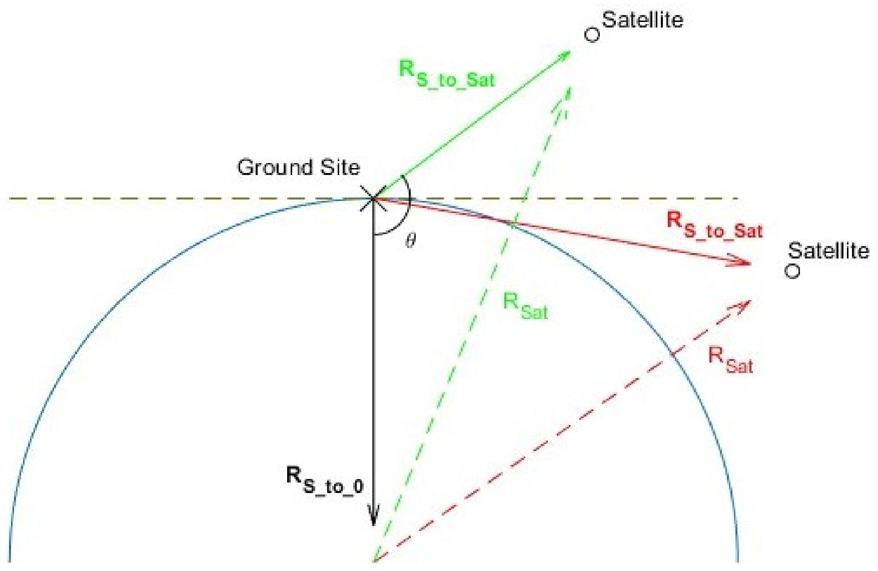

Figure 5: 2D Representation of Tracking Custody Analysis

3.2 Equations for the Addition Analysis:

$$R_S(t) = \begin{bmatrix} cos(\theta(t)) & -sin(\theta(t)) & 0 \\ sin(\theta(t)) & cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{xS} \\ R_{yS} \\ R_{zS} \end{bmatrix} \tag{27}$$

$$\mathbf{R_{S\_to\_0}} = \mathbf{0} - R_S \tag{28}$$

$$\mathbf{R_{S\_to\_Sat}} = R_{Sat} - R_S \tag{29}$$

$$\mathbf{R_{S\_to\_0}} \cdot \mathbf{R_{S\_to\_Sat}} = |\mathbf{R_{S\_to\_0}}||\mathbf{R_{S\_to\_Sat}}|cos(\theta(t)) \tag{30}$$

3.3 Use of Equations

The first step of this analysis is to propagate the satellite's orbit. Using the same approach as the previous analysis, integrate the EOMs over the new, 1 week time interval. Then, the initial position of each ground site is rotated using Eqs. (27) and (6) according to each time step. Once the all of position vectors are known for the satellite and the ground sites, the next step is to calculate the vectors from the ground sites to the satellite and the vector from the ground site to the center of the Earth. This can be seen in Figure 5 where the elevation angle about the ground site is measured from the vector to Earth's center to the vector pointing at the satellite. To get the vector pointing to Earth's center, simply subtract the ground station's position from a 0 vector, Eq. (28). Then get the vector from the ground site to the satellite, each of the position vectors from the two objects need to be subtracted, Eq. (29). After calculating these two new vectors take the dot product. The result is then used in Eq. (30) to find the angle between the two vectors.

After performing the vector calculations for all time steps, the next step is to find all of the times where there separation angle, θ, is greater than 90 degrees. If this angle is less than 90 degrees, then the satellite it below the local horizon and cannot be seen by the ground site. Looking back at Figure 5, the green satellite is above the gray dashed line, the local horizon, and can be seen by the ground site. The red satellite on the other hand is below the local horizon and cannot be seen.

3.4 Analysis Results

After performing the calculations described in section 5.3 Use of Equations, the following results were collected.

	Station 101	Station 337	Station 394
Number of Observation Intervals	28	38	97
Average Duration Observed(s)	717.857142857143	697.894736842105	798.144329896907
Max Duration Observed (s)	900	880	900
Total Time Observed (s)	20100	26520	77420
Percentage of Time Observed	3.32352%	4.38506%	12.8013%

Table 12: Predicted Tracking Results After 1 Week

3.5 Additional Analysis Conclusions

Prior to receive the results of the tracking custody analysis, the expectation that was that Station 394, Thule, Greenland, would out perform the other two stations in terms of overall tracking time due to its location at a high latitude. Due to the satellite's high inclination, the tracking site at Thule provides the advantage of frequent observations each time the satellite crosses the North Pole. What was interesting to discover from the results of the analysis is that about only 20% of the time can one of the three ground sites observe the satellite. Initial expectations placed the observation percentage into the 50-60% range, but without additional ground sites in the Southern hemisphere the lower value is justifiable.

4 Lessons Learned (Part 5e)

The biggest lesson learned during this project was the proper way to build a batch processor in MATLAB. My biggest issue during the development of my algorithm was that it was slow and my RMS values would not converge. My data would match the expected values of the first residuals and RMSs, but after the first iteration through the processor my values would not converge. Tracing through my code, I had a few typographical errors in my equations for the A and $\tilde{H}$ matrices but what I discovered that was my bigger issue was that my information matrix was not positive definite. When performing the Cholesky decomposition, I would produce imaginary solutions for the best estimate and my RMS values would increase each iteration and not converge. I spent several days looking through my code but I could not seem to find a solution and I decided to rebuild the main body of my batch processor and make it more compact. In the previous version of my code I called ode45 twice for each time step, once to update the satellite dynamics and once to update the STM for the new time step. In the current version, I changed it to one ode45 call per iteration. Not only did this resolve the speed issue of my processor, but it also resolved the problem with the information matrix not being positive definite. I was able to get good results and data convergence within the first few iterations of the processor. I would like to get more practice using both a batch processor and a sequential processor, but I think it is safe to say I feel much more comfortable working with state estimation algorithms.

A MATLAB Code (Part 5b)

```
%%ASTE 583 - Orbital Determination Project
%% Program Formatting
clear;
clc;
close all;

format long g;

options = odeset('RelTol', 1e-12, 'AbsTol', 1e-12);
% Option set used in ode45 to improve results
%% Input Data
global thetaDot rEarth cs m rho0 r0 Hscale
% Variables to be used in supporting, in-script functions

% Earth Orientation
thetaDot = 7.2921158543e-5; % rad/s

% Earth Gravity Model
muEarth = 3.986004415e14; % m^3/s^2
J2 = 1.082626925638815e-3;

rEarth = 6378136.3; % m

% Atmospheric Drag Model
rho0 = 3.614e-13; % kg/m^3
r0 = 7e5 + rEarth; % m, satellite radius magnitude at t0
Hscale = 88667; % m

cs = 3; % m^2
m = 970; % kg
cD = 2;

% Tracking Station Locations (Body-Fixed) [x y z]
station101 = [-5127510.0 -3794160.0 0.0];
station337 = [3860910.0 3238490.0 3898094.0];
station394 = [549505.0 -1380872.0 6182197.0];

% Satellite Inital Conditions (Inertial)
t0 = 0; % seconds
R0 = [757700 5222607 4851500]; % [x y z] m
V0 = [2213.21 4678.34 -5371.30]; % [vX vY vZ] m/second

% Observation Data
trackingDataInput = readtable("OD Tracking Data - Sheet1 (1).csv");
% Loads in observation data

tTrack = table2array(trackingDataInput(:,"Epoch"));
% Saves the time stamp for each observation

station = table2array(trackingDataInput(:,"Station_Number"));
% Saves the station number observing the spacecraft at each time stamp
```

```
rangeTrack = table2array(trackingDataInput(:,"Range"));
% Saves the ranges at each observation time stamp

rangeRateTrack = table2array(trackingDataInput(:,"Range_Rate"));
% Saves the range rate at each observation time step

trackingData = [rangeTrack rangeRateTrack];
% Variable used later to calculate the observation residuals
%% Algorithm Equations

% Weighting Matrix
stdRange = 0.01; % m
stdRangeRate = 0.001; % m/s
W = [stdRange^-2 0; 0 stdRangeRate^-2]; % Weighting matrix

% A priori State Deviation
x_APriori = zeros(18,1);

% A priori Covariance
P_APriori = diag([1e6,1e6,1e6,...
                  1e6,1e6,1e6,...
                  1e20,1e6,1e6,...
                  1e-10,1e-10,1e-10,...
                  1e6,1e6,1e6, ...
                  1e6,1e6,1e6]);

% Variable that produces the calculated range and range-rate information
G_ref = @(X,rS,t) ...
    [rhoRange(X(1:3),rS,t) rhoRangeRate(X(1:3),X(4:6),rS,t)];
%% Batch Processor
maxLoops = 20; % Max loops of the batch processor
eTol = 1e-6; % Error tolerance to determine if the RMS has converged

% The initial state vector
X0_update = [R0 V0 muEarth J2 cD ...
    station101 station337 station394, reshape(eye(18),1,18^2)]';
% X(1:3) = Satellite x, y, z in inertial coordinates
% X(4,6) = Satellite vX, vY, vZ in inertial coordinates
% X(7) = muEarth
% X(8) = J2
% X(9) = cD
% X(10:12) = Station 101 x, y, z in body-fixed coordinate system
% X(13:15) = Station 337 x, y, z in body-fixed coordinate system
% X(16:18) = Station 394 x, y, z in body-fixed coordinate system
% X(19:342) = State transition matrix elements reshaped into a row-vector
% *** State transition matrix = STM

% Storage variable for all range and range-rate residuals
yLoop = {};

% Storage variable used to calculate all standard deviations and
% correlation coefficients
% *** Standard deviation = STDs
varCovarMatrix = {};
```

```
% The Batch Processor
for loop = 1:maxLoops
    M = P_APriori^-1; % Information matrix
    N = P_APriori^-1 * x_APriori;

    y = zeros(length(tTrack),2);
    % Storage for the observation residuals

    X0 = X0_update; % Update of the intial state vector for this loop
    stateEstimate{loop} = X0;

    % Solve the satellite's EOM for each observation time stamp
    [~, Xcalc] = ode45(@dXdotdt,tTrack, X0, options);

    % Iterating through each observation and determining the sum of the
    % residual error to improve the intial estimate of the state
    for i = 1:length(tTrack)

        X0 = Xcalc(i,1:18)';
        % The calculated state vector for the ith time step

        phi0 = reshape(Xcalc(i,19:end),18,18);
        % The STM for the ith time step

        % Depending on the ground site observing the satellite, update the
        % coordinates of station's position
        if station(i) == 101
            rS = X0(10:12)';
        elseif station(i) == 337
            rS = X0(13:15)';
        else
            rS = X0(16:18)';
        end

        % Mapping matrix
        Htilde = Htilde_func(X0,tTrack(i),station(i));

        % H-matrix
        H{i} = Htilde * phi0;

        % Calculation of the residual fo this time step
        y(i,:) = trackingData(i,:) - G_ref(X0,rS,tTrack(i));

        % Updating the values of the information and N matrices based on
        % the new values of the observation residuals and H-matrix
        M = M + H{i}.' * W * H{i};
        N = N + H{i}.' * W * y(i,:).';
    end

    % Using the in-script Cholesky decomposition function to find the
    % state deviation vector
    xDeviation = cholesky(M,N);
```

```
% Used to check accuracy of homemade Cholesky decomp.
% R2 = chol(M);
% xBestEst2 = R2\(R2'\N);

% Summing the squares of the observation residuals for the range
rhoRMSsum = sum(y(:,1).^2);

% Calculating the RMS for range in this iteration
rhoRMS(loop) = sqrt(rhoRMSsum/length(y));

% Summing the squares of the observation residuals for the range-rate
rhoDotRMSsum = sum(y(:,2).^2);

% Calculating the RMS for range-rate in this iteration
rhoDotRMS(loop) = sqrt(rhoDotRMSsum/length(y));

% Saving observation residuals of this iteration
yLoop{loop} = y;

% Saving covariance matrix for this iteration
% P = M^-1 = R^-1*R^-T
% Cholesky decomposition performs more accurate operation than simply
% inversing the information matrix
R = zeros(size(M));
for i = 1:length(M)
    for j = i:length(M)
        if i == j
            R(i,i) = sqrt(M(i,i) - sum(R(1:i-1,i).^2));
        else
            R(i,j) = (M(i,j) - sum(R(1:i-1,i).*R(1:i-1,j)))/R(i,i);
        end
    end
end
varCovarMatrix{loop} = R^-1 * R'^-1;

figure(loop)
subplot(2,1,1)
plot(1:length(tTrack),y(:,1),'.')
ylabel("Range (meters)")
grid on

subplot(2,1,2)
plot(1:length(tTrack),y(:,2),'.')
ylabel("Range-rate (m/s)")
grid on

sgtitle("Range & Range-rate Residuals: Iteration " + loop)

% After two iterations, begin checking to see if weighted RMS meet
% local error tolerances
if loop >= 2
    if abs((rhoRMS(loop)-rhoRMS(loop-1))/rhoRMS(loop-1)) < eTol
        if abs((rhoDotRMS(loop)-rhoDotRMS(loop-1))/rhoDotRMS(loop-1)) < eTol
            break
```

```
                % If the relative error of the weighted RMS is less than the
                % assigned error tolerance, stop the processor
            end
        end
    end

    % Loop used to help print out table values
    fprintf( loop + " & ");
    for k = 1:18
        fprintf(X0_update(k) + " & ");
    end
    fprintf("\n")

    % Otherwise, improve initial state vector and run the processor
    % again
    X0_update(1:18) = X0_update(1:18) + xDeviation;
    x_APriori = x_APriori - xDeviation;

    % Plotting the range and range-rate residuals for each iteration of the
    % processor

end

figure(loop+1)
clf
subplot(2,1,1)
plot(1:loop,rhoRMS(1:end))
ylabel("Range RMS (meters)")
grid on

subplot(2,1,2)
plot(1:loop-1,rhoDotRMS(1:end-1))
ylabel("Range-rate RMS (m/s)")
grid on

sgtitle("RMS of the Range and Range-Rate Residuals")

yActual = yLoop;
%% Part 4b: Comparing the Strengths of the Observation Data
% This section compares the strengths of the range and range-rate data by
% only considering the values tied to either the range or range-rate data.
% After the outer for-loop completes, I will compare the data produced by
% the two approaches.
clear rhoRMS rhoDotRMS stateEstimate H RMS X0 X xDeviation
clear varCovarMatrix_part4
yLoop = {};
for type = [1 2]
    maxLoops = 20; % Max loops of the batch processor
    eTol = 1e-6; % Error tolerance to determine if the RMS has converged

    % The initial state vector
    X0 = [R0 V0 muEarth J2 cD ...
        station101 station337 station394, reshape(eye(18),1,18^2)]';
    % X(1:3) = Satellite x, y, z in inertial coordinates
% X(4,6) = Satellite vX, vY, vZ in inertial coordinates
% X(7) = muEarth
% X(8) = J2
% X(9) = cD
% X(10:12) = Station 101 x, y, z in body-fixed coordinate system
% X(13:15) = Station 337 x, y, z in body-fixed coordinate system
% X(16:18) = Station 394 x, y, z in body-fixed coordinate system
% X(19:342) = State transition matrix elements reshaped into a row-vector
% *** State transition matrix = STM

x_APriori = zeros(18,1);
% Storage variable for all range and range-rate residuals

% The Batch Processor
for loop = 1:maxLoops
    M = P_APriori^-1; % Information matrix
    N = P_APriori^-1 * x_APriori;

    y = zeros(length(tTrack),2);
    % Storage for the observation residuals

    X = X0; % Update of the intial state vector for this loop
    stateEstimate{type,loop} = X;
    % Solve the satellite's EOM for each observation time stamp
    [~, Xcalc] = ode45(@dXdotdt,tTrack, X, options);

    % Iterating through each observation and determining the sum of the
    % residual error to improve the intial estimate of the state
    for i = 1:length(tTrack)

        X = Xcalc(i,1:18)';
        % The calculated state vector for the ith time step

        phi0 = reshape(Xcalc(i,19:end),18,18);
        % The STM for the ith time step

        % Depending on the ground site observing the satellite, update the
        % coordinates of station's position
        if station(i) == 101
            rS = X(10:12)';
        elseif station(i) == 337
            rS = X(13:15)';
        else
            rS = X(16:18)';
        end

        % Mapping matrix
        Htilde = Htilde_func(X,tTrack(i),station(i));

        % H-matrix
        H{i} = Htilde * phi0;

        % Calculation of the residual fo this time step
        y(i,:) = trackingData(i,:) - G_ref(X,rS,tTrack(i));
        % Updating the values of the information and N matrices based on
        % the new values of the observation residuals and H-matrix
        M = M + H{i}(type,:).' * W(type,type) * H{i}(type,:);
        N = N + H{i}(type,:).' * W(type,type) * y(i,type).';
    end

    % Using the in-script Cholesky decomposition function to find the
    % state deviation vector
    xDeviation = cholesky(M,N);

    % Used to check accuracy of homemade Cholesky decomp.
    % R2 = chol(M);
    % xBestEst2 = R2\(R2'\N);

    % Summing the squares of the observation residuals for the range
    rhoRMSsum = sum(y(:,1).^2);

    % Calculating the RMS for range in this iteration
    rhoRMS(loop) = sqrt(rhoRMSsum/length(y));

    % Summing the squares of the observation residuals for the range-rate
    rhoDotRMSsum = sum(y(:,2).^2);

    % Calculating the RMS for range-rate in this iteration
    rhoDotRMS(loop) = sqrt(rhoDotRMSsum/length(y));

    RMS{type}(1,loop) = rhoRMS(loop);
    RMS{type}(2,loop) = rhoDotRMS(loop);

    % Saving observation residuals of this iteration
    yLoop{type,loop} = y;

    % Saving covariance matrix for this iteration
    % P = M^-1 = R^-1*R^-T
    % Cholesky decomposition performs more accurate operation than simply
    % inversing the information matrix
    R = zeros(size(M));
    for i = 1:length(M)
        for j = i:length(M)
            if i == j
                R(i,i) = sqrt(M(i,i) - sum(R(1:i-1,i).^2));
            else
                R(i,j) = (M(i,j) - sum(R(1:i-1,i).*R(1:i-1,j)))/R(i,i);
            end
        end
    end
    varCovarMatrix_part4{type,loop} = R^-1 * R'^-1;

    % After two iterations, begin checking to see if weighted RMS meet
    % local error tolerances
    if loop >= 2
        if abs((RMS{type}(type,loop)-RMS{type}(type,loop-1))/...
                RMS{type}(type,loop-1)) < eTol
```

```
                break
                % If the relative error of the weighted RMS is less than the
                % assigned error tolerance, stop the processor
            else
                X0(1:18) = X0(1:18) + xDeviation;
                x_APriori = x_APriori - xDeviation;
            end
        else
            % Otherwise, improve initial state vector and run the processor
            % again
            X0(1:18) = X0(1:18) + xDeviation;
            x_APriori = x_APriori - xDeviation;
        end
    end
end
%
for i = 1:length(yLoop)
    if isempty(yLoop{1,i}) || isempty(yLoop{2,i})
        break;
    else
        figure(10+2*(i-1)+1)
        plot(1:length(tTrack),yLoop{1,i}(:,1),'.',1:length(tTrack),yActual{i}(:,1),'.')
        ylabel("Range (meters)")
        grid on
        title("Range Residuals: Iteration " + i)

        figure(10+2*i)
        plot(1:length(tTrack),yLoop{2,i}(:,2),'.',1:length(tTrack),yActual{i}(:,2),'.')
        ylabel("Range-rate (m/s)")
        grid on
        title("Range-rate Residuals: Iteration " + i)
    end
end


%% Part 5: State Standard Deviations and Covariance

for i = 1:length(varCovarMatrix)-1
    P = varCovarMatrix{i};
    % Use saved information matrix from batch processor above

    std{i} = sqrt(diag(P))';
    % The STDs are the square root of the diagonal of the
    % variance-covariance matrix

    % Loop used to help print out table values
    % fprintf( i + " & ");
    % for k = 1:18
    %     fprintf(std{i}(k) + " & ");
    % end
    % fprintf("\n")

end
%% Part 6: Additional Analysis
```

```
% Determine ground site tracking custody over 1 week and determine which
% site provides the greatest amount of coverage time. The process:
% propagate the satellite's orbit for 1 week starting at t+5 hours. The
% goal: predict which station sees the satellite the most often and which
% has the longest average tracking time. Approved by Dr. Hintz on
% 11/29/2023



rotateZ = @(t) [cos(theta(t)) -sin(theta(t)) 0;
                 sin(theta(t))  cos(theta(t)) 0;
                 0              0             1];
% To account for Earth's rotation, the ground station positions will need
% to be rotated counter-clockwise by some angle theta


vectAngle = [];
% Storage for the angle between the vector from the ground site to the
% satellite and the ground site to the center of the Earth

% When tracking the satellite the angle with be acute if the ground site
% cannot see the spacecraft at the current time stamp. If the angle is
% obtuse, the the spacecraft is above the station's local horizon.
minOBSangle = 90;

nDays = 7; % number of days
startOffset = 5; % number of hours
t = [0 startOffset*3600:20:nDays*24*3600-20]; % array of time stamps

[~, Xpropagate] = ode45(@dXdotdt,t, X0_update, options);
% Propagation of the best intial estimate, found by the batch processor
% above

for i = 1:length(t)
    % Rotation of the coordinates of the ground sites to account for the
    % Earth's orientation at the ith time step.
    rS1 = (rotateZ(t(i)) * Xpropagate(i,10:12)')';
    rS2 = (rotateZ(t(i)) * Xpropagate(i,13:15)')';
    rS3 = (rotateZ(t(i)) * Xpropagate(i,16:18)')';

    % The satellites interial coordinates at the ith time step
    r = Xpropagate(i,1:3);

    % Need to determine the vector from the ground site to the satellite
    rS1toSat = r - rS1;
    rS2toSat = r - rS2;
    rS3toSat = r - rS3;

    % Need to reorient the vector from the ground site to the center of the
    % Earth
    rS1to0 =- rS1;
    rS2to0 =- rS2;
    rS3to0 =- rS3;
    % Perfoming a dot product to find the angle between the two vectors
    % dot(-rS,rStoSat) = |rS|*|rStoSat|cos(theta)
    % The rS vector in the dot product need to be negative because we want
    % the vector from the ground site to the center of the Earth.
    vectAngle(1,i) = acosd(dot(rS1to0,rS1toSat)/ ...
        norm(rS1to0)/norm(rS1toSat));
    vectAngle(2,i) = acosd(dot(rS2to0,rS2toSat)/ ...
        norm(rS1to0)/norm(rS2toSat));
    vectAngle(3,i) = acosd(dot(rS3to0,rS3toSat)/ ...
        norm(rS1to0)/norm(rS3toSat));
end

% Since we determined that the ground site would be able to see satellite
% if the angle between the two vectors was greater than 90 degrees, we find
% all the times t where this is true
rS1Custody = t(vectAngle(1,:) >= minOBSangle);
rS2Custody = t(vectAngle(2,:) >= minOBSangle);
rS3Custody = t(vectAngle(3,:) >= minOBSangle);

% Starting the total tracking time at 0 seconds and 1 interval
% Whenever there is a break greater than 20 seconds, there is a new
% interval and the counter increases. The total duration will increase when
% there are consecutive tracking times
rS1CustodyTime = 0;
rS1CustodyIntervals = 1;
rS2CustodyTime = 0;
rS2CustodyIntervals = 1;
rS3CustodyTime = 0;
rS3CustodyIntervals = 1;

% Since each site will likely have different amount of tracking intervals,
% the for-loop is split up into 3 parts to account for it
for i = 2:length(rS1Custody)
    if rS1Custody(i) - rS1Custody(i-1) == 20
        rS1CustodyTime(end) = rS1CustodyTime(end) + 20;
    else
        rS1CustodyIntervals = rS1CustodyIntervals + 1;
        rS1CustodyTime = [rS1CustodyTime 0];
    end
end

for i = 2:length(rS2Custody)
    if rS2Custody(i) - rS2Custody(i-1) == 20
        rS2CustodyTime(end) = rS2CustodyTime(end) + 20;
    else
        rS2CustodyIntervals = rS2CustodyIntervals + 1;
        rS2CustodyTime = [rS2CustodyTime 0];
    end
end

for i = 2:length(rS3Custody)
    if rS3Custody(i) - rS3Custody(i-1) == 20
        rS3CustodyTime(end) = rS3CustodyTime(end) + 20;
    else
```

```
            rS3CustodyIntervals(end) = rS3CustodyIntervals(end) + 1;
            rS3CustodyTime = [rS3CustodyTime 0];
        end
end
%% Functions

function X0 = dXdotdt(~,y)
% This function is used in ode45 to numerically integrate the EOMs of the
% satellite's orbit
global rEarth rho0 m r0 Hscale cs

vDot = @(r,v,muEarth,J2,cD) -muEarth * ( ...
     1/norm(r)^3*eye(3) + 3/2*J2*rEarth^2/norm(r)^5 *( ...
     diag([1,1,3]) - 5*(r(3)/norm(r))^2*eye(3)))*r - ...
     rho0/(2*m) * exp((norm(r)-r0)/-Hscale)*cs*cD*norm(vA(r,v))*vA(r,v);

A = Amatrix(y(1:18));
phiDot = A * reshape(y(19:end),18,18);
X0 = [y(4:6);
     vDot(y(1:3),y(4:6),y(7),y(8),y(9));
     zeros(12,1);
     reshape(phiDot,18^2,1)];
end

function rho = rhoDensity(r)
% This function calculates the local air density for a given r
global rho0 r0 Hscale
rho = rho0*exp(-1*(r - r0)/Hscale);
end

function rad = theta(t)
% Calculates the Earth's orientation t seconds after October 3, 1999,
% 23:11:09.1814 UTC
global thetaDot
rad = thetaDot * t; % radians
end

function vAvect = vA(r,v)
% Calculates thesatellite's velocity relative to the Earth's atmosphere
% given the satellites position and velocity vector
global thetaDot
vAvect = v - thetaDot * cross([0;0;1], r);
end

function rho = rhoRange(r,rS,t)
% Calculates the distance between the satellite and a ground site, given
% the satellite and graound site's position vectors and the time
x = r(1);
y = r(2);
z = r(3);

xS = rS(1);
yS = rS(2);
zS = rS(3);
```

```
rho = sqrt(...
    x^2 + y^2 + z^2 + xS^2 + yS^2 + zS^2 ...
    - 2*(x*xS + y*yS)*cos(theta(t)) ...
    + 2*(x*yS - y*xS)*sin(theta(t)) ...
    - 2*z*zS ...
    );
end

function rhoRate = rhoRangeRate(r,v,rS,t)
% Calculates the rate at which the distance between the satellite and the
% ground site is changing. This requires the satellite's position and
% velocity vectors, the position vector of the ground site, and the current
% time
global thetaDot
x = r(1);
y = r(2);
z = r(3);
vx = v(1);
vy = v(2);
vz = v(3);

xS = rS(1);
yS = rS(2);
zS = rS(3);

rhoRate = 1/rhoRange(r,rS,t) * ( ...
    x*vx + y*vy + z*vz - ( ...
    vx*xS + vy*yS)*cos(theta(t)) + ...
    thetaDot*(x*xS + y*yS)*sin(theta(t)) + ( ...
    vx*yS - vy*xS)*sin(theta(t)) + ...
    thetaDot*(x*yS - y*xS)*cos(theta(t)) - vz*zS);
end

function A = Amatrix(X0)
% This function calculates the partial derivatives of the A matrix given
% the elements of the state vector
global thetaDot rEarth cs m Hscale

r = X0(1:3);
v = X0(4:6);
muEarth = X0(7);
J2 = X0(8);
cD = X0(9);

x = r(1);
y = r(2);
z = r(3);
vx = v(1);
vy = v(2);
vz = v(3);

R = norm(r);
rhoA = rhoDensity(R);
```

```
VA = norm(vA(r,v));

A = zeros(18);

% dxDot/dxDot
A(1,4) = 1;

% dyDot/dyDot
A(2,5) = 1;

% dzDot/dzDot
A(3,6) = 1;

% dxDotDot/dx
A(4,1) = -muEarth/R^3 *  ( ...
    1 - 3/2*J2*(rEarth/R)^2 * (5*(z/R)^2 - 1)) + ...
    3*muEarth*x^2/R^5 * ( ...
    1 - 5/2*J2*(rEarth/R)^2 * (7*(z/R)^2 - 1)) + ...
    1/2*cD*cs/m*rhoA*VA * ...
    x * (vx + thetaDot*y)/R/Hscale - ...
    1/2*cD*cs/m*rhoA * ( ...
    -thetaDot*vy + thetaDot^2*x)*(vx + thetaDot*y)/VA;

% dxDotDot/dy
A(4,2) = 3*muEarth*x*y/R^5 * ( ...
    1 - 5/2*J2*(rEarth/R)^2 * (7*(z/R)^2 - 1)) + ...
    1/2*cD*cs/m*rhoA*VA * ...
    y*(vx + thetaDot*y)/R/Hscale - ...
    1/2*cD*cs/m*rhoA * ( ...
    thetaDot*vx + thetaDot^2*y) * ( ...
    vx + thetaDot*y)/VA - ...
    1/2*cD*cs/m*rhoA*VA*thetaDot;

% dxDotDot/dz
A(4,3) = 3*muEarth*x*z/R^5 * ( ...
    1 - 5/2*J2*(rEarth/R)^2 * (7*(z/R)^2 - 3)) + ...
    1/2*cD*cs/m*rhoA*VA * ...
    z*(vx+thetaDot*y)/R/Hscale;

% dxDotDot/dxDot
A(4,4) = -1/2*cD*cs/m*rhoA * ...
    (vx+thetaDot*y)^2/VA - ...
    1/2*cD*cs/m*rhoA*VA;

% dxDotDot/dyDot
A(4,5) = -1/2*cD*cs/m*rhoA * ...
    (vy-thetaDot*x)*(vx+thetaDot*y)/VA;

% dxDotDot/dzDot
A(4,6) = -1/2*cD*cs/m*rhoA * vz *...
    (vx+thetaDot*y)/VA;

% dxDotDot/dMu
A(4,7) = -x/R^3 * ...
```

```
          (1 - 3/2*J2*(rEarth/R)^2 * (5*(z/R)^2 - 1));

% dxDotDot/dJ2
A(4,8) = 3*muEarth*x/2/R^3 * ...
          (rEarth/R)^2 * (5*(z/R)^2-1);

% dxDotDot/dCd
A(4,9) = -1/2*cs/m*rhoA*VA*(vx+thetaDot*y);


% dyDotDot/dx
A(5,1) = 3*muEarth*x*y/R^5 * ( ...
          1 - 5/2*J2*(rEarth/R)^2 * (7*(z/R)^2-1)) + ...
          1/2*cD*cs/m*rhoA*VA * ( ...
          vy - thetaDot*x)*x/R/Hscale - ...
          1/2*cD*cs/m*rhoA * ( ...
          thetaDot^2*x - thetaDot*vy) * ( ...
          vy-thetaDot*x)/VA + ...
          1/2*cD*cs/m*rhoA*VA*thetaDot;

% dyDotDot/dy
A(5,2) = -muEarth/R^3 * ( ...
          1 - 3/2*J2*(rEarth/R)^2 * (5*(z/R)^2 - 1)) + ...
          3*muEarth*y^2/R^5 * ( ...
          1 - 5/2*J2*(rEarth/R)^2 * (7*(z/R)^2 - 1)) + ...
          1/2*cD*cs/m*rhoA*VA * ...
          y*(vy - thetaDot*x)/R/Hscale - ...
          1/2*cD*cs/m*rhoA * ...
          (thetaDot*vx + thetaDot^2*y) * ...
          (vy - thetaDot*x)/VA;

% dyDotDot/dz
A(5,3) = 3*muEarth*y*z/R^5 * ( ...
          1 - 5/2*J2*(rEarth/R)^2 * (7*(z/R)^2 - 3)) + ...
          1/2*cD*cs/m*rhoA*VA * ...
          z*(vy - thetaDot*x)/R/Hscale;

% dyDotDot/dxDot
A(5,4) = -1/2*cD*cs/m*rhoA * ( ...
          vy - thetaDot*x) * (vx + thetaDot*y)/VA;

% dyDotDot/dyDot
A(5,5) = -1/2*cD*cs/m*rhoA * ( ...
          vy - thetaDot*x)^2/VA - ...
          1/2*cD*cs/m*rhoA*VA;

% dyDotDot/dzDot
A(5,6) = -1/2*cD*cs/m*rhoA * ...
          vz*(vy - thetaDot*x)/VA;

% dyDotDot/dMu
A(5,7) = -y/R^3 * ...
          (1 - 3/2*J2*(rEarth/R)^2 * (5*(z/R)^2 - 1));
```

```
% dyDotDot/dJ2
A(5,8) = 3/2*muEarth*y/R^3 * ( ...
    rEarth/R)^2 * (5*(z/R)^2 - 1);

% dyDotDot/dCd
A(5,9) = -1/2*cs/m*rhoA*VA*(vy-thetaDot*x);


% dzDotDot/dx
A(6,1) = 3*muEarth*x*z/R^5 * ( ...
    1 - 5/2*J2*(rEarth/R)^2 * (7*(z/R)^2 - 3)) + ...
    1/2*cD*cs/m*rhoA*VA*vz*x/R/Hscale - ...
    1/2*cD*cs/m*rhoA * ...
    vz*(thetaDot^2*x - thetaDot*vy)/VA;

% dzDotDot/dy
A(6,2) = 3*muEarth*y*z/R^5 * ( ...
    1 - 5/2*J2*(rEarth/R)^2 * (7*(z/R)^2 - 3)) + ...
    1/2*cD*cs/m*rhoA*VA*vz*y/R/Hscale - ...
    1/2*cD*cs/m*rhoA * ...
    vz*(thetaDot*vx + thetaDot^2*y)/VA;

% dzDotDot/dz
A(6,3) = -muEarth/R^3 * ( ...
    1 - 3/2*J2*(rEarth/R)^2 * (5*(z/R)^2 - 3)) + ...
    3*muEarth*z^2/R^5 * ( ...
    1 - 5/2*J2*(rEarth/R)^2 * (7*(z/R)^2 - 5)) + ...
    1/2*cD*cs/m*rhoA*VA*z*vz/R/Hscale;

% dzDotDot/dxDot
A(6,4) = -1/2*cD*cs/m*rhoA * ...
    vz*(vx + thetaDot*y)/VA;

% dzDotDot/dyDot
A(6,5) = -1/2*cD*cs/m*rhoA * ...
    vz*(vy - thetaDot*x)/VA;

% dzDotDot/dzDot
A(6,6) = -1/2*cD*cs/m*rhoA * ...
    vz^2/VA - 1/2*cD*cs/m*rhoA*VA;

% dzDotDot/dMu
A(6,7) = -z/R^3 * ( ...
    1 - 3/2*J2*(rEarth/R)^2 * (5*(z/R)^2 - 3));

% dzDotDot/dJ2
A(6,8) = 3*muEarth*z/2/R^3 * ( ...
    rEarth/R)^2 * (5*(z/R)^2 - 3);

% dzDotDot/dCd
A(6,9) = -1/2*cs/m*rhoA*VA*vz;
end

function H = Htilde_func(X0,t,station)
```

```
% This function calculates the mapping matrix H^tilde given the elements of
% the state vector, the time, and the station number tracking the satellite
% at time t.
global thetaDot

r = X0(1:3);
v = X0(4:6);
rS1 = X0(10:12);
rS2 = X0(13:15);
rS3 = X0(16:18);

x = r(1);
y = r(2);
z = r(3);
vx = v(1);
vy = v(2);
vz = v(3);

H = zeros(2,18);

% Depending on the current station tracking the satellite, select the
% corresponding coordinates
if station == 101
    rS = rS1;
elseif station == 337
    rS = rS2;
else
    rS = rS3;
end

xS = rS(1);
yS = rS(2);
zS = rS(3);

rho = rhoRange(r,rS,t);
rhoRate = rhoRangeRate(r,v,rS,t);

% d rho/d x
H(1,1) = (x - xS*cos(theta(t)) + yS*sin(theta(t)))/rho;

% d rho/d y
H(1,2) = (y - yS*cos(theta(t)) - xS*sin(theta(t)))/rho;

% d rho/d z
H(1,3) = (z - zS)/rho;

% d rho/d xdot = 0
% d rho/d ydot = 0
% d rho/d zdot = 0
% d rho/d muEarth = 0
% d rho/d J2 = 0
% d rho/d cD = 0

% Depending on the station, store the partial derivative of rho in its
% corresponding location in the mapping matrix for the given time
if station == 101
    % d rho/d xS1
    H(1,10) = (xS - x*cos(theta(t)) - y*sin(theta(t)))/rho;

    % d rho/d yS1
    H(1,11) = (yS - y*cos(theta(t)) + x*sin(theta(t)))/rho;

    % d rho/d zS1
    H(1,12) = (zS - z)/rho;
elseif station == 337
    % d rho/d xS2
    H(1,13) = (xS - x*cos(theta(t)) - y*sin(theta(t)))/rho;

    % d rho/d yS2
    H(1,14) = (yS - y*cos(theta(t)) + x*sin(theta(t)))/rho;

    % d rho/d zS2
    H(1,15) = (zS - z)/rho;
else
    % d rho/d xS3
    H(1,16) = (xS - x*cos(theta(t)) - y*sin(theta(t)))/rho;

    % d rho/d yS3
    H(1,17) = (yS - y*cos(theta(t)) + x*sin(theta(t)))/rho;

    % d rho/d zS3
    H(1,18) = (zS - z)/rho;
end

% d rhodot/d x
H(2,1) = (vx + thetaDot*xS*sin(theta(t)) + ...
    thetaDot*yS*cos(theta(t)))/rho - ...
    rhoRate*(x - xS*cos(theta(t)) + yS*sin(theta(t)))/rho^2;

% d rhodot/d y
H(2,2) = (vy + thetaDot*yS*sin(theta(t)) - ...
    thetaDot*xS*cos(theta(t)))/rho - ...
    rhoRate*(y - yS*cos(theta(t)) - xS*sin(theta(t)))/rho^2;

% d rhodot/d z
H(2,3) = vz/rho - rhoRate*(z - zS)/rho^2;

% d rhodot/d xdot
H(2,4) = (x - xS*cos(theta(t)) + yS*sin(theta(t)))/rho;

% d rhodot/d ydot
H(2,5) = (y - yS*cos(theta(t)) - xS*sin(theta(t)))/rho;

% d rhodot/d zdot
H(2,6) = (z - zS)/rho;

% d rhodot/d muEarth = 0
% d rhodot/d J2 = 0
% d rhodot/d cD = 0

% Depending on the station, store the partial derivative of rhodot in its
% corresponding location in the mapping matrix for the given time
if station == 101
    % d rhodot/d xS1
    H(2,10) = (-vx*cos(theta(t)) + thetaDot*x*sin(theta(t)) - ...
        vy*sin(theta(t)) - thetaDot*y*cos(theta(t)))/rho - ...
        rhoRate*(xS - x*cos(theta(t)) - y*sin(theta(t)))/rho^2;

    % d rhodot/d yS1
    H(2,11) = (-vy*cos(theta(t)) + thetaDot*y*sin(theta(t)) + ...
        vx*sin(theta(t)) + thetaDot*x*cos(theta(t)))/rho - ...
        rhoRate*(yS - y*cos(theta(t)) + x*sin(theta(t)))/rho^2;

    % d rhodot/d zS1
    H(2,12) = -vz/rho - rhoRate*(zS - z)/rho^2;
elseif station == 337
    % d rhodot/d xS2
    H(2,13) = (-vx*cos(theta(t)) + thetaDot*x*sin(theta(t)) - ...
        vy*sin(theta(t)) - thetaDot*y*cos(theta(t)))/rho - ...
        rhoRate*(xS - x*cos(theta(t)) - y*sin(theta(t)))/rho^2;

    % d rhodot/d yS2
    H(2,14) = (-vy*cos(theta(t)) + thetaDot*y*sin(theta(t)) + ...
        vx*sin(theta(t)) + thetaDot*x*cos(theta(t)))/rho - ...
        rhoRate*(yS - y*cos(theta(t)) + x*sin(theta(t)))/rho^2;

    % d rhodot/d zS2
    H(2,15) = -vz/rho - rhoRate*(zS - z)/rho^2;
else
    % d rhodot/d xS3
    H(2,16) = (-vx*cos(theta(t)) + thetaDot*x*sin(theta(t)) - ...
        vy*sin(theta(t)) - thetaDot*y*cos(theta(t)))/rho - ...
        rhoRate*(xS - x*cos(theta(t)) - y*sin(theta(t)))/rho^2;

    % d rhodot/d yS3
    H(2,17) = (-vy*cos(theta(t)) + thetaDot*y*sin(theta(t)) + ...
        vx*sin(theta(t)) + thetaDot*x*cos(theta(t)))/rho - ...
        rhoRate*(yS - y*cos(theta(t)) + x*sin(theta(t)))/rho^2;

    % d rhodot/d zS3
    H(2,18) = -vz/rho - rhoRate*(zS - z)/rho^2;
end

end
function xDeviation = cholesky(M,N)
% Cholesky Decomposition
% Need to calculate R such that R' * R = M
% Process produces more accurate solutions that xBestEst = M^-1 * N
R = zeros(size(M));
z = zeros(size(N));
for i = 1:length(M)
    for j = i:length(M)
```

```
          if i == j
              R(i,i) = sqrt(M(i,i) - sum(R(1:i-1,i).^2));
          else
              R(i,j) = (M(i,j) - sum(R(1:i-1,i).*R(1:i-1,j)))/R(i,i);
          end
     end
end

% Need to calculate z such that R' * z = N
for i = 1:length(N)
     z(i) = (N(i) - sum(R(1:i-1,i).*z(1:i-1)))/R(i,i);
end

xDeviation = zeros(size(N));
% Creating a zero vector equal in size to the x a priori values

n = length(xDeviation);
for i = n:-1:1
     xDeviation(i) = (z(i) - sum(R(i,i+1:n).*xDeviation(i+1:n)'))/R(i,i);
end
end
```