ASEN 5044
Homework 8

ASEN 5044 | HW 8

Problem 1

a) Specify the full DT LTI stochastic dynamics model for each Ak.

$$\Gamma_A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \qquad A_A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\Omega_A \\ 0 & 0 & 0 & 1 \\ 0 & \Omega_A & 0 & 0 \end{bmatrix} \qquad dt = 0.5 \text{ sec}, \quad \Omega_A = 0.045 \text{ rad/s}$$

$$Z_A = dt \cdot \begin{bmatrix} -A_A & \Gamma_A W \Gamma_A' \\ 0 & A_A' \end{bmatrix} \Rightarrow e^{Z_A} = \exp(Z_A)$$

$$e^{Z_A} = \begin{bmatrix} \cdots & F^{-1} Q \\ 0 & F^T \end{bmatrix} \Rightarrow Q_A = (F_A^T)^T \cdot [F_A^{-1} Q_A] \quad (\text{from } e^{Z_A} \text{ matrix})$$

$$Q_A = \begin{bmatrix} 0.83 & 2.50 & 0.03 & 0.10 \\ 2.50 & 10.00 & 0.07 & 0.33 \\ 0.03 & 0.07 & 0.21 & 0.63 \\ 0.10 & 0.33 & 0.63 & 2.51 \end{bmatrix}$$

$$W = q_w \cdot \begin{bmatrix} 2 & 0.05 \\ 0.05 & 0.5 \end{bmatrix}, \quad q_w = 10 \text{ (m/s)}^2$$

$$F_A = \begin{bmatrix} 1 & \sin(\Omega_A dt)/\Omega_A & 0 & -(1-\cos(\Omega_A dt))/\Omega_A \\ 0 & \cos(\Omega_A dt) & 0 & -\sin(\Omega_A dt) \\ 0 & (1-\cos(\Omega_A dt))/\Omega_A & 1 & \sin(\Omega_A dt)/\Omega_A \\ 0 & \sin(\Omega_A dt) & 0 & \cos(\Omega_A dt) \end{bmatrix}$$

$$F_A = \begin{bmatrix} 1 & 0.50 & 0 & -0.01 \\ 0 & 1.00 & 0 & -0.02 \\ 0 & 0.01 & 1 & 0.50 \\ 0 & 0.02 & 0 & 1.00 \end{bmatrix}$$

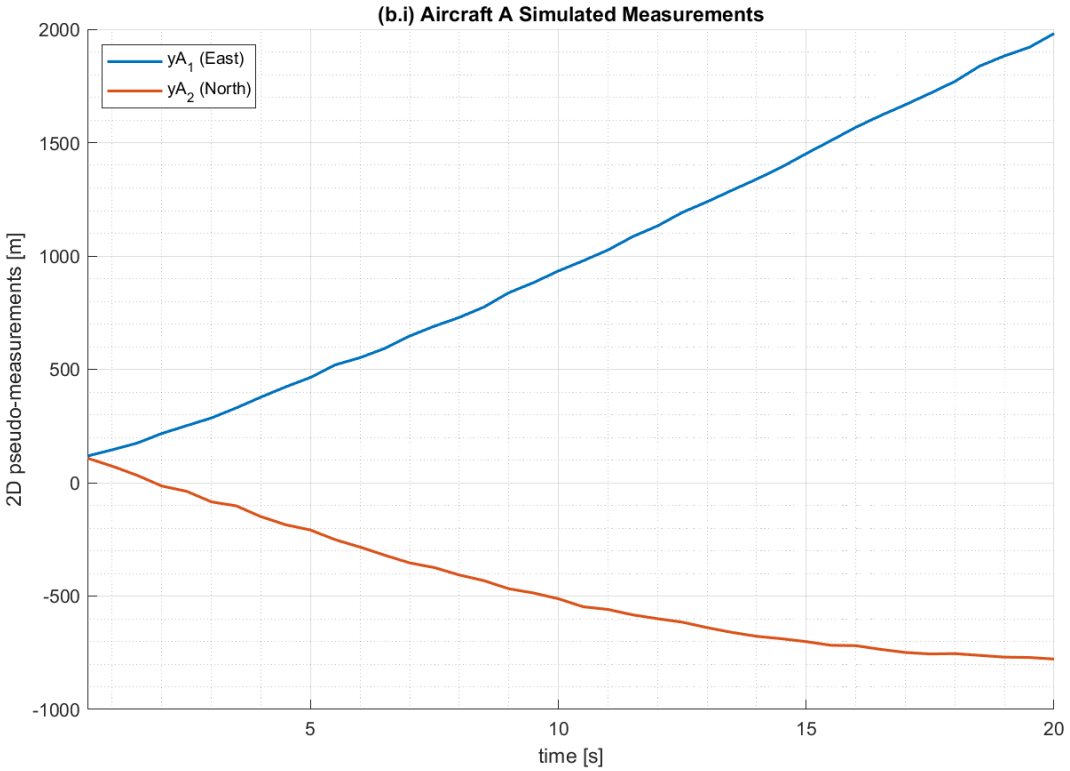
$$\Gamma_B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \qquad A_B = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\Omega_B \\ 0 & 0 & 0 & 1 \\ 0 & \Omega_B & 0 & 0 \end{bmatrix} \qquad dt = 0.5 \text{ sec}, \quad \Omega_B = -0.045 \text{ rad/s}$$

$$Z_A = dt \cdot \begin{bmatrix} -A_B & \Gamma_B W \Gamma_B' \\ 0 & A_B' \end{bmatrix} \Rightarrow e^{Z_B} = \exp(Z_B)$$

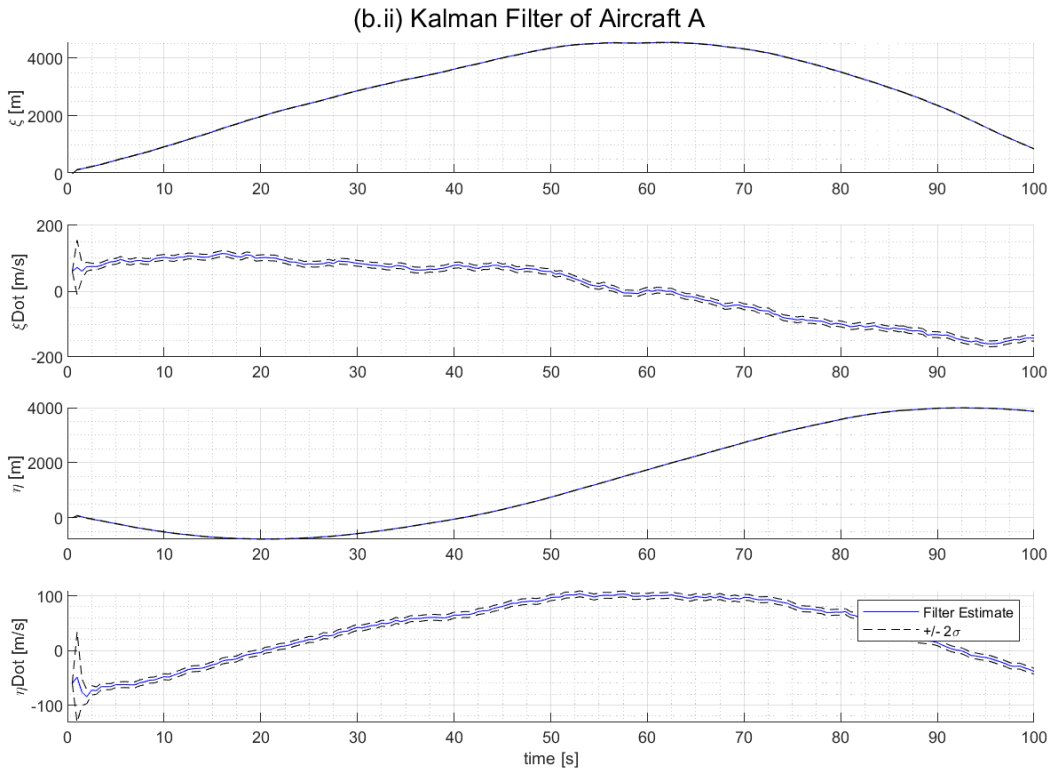
$$Q_B = (F_B^T)^T [F_B^{-1} Q_B] = \begin{bmatrix} 0.83 & 2.50 & 0.02 & 0.03 \\ 2.50 & 10.00 & 0.05 & 0.17 \\ 0.02 & 0.05 & 0.21 & 0.62 \\ 0.03 & 0.17 & 0.62 & 2.50 \end{bmatrix} = Q_B$$

$$F_B = \begin{bmatrix} 1 & \sin(\Omega_B dt)/\Omega_B & 0 & -(1-\cos(\Omega_B dt))/\Omega_B \\ 0 & \cos(\Omega_B dt) & 0 & -\sin(\Omega_B dt) \\ 0 & (1-\cos(\Omega_B dt))/\Omega_B & 1 & \sin(\Omega_B dt)/\Omega_B \\ 0 & \sin(\Omega_B dt) & 0 & \cos(\Omega_B dt) \end{bmatrix} = \begin{bmatrix} 1 & 0.50 & 0 & 0.01 \\ 0 & 1.00 & 0 & 0.02 \\ 0 & -0.01 & 1 & 0.50 \\ 0 & -0.02 & 0 & 1.00 \end{bmatrix} = F_B$$

Problem 1.b.i


(b.i) Aircraft A Simulated Measurements
yA1 (East)
yA2 (North)
2D pseudo-measurements [m]
time [s]
2000
1500
1000
500
0
-500
-1000
5
10
15
20

Problem 1.b.ii



The Kalman filter results shown above have some interesting characteristics. The plots are shown in the following order - East position, East velocity, North position, North Velocity. Very quickly, it is clear that the velocity states show less certainty than the position states in comparison to the change over the course of 100 seconds. This is observation is made both throughout the 100 second period, but more significantly during the first few seconds of filtering. The velocity states in the first few seconds see the 2σ bounds balloon and then converge back towards the estimate. This indicates a minimum amount of data required for the filter to converge towards the long term behavior. The filter appears to behave properly, neither converging inappropriately, or diverging too much.

Problem 1.c.i

ASEN 5044	HW 8	

Problem 1

c) i) Implement a KF to estimate the joint augmented aircraft states.

First, combine the Aircraft A and Aircraft B DT LTI matrices.

F_S

$$F_S = \left[\begin{array}{c|c} F_A & 0 \\ \hline 0 & F_B \end{array}\right] \qquad A_S = \left[\begin{array}{c|c} A_A & 0 \\ \hline 0 & A_B \end{array}\right] \qquad W_S = \left[\begin{array}{c|c} W & 0 \\ \hline 0 & W \end{array}\right] \qquad \Gamma_S = \left[\begin{array}{c|c} \Gamma_A & 0 \\ \hline 0 & \Gamma_B \end{array}\right]$$

Now, form a new Q_S matrix:

$$Z_S = dt \left[\begin{array}{c|c} -A_S & \Gamma_S W_S \Gamma_S' \\ \hline 0 & A_S' \end{array}\right]$$

$$e^{Z_S} = \exp(Z_S) = \left[\begin{array}{c|c} (\cdots) & F_S^{-1} Q_S \\ \hline 0 & F' \end{array}\right] \Rightarrow Q_S = (F_S')' \left[F_S^{-1} Q_S\right] \text{ (from } e^{Z_S} \text{ matrix)}$$

Q_S

$$Q_S = \begin{bmatrix} 0.83 & 2.50 & 0.03 & 0.10 & 0 & 0 & 0 & 0 \\ 2.50 & 10.00 & 0.07 & 0.33 & 0 & 0 & 0 & 0 \\ 0.03 & 0.07 & 0.21 & 0.63 & 0 & 0 & 0 & 0 \\ 0.10 & 0.33 & 0.63 & 2.51 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.83 & 2.50 & 0.03 & 0.03 \\ 0 & 0 & 0 & 0 & 2.50 & 10.00 & 0.03 & 0.17 \\ 0 & 0 & 0 & 0 & 0.03 & 0.03 & 0.21 & 0.63 \\ 0 & 0 & 0 & 0 & 0.03 & 0.17 & 0.63 & 2.50 \end{bmatrix}$$

H_S

$$H_S = \left[\begin{array}{c|c} H & 0 \\ \hline H & -H \end{array}\right] \Rightarrow Y_S(k) = H X_S(k) + V_D(k)$$

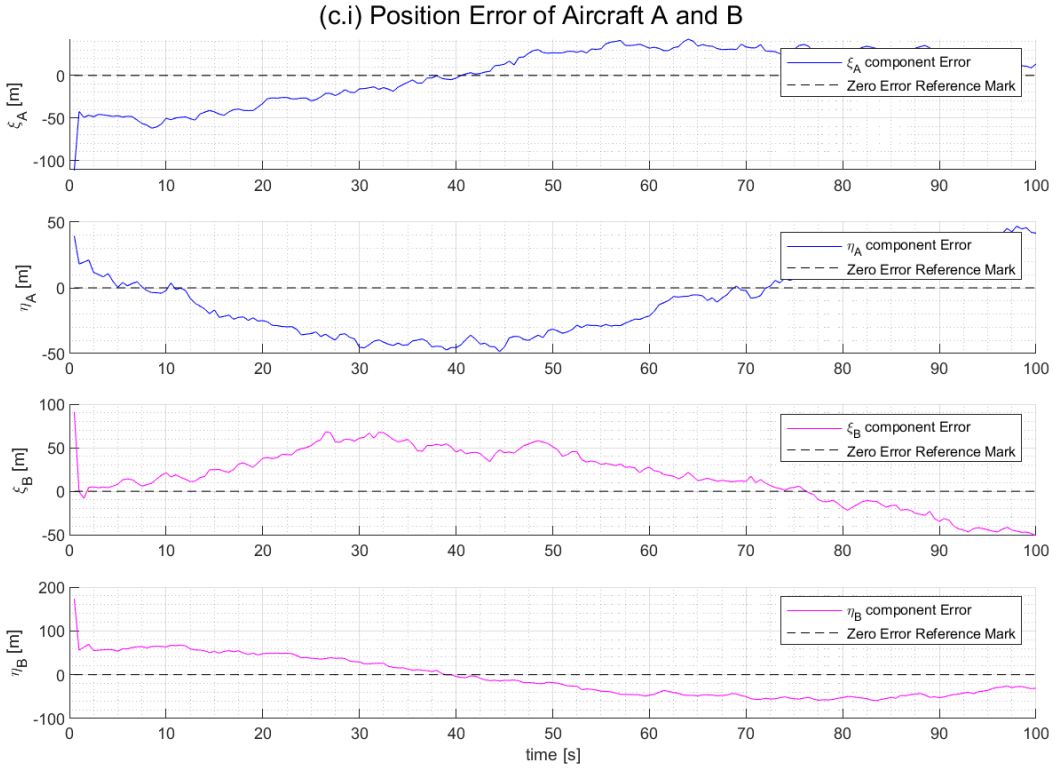
Top "row" handles original Aircraft A "direct" measurements, bottom "row" is the result of the Aircraft B measurements being a sum of Aircraft A measurement plus transponder measurements.

$$y_D(k) = r_A(k) - r_B(k)$$
$$\hookrightarrow H \qquad \hookrightarrow -H$$

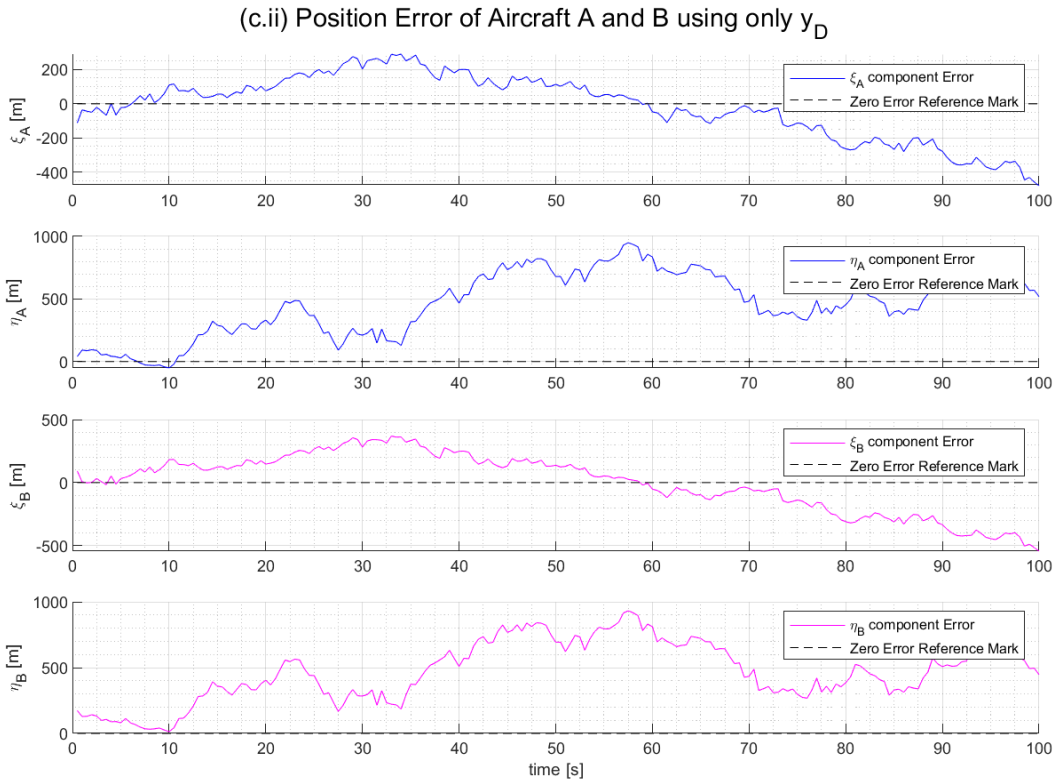
R_S

$$R_S = \left[\begin{array}{c|c} R_A & 0 \\ \hline 0 & R_D \end{array}\right]$$

$$R_S = \begin{bmatrix} 20 & 0.05 & 0 & 0 \\ 0.05 & 20 & 0 & 0 \\ 0 & 0 & 10 & 0.15 \\ 0 & 0 & 0.15 & 10 \end{bmatrix} \ (m^2)$$


(c.i) Position Error of Aircraft A and B
ξ_A [m]
ξ_A component Error
Zero Error Reference Mark
η_A [m]
η_A component Error
Zero Error Reference Mark
ξ_B [m]
ξ_B component Error
Zero Error Reference Mark
η_B [m]
η_B component Error
Zero Error Reference Mark
time [s]

Problem 1.c.ii



Problem 1.c.iii

We believe the primarily interesting behavior of the covariance matrices is that that they converge to a value. with the influence of measurements. Under only pure prediction updates, the covariance matrices only grow in magnitude, indicating that the underlying physics model alone is not enough to confidently predict the system state. It is here that we can see the true utility of utilizing measurements as a way to inform our dynamic models, as they provide inferences into how the dynamics of the system may be misunderstood.

Problem 2.a - Find the required CT Jacobians

ASEN 5044 | HW 8

Problem 2

a) Find the required CT Jacobians.

$$f_1 = x_2 \quad f_2 = -\frac{\mu x_1}{(x_1^2+x_3^2)^{3/2}} + u_1 \quad f_3 = x_4 \quad f_4 = -\frac{\mu x_3}{(x_1^2+x_3^2)^{3/2}} + u_2$$

$$\bar{x} = [x, \dot{x}, y, \dot{y}]^T = [x_1, x_2, x_3, x_4]^T$$

$$\tilde{A} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & & & \\ \frac{\partial f_3}{\partial x_1} & & \ddots & \\ \frac{\partial f_4}{\partial x_1} & \cdots & & \frac{\partial f_4}{\partial x_4} \end{bmatrix}$$

$$\frac{\partial f_1}{\partial x_2} = 1 \qquad \frac{\partial f_1}{\partial x_1} = \frac{\partial f_1}{\partial x_3} = \frac{\partial f_1}{\partial x_4} = 0$$

$$\frac{\partial f_3}{\partial x_4} = 1 \qquad \frac{\partial f_3}{\partial x_1} = \frac{\partial f_1}{\partial x_2} = \frac{\partial f_1}{\partial x_3} = 0$$

$$\frac{\partial f_2}{\partial x_2} = \frac{\partial f_2}{\partial x_4} = 0 \qquad \frac{\partial f_4}{\partial x_2} = \frac{\partial f_4}{\partial x_4} = 0$$

$$\frac{\partial f_2}{\partial x_1} = \frac{\partial}{\partial x_1}\left(-\mu x_1 (x_1^2+x_3^2)^{-3/2} + u_1\right) = -\mu (x_1^2+x_3^2)^{-3/2} + 3\mu x_1^2 (x_1^2+x_3^2)^{-5/2}$$

$$\frac{\partial f_2}{\partial x_3} = \frac{\partial}{\partial x_3}\left(-\mu x_1 (x_1^2+x_3^2)^{-3/2} + u_1\right) = 3\mu x_1 x_3 (x_1^2+x_3^2)^{-5/2}$$

$$\frac{\partial f_4}{\partial x_1} = \frac{\partial}{\partial x_1}\left(-\mu x_3 (x_1^2+x_3^2)^{-3/2} + u_2\right) = 3\mu x_1 x_3 (x_1^2+x_3^2)^{-5/2}$$

$$\frac{\partial f_4}{\partial x_3} = \frac{\partial}{\partial x_3}\left(-\mu x_3 (x_1^2+x_3^2)^{-3/2} + u_2\right) = -\mu (x_1^2+x_3^2)^{-3/2} + 3\mu x_3^2 (x_1^2+x_3^2)^{-5/2}$$

$$\tilde{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-\mu}{r_0^3} + \frac{3\mu x_1^2}{r_0^5} & 0 & \frac{3\mu x_1 x_3}{r_0^5} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{3\mu x_1 x_3}{r_0^5} & 0 & \frac{-\mu}{r_0^3} + \frac{3\mu x_3^2}{r_0^5} & 0 \end{bmatrix}$$

$$\tilde{B} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} \\ \frac{\partial f_3}{\partial u_1} & \frac{\partial f_3}{\partial u_2} \\ \frac{\partial f_4}{\partial u_1} & \frac{\partial f_4}{\partial u_2} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\frac{\partial f_1}{\partial u_1} = \frac{\partial f_1}{\partial u_2} = 0 \qquad \frac{\partial f_2}{\partial u_1} = 1 \qquad \frac{\partial f_2}{\partial u_2} = 0$$

$$\frac{\partial f_3}{\partial u_1} = \frac{\partial f_3}{\partial u_2} = 0 \qquad \frac{\partial f_4}{\partial u_1} = 0 \qquad \frac{\partial f_4}{\partial u_2} = 1$$

ASEN 5044 | HW 8

Problem 2

a) Find the required CT Jacobians:

$$g_1 = \left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{1/2}$$

$$g_2 = \left((x_1 - x_s^i)(x_2 - \dot{x}_s^i) + (x_3 - y_s^i)(x_4 - \dot{y}_s^i)\right)\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{-1/2}$$

$$g_3 = \tan^{-1}\left(\frac{(x_3 - y_s^i)}{(x_1 - x_s^i)}\right)$$

$$\tilde{C} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} & \frac{\partial g_1}{\partial x_4} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \frac{\partial g_3}{\partial x_3} & \frac{\partial g_4}{\partial x_4} \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} & \frac{\partial g_4}{\partial x_4} \end{bmatrix} \qquad \frac{\partial g_1}{\partial x_2} = \frac{\partial g_1}{\partial x_4} = 0 \qquad \frac{\partial g_3}{\partial x_2} = \frac{\partial g_3}{\partial x_4} = 0$$

$$\frac{\partial g_1}{\partial x_1} = \frac{\partial}{\partial x_1}\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{1/2} = \frac{1}{2}\left(2(x_1 - x_s^i)\right)\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{-1/2}$$

$$\frac{\partial g_1}{\partial x_1} = \frac{x_1 - x_s^i}{\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{1/2}}$$

$$\frac{\partial g_1}{\partial x_3} = \frac{\partial}{\partial x_3}\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{1/2} = \frac{1}{2}\left(2(x_3 - y_s^i)\right)\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{-1/2}$$

$$\frac{\partial g_1}{\partial x_3} = \frac{x_3 - y_s^i}{\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{1/2}}$$

$$\frac{\partial g_2}{\partial x_1} = \frac{\partial}{\partial x_1}\left(\frac{(x_1 - x_s^i)(x_2 - \dot{x}_s^i) + (x_3 - y_s^i)(x_4 - \dot{y}_s^i)}{\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{1/2}}\right) \qquad \text{Product rule: } \frac{\partial}{\partial x}(uv) = v\frac{\partial u}{\partial x} + u\frac{\partial v}{\partial x}$$

$$= \left((x_1 - x_s^i)(x_2 - \dot{x}_s^i) + (x_3 - y_s^i)(x_4 - \dot{y}_s^i)\right)\left(-\frac{1}{2}\cdot 2(x_1 - x_s^i)\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{-3/2}\right)$$

$$+ \left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{-1/2}(x_2 - \dot{x}_s^i)$$

$$\frac{\partial g_2}{\partial x_1} = \frac{(x_2 - \dot{x}_s^i)}{\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{1/2}} - \frac{(x_1 - x_s^i)\left((x_1 - x_s^i)(x_2 - \dot{x}_s^i) + (x_3 - y_s^i)(x_4 - \dot{y}_s^i)\right)}{\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{3/2}}$$

$$\frac{\partial g_2}{\partial x_3} = \frac{\partial}{\partial x_3}\left(\frac{(x_1 - x_s^i)(x_2 - \dot{x}_s^i) + (x_3 - y_s^i)(x_4 - \dot{y}_s^i)}{\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{1/2}}\right) \qquad \text{Product Rule: } \frac{\partial}{\partial x}(uv) = v\frac{\partial u}{\partial x} + u\frac{\partial v}{\partial x}$$

$$= \left((x_1 - x_s^i)(x_2 - \dot{x}_s^i) + (x_3 - y_s^i)(x_4 - \dot{y}_s^i)\right)\left(-\frac{1}{2}\cdot 2(x_3 - y_s^i)\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{-3/2}\right)$$

$$+ \left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{-1/2}(x_4 - \dot{y}_s^i)$$

$$\frac{\partial g_2}{\partial x_3} = \frac{(x_4 - \dot{y}_s^i)}{\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{1/2}} - \frac{(x_3 - y_s^i)\left((x_1 - x_s^i)(x_2 - \dot{x}_s^i) + (x_3 - y_s^i)(x_4 - \dot{y}_s^i)\right)}{\left((x_1 - x_s^i)^2 + (x_3 - y_s^i)^2\right)^{3/2}}$$

Next page

ASEN 5044 | HW 8

Problem 2

a) Find the required 9 Jacobians:

$$\frac{\partial g_2}{\partial x_2} = \frac{\partial}{\partial x_2}\left(\frac{(x_1-x_s(i))(x_2-\dot{x}_s(i))+(x_3-y_s(i))(x_4-\dot{y}_s(i))}{((x_1-x_s(i))^2+(x_3-y_s(i))^2)^{1/2}}\right)$$

Product Rule: $\frac{\partial}{\partial x}(uv) = v\frac{\partial u}{\partial x} + u\frac{\partial v}{\partial x}$

$$= \left((x_1-x_s(i))(x_2-\dot{x}_s(i))+(x_3-y_s(i))(x_4-\dot{y}_s(i))\right)(0)$$
$$+ \left((x_1-x_s(i))^2+(x_3-y_s(i))^2\right)^{1/2}(x_1-x_s(i))$$

$$\frac{\partial g_2}{\partial x_2} = \frac{(x_1-x_s(i))}{((x_1-x_s(i))^2+(x_3-y_s(i))^2)^{1/2}}$$

$$\frac{\partial g_2}{\partial x_4} = \frac{\partial}{\partial x_4}\left(\frac{(x_1-x_s(i))(x_2-\dot{x}_s(i))+(x_3-y_s(i))(x_4-\dot{y}_s(i))}{((x_1-x_s(i))^2+(x_3-y_s(i))^2)}\right)$$ Product Rule

$$= \left((x_1-x_s(i))(x_2-\dot{x}_s(i))+(x_3-y_s(i))(x_4-\dot{y}_s(i))\right)(0)$$
$$+ \left((x_3-x_s(i))^2+(x_3-y_s(i))^2\right)^{1/2}(x_3-y_s(i))$$

$$\frac{\partial g_2}{\partial x_4} = \frac{(x_3-y_s(i))}{((x_1-x_s(i))^2+(x_3-y_s(i))^2)^{1/2}}$$

$$\frac{\partial g_3}{\partial x_1} = \frac{\partial}{\partial x_1}\left(\tan^{-1}\left(\frac{(x_3-y_s(i))}{(x_1-x_s(i))}\right)\right)$$

Chain Rule: $\frac{\partial}{\partial x}(\tan^{-1}(u)) = \frac{\partial}{\partial u}(\tan^{-1}(u))\frac{\partial u}{\partial x}$

$$= \frac{1}{1+u^2}\cdot\frac{\partial u}{\partial x_1}$$

$$= \frac{1}{1+\frac{(x_3-y_s(i))^2}{(x_1-x_s(i))^2}}\cdot\frac{-(x_3-y_s(i))}{(x_1-x_s(i))^2}$$

$$u = \frac{x_3-y_s(i)}{x_1-x_s(i)}$$

$$\frac{\partial u}{\partial x_1} = (x_3-y_s(i))\frac{\partial}{\partial x_1}\left((x_1-x_s(i))^{-1}\right)$$
$$= -\frac{(x_3-y_s(i))}{(x_1-x_s(i))^2}$$

$$\frac{\partial g_3}{\partial x_1} = \frac{-(x_3-y_s(i))}{(x_1-x_s(i))^2+(x_3-y_s(i))^2}$$

$$\frac{\partial g_3}{\partial x_3} = \frac{\partial}{\partial x_3}\left(\tan^{-1}\left(\frac{(x_3-y_s(i))}{(x_1-x_s(i))}\right)\right)$$

Chain Rule: $\frac{\partial}{\partial x}(\tan^{-1}(u)) = \frac{\partial}{\partial u}(\tan^{-1}(u))\cdot\frac{\partial u}{\partial x}$

$$= \frac{1}{1+u^2}\frac{\partial v}{\partial x_3}$$

$$= \frac{1}{1+\frac{(x_3-y_s(i))^2}{(x_1-x_s(i))^2}}\cdot(x_1-x_s(i))^{-1}$$

$$= \frac{1}{(x_1-x_s(i))+\frac{(x_3-y_s(i))^2}{(x_1-x_s(i))}}$$

$$u = \frac{x_3-y_s(i)}{x_1-x_s(i)}$$

$$\frac{\partial v}{\partial x_3} = (x_1-x_s(i))^{-1}\frac{\partial}{\partial x_3}(x_3-x_s(i))$$
$$= (x_1-x_s(i))^{-1}$$

$$\frac{\partial g_3}{\partial x_3} = \frac{(x_1-x_s(i))}{(x_1-x_s(i))^2+(x_3-y_s(i))^2}$$

Problem 2.b - Linearize the system about the nominal trajectory

To evaluate the F matrix at every step in time and with constantly changing position, we wrote a function to calculate the F matrix based on the nominal trajectory conditions at every step in time. That function is presented below.

```
function [ F ] = F_variant(X,Y)
mu = 398600;         % km^3/s^2
r0_nom = 6678;           % km
dt = 10;

F = expm(dt*[0, 1, 0, 0;
        (-mu*(r0_nom)^(-3))+(3*mu*X^2*r0_nom^(-5)), 0, 3*mu*X*Y*r0_nom^(-5), 0;
        0, 0, 0, 1;
        (3*mu*X*Y)*r0_nom^(-5), 0, (-mu*r0_nom^(-3))+(3*mu*Y^2*r0_nom^(-5)), 0]);
end
```

To evaluate the H matrix at every step in time and with constantly changing position and velocity, we wrote a function to calculate H based on eight separate inputs. The code for this is shown below.

```
function [ H ] = H_variant(X,Xdot,Y,Ydot,Xs,Xsdot,Ys,Ysdot)
%initialize H
H = zeros(3,4);

%first row
H(1,1) = (X-Xs)/sqrt((X-Xs)^2+(Y-Ys)^2);
H(1,2) = 0;
H(1,3) = (Y-Ys)/sqrt((X-Xs)^2+(Y-Ys)^2);
H(1,4) = 0;

%second row
H(2,1)=(Xdot-Xsdot)/sqrt((X-Xs)^2+(Y-Ys)^2)-(X-Xs)*((X-Xs)*(Xdot-Xsdot)+(Y-Ys)*(Ydot-Ysdot))...
/ ((X-Xs)^2+(Y-Ys)^2)^(3/2);
H(2,2) = (X-Xs)/sqrt((X-Xs)^2+(Y-Ys)^2);
H(2,3)=(Ydot-Ysdot)/sqrt((X-Xs)^2+(Y-Ys)^2)-(Y-Ys)*((X-Xs)*(Xdot-Xsdot)+(Y-Ys)*(Ydot-Ysdot))...
/ ((X-Xs)^2+(Y-Ys)^2)^(3/2);
H(2,4) = (Y-Ys)/sqrt((X-Xs)^2+(Y-Ys)^2);

%third row
H(3,1) = ((Y-Ys)/((X-Xs)^2+(Y-Ys)^2));
H(3,2) = 0;
H(3,3) = ((-X+Xs)/((X-Xs)^2+(Y-Ys)^2));
H(3,4) = 0;
end
```

It is noted that the result is time-varying.

Problem 2.c - Simulate the Linearized DT Dynamics and Measurements

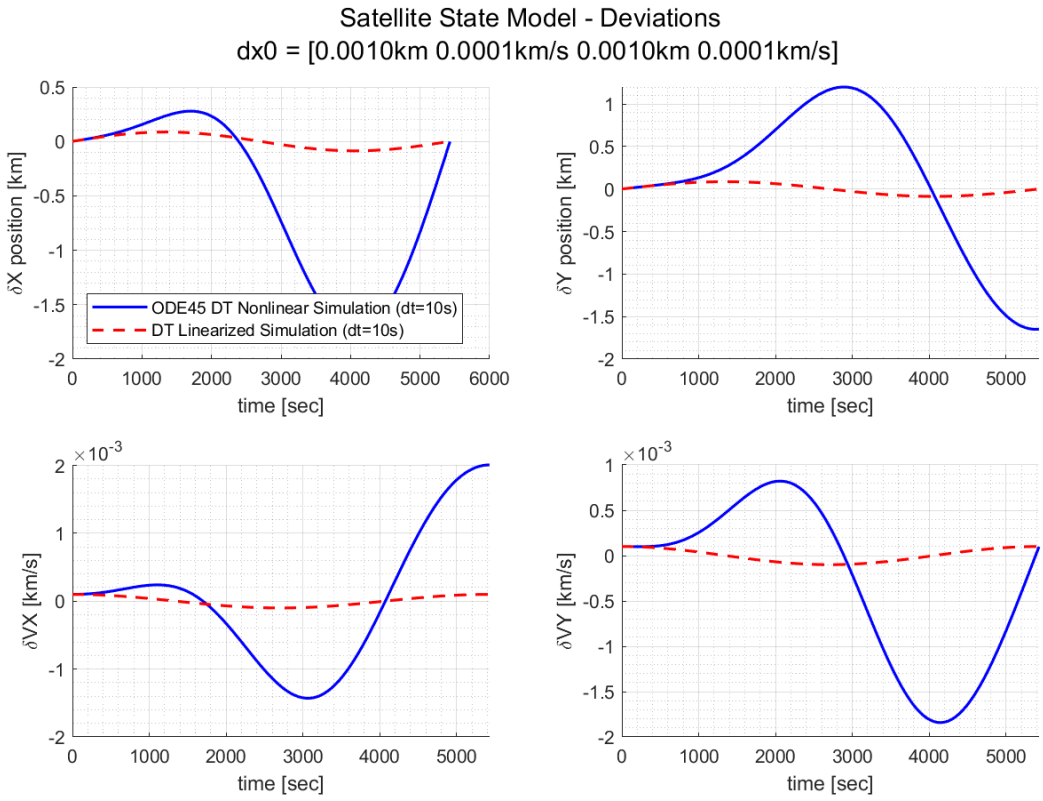


Figure 1: State Model Deviations

Figure 1 shows the linearized DT dynamics around the nominal trajectory with $r_0 = 6678km$ and an initial state perturbation of $[1m, 0.1m/s, 1m, 0.1m/s]$. The linearized deviation from the nominal trajectory is plotted in red and the nonlinear deviation from the nominal trajectory is plotted in blue. While the linearized simulation captures the overall behavior, it does not quite capture magnitude of the deviation.

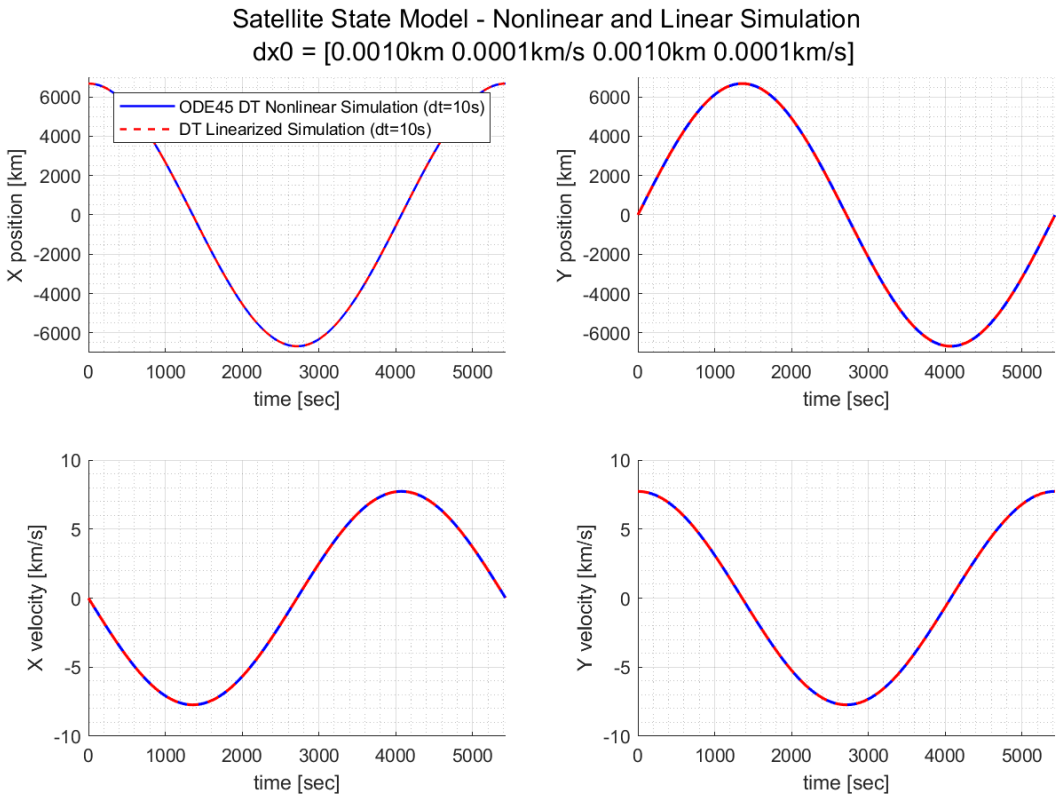


Figure 2: State Model Simulation

Figure 2 shows the linearized DT dynamics added to the nominal trajectory (red) and the full nonlinear dynamics (blue). The linearized DT dynamics deviate slightly from the full nonlinear dynamics as seen in the pure deviation plots above. In the context of the orbit dimensions, the deviations are relatively small for the given perturbations. For small perturbations, the linearized model appears adequate, but as the perturbation increases in magnitude, the difference between the nonlinear and linearized simulations grows.

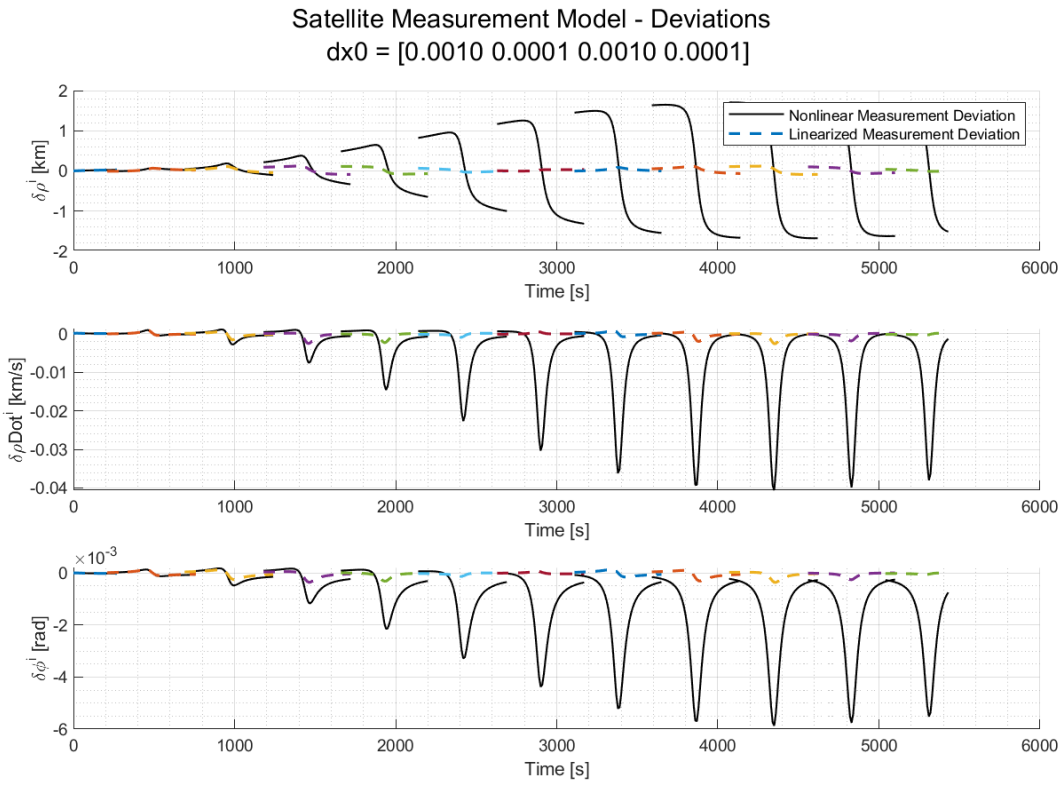


Figure 3: Measurement Model Deviations

Figure 3 shows the linearized DT measurement model around the nominal trajectory with $r_0 = 6678km$ and an initial state perturbation of $[1m, 0.1m/s, 1m, 0.1m/s]$. The linearized measurement deviation from the nominal measurement is plotted with colors corresponding to the station and the nonlinear measurement deviation from the nominal trajectory is plotted in black. Similar to the measurement model deviations, the linearized model does not capture the full magnitude of the deviations, but does capture the general trend.

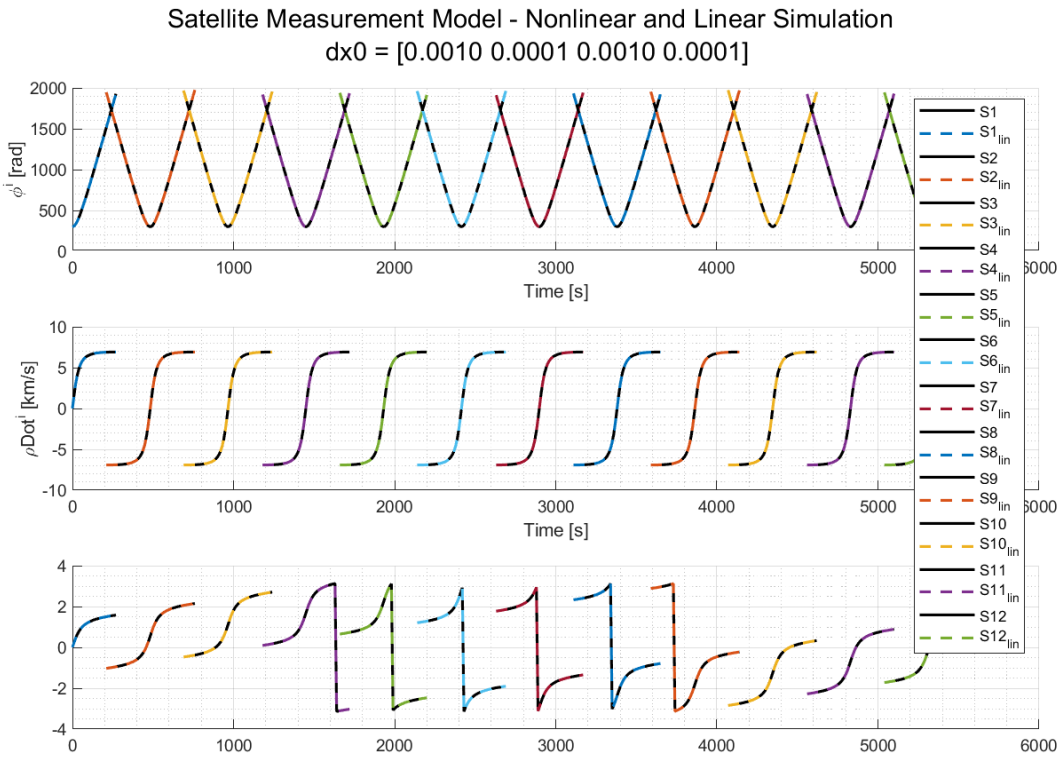


Figure 4: Measurement Model Simulation

Figure 4 shows the linearized DT measurement model added to the nominal trajectory in colors corresponding to the station and the nonlinear measurement model in black. In the context of the measurement value magnitudes, the deviations are small for a small perturbation. As such, our linearized model is likely useful for filtering and orbit estimation.

Code

```
%=========================================================================
% Jake Vendl | Jack Toland
% ASEN 5044
% Homework 8
% 12/3/2019
%=========================================================================
close all; clear all; clc

%% Problem setup

qw = 10;            % (m/s)^2
dt = 0.5;           % sec
OmegaA = 0.045;     % rad/s
```

```
OmegaB = -0.045;    % rad/s

FA = [1, sin(OmegaA*dt)/OmegaA, 0, -(1-cos(OmegaA*dt))/OmegaA;
    0, cos(OmegaA*dt), 0, -sin(OmegaA*dt);
    0, (1-cos(OmegaA*dt))/OmegaA, 1, sin(OmegaA*dt)/OmegaA;
    0, sin(OmegaA*dt), 0, cos(OmegaA*dt)];

FB = [1, sin(OmegaB*dt)/OmegaB, 0, -(1-cos(OmegaB*dt))/OmegaB;
    0, cos(OmegaB*dt), 0, -sin(OmegaB*dt);
    0, (1-cos(OmegaB*dt))/OmegaB, 1, sin(OmegaB*dt)/OmegaB;
    0, sin(OmegaB*dt), 0, cos(OmegaB*dt)];

AA = [0 1 0 0;
    0 0 0 -OmegaA;
    0 0 0 1;
    0 OmegaA 0 0];

AB = [0 1 0 0;
    0 0 0 -OmegaB;
    0 0 0 1;
    0 OmegaB 0 0];

%% Problem 1
%====== a ======%
GammaA = [0 0; 1 0; 0 0; 0 1];
GammaB = [0 0; 1 0; 0 0; 0 1];

W = qw * [2 0.05; 0.05 0.5];

%find QA and QB using Van Loan's method
%first, find QA
ZA = dt.*[-AA GammaA*W*GammaA';
    zeros(4,4) AA'];
ezA = expm(ZA);
QA = ezA(5:8,5:8)' * ezA(1:4,5:8); %Q = (F')' * (inv(F)*Q)

ZB = dt.*[-AB GammaB*W*GammaB';
    zeros(4,4) AB'];
ezB = expm(ZB);
QB = ezB(5:8,5:8)' * ezB(1:4,5:8);

%====== b ======%
rng(100);

%b)i
H = [1 0 0 0; 0 0 1 0];
RA = [20 0.05; 0.05 20];
p=2;
T=200;

load('hw8problem1_data.mat');
```

```
Sv = chol(RA,'lower');
q = zeros(p,T); yA = zeros(p,T);
for k = 1:T
    qk = randn(2,1);
    vk = (Sv*qk);
    yA(:,k) = H*xasingle_truth(:,k) + vk;
end

fig = figure; hold on; grid on; grid minor;
set(fig, 'Position', [100 100 900 600]);
title('(b.i) Aircraft A Simulated Measurements')
xlabel('time [s]'); ylabel('2D pseudo-measurements [m]')
xticks([0 10 20 30 40])
xticklabels({'0','5','10','15','20'})
plot(1:40,yA(1,1:40),'-','LineWidth',1.5)
plot(1:40,yA(2,1:40),'-','LineWidth',1.5)
xlim([1 40]);
legend('yA_1 (East)','yA_2 (North)','Location','Northwest')
saveas(fig,'ASEN5044_HW8_P1_bi.png','png');

%b)i
muA = [0, 85*cos(pi/4), 0, -85*sin(pi/4)]';
PA = 900*diag([10,2,10,2]);
% IMPLEMENT KALMAN HERE %
%define initial conditions
x_plus(:,1) = muA;

%handle the first time step outside of the loop, for indexing purposes
x_minus(:,1) = FA*muA;
P_minus(:,:,1) = FA*PA*FA' + QA;
P_plus(:,:,1) = PA;

%now iterate through the k's
sigma = zeros(4,length(T)); K = zeros(4,2,length(T));
for k=1:(T-1)
    x_minus(:,k+1) = FA*x_plus(:,k);
    P_minus(:,:,k+1) = FA*P_plus(:,:,k)*FA' + QA;
    K(:,:,k+1) = P_minus(:,:,k+1) * H' * inv(H*P_minus(:,:,k+1)*H'+RA);

    x_plus(:,k+1) = x_minus(:,k+1) + K(:,:,k+1) * (yA(:,k+1)-H*x_minus(:,k+1));
    P_plus(:,:,k+1) = (eye(4)-K(:,:,k+1)*H)*P_minus(:,:,k+1);

    sigma(1,k+1) = sqrt(P_plus(1,1,k+1));
    sigma(2,k+1) = sqrt(P_plus(2,2,k+1));
    sigma(3,k+1) = sqrt(P_plus(3,3,k+1));
    sigma(4,k+1) = sqrt(P_plus(4,4,k+1));
end

fig = figure; hold on;
set(fig, 'Position', [100 100 900 600]);
```

```
sgtitle('(b.ii) Kalman Filter of Aircraft A')
for i=1:4
    subplot(4,1,i); hold on; grid on; grid minor;
    plot(1:200,x_plus(i,:),'b-')
    plot(1:200,x_plus(i,:)+2*sigma(i,:),'k--')
    plot(1:200,x_plus(i,:)-2*sigma(i,:),'k--')
    xticks([0 20 40 60 80 100 120 140 160 180 200])
    xticklabels({'0','10','20','30','40','50','60','70','80','90','100'})
end
xlabel('time [s]');
subplot(4,1,1); ylabel('\xi [m]')
subplot(4,1,2); ylabel('\xiDot [m/s]')
subplot(4,1,3); ylabel('\eta [m]')
subplot(4,1,4); ylabel('\etaDot [m/s]')
legend('Filter Estimate','+/- 2\sigma')
saveas(fig,'ASEN5044_HW8_P1_bii.png','png');


%====== c ======%
muB = [3200, 85*cos(pi/4), 3200, -85*sin(pi/4)]';
PB = 900*diag([11,4,11,4]);
RD = [10 0.15;
    0.15 10];

%c)i
FS = [FA zeros(4);
    zeros(4) FB];
AS = [AA zeros(4);
    zeros(4) AB];
WS = [W zeros(2);
    zeros(2) W];
GammaS = [GammaA zeros(4,2);
    zeros(4,2) GammaB];
ZS = dt*[-AS GammaS*WS*GammaS';
    zeros(8) AS'];
eZS = expm(ZS);
QS = eZS(9:16,9:16)' * eZS(1:8,9:16);

HS = [1 0 0 0 0 0 0 0;
    0 0 1 0 0 0 0 0;
    1 0 0 0 -1 0 0 0;
    0 0 1 0 0 0 -1 0];
RS = [RA zeros(2);
    zeros(2) RD];

%simulate measurements for yAprime and yD
SVA = chol(RA,'lower');
yAprime = zeros(p,T);
for k = 1:T
    qk = randn(2,1);
    vk = (SVA*qk);
```

```
    yAprime(:,k) = H*xadouble_truth(:,k) + vk;
end

SVD = chol(RD,'lower');
yD = zeros(p,T);
for k=1:T
    qk = randn(2,1);
    vk = (SVD*qk);
    yD(:,k) = [xadouble_truth(1,k); xadouble_truth(3,k)] - [xbdouble_truth(1,k); xbdouble_truth(3,k)]
end

yS = [yAprime; yD];

%now we are ready to get filtering
clear x_plus x_minus P_minus P_plus K

%define initial time step conditions prior to entering the loop
x_plus(:,1) = [muA; muB];
x_minus(:,1) = FS*x_plus(:,1);
P_minus(:,:,1) = FS*[PA zeros(4); zeros(4) PB]*FS' + QS;
P_plus(:,:,1) = [PA zeros(4); zeros(4) PB];

K = zeros(8,4,length(T));
for k=1:(T-1)
    x_minus(:,k+1) = FS*x_plus(:,k);
    P_minus(:,:,k+1) = FS*P_plus(:,:,k)*FS' + QS;
    K(:,:,k+1) = P_minus(:,:,k+1) * HS' * inv(HS*P_minus(:,:,k+1)*HS'+RS);

    x_plus(:,k+1) = x_minus(:,k+1) + K(:,:,k+1) * (yS(:,k+1)-HS*x_minus(:,k+1));
    P_plus(:,:,k+1) = (eye(8)-K(:,:,k+1)*HS)*P_minus(:,:,k+1);

    sigma(1,k+1) = sqrt(P_plus(1,1,k+1));
    sigma(2,k+1) = sqrt(P_plus(2,2,k+1));
    sigma(3,k+1) = sqrt(P_plus(3,3,k+1));
    sigma(4,k+1) = sqrt(P_plus(4,4,k+1));
    sigma(5,k+1) = sqrt(P_plus(5,5,k+1));
    sigma(6,k+1) = sqrt(P_plus(6,6,k+1));
    sigma(7,k+1) = sqrt(P_plus(7,7,k+1));
    sigma(8,k+1) = sqrt(P_plus(8,8,k+1));
end

fig = figure; hold on;
set(fig, 'Position', [100 100 900 600]);
sgtitle('(c.i) Position Error of Aircraft A and B')
for i=1:4
    subplot(4,1,i); hold on; grid on; grid minor;
    if i == 1
        plot(1:200, x_plus(2*i-1,:)-xadouble_truth(1,2:201),'b-');
    elseif i == 2
        plot(1:200, x_plus(2*i-1,:)-xadouble_truth(3,2:201),'b-');
    elseif i == 3
```

```
        plot(1:200, x_plus(2*i-1,:)-xbdouble_truth(1,2:201),'m-');
    else
        plot(1:200, x_plus(2*i-1,:)-xbdouble_truth(3,2:201),'m-');
    end
    plot([1 200],[0 0],'k--');
    xticks([0 20 40 60 80 100 120 140 160 180 200])
    xticklabels({'0','10','20','30','40','50','60','70','80','90','100'})
end
subplot(4,1,1); ylabel('\xi_A [m]');
legend('\xi_A component Error','Zero Error Reference Mark')
subplot(4,1,2); ylabel('\eta_A [m]');
legend('\eta_A component Error','Zero Error Reference Mark')
subplot(4,1,3); ylabel('\xi_B [m]');
legend('\xi_B component Error','Zero Error Reference Mark')
subplot(4,1,4); ylabel('\eta_B [m]'); xlabel('time [s]');
legend('\eta_B component Error','Zero Error Reference Mark')
saveas(fig,'ASEN5044_HW8_P1_ci.png','png');


%c)ii
HS = [1 0 0 0 -1 0 0 0;
      0 0 1 0 0 0 -1 0];

%now we are ready to get filtering
clear yS x_plus x_minus P_minus P_plus K

%define initial time step conditions prior to entering the loop
x_plus(:,1) = [muA; muB];
x_minus(:,1) = FS*x_plus(:,1);
P_minus(:,:,1) = FS*[PA zeros(4); zeros(4) PB]*FS' + QS;
P_plus(:,:,1) = [PA zeros(4); zeros(4) PB];

yS = yD;
RS = RD;
%now loop through to filter
for k=1:(T-1)
    x_minus(:,k+1) = FS*x_plus(:,k);
    P_minus(:,:,k+1) = FS*P_plus(:,:,k)*FS' + QS;
    K(:,:,k+1) = P_minus(:,:,k+1) * HS' * inv(HS*P_minus(:,:,k+1)*HS'+RS);

    x_plus(:,k+1) = x_minus(:,k+1) + K(:,:,k+1) * (yS(:,k+1)-HS*x_minus(:,k+1));
    P_plus(:,:,k+1) = (eye(8)-K(:,:,k+1)*HS)*P_minus(:,:,k+1);

    sigma(1,k+1) = sqrt(P_plus(1,1,k+1));
    sigma(2,k+1) = sqrt(P_plus(2,2,k+1));
    sigma(3,k+1) = sqrt(P_plus(3,3,k+1));
    sigma(4,k+1) = sqrt(P_plus(4,4,k+1));
    sigma(5,k+1) = sqrt(P_plus(5,5,k+1));
    sigma(6,k+1) = sqrt(P_plus(6,6,k+1));
    sigma(7,k+1) = sqrt(P_plus(7,7,k+1));
    sigma(8,k+1) = sqrt(P_plus(8,8,k+1));
```

```
end

fig = figure; hold on;
set(fig, 'Position', [100 100 900 600]);
sgtitle('(c.ii) Position Error of Aircraft A and B using only y_D')
for i=1:4
    subplot(4,1,i); hold on; grid on; grid minor;
    if i == 1
        plot(1:200, x_plus(2*i-1,:)-xadouble_truth(1,2:201),'b-');
    elseif i == 2
        plot(1:200, x_plus(2*i-1,:)-xadouble_truth(3,2:201),'b-');
    elseif i == 3
        plot(1:200, x_plus(2*i-1,:)-xbdouble_truth(1,2:201),'m-');
    else
        plot(1:200, x_plus(2*i-1,:)-xbdouble_truth(3,2:201),'m-');
    end
    plot([1 200],[0 0],'k--');
    xticks([0 20 40 60 80 100 120 140 160 180 200])
    xticklabels({'0','10','20','30','40','50','60','70','80','90','100'})
end
subplot(4,1,1); ylabel('\xi_A [m]');
legend('\xi_A component Error','Zero Error Reference Mark')
subplot(4,1,2); ylabel('\eta_A [m]');
legend('\eta_A component Error','Zero Error Reference Mark')
subplot(4,1,3); ylabel('\xi_B [m]');
legend('\xi_B component Error','Zero Error Reference Mark')
subplot(4,1,4); ylabel('\eta_B [m]'); xlabel('time [s]');
legend('\eta_B component Error','Zero Error Reference Mark')
saveas(fig,'ASEN5044_HW8_P1_cii.png','png');


%=========================================================================
% Jake Vendl | Jack Toland
% ASEN 5044
% Homework 8
% 12/3/2019
%=========================================================================
close all; clear all; clc

mu = 398600;        % km^3/s^2
r0 = 6678;          % km
rE = 6378;          % km
wE = 2*pi/86400;    % rad/s
dt = 10;


%% A - Find the Required CT Jacobians
% No MATLAB Required

%% B - Linearize the System about the Nominal Trajectory
% See functions at end of script:
```

```
% function [ F ] = F_variant(X,Y)
% function [ H ] = H_variant(X,Xdot,Y,Ydot,Xs,Xsdot,Ys,Ysdot)

% Nominal Trajectory
r0_nom = 6678;

% Initial State
x1_init = 6678;
x2_init = 0;
x3_init = 0;
x4_init = r0*sqrt(mu/r0^3);
x0 = [x1_init, x2_init, x3_init, x4_init]';
P = 2*pi*sqrt(r0^3/mu);

% Initial Perturbation
dx0 = 1e-2*[0.1, 0.01, 0.1, 0.01]';

% Discrete Time Vector
t_vec = 0:dt:P;


%% C - Simulate Linearized and Nonlinear DT Dynamics near Nominal

%%%%%%%%%%%%%%%%%%%%%%%%%
% NONLINEAR
%%%%%%%%%%%%%%%%%%%%%%%%%

% Nominal Trajectory Nonlinear Model Simulation
s0 = x0;
opts = odeset('RelTol',1e-12,'AbsTol',1e-12);
[T, S] = ode45(@(t,s)orbit_prop_func(t,s),t_vec,s0,opts);


% Perturbed Trajectory Nonlinear Model Simulation
s0 = x0 + dx0;
opts = odeset('RelTol',1e-12,'AbsTol',1e-12);
[T_per, S_per] = ode45(@(t,s)orbit_prop_func(t,s),t_vec,s0,opts);


% Nominal Trajectory Nonlinear Measurement Simulation
X=S(:,1); Y=S(:,3); XD=S(:,2); YD=S(:,4);
Xs = zeros(12,length(T));
Ys = zeros(12,length(T));
XDs = zeros(12,length(T));
YDs = zeros(12,length(T));
rho = zeros(12,length(T));
rhoDot = zeros(12,length(T));
phi = zeros(12,length(T));
%now simulate the measurements for all time
for i=1:12 %stations
    theta = (i-1)*pi/6;
```

```
    for t=1:length(T) %loop through one orbit period
        currentTime = T(t);

        %find station position and velocity
        Xs(i,t) = rE*cos(wE*currentTime + theta);
        Ys(i,t) = rE*sin(wE*currentTime + theta);
        XDs(i,t) = -rE*wE*sin(wE*currentTime + theta);
        YDs(i,t) = rE*wE*cos(wE*currentTime + theta);

        %peform check at given time to see if s/c is visible
        phi(i,t) = atan2((Y(t)-Ys(i,t)),(X(t)-Xs(i,t)));
        thetaCheck = atan2(Ys(i,t),Xs(i,t));
        if (thetaCheck-pi/2) > (thetaCheck+pi/2)
            upperBound = thetaCheck-pi/2;
            lowerBound = thetaCheck+pi/2;
        else
            upperBound = thetaCheck+pi/2;
            lowerBound = thetaCheck-pi/2;
        end
        if (lowerBound <= phi(i,t) && phi(i,t) <= upperBound) ...
            || (lowerBound-2*pi <= phi(i,t) && phi(i,t)<=upperBound-2*pi)...
            || (lowerBound+2*pi <= phi(i,t) && phi(i,t)<=upperBound+2*pi)

            rho(i,t) = sqrt((X(t)-Xs(i,t))^2 + (Y(t)-Ys(i,t))^2);
            rhoDot(i,t) = ((X(t)-Xs(i,t))*(XD(t)-XDs(i,t)) + (Y(t)-Ys(i,t))*(YD(t)-YDs(i,t)))...
                / rho(i,t);
        else
            rho(i,t) = nan;
            rhoDot(i,t) = nan;
            phi(i,t)=nan;
        end
    end
end


% Perturbed Trajectory Nonlinear Measurement Simulation
X=S_per(:,1); Y=S_per(:,3); XD=S_per(:,2); YD=S_per(:,4);
Xs = zeros(12,length(T));
Ys = zeros(12,length(T));
XDs = zeros(12,length(T));
YDs = zeros(12,length(T));
rho_per = zeros(12,length(T));
rhoDot_per = zeros(12,length(T));
phi_per = zeros(12,length(T));
%now simulate the measurements for all time
for i=1:12 %stations
    theta = (i-1)*pi/6;
    for t=1:length(T) %loop through one orbit period
        currentTime = T(t);

        %find station position and velocity
```

```
        Xs(i,t) = rE*cos(wE*currentTime + theta);
        Ys(i,t) = rE*sin(wE*currentTime + theta);
        XDs(i,t) = -rE*wE*sin(wE*currentTime + theta);
        YDs(i,t) = rE*wE*cos(wE*currentTime + theta);

        %peform check at given time to see if s/c is visible
        phi_per(i,t) = atan2((Y(t)-Ys(i,t)),(X(t)-Xs(i,t)));
        thetaCheck = atan2(Ys(i,t),Xs(i,t));
        if (thetaCheck-pi/2) > (thetaCheck+pi/2)
            upperBound = thetaCheck-pi/2;
            lowerBound = thetaCheck+pi/2;
        else
            upperBound = thetaCheck+pi/2;
            lowerBound = thetaCheck-pi/2;
        end
        if (lowerBound <= phi_per(i,t) && phi_per(i,t) <= upperBound) ...
            || (lowerBound-2*pi <= phi_per(i,t) && phi_per(i,t)<=upperBound-2*pi)...
            || (lowerBound+2*pi <= phi_per(i,t) && phi_per(i,t)<=upperBound+2*pi)

            rho_per(i,t) = sqrt((X(t)-Xs(i,t))^2 + (Y(t)-Ys(i,t))^2);
            rhoDot_per(i,t) = ((X(t)-Xs(i,t))*(XD(t)-XDs(i,t)) + (Y(t)-Ys(i,t))*(YD(t)-YDs(i,t)))...
                / rho_per(i,t);
        else
            rho_per(i,t) = nan;
            rhoDot_per(i,t) = nan;
            phi_per(i,t)=nan;
        end
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%
% LINEAR
%%%%%%%%%%%%%%%%%%%%%%%%%%

% Perturbed Trajectory Nonlinear Model and Measurement Simulation
dx_lin = [];
dy_lin = zeros(36,length(T));
dx = dx0;
for t = 1:length(t_vec)
    % Model Simulation
    dx_lin = horzcat(dx_lin, dx(:,t));
    F = F_variant(dx_lin(1,t),dx_lin(3,t));
    dx(:,t+1) = F*dx(:,t);

    % Measurement Simulation
    for i=1:12
        if ~isnan(rho_per(i,t))
            H = H_variant(X(t),XD(t),Y(t),YD(t),Xs(i,t),XDs(i,t),Ys(i,t),YDs(i,t));
            dy(:,t) = H*dx(:,t);
```

```
            dy_lin(3*i-2:3*i,t) = dy(:,t);
        else
            dy_lin(3*i-2:3*i,t) = [nan nan nan]';
        end
    end
end
dx_lin = dx_lin';


%%%%%%%%%%%%%%%%%%%%%%%%%
% Plots
%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('Plotting Satellite for 1 Orbit: \n');
fprintf("x0 = [6678, 0, 0, 7.7258]' \n");


% Satellite State Model - Nonlinear and Linear
fig = figure('visible','on');
set(fig,'Position',[100 100 900 600]);
sgtitle(sprintf('Satellite State Model - Nonlinear and Linear Simulation \n dx0 = [%.4fkm %.4fkm/s %

subplot(2,2,1); hold on; grid on; grid minor;
plot(T_per,S_per(:,1),'b-','LineWidth',1.2);
plot(t_vec,S(:,1)+dx_lin(:,1),'r--','LineWidth',1.2);
ylim([-7000 7000]);
xlabel('time [sec]');
ylabel('X position [km]');
legend('ODE45 DT Nonlinear Simulation (dt=10s)','DT Linearized Simulation (dt=10s)');
xlim([0 P]);

subplot(2,2,2); hold on; grid on; grid minor;
plot(T_per,S_per(:,3),'b-','LineWidth',1.5);
plot(t_vec,S(:,3)+dx_lin(:,3),'r--','LineWidth',1.5);
ylim([-7000 7000]);
xlabel('time [sec]');
ylabel('Y position [km]');
xlim([0 P]);

subplot(2,2,3); hold on; grid on; grid minor;
plot(T_per,S_per(:,2),'b-','LineWidth',1.5);
plot(t_vec,S(:,2)+dx_lin(:,2),'r--','LineWidth',1.5);
ylim([-10 10]);
xlabel('time [sec]');
ylabel('X velocity [km/s]');
xlim([0 P]);

subplot(2,2,4); hold on; grid on; grid minor;
plot(T_per,S_per(:,4),'b-','LineWidth',1.5);
plot(t_vec,S(:,4)+dx_lin(:,4),'r--','LineWidth',1.5);
ylim([-10 10]);
xlabel('time [sec]');
```

```
ylabel('Y velocity [km/s]');
xlim([0 P]);
saveas(fig,'ASEN5044_HW8_P2_StateModelSimulation.png','png');


% Satellite State Model - Deviations
fig = figure('visible','on');
set(fig,'Position',[100 100 900 600]);
sgtitle(sprintf('Satellite State Model - Deviations \n dx0 = [%.4fkm %.4fkm/s %.4fkm %.4fkm/s]',dx(1,

subplot(2,2,1); hold on; grid on; grid minor;
plot(t_vec,S_per(:,1)-S(:,1),'b-','LineWidth',1.5);
plot(t_vec,dx_lin(:,1),'r--','LineWidth',1.5);
xlabel('time [sec]');
ylabel('\deltaX position [km]');
legend('ODE45 DT Nonlinear Simulation (dt=10s)','DT Linearized Simulation (dt=10s)','Location','South

subplot(2,2,2); hold on; grid on; grid minor;
plot(t_vec,S_per(:,3)-S(:,3),'b-','LineWidth',1.5);
plot(t_vec,dx_lin(:,3),'r--','LineWidth',1.5);
xlabel('time [sec]');
ylabel('\deltaY position [km]');
xlim([0 P]);

subplot(2,2,3); hold on; grid on; grid minor;
plot(t_vec,S_per(:,2)-S(:,2),'b-','LineWidth',1.5);
plot(t_vec,dx_lin(:,2),'r--','LineWidth',1.5);
xlabel('time [sec]');
ylabel('\deltaVX [km/s]');
xlim([0 P]);

subplot(2,2,4); hold on; grid on; grid minor;
plot(t_vec,S_per(:,4)-S(:,4),'b-','LineWidth',1.5);
plot(t_vec,dx_lin(:,4),'r--','LineWidth',1.5);
xlabel('time [sec]');
ylabel('\deltaVY [km/s]');
xlim([0 P]);
saveas(fig,'ASEN5044_HW8_P2_StateModelDeviations.png','png');


% Satellite Measurement Model - Nonlinear and Linear
fig = figure('visible','on');
set(fig,'Position',[100 100 900 600]);
sgtitle(sprintf('Satellite Measurement Model - Nonlinear and Linear Simulation \n dx0 = [%.4f %.4f %
subplot(3,1,2); hold on; grid on; grid minor;
for i=1:12
    plot(T,rhoDot_per(i,:),'k-','LineWidth',1.5)
    plot(T,rhoDot_per(i,:) + dy_lin(3*i-1,:),'--','LineWidth',1.5)
end
xlabel('Time [s]'); ylabel('\rhoDot^i [km/s]')
```

```
subplot(3,1,3); hold on; grid on; grid minor;
for i=1:12
    plot(T,phi_per(i,:),'k-','LineWidth',1.5)
    plot(T,phi_per(i,:) + dy_lin(3*i,:),'--','LineWidth',1.5)
end
subplot(3,1,1); hold on; grid on; grid minor;
for i=1:12
    plot(T,rho_per(i,:),'k-','LineWidth',1.5)
    plot(T,rho_per(i,:) + dy_lin(3*i-2,:),'--','LineWidth',1.5)
end
xlabel('Time [s]'); ylabel('\rho^i [km]')
legend('S1','S1_{lin}','S2','S2_{lin}','S3','S3_{lin}','S4','S4_{lin}','S5','S5_{lin}'...
    ,'S6','S6_{lin}','S7','S7_{lin}','S8','S8_{lin}','S9','S9_{lin}'...
    ,'S10','S10_{lin}','S11','S11_{lin}','S12','S12_{lin}')
xlabel('Time [s]'); ylabel('\phi^i [rad]')
saveas(fig,'ASEN5044_HW8_P2_MeasurementModelSimulation.png','png');


% Satellite Measurement Model - Deviations
fig = figure('visible','on');
set(fig,'Position',[100 100 900 600]);
sgtitle(sprintf('Satellite Measurement Model - Deviations \n dx0 = [%.4f %.4f %.4f %.4f]',dx(1,1),dx
subplot(3,1,2); hold on; grid on; grid minor;
for i=1:12
    plot(T,rhoDot_per(i,:)-rhoDot(i,:),'k-','LineWidth',1)
    plot(T,dy_lin(3*i-1,:),'--','LineWidth',1.5)
end
xlabel('Time [s]'); ylabel('\delta\rhoDot^i [km/s]')
subplot(3,1,3); hold on; grid on; grid minor;
for i=1:12
    plot(T,phi_per(i,:)-phi(i,:),'k-','LineWidth',1)
    plot(T,dy_lin(3*i,:),'--','LineWidth',1.5)
end
xlabel('Time [s]'); ylabel('\delta\phi^i [rad]')
subplot(3,1,1); hold on; grid on; grid minor;
for i=1:12
    plot(T,rho_per(i,:)-rho(i,:),'k-','LineWidth',1)
    plot(T,dy_lin(3*i-2,:),'--','LineWidth',1.5)
end
xlabel('Time [s]'); ylabel('\delta\rho^i [km]')
legend('Nonlinear Measurement Deviation','Linearized Measurement Deviation')
saveas(fig,'ASEN5044_HW8_P2_MeasurementModelDeviations.png','png');


%make movie simulation
% fig=figure; hold on; grid on; grid minor;
% for t=1:length(T)
%     fig; hold on; grid on; grid minor;
%     plot(X(t),Y(t),'b*')
%     for i=1:12
%         plot(Xs(i,t),Ys(i,t),'k.')
```

```
%         if ~isnan(rho(i,t))
%             line([Xs(i,t) X(t)],[Ys(i,t) Y(t)])
%         end
%     end
%
%     axis equal; axis([-8000 8000 -8000 8000]);
%     xlabel('x');ylabel('y');title(sprintf('simulation, t=%.0fs',T(t)))
%     M(t) = getframe(fig);
%     clf(fig);
% end
% video = VideoWriter('visualization', 'Uncompressed AVI');
% video.FrameRate = 60;
% open(video)
% writeVideo(video, M);
% close(video);

function [ F ] = F_variant(X,Y)
mu = 398600;        % km^3/s^2
r0_nom = 6678;          % km
dt = 10;

F = expm(dt*[0, 1, 0, 0;
        (-mu*(r0_nom)^(-3))+(3*mu*X^2*r0_nom^(-5)), 0, 3*mu*X*Y*r0_nom^(-5), 0;
        0, 0, 0, 1;
        (3*mu*X*Y)*r0_nom^(-5), 0, (-mu*r0_nom^(-3))+(3*mu*Y^2*r0_nom^(-5)), 0]);
end

function [ H ] = H_variant(X,Xdot,Y,Ydot,Xs,Xsdot,Ys,Ysdot)
%initialize H
H = zeros(3,4);

%first row
H(1,1) = (X-Xs)/sqrt((X-Xs)^2+(Y-Ys)^2);
H(1,2) = 0;
H(1,3) = (Y-Ys)/sqrt((X-Xs)^2+(Y-Ys)^2);
H(1,4) = 0;

%second row
H(2,1) = (Xdot-Xsdot)/sqrt((X-Xs)^2+(Y-Ys)^2) - (X-Xs)*((X-Xs)*(Xdot-Xsdot)+(Y-Ys)*(Ydot-Ysdot)) / (
H(2,3) = (Ydot-Ysdot)/sqrt((X-Xs)^2+(Y-Ys)^2) - (Y-Ys)*((X-Xs)*(Xdot-Xsdot)+(Y-Ys)*(Ydot-Ysdot)) / (
H(2,2) = (X-Xs)/sqrt((X-Xs)^2+(Y-Ys)^2);
H(2,4) = (Y-Ys)/sqrt((X-Xs)^2+(Y-Ys)^2);

%third row
H(3,1) = ((-Y+Ys)/((X-Xs)^2+(Y-Ys)^2));
H(3,2) = 0;
H(3,3) = ((X-Xs)/((X-Xs)^2+(Y-Ys)^2));
H(3,4) = 0;
end

function [ ds ] = orbit_prop_func(t,s)
```

```
mu = 398600;

x = s(1);
y = s(3);
r = sqrt(x^2+y^2);

xdot = s(2);
ydot = s(4);

xddot = -mu/r^3 * x;
yddot = -mu/r^3 * y;

ds = [xdot, xddot, ydot, yddot]';
end
```