AE4872
Satellite Orbit Determination

Homework Assignment 4

Application of Kalman Filter

Ali Nawaz

Time Spent
20 Hours

TU Delft Delft University of Technology

List of symbols

Table 1: List of symbols

Symbol/Acronym	*Description*	*Unit*
A	Information/Data Matrix	-
LSQ	Least Squares	-
WLSQ	Weighted Least Squares	-
MSL	Matlab Script Lines	-
r	Distance vector	m
pinv.	Pseudo Inverse	-
SVD	Singular Value Decomposition	-
t	Time	s
var.	Variance	Hz^2
x	State vector	-
$\bar{y}$	Observation vector	-
$\hat{y}$	Estimated observation vector	-
$\hat{x}$	Estimated state vector	-
ρ	Range measurement	m
$\tilde{\rho}$	Pseudo range	m
t_T	Transmitter time	s
t_R	Receiver time	s
x_{gps}	GPS x location	m
y_{gps}	GPS y location	m
z_{gps}	GPS z location	m
s	State vector	-
s_0	Initial state vector	-
c	speed of light, 299792458	m/s
P_{xx}	State covariance matrix	-
P_{yy}	Observation vector covariance matrix	-
$\bar{S}$	State vector	-
s	Estimated state vector	-
μ	Standard gravitational parameter	$3.986004419\times10^{14}m^3s^{-2}$
R	Observation co-variance matrix	-
P	State co-variance matrix	-
Q	Noise co-variance matrix	-

Cover Image: 1895 Antique map of constellations:
https://nl.pinterest.com/pin/361765782537705142/.

Contents

1 Position of SWARM A with EKF **1**
- 1.1 Problem definition . . . 1
- 1.2 Comparison estimated vs precise radial direction. . . . 4
- 1.3 Effect of noise. . . . 5

2 Sensitivity Analysis **8**
- 2.1 Effect of weights on initial state vector and observation . . . 8
- 2.2 Effect of numerical integration for propagation . . . 9
- 2.3 Discussion on getting good estimation for initial epochs . . . 10

A Matlab Script **11**

1

Position of SWARM A with EKF

This chapter focuses on estimating the position SWARM A satellite for 11 epochs with the aid of Extended Kalman Filter(EKF). Section 1.1 outlines the algorithm used to achieve this objective. Section 1.2 compares the estimated and precise radial direction of SWARM A satellite. Section 1.3 discusses the effect of noise on the saturation of Kalman filter. Finally the section presents a good choice of system noise before concluding on the overall results.

1.1. Problem definition

Figure 1.1 outlines the steps taken to compute the position of SWARM A satellite with a sequential LSQ process guided by EKF. The choice of initial S, S_0 follows from the final value of S for epoch 1 obtained in Assignment 1. The initialisation vector is presented with the aid of Equation 1.1.

$$\bar{S}_0 = \begin{bmatrix} x(1) \\ y(1) \\ z(1) \\ \frac{x(2)-x(1)}{dt} \\ \frac{y(2)-y(1)}{dt} \\ \frac{z(2)-z(1)}{dt} \end{bmatrix} = \begin{bmatrix} 1.9393e6 \\ -5.8377e6 \\ 2.9332e6 \\ 0.0010e6 \\ -0.0031e6 \\ -0.0069e6 \end{bmatrix} \tag{1.1}$$

Adding the vector [1000m 1000m 1000m 100 m/s 100 m/s 100m/s]' for visualising convergence.

$$\bar{S}_0 = \begin{bmatrix} 1.9403e6 \\ -5.8367e6 \\ 2.9342e6 \\ 0.0011e6 \\ -0.0030e6 \\ -0.0068e6 \end{bmatrix}$$

ϕ is initialised to an identity matrix of size 6x6. While P is the state co variance matrix. Even though position and velocity are correlated, for the sake of simplicity it was assumed that no correlation exists between position and velocity. The initial co variance matrix P_0, thus simplifies into a diagonal matrix, with variance of the states in the diagonal. The initial estimate $\bar{S}_0$ and the precise orbital position for epoch 1 and 2 is known. The difference between them provides a minimum estimate of the standard deviation. This is outlined with the aid of Equation 1.2.

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{\dot{x}} \\ \sigma_{\dot{y}} \\ \sigma_{\dot{z}} \end{bmatrix} = \bar{S}_0 - \begin{bmatrix} x_{precise}(1) \\ y_{precise}(1) \\ z_{precise}(1) \\ \frac{x_{precise}(2)-x_{precise}(1)}{dt} \\ \frac{y_{precise}(2)-y_{precise}(1)}{dt} \\ \frac{z_{precise}(2)-z_{precise}(1)}{dt} \end{bmatrix} \tag{1.2}$$

With this information of standard deviation, the initial co-variance matrix becomes:

$P_0 = \text{diag}\ ([\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_{\dot{x}}^2, \sigma_{\dot{y}}^2, \sigma_{\dot{z}}^2])$

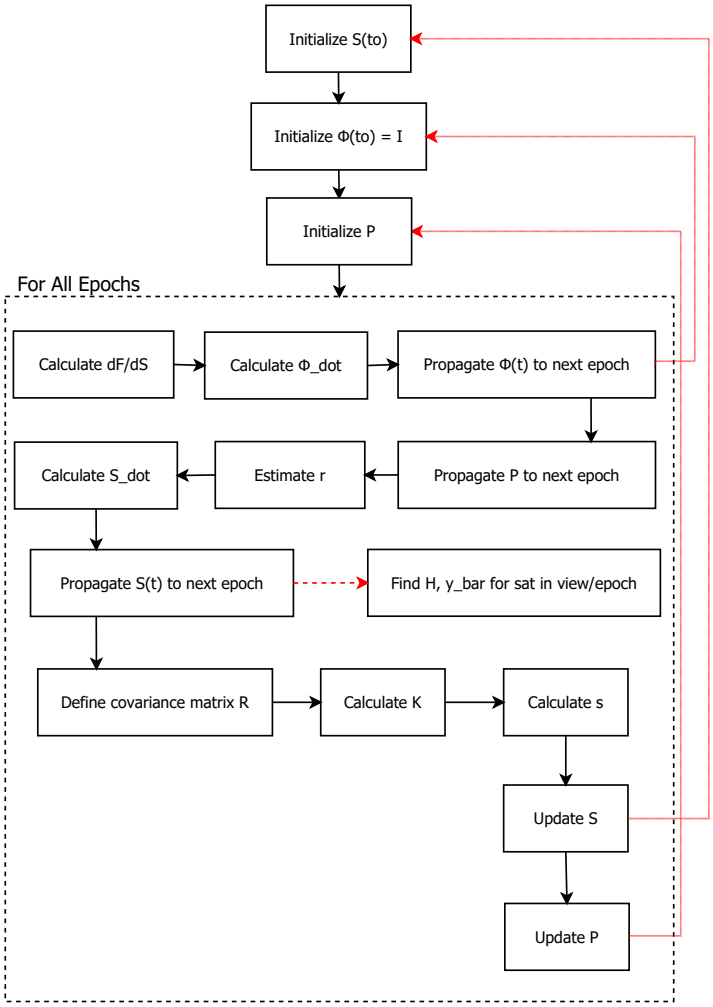


Figure 1.1: Flowchart illustrating steps undertaken to conduct sequential LSQ with EKF for a given initial state vector.

$\frac{dF}{dS}$ is estimated using the same derivations presented in Assignment 2[1]. This is outlined with the aid of Equations 1.3-1.6.

The gradient of F(t) wrt S(t) can be expressed as:

$$\frac{\delta F(t)}{\delta S(t)} = \begin{bmatrix} \frac{\delta F_1}{\delta S_1} & \cdots & \frac{\delta F_n}{\delta S_1} \\ \vdots & \ddots & \vdots \\ \frac{\delta F_1}{\delta S_k} & \cdots & \frac{\delta F_n}{\delta S_k} \end{bmatrix} \tag{1.3}$$

Where, k represents the vector size of states S.

While, n represents the vector size of time differentiation of S, F.

In this case n = k = 6.

$$\frac{dF_4}{dS_1} = \frac{d}{dx}\left(\frac{-\mu \cdot x}{r^3}\right) = \frac{\mu}{r^3}\left(\frac{3 \cdot x^2}{r^2} - 1\right) \tag{1.4}$$

$$\frac{dF_4}{dS_2} = \frac{d}{dy}\left(\frac{-\mu \cdot x}{r^3}\right) = \frac{3 \cdot \mu \cdot x \cdot y}{r^5} \tag{1.5}$$

[1]Git Repository: https://github.com/Alixir/Satellite-Orbit-Determination

$$\frac{dF(S,t)}{dS} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{\mu}{r^3}\left(\frac{3\cdot x^2}{r^2} - 1\right) & \frac{3\cdot\mu\cdot x\cdot y}{r^5} & \frac{3\cdot\mu\cdot x\cdot z}{r^5} & 0 & 0 & 0 \\ \frac{3\cdot\mu\cdot x\cdot y}{r^5} & \frac{\mu}{r^3}\left(\frac{3\cdot y^2}{r^2} - 1\right) & \frac{3\cdot\mu\cdot y\cdot z}{r^5} & 0 & 0 & 0 \\ \frac{3\cdot\mu\cdot x\cdot z}{r^5} & \frac{3\cdot\mu\cdot y\cdot z}{r^5} & \frac{\mu}{r^3}\left(\frac{3\cdot z^2}{r^2} - 1\right) & 0 & 0 & 0 \end{bmatrix} \tag{1.6}$$

Here, ϕ represents the state transition matrix. ϕ can be propagated iteratively with the aid of basic using $\dot{\phi}$. Derivation of $\dot{\phi}$ is presented in Equation 1.7.

$$\begin{aligned} \phi(\dot{t}, t_0) &= \frac{\delta F(t)}{\delta S(t)|S^*}\phi(t, t_0) \\ \phi_{t,t_0} &= \phi_{t_0} + \dot{\phi}(t, t_0)dt \end{aligned} \tag{1.7}$$

This information of ϕ is used to propagate state co-variance matrix P to the next epoch with the aid of Equation 1.8.

$$P_k = \phi_{kj} \cdot P_j \cdot \phi_{kj}^T \tag{1.8}$$

Next is propagated with the aid of $\dot{S}$ or with the aid of state transition matrix ϕ. Both of the processes are outlined in Equation 1.9.

To propagate S with the aid of $\dot{S}$, the norm r of the position values of $\bar{S}$ is required.

$$\dot{\bar{S}} = F(\bar{S}, t) = \dot{\bar{S}}(t) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ -\frac{\mu\cdot x}{r^3} \\ -\frac{\mu\cdot y}{r^3} \\ -\frac{\mu\cdot z}{r^3} \end{bmatrix}$$

The latter follows from the Equations of motion:

Equations of motion can be expressed as: (1.9)

$$\frac{\delta x}{\delta t} = \dot{x}, \quad \frac{\delta y}{\delta t} = \dot{y}, \quad \frac{\delta z}{\delta t} = \dot{z}$$

$$\frac{\delta \dot{x}}{\delta t} = \ddot{x} = \frac{-\mu x}{r^3}, \quad \frac{\delta \dot{y}}{\delta t} = \ddot{y} = \frac{-\mu y}{r^3}, \quad \frac{\delta \dot{z}}{\delta t} = \ddot{z} = \frac{-\mu z}{r^3}$$

Where r = $\sqrt{x^2 + y^2 + z^2}$

$$S = S + \dot{S}$$

To propagate S with the aid of ϕ:

$$S = \phi_{t,t_0} S$$

The design matrix for each Epoch is updated using Equation 1.10. The size of H matrix depends on the no. of satellites observed per epoch. Size of H = no. of sats in view by 6. The definition of H does not poses full rank by definition. Thus singularity warnings are present in Matlab numerical calculations.

$$H(t) = \begin{bmatrix} \frac{x - x_{GPS}}{\rho} & \frac{y - y_{GPS}}{\rho} & \frac{z - z_{GPS}}{\rho} & 0 & 0 & 0 \end{bmatrix} \tag{1.10}$$

The observation vector $\bar{y}$ for each epoch is estimated with the aid of Equation 1.11.

$$\bar{y} = \tilde{\rho} - (c \cdot t_R - c \cdot t_T) - \rho$$

Where, ρ represents: (1.11)

$$\rho = \sqrt{(x - x_{GPS})^2 + (y - y_{GPS})^2 + (z - z_{GPS})^2}$$

During Assignment 2 it was provided that the observations are uncorrelated with standard deviation of 5m. Thus R is a square co-variance matrix with dimension equal to vector length of observations per epoch. Each diagonal element of R is 25. Using the information provided so far, the Kalman filter can be calculated with the aid of Equation 1.12.

$$K_k = P_k H_k^T [H_k P_k H_k^T + R]^{-1} \tag{1.12}$$

With the aid of Kalman filter state difference s is estimation as illustrated in Equation 1.13. The state difference $\hat{s}$, is used to update $\bar{S}$. While Kalman gain K is used to update $\hat{P}$ for current epoch. This is also illustrated in Equation 1.13

$$\hat{s} = K\bar{y}$$
$$S = S + \hat{s} \text{ And, } \hat{P} = (I - KH)P \tag{1.13}$$

Where, "^" represents the information following from a least squares optimisation.

The $\hat{P}$, $\hat{S}$ and ϕ shown above is used as intializing parameters for the next epoch, and hence the name "Sequential LSQ with EKF".

1.2. Comparison estimated vs precise radial direction.

Figures 1.2 and 1.3 illustrate the position and differences in position for the case where $\bar{S}$ is propagated to the next epoch by basic Euler extrapolation. Positions of SWARM A satellite for the two approaches outlined in Assignment 2 are also presented, to check if EKF actually performs well upto it's reputation. And it does. While Figures 1.4 and 1.5 illustrate the same scenario but now for the case where $\bar{S}$ is propagated with the aid of state transition matrix ϕ.

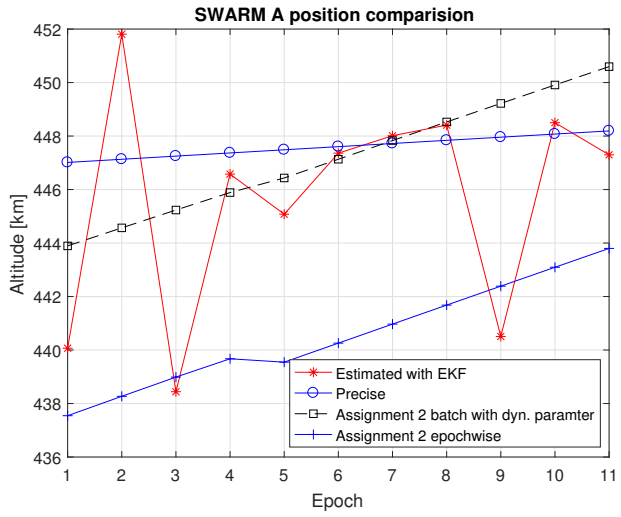


Figure 1.2: Measured and precise orbit direction for SWARM A satellite for $\bar{S}$ propagation with $\dot{\bar{S}}$.

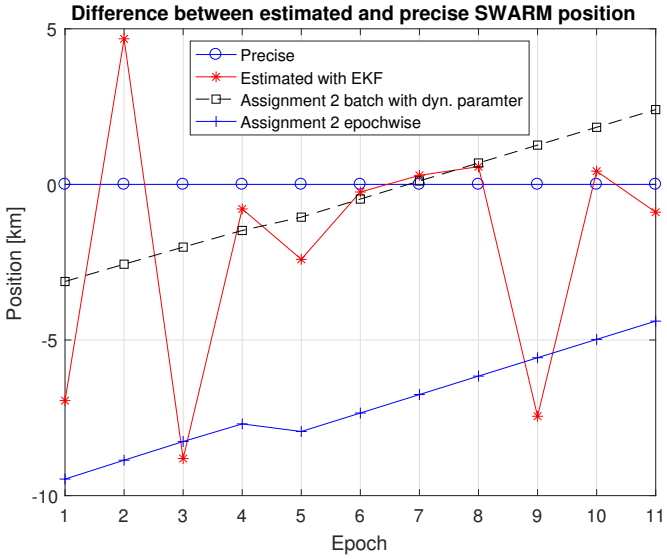


Figure 1.3: Difference between the measured and precise orbit direction for SWARM A satellite for $\bar{S}$ propagation with $\dot{\bar{S}}$.

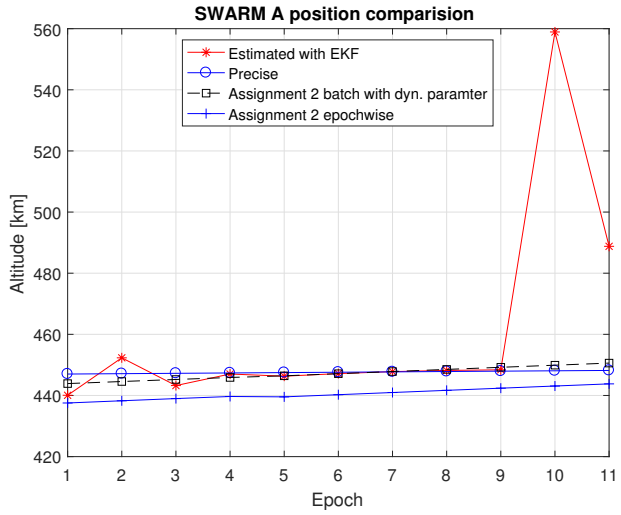


Figure 1.4: Measured and precise orbit direction for SWARM A satellite for $\bar{S}$ propagation with $\dot{\bar{S}}$.

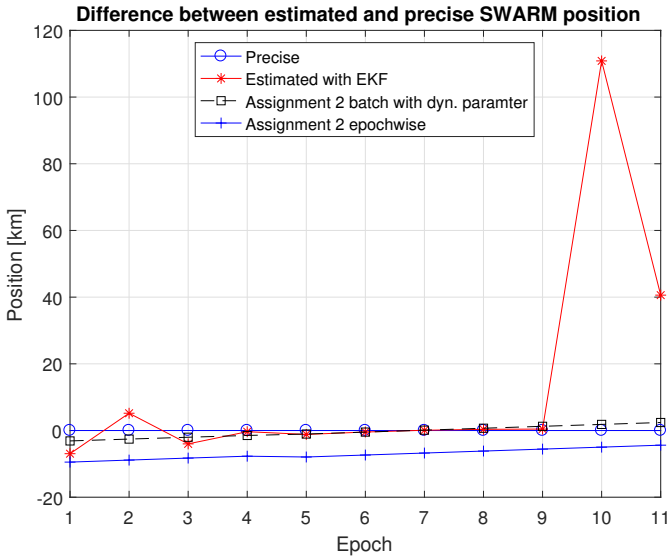


Figure 1.5: Difference between the measured and precise orbit direction for SWARM A satellite for $\bar{S}$ propagation with $\dot{\bar{S}}$.

Definitely, the ϕ propagated $\bar{S}$ presents better solution. This might be hard to see for the jump in position after 9th Epoch. Thus a zoomed in view is presented in Figure 1.6.However, the results are not that accurate after 9th epoch, this could be due to saturation of the Kalman filter. To avoid saturation of Kalman filter, the effect of adding noise is analysed in the next section.

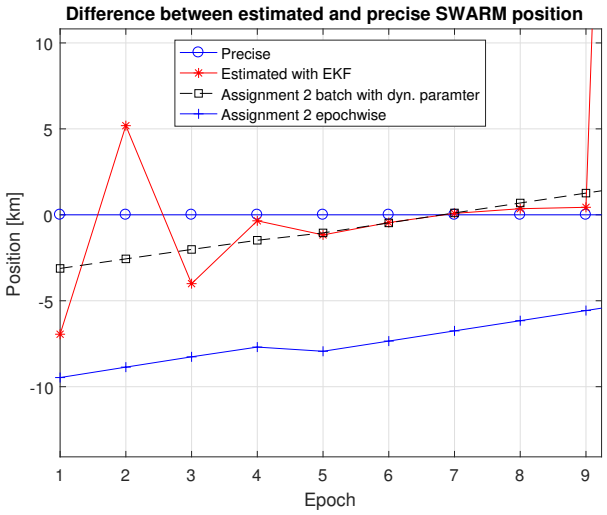


Figure 1.6: Zoomed in difference between the measured and precise orbit direction for SWARM A satellite for $\bar{S}$ propagation with $\dot{\bar{S}}$.

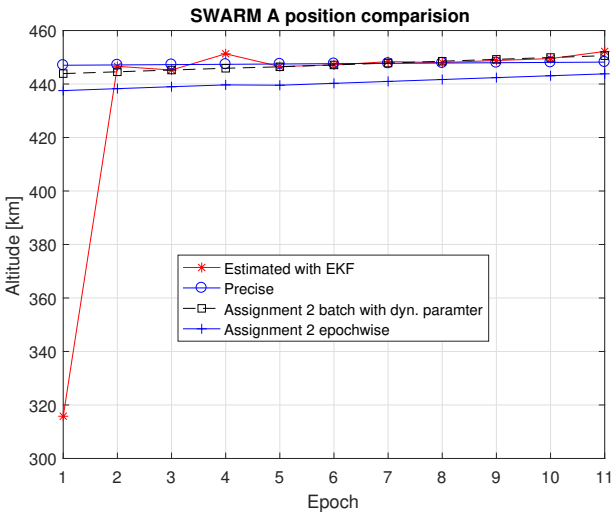


Figure 1.7: Position of SWARM A satellite for added noise obtained via trial and error. Other position estimations are provided to compare the level of convergence.

1.3. Effect of noise

Previous analysis assumed that the Kalman filter is perfect, and that all the dynamical parameters are taken care of by the state transition matrix ϕ. However, this is not the case, which results in saturation of the Kalman filter for increasing propagations. The state vector and state covariance are assumed to be perfect after increasing iterations and new observations are paid less priority to. To compensate for this state noise compensation is applied. Usually the noise is added as a zero mean Gaussian noise in the state vector equation, in form of a disturbance input. However, since this is already present it will not be taken into account. Rather here, an attempt is taken to model it directly into the co-variance matrix. This is done with the aid of Equation 1.14.

$$P_k = \phi_{kj} P_j \phi_{kj}^T + \Gamma_{kj} Q_k \Gamma_{kj}^T$$
Where Γ is taken as identity and Q_k for the sake of simplicity is taken as constant.: (1.14)
$$Q = diag([\lambda_{x_{pos}}\ \lambda_{y_{pos}}\ \lambda_{z_{pos}}\ \lambda_{\dot{x}_{pos}}\ \lambda_{\dot{y}_{pos}}\ \lambda_{\dot{z}_{pos}}])$$

From Equation 1.14 it is evident that Q is coupled with P, which mean for a specific value of P, Q should depend on it. However, since it is assumed that Q is constant, one Q is used for variable P, this will definitely not be the optimum value. Two cases will be analysed here **1.** Trial and error **2.** Using observation for a case where Q is 0. **3.** Using the information found in method 2, to design a Q which updates itself every iteration.

1. The first attempt is simple. Higher values for positional λ and velocity λ resulted in increased deviations from the case where all λs are set to zero. Value with negative order of magnitudes seemed to perform better (i.e. 10^{-x} where x is positive). After multiple trial and error attempts, for λ_{pos} (represents λ for all positions x, y and z) a value of 0.001, combined with λ_{vel} of 0.1 resulted in better convergence for later epochs. The result is presented with the aid of Figures 1.7, 1.8 and 1.9. Since the positional plot does not provide much insight, only the difference plots will be provided for the next examples. Comparing Figures 1.6 and 1.9, it can be observed that the initial jump at epoch 1 is significant, however after 9th epoch the solutions tend to converge rather than diverge, However comparing epochs 3-9 it can be argued that adding this constant λ, in form of Q, has impact on the performance. However, it's great to see how trying random numbers with little logic attached to it can provide such convergence for increasing epoch. In method 2 a rather logical method is implemented.

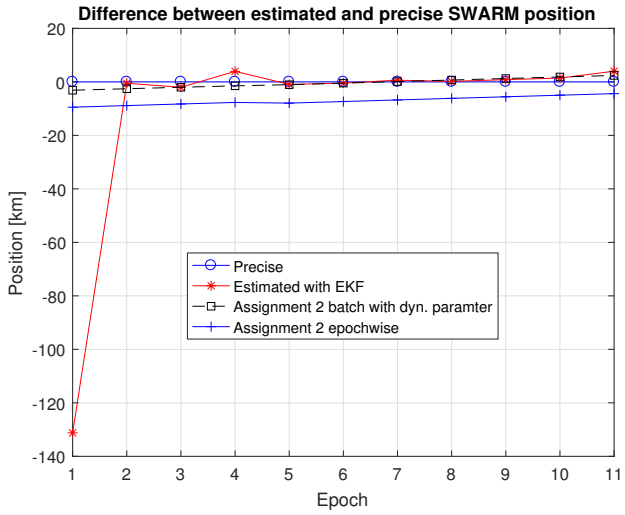


Figure 1.8: Difference of SWARM A satellite from precise orbit for added noise obtained via trial and error. Other positional differences are provided to compare the level of convergence.

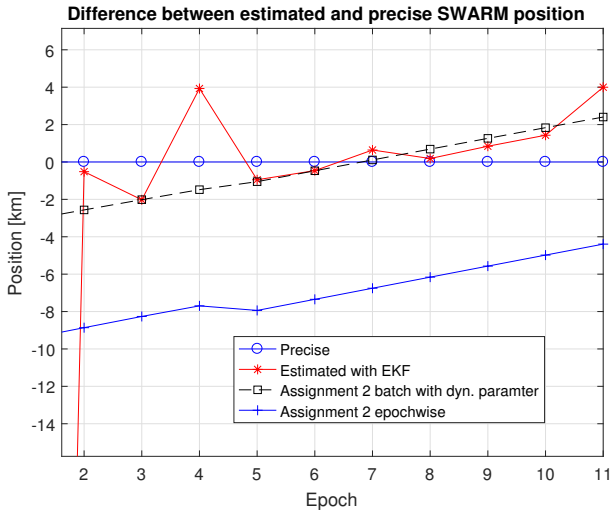


Figure 1.9: Zoomed in difference of SWARM A satellite from precise orbit for added noise obtained via trial and error. Other positional differences are provided to compare the level of convergence.

2. One would expect the standard deviation of the noise to be in the same units as the standard deviation of state vector. Thus the units of diagonal terms of Q matrix should be in m^2 and $(m/s)^2$. Measuring the distance in the 9th epoch with the aid of Figure 1.6 before the measured position blows up, the distance is off by 436.6m and that for 8th epoch is 354.9m. Thus the the value of positional λ should be around 400^2, since adding this to P of 8th and 9th epoch fixes the co-variance at that point. By fixing the co-variance it is meant that an attempt is taken to add noise in a manner that P represents the actual variance of positions and velocity at a given epoch. The velocity terms should be around $\frac{(436.6-354.9)^2}{dt^2}$ $(m/s)^2$. For simplicity all x,y and z positions are given the same λ_{pos} weights, equivalent statement holds for λ_{vel}. Implementing Q = diag([400^2, 400^2, 400^2, $\frac{(436.6-354.9)^2}{dt^2}$, $\frac{(436.6-354.9)^2}{dt^2}$, $\frac{(436.6-354.9)^2}{dt^2}$]) results in much better convergence as shown in Figures1.10 and 1.11. Furthermore, the initial jump is reasonably reduced. This also proves how the reasoning presented in Method **1.**, that higher orders of values lead to increasing deviation, is wrong.

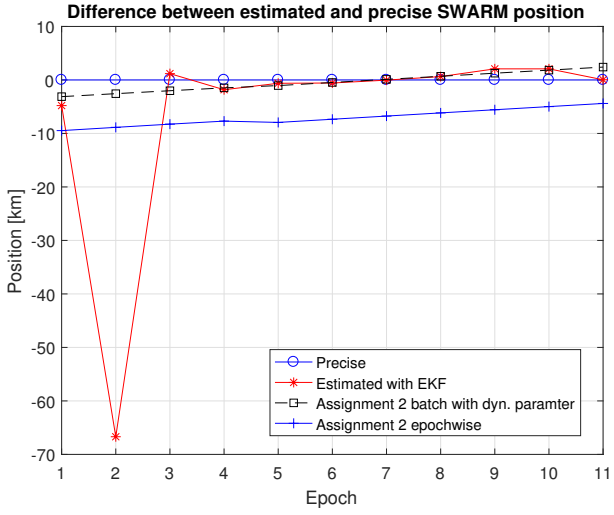


Figure 1.10: Difference of SWARM A satellite from precise orbit for added noise obtained for Q=diag([400^2, 400^2, 400^2, $\frac{(436.6-354.9)^2}{dt^2}$, $\frac{(436.6-354.9)^2}{dt^2}$, $\frac{(436.6-354.9)^2}{dt^2}$]). Other positional differences are provided to compare the level of convergence.

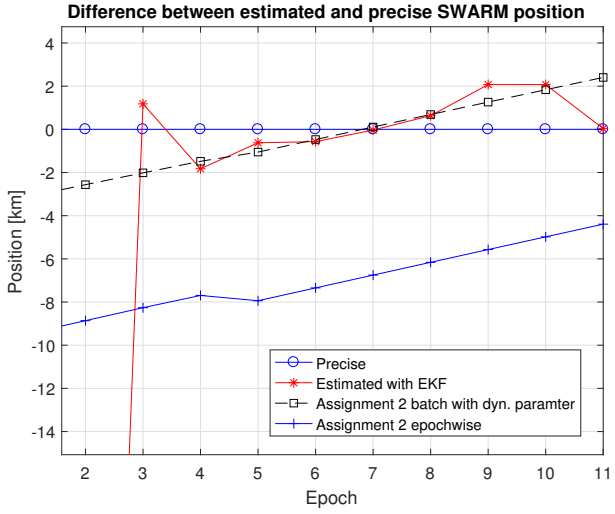


Figure 1.11: Zoomed in difference of SWARM A satellite from precise orbit for added noise obtained for diag([400^2, 400^2, 400^2, $\frac{(436.6-354.9)^2}{dt^2}$, $\frac{(436.6-354.9)^2}{dt^2}$, $\frac{(436.6-354.9)^2}{dt^2}$]). Other positional differences are provided to compare the level of convergence.

3.However, the Q constructed in Method **2.** is still constant for all for epochs. This reasoning is fundamentally in contradiction with the method in which the diagonal values of Q is estimated. A better approach is an iterative one. A strict assumption is that at the end of every epoch the precise position at that epoch is also known. The square of the difference in between the estimated and precise position at this epoch is used as λ_{pos} in noise, while the square of the difference between previous and current position estimate divided by

the square of the time step is used as λ_{vel}. The algorithm flow is presented in Figure 1.12. While the results of this process is outlined in Figures 1.14 and 1.15. It can be observed that the initial jump is is much smaller compared to the results in Figures 1.8 and 1.10. Overtime the results also tend to converge closer to the precise orbit.

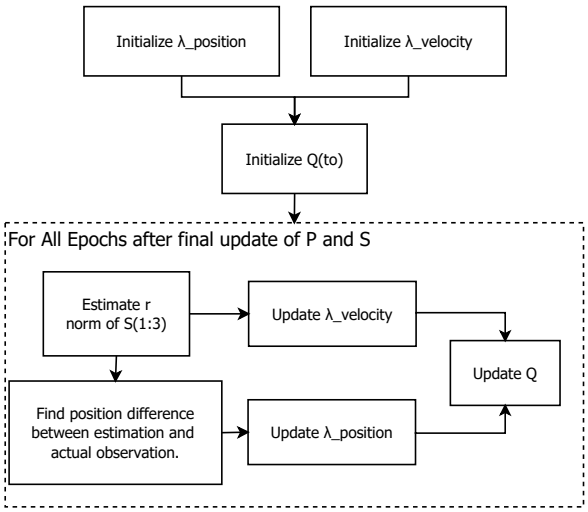


Figure 1.12: Flowchart illustrating steps undertaken to update Q matrix.

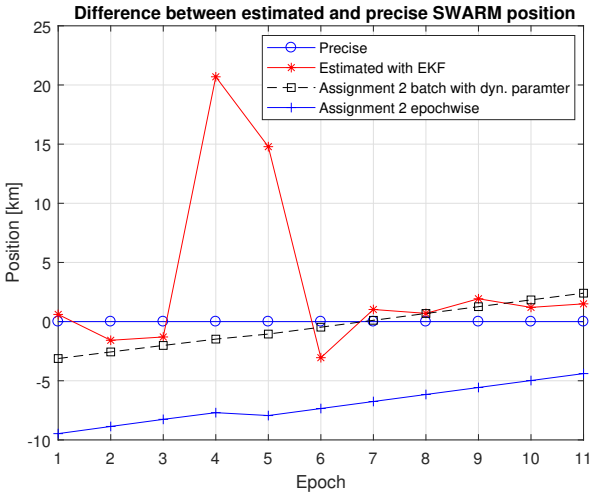


Figure 1.13: Effect of P_0 = P_0 + diag([10^{14} 10^{14} 10^{14} 10^6 10^6 10^6]) on the difference between estimated and precise position.

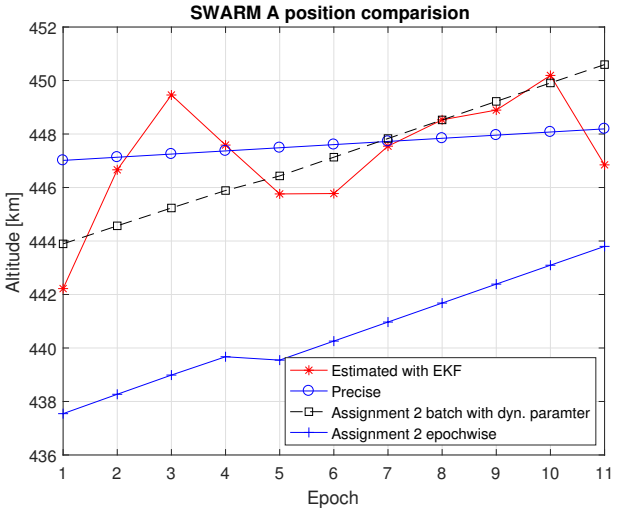


Figure 1.14: Difference of SWARM A satellite from precise orbit for added noise obtained for iterative Q. Other positional differences are provided to compare the level of convergence.

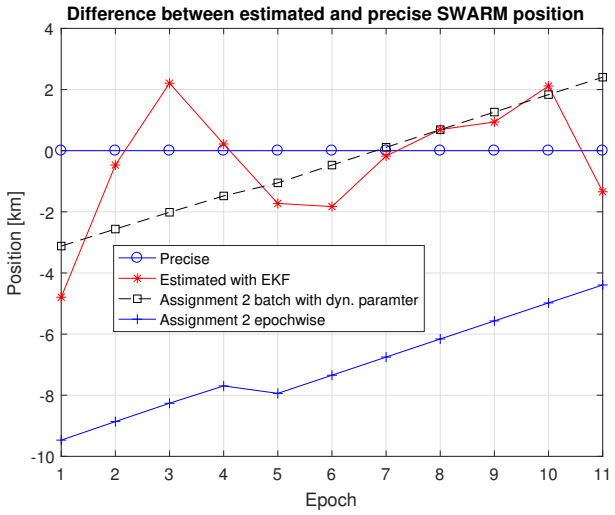


Figure 1.15: Zoomed in difference of SWARM A satellite from precise orbit for added noise obtained for iterative Q. Other positional differences are provided to compare the level of convergence.

After analysing the above three scenarios, which value of Q shall be considered optimum? The answer is: it's hard to tell. The number of epochs presented is not significant enough. The results from technique 1 outlined in Figure 1.9 shows a divergence for the last epoch. It could be that the solutions converge to the precise orbit with increasing epochs, but it might as well be that they diverge. Especially, since the weights are random it might just as well a strike of luck. The choice now comes down to methods 2. and 3. The advantage of method 3 is that it is iterative so eventually as the position differences go to zero, no weight will be added. But the definition for λ_{vel} indicates that they might not go to zero. In that case the idea is still flawed. While Method 2. presents an initial jump which however converge reasonably well with increasing epochs. If the initial jump is not an issue, Method 2. is preferred based on the given 11 epochs, especially since it converges in the final epoch. However, the sum of error residuals over the given 11 epochs is considered than Method 3. becomes a good choice.

2

Sensitivity Analysis

This chapter focuses on effect of increasing/decreasing weights on the initial state and observation vector. Next the effect of propagating $\bar{S}$ with the aid of Euler integration, state transition matrix and ode45 is analysed. As a conclusion, a discussion is outlined to get good initial performance with Kalman filter.

2.1. Effect of weights on initial state vector and observation

The state co-variance matrix P, is a good measure of how well we know the states. Putting more weight on the initial state vector means, putting less weight on the observation. Similarly, putting more weight on the observation means using less weight on the initial state vector. Increasing more weight on observation would mean decreasing the diagonal elements of R. While, increasing the diagonal terms of P puts more weight on the observation, since it says we know less about the initial state. Decreasing the diagonal terms of P puts more weight on the initial state. A test is conducted to prove the above statement. Since P and R form, K changing both effects the outcome. However, for simplicity in observation only one variable, P is changed. It is assumed that R is provided along with the observations. For further simplification, the dynamics of an evolving Q is ignored. Constant Q outlined in method 2 of previous Section is implemented. In this case, the results of P without any modifications is the same as the result presented in Figures 1.10 and 1.11. First the diagonal values of P is increased by diag($[10^{14}\ 10^{14}\ 10^{14}\ 10^{6}\ 10^{6}\ 10^{6}]$). Based on previous analysis, one would expect less weight on initial state vector and more weight on observations. Since now P indicates more uncertainty about the initial state vector. The result of this should be values of estimation being closer to observation. The result is outlined in Figure 1.13. Secondly, P matrix is assumed to zero. Which emphasises a scenario where all information above the state is known. Based on the above analysis, vice versa is expected. i.e. more weight on initial state vector and less on observations. The estimations are expected to be further away from the observation. The result is presented in Figure 2.1. From Figure 2.1 one might get the feeling that it converges, thus a zoomed in version in provided in Figure 2.2 to prove otherwise.

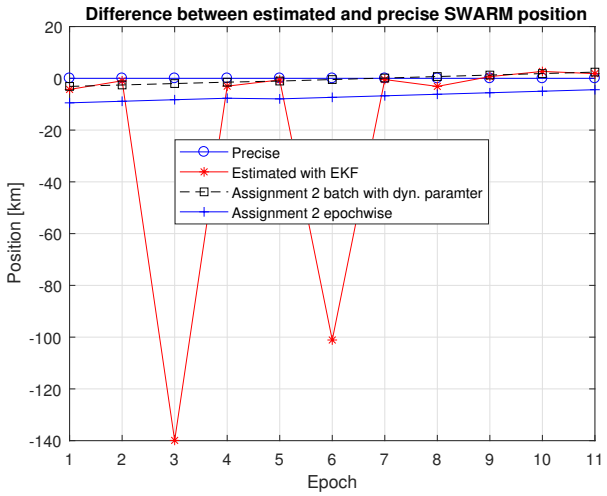


Figure 2.1: Effect of P_0 = zero matrix on the difference between estimated and precise position.

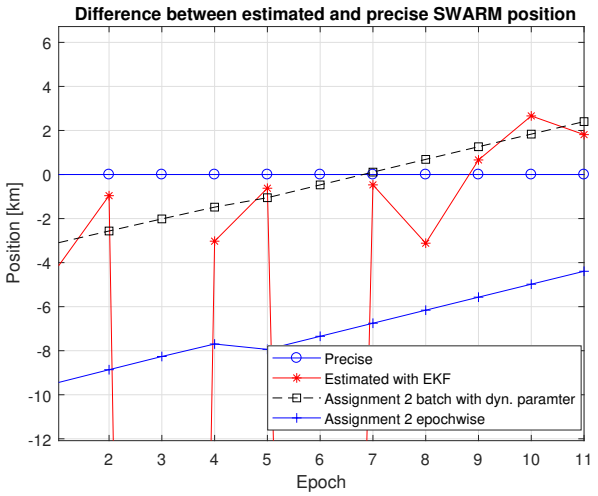


Figure 2.2: Zoomed in effect of P_0 = zero matrix on the difference between estimated and precise position.

The observations are in line with the statements provided. However to put them in numbers, the sum of

squared error residuals (Sum of squared difference from precise orbit for all epochs) is also presented. For P_0 sum of squared error residuals = 4.4858e+03 km^2. For P_0 = P_0 + diag([10^{14} 10^{14} 10^{14} 10^6 10^6 10^6]), this results in 26.3126 km^2. For P_0 = zero matrix, sum of squared error residuals is 2.9842e+04 km^2.

2.2. Effect of numerical integration for propagation

An analysis is carried out for three different ways of propagating $\bar{S}$. P is taken as P_0 and Q from method 2 is chosen. Results of first propagation with the aid of state transition matrix, i.e. $\bar{S}$= $\phi \cdot \bar{S}$ was outlined in Figures 1.8 and 1.9. The sum square of error residuals was obtained to be 4.4858e+03 km^2. To be fair, and avoid the initial jump due to the effect of P initialisation, the sum of square of error residuals from epoch 3 onward is 14.4454km^2. Second way of propagation is simple Euler integration. i.e. $\bar{S}$ =$\bar{S}$+ $\dot{\bar{S}}$dt, where $\dot{\bar{S}}$ is obtained with the aid of Equation 1.9. The results of this is presented in Figure2.3. The sum of squared error residuals for all epochs was obtained to be 199.876 km^2. While the sum of squared error residuals from epochs 3 onward gives 70.3856 km^2. The last case is the one with ode45. A separate function is constructed, which is later used with the aid of odeset in Matlab. The function is presented as follows:

```
function dSdt = Sdot(T,S);
    r = norm(S(1:3));
    mu = 3.986004419*10^(14);
    dSdt = zeros(6,1);
    dSdt(1) = S(4);
    dSdt(2) = S(5);
    dSdt(3) = S(6);
    dSdt(4) = -mu*S(1)/r^(3);
    dSdt(5) = -mu*S(2)/r^(3);
    dSdt(6) = -mu*S(3)/r^(3);
end
```

The following is used to implement it with the aid of ode45

```
    options = odeset('AbsTol',1e-10, 'RelTol',1e-10);
    [T,S] = ode45(@Sdot, [time(e)-dt, time(e)],S,options);
    S = S(end,:) ';
```

The result of implementing ode45 on the difference in precise and estimated position is outlined in Figure 2.4. The sum of squared error residuals for all epochs was obtained to be : 231.2208 km^2. While the sum of squared error of residuals for epoch 3 onward is obtained to be 28.0090km^2.

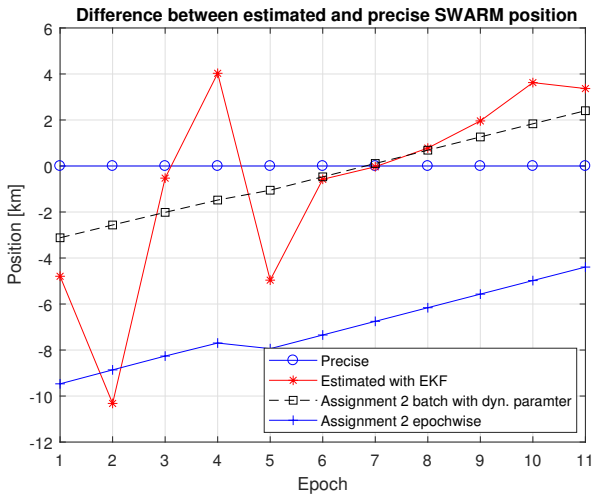


Figure 2.3: Effect of Euler integration on difference between estimated and actual satellite position.

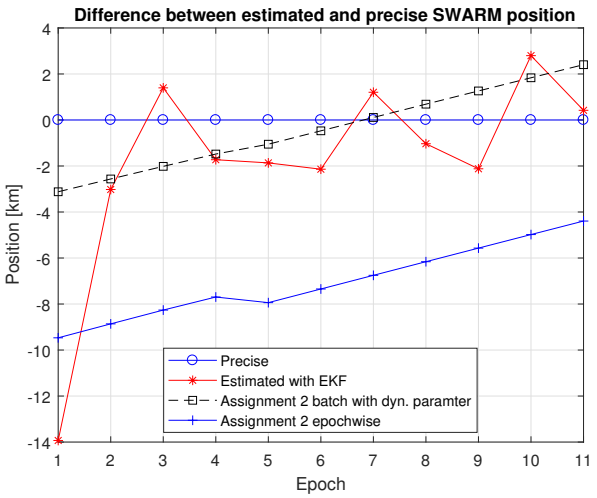


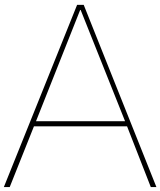
Figure 2.4: Effect of ode45 on difference between estimated and actual satellite position.

Comparing all of the above results, when all epochs are considered state propagation with the aid of Euler integration presents the best results in term of the sum of squared error residuals. However, the initial jump contributes significantly in the performance if all epochs are considered. The results are more due to the initialisation and less emphasis is provided on the propagation technique. However, if the later epochs are considered than more emphasis is weighted on the propagation technique. Since Kalman filter updates the initial parameters to a reasonable level. Thus, comparing epoch 3 onward provides a better overview on the

effect of the choice of propagation technique. From this view, propagation with the aid of state transition matrix performs the best (in terms of sum of squared error residuals) followed by ode45, while Euler integration performs the worst.

2.3. Discussion on getting good estimation for initial epochs

It was observed in Figure 1.8 that, the initial jump is very high for the EKF estimate. This is very much due to the different initialisation conditions used. This was improved in Figures 1.13 where more weight was provided on the state by increasing the diagonal terms on P. This was also improved by using different state propagation schemes. As shown in Figures 2.3 and 2.4. Choosing a good value of Q also leads to a better initial performance as seen in Figures 1.14 and 1.15. But these factors directly influence the Kalman gain K, which is the reason the estimation behave the way they do. From fundamentals of control theory, it is known that for large fixed K fast initial convergence is expected, but this results in large steady state variance. While for small fixed K, slow initial convergence is expected but better performance for steady state errors is observed. The case scenario illustrated implements a variable Kalman gain K, which relies on the choice of H, P and R. Since R and H is predefined, P_0 should be selected which results in high initial Kalman gain, which will result in good initial performance. Furthermore, the Kalman gain is iterative which means eventually it will adapt to reasonable value to provide better steady state error, compared to a scenario where fixed high gain K is implemented.



Matlab Script

Following script can be also be found at this Github Repository:
https://github.com/Alixir/Satellite-Orbit-Determination

```
% Single point positioning
% Wouter van der Wal
% Delft University of Technology

clear all
close all
clc

% Fundamental GPS constants
% gpsconst;

dt                  = 10;       % time step
number_of_epochs    = 11;
order               = 11;       % order of interpolation of precise orbits GPS satellites
order_clk           = 3;        % order of interpolation of clock correction
t_orbit             = 60*[-78:15:73];  % time vector for precise orbits for GPS satellite ...
    provided by IGS every 15 minutes
time                = 0:dt:(number_of_epochs-1)*dt; % time vector for SWARM satellite
c                   = 299792458; % velocity of light m/sec
earth_radius        = 6371; % in km
%% Read the IGS coordinates and clock corrections for all the GPSsatellites
[x_gps,y_gps,z_gps,clock_correction] = readsp3('igs19301.sp3');


%% Read the Swarm GPS data
fid_rinex = fopen('Data_SWARM_A.dat','r');
for ii = 1:15
    line = fgets(fid_rinex);
end
% Read the observations
for epoch = 1:number_of_epochs
    line = fgets(fid_rinex);
    ft = sscanf(line,'%c %f %f %f %f %f %f %f %f');
    no_sat(epoch) = ft(9);
    si = 0;
    for no = 1:no_sat(epoch)
        si = si + 1;
        line = fgets(fid_rinex);
        ft = sscanf(line,'%c %f %f %f %f %f %f %f %f');
        prns(epoch,no) = ft(2);        % PRN number of satellite
        n = prns(epoch,no);
        C1(epoch,si) = ft(5);
        % Interpolate the coordinates of the GPS satellites to the current epoch
        [x_gps_interpolated(epoch,si),x_error(epoch,si)] = polint(t_orbit,x_gps(:,n),time(...
    epoch),order);
        [y_gps_interpolated(epoch,si),y_error(epoch,si)] = polint(t_orbit,y_gps(:,n),time(...
    epoch),order);
        [z_gps_interpolated(epoch,si),z_error(epoch,si)] = polint(t_orbit,z_gps(:,n),time(...
    epoch),order);
        [clock_correction_interpolated(epoch,si),t_error(epoch,si)] = polint(t_orbit,...
```

```
        clock_correction(:,n),time(epoch),order_clk);
    end
end
fclose(fid_rinex);
ccGPS = clock_correction_interpolated*c;
x_gps_interpolated = x_gps_interpolated*1e3;                    % Convert to meters
y_gps_interpolated = y_gps_interpolated*1e3;
z_gps_interpolated = z_gps_interpolated*1e3;
r_gps_interpolated = sqrt(x_gps_interpolated(:,1).^2+y_gps_interpolated(:,1).^2+...
    z_gps_interpolated(:,1).^2);
% Plot for checking
% figure; plot((r_gps_interpolated-6371e3)/1e3); title('altitude PRN 7 [km]')


%% Read the precise Swarm orbit
fid_obs = fopen('PreciseOrbit_SWARM_A.dat','r');
for ii = 1:22
    line = fgets(fid_obs);
end
for epoch = 1:11
        line = fgets(fid_obs);
        line = fgets(fid_obs);
        x_precise(epoch) = read_strval(line(5:18)); % These are in km from the center of the ...
    earth - Ali
        y_precise(epoch) = read_strval(line(19:32));
        z_precise(epoch) = read_strval(line(33:46));
        line = fgets(fid_obs);
end
fclose(fid_obs);
% Convert to spherical coordinates for checking
r_precise = sqrt( x_precise.^2+y_precise.^2+z_precise.^2 );
% Plot for checking
% figure; plot(r_precise-6371); title('altitude Swarm [km]')
%% ========== START OF STUDENT SCRIPT ==============
%% Author: Ali Nawaz
% Course: AE4872 Satellite Orbit Determination
% Homework Assignment 4: Extended Kalman Filter
%%=================================================
%% Data retrieval
xg = x_gps;
yg = y_gps;
zg = z_gps;
cc = clock_correction;

xgi = x_gps_interpolated;
ygi = y_gps_interpolated;
zgi = z_gps_interpolated;
rgi = r_gps_interpolated;
cci = clock_correction_interpolated;
ccf = ccGPS; % Final clock correction = cci*c

xp = x_precise;
yp = y_precise;
zp = z_precise;
rp = r_precise;

%% Task A - Data Initialisation from Assignment 2
% Final values of the epochwise distance is stored in the variable
% s_final_rcc. Please refer to previous assignment or follow the github
% link. While the clock error is stored in del_t variable. Both of them are
% used to reconstruct the initial state vector.

% Loading the initial state vector from previous script
pos =  load('s_final_rcc','-mat'); % Position in m
clock_errors = load('del_t','-mat'); % Clock errors in s

% s_final_rcc = pos.s_final_rcc(end,:); % Final x,y,z position for last epoch with clock ...
    correction implemented
s_final_rcc = pos.s_final_rcc(1,:); % Final x,y,z position for first epoch with clock ...
    correction implemented
final_clock_error = clock_errors.del_t(1,end); % Final receiver clock correction

% Basic backward Euler scheme to evaluate velocity terms in m/s
% vel = (pos.s_final_rcc(end,:) - pos.s_final_rcc(end-1,:))/dt;
vel = (pos.s_final_rcc(2,:) - pos.s_final_rcc(1,:))/dt;
```

```
% Intializing state vector: [ x, y, z, x_dot, y_dot, z_dot] with clock
% correction implemented
s0_buf = [ s_final_rcc, vel] +  [ 0 0 0 0 0 0 ];

% Shifting initial vector slightly to visualize convergence.
% s0 = s0_buf + [ 1000 1000 1000 100 100 100];
s0 = s0_buf;

%% Task A

% Sequentially scan through all Epochs
S = s0_buf';
mu = 3.986004419e14; % [ m^3 s^-2]
% Minimum standard deviation in the state vector, using the known precise
% values for first epoch.
init_diff = s0_buf - [xp(1) yp(1) zp(1) (xp(2)-xp(1))/dt (yp(2)-yp(1))/dt (zp(2)-zp(1))/dt];
P = diag([init_diff.^2]);
% Effect of changing weights on observation vs
% P = P + diag([ 10^20 10^20 10^20 10^10  10^10 10^10 ]);
% P = diag([ 0 0 0 0 0 0 ]);
dist = [];
phi = eye(6,6);
% Lambda parameter choices
% lambda_pos = 0.001;
% lambda_vel = 0.1;
% lambda_pos = 0;
% lambda_vel = 0;
% lambda_pos = mean([(0.4366-0.3549)^2,(0.3549-0.1093)^2]);
% lambda_vel = lambda_pos/(dt^2);
% lambda_pos = mean([(0.4366-0.3549),(0.3549-0.1093)]);
% lambda_vel = lambda_pos/(dt);
% lambda_pos = mean([((0.4366-0.3549)*10^3)^2,((0.3549-0.1093)*10^3)^2]);
% lambda_vel = lambda_pos/(dt)^2;
% lambda_pos = mean([((0.4366-0.3549)*10^3),((0.3549-0.1093)*10^3)]);
% lambda_vel = lambda_pos/(dt);
% lambda_pos = ((436.6+354.9)/2)^2;
lambda_pos = (400)^2;
lambda_vel = (436.6-354.9)^2/dt^2;
% Noise matrix added to state covariance matrix P to avoid filter
% saturation.
Q = diag([lambda_pos,lambda_pos,lambda_pos,lambda_vel,lambda_vel,lambda_vel]);
% Error co-variance parameters
lvl = [lambda_vel];
lpl = [lambda_pos];
% Collect estimated radial distances
S_collect = [norm(S)];
for e = 1:epoch
        clear H H_tilde dx4 dy4 dz4 drho4;
        r = norm(S(1:3)); % radial distance sqrt(x0, y0, z0) in meters
        % Calculating dF/dS matrix for s0 = [ x, y, z, x_dot, y_dot, z_dot ]
        dF_dS = [ 0 0 0 1 0 0; 0 0 0 0 1 0 ; 0 0 0 0 0 1 ;...
                  (mu/r^3)*( (3*S(1)^2)/(r^2) - 1),    3*mu*S(1)*S(2)/r^5,   3*mu*S(1)*S(3)/r...
    ^5, 0, 0, 0;...
                  3*mu*S(1)*S(2)/r^5, (mu/r^3)*( (3*S(2)^2)/(r^2) -1),     3*mu*S(2)*S(3)/r^5,...
     0, 0, 0   ;...
                  3*mu*S(1)*S(3)/r^5,  3*mu*S(2)*S(3)/r^5, (mu/r^3)*( (3*S(3)^2)/(r^2) -1), 0,...
     0, 0];

        % Calculating phi_dot
        phi_dot = dF_dS*phi;
        % Updating phi
        phi = phi + phi_dot*dt;

        % Propagating state co-variance matrix P
        P = phi*P*phi' + Q ;

        % Calculating S_dot for s0 = [ x, y, z, x_dot, y_dot, z_dot ]
        S_dot = [ S(4); S(5); S(6); -mu*S(1)/r^3; -mu*S(2)/r^3 ; -mu*S(3)/r^3 ];

% Propagate Sj to Sk
% Euler integration
%         S = S + S_dot.*dt;
% State transition matrix
          S = phi*S;
% ode45
```

```
%         options = odeset('AbsTol',1e-10, 'RelTol',1e-10);
%         [T,S] = ode45(@Sdot, [time(e)-dt, time(e)],S,options);
%         S = S(end,:)';

% Design matrix generation
        for s = 1:no_sat(e)
            dx4(s) = S(1) - xgi(e,s); % x - x_gps for sth satellite in view
            dy4(s) = S(2) - ygi(e,s); % y - y_gps for sth satellite in view
            dz4(s) = S(3) - zgi(e,s); % z - z_gps for sth satellite in view

            drho4(s) = sqrt(dx4(s)^2 + dy4(s)^2 + dz4(s)^2); % measured range for nth ...
    satellite in view
            H(s,:) = [ dx4(s)/drho4(s) dy4(s)/drho4(s) dz4(s)/drho4(s) 0 0 0]; % Assembling ...
    epochwise information matrix.
        end
        H_tilde = H*phi;
        y_bar = transpose(C1(e,1:no_sat(e)) + (ccGPS(e,1:no_sat(e))- final_clock_error*c) - ...
    drho4);

        % Rk, Error covariance matrix R at kth epoch
        R = 5^2 * eye(size(y_bar));
        % Kalman Gain K for epoch k
        K = P*H'/(H*P*H' +R);
        % Filter update for x_hat
        x_hat = K*(y_bar);
        % Update S for current epoch
        S = S + x_hat;
        % Update P for current epoch
        P = (eye(size(K*H)) - K*H)*P;
        dist = [dist, norm(S(1:3))]; % Distance from the center of the ref. frame in m

        % Iterative Q explained in the report.
        S_collect = [S_collect,norm(S(1:3))];
        lpl = [lpl,(S_collect(e)-r_precise(e)*10^3)^2];
        lvl = [lvl, ((S_collect(end)-S_collect(end-1))^2)/dt^2];
        lambda_pos = lpl(end);
        lambda_vel = lvl(end);
        Q = diag([lambda_pos,lambda_pos,lambda_pos,lambda_vel,lambda_vel,lambda_vel]);
end
radial_dist = dist*10^(-3); % Altitude in km
dist_a2_data =  load('dist_a2_rcc','-mat'); % Position in m for batch with receiver clock ...
    correction
dist_a2 = dist_a2_data.dist_measured_rcc;
dist_a2seq_data =  load('dist_a2_seq','-mat'); % Position in km for sequential with receiver ...
    clock correction
dist_a2_seq = dist_a2seq_data.dist_measured;

figure(1)
plot(1:length(radial_dist),radial_dist-earth_radius,'r-*',1:length(r_precise),r_precise-...
    earth_radius,'b-o',1:length(dist_a2),dist_a2,'k--s',1:length(dist_a2_seq),dist_a2_seq,'b-+...
    ');
grid on
legend('Estimated with EKF','Precise','Assignment 2 batch with dyn. paramter','Assignment 2 ...
    epochwise','Location','Best');
xlabel('Epoch');
ylabel('Altitude [km]');
title('SWARM A position comparision');

% Altitudes
diff_radial = radial_dist-r_precise; % Difference in km
diff_radial_a2 = dist_a2 - (r_precise-6371)';
diff_radial_a2seq = dist_a2_seq - (r_precise-6371)';

figure(2)
plot(1:11,zeros(1,11),'b-o',1:length(diff_radial),diff_radial,'r-*',1:length(diff_radial_a2),...
    diff_radial_a2,'k--s',1:length(diff_radial_a2seq),diff_radial_a2seq,'b-+');
title('Difference between estimated and precise SWARM position');
legend('Precise','Estimated with EKF','Assignment 2 batch with dyn. paramter','Assignment 2 ...
    epochwise','Location','Best');
xlabel('Epoch');
ylabel('Position [km]');
grid on

% Sum of squared error residual:
sser = diff_radial*diff_radial';
```

```
function dSdt = Sdot(T,S);
    r = norm(S(1:3));
    mu = 3.986004419*10^(14);
    dSdt = zeros(6,1);
    dSdt(1) = S(4);
    dSdt(2) = S(5);
    dSdt(3) = S(6);
    dSdt(4) = -mu*S(1)/r^(3);
    dSdt(5) = -mu*S(2)/r^(3);
    dSdt(6) = -mu*S(3)/r^(3);
end
```