AE4872
Satellite Orbit Determination

Homework Assignment 3

Least squares estimation of an initial state vector and a dynamical parameter

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	-
$\hat{S}$	State vector	-
s	Estimated state vector	-
μ	Standard gravitational parameter	$3.986004419\times10^{14}m^3s^{-2}$

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

Contents

1

LSQ Estimate for initial state vector

Previously in assignment 2, the position of one of the SWARM satellites was analysed without taking into account the dynamics. Previous assignments along with the supporting Matlab files can be accessed at the git repository [1]. For this assignment same data is used, but a dynamical parameter is also estimated. Thus the model under consideration is dynamical which requires solution via numerical techniques. The purpose of this chapter is to outline dynamical model used followed by a discussion of the results obtained.

1.1. Problem definition

The observation vector and the state vector used in Assignment 2, is illustrated with the aid of Equations 1.1 and 1.2. Equation 1.1 illustrates the dependency of the observation vector on measured and estimated range measurements ρ along with the corrections for receiver (t_R) and transmitter clock times(t_T). This observation vector is further expanded in the linear matrix format in Equation 1.2.

$$\bar{y} = \tilde{\rho} - (c \cdot t_R - c \cdot t_T) - \rho \tag{1.1}$$

$$\begin{aligned}
&\bar{y} = A \cdot \bar{x} \\
&\text{Where, the information matrix A is:} \\
&A = \begin{bmatrix} \frac{x - x_{GPS}}{\rho} & \frac{y - y_{GPS}}{\rho} & \frac{z - z_{GPS}}{\rho} \end{bmatrix} \\
&\text{Where, } \rho \text{ represents:} \\
&\rho = \sqrt{(x - x_{GPS})^2 + (y - y_{GPS})^2 + (z - z_{GPS})^2} \\
&\text{And finally the state vector is represented by:} \\
&\bar{x} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}
\end{aligned} \tag{1.2}$$

Least squares approach undertaken in this section involves using an initial state vector which consists of position and velocity. This is expressed with the aid of Equation 1.3. The state vector is significantly different than the one represented in Equation 1.2. However, The information presented in Equation 1.2 can be manipulated to accommodate the new state vector presented in Equation 1.3.

$$\bar{S} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{1.3}$$

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

Differentiating the new state vector results in the expression presented in Equation 1.4.

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

The latter follows from the Equations of motion:

Equations of motion can be expressed as:

$$\begin{aligned} \frac{\delta x}{\delta t} &= \dot{x} \\ \frac{\delta y}{\delta t} &= \dot{y} \\ \frac{\delta z}{\delta t} &= \dot{z} \\ \frac{\delta \dot{x}}{\delta t} &= \ddot{x} = \frac{-\mu x}{r^3} \\ \frac{\delta \dot{y}}{\delta t} &= \ddot{y} = \frac{-\mu y}{r^3} \\ \frac{\delta \dot{z}}{\delta t} &= \ddot{z} = \frac{-\mu z}{r^3} \end{aligned} \tag{1.4}$$

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

$\bar{S}$ can be re-written to facilitate iterative propagation with the aid of Euler integration. This is presented with the aid of Equation 1.5.

$$\dot{\bar{S}} = F(S^*, t) + \frac{\delta F(t)}{\delta S(t)|S^*}[S(t) - S^*(t)]$$

$$\dot{\bar{S}} \approx S(t_0 + \Delta) - S(t_0)$$

Here, $S(t) - S^*(t)$ can be represented as the state vector used in the previous assignment.

However, the previous state vector will now include velocity terms as well.

The updates state vector, s can now be written as:

$$s = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta \dot{x} \\ \Delta \dot{y} \\ \Delta \dot{z} \end{bmatrix} \tag{1.5}$$

Thus the above equation for $\dot{\bar{S}}$ simplifies to:

$$s(t) = \phi(t, t_0)s(t_0)$$

$$\dot{s}(t) = \dot{\phi}(t, t_0)s(t_0)$$

Here, ϕ represents the state transition matrix.

ϕ can be propagated iteratively using:

$$\dot{\phi(t, t_0)} = \frac{\delta F(t)}{\delta S(t)|S^*}\phi(t, t_0)$$

Using the expressions outlined in Equation 1.5 the linear matrix expression for one epoch can now be written as shown in 1.6, which relates observations to the initial state.

$$\bar{y}_1 = H_1\phi(t_1, t_0)s(t_0) + \epsilon_1$$

Where H is the updated information matrix, to facilitate the new state vector s

$$H(t_{i+1}) = \begin{bmatrix} \frac{x - x_{GPS}}{\rho} & \frac{y - y_{GPS}}{\rho} & \frac{z - z_{GPS}}{\rho} & 0 & 0 & 0 \end{bmatrix} \tag{1.6}$$

A key relation in the above expression is $\frac{\delta F}{\delta S}$. This is derived with the aid of equation of motion presented earlier in Equation 1.4. Equations 1.7 - 1.10 derive this key expression.

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

$$\frac{\delta F(t)}{\delta S(t)} = \begin{bmatrix} \frac{\delta F_1}{\delta S_1} & \cdots & \frac{\delta F_n}{\delta S_1} \\ \vdots & \ddots & \vdots \\ \frac{\delta F_1}{\delta S_k} & \cdots & \frac{\delta F_n}{\delta S_k} \end{bmatrix} \tag{1.7}$$

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.

$$\frac{dF_4}{dS_1} = \frac{d}{dx}\left(\frac{-\mu \cdot x}{r^3}\right) = \frac{\mu}{r^3}\left(\frac{3 \cdot x^2}{r^2} - 1\right) \tag{1.8}$$

$$\frac{dF_4}{dS_2} = \frac{d}{dy}\left(\frac{-\mu \cdot x}{r^3}\right) = \frac{3 \cdot \mu \cdot x \cdot y}{r^5} \tag{1.9}$$

$$\frac{dF(S,t)}{dS} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{\mu}{r^3}\left(\frac{3 \cdot x^2}{r^2} - 1\right) & \frac{3 \cdot \mu \cdot x \cdot y}{r^5} & \frac{3 \cdot \mu \cdot x \cdot z}{r^5} & 0 & 0 & 0 \\ \frac{3 \cdot \mu \cdot x \cdot y}{r^5} & \frac{\mu}{r^3}\left(\frac{3 \cdot y^2}{r^2} - 1\right) & \frac{3 \cdot \mu \cdot y \cdot z}{r^5} & 0 & 0 & 0 \\ \frac{3 \cdot \mu \cdot x \cdot z}{r^5} & \frac{3 \cdot \mu \cdot y \cdot z}{r^5} & \frac{\mu}{r^3}\left(\frac{3 \cdot z^2}{r^2} - 1\right) & 0 & 0 & 0 \end{bmatrix} \tag{1.10}$$

Now that the fundamental expressions are outlined, Figure 1.1 outlines the algorithm used to conduct the Least Squares estimate for a given initial vector.

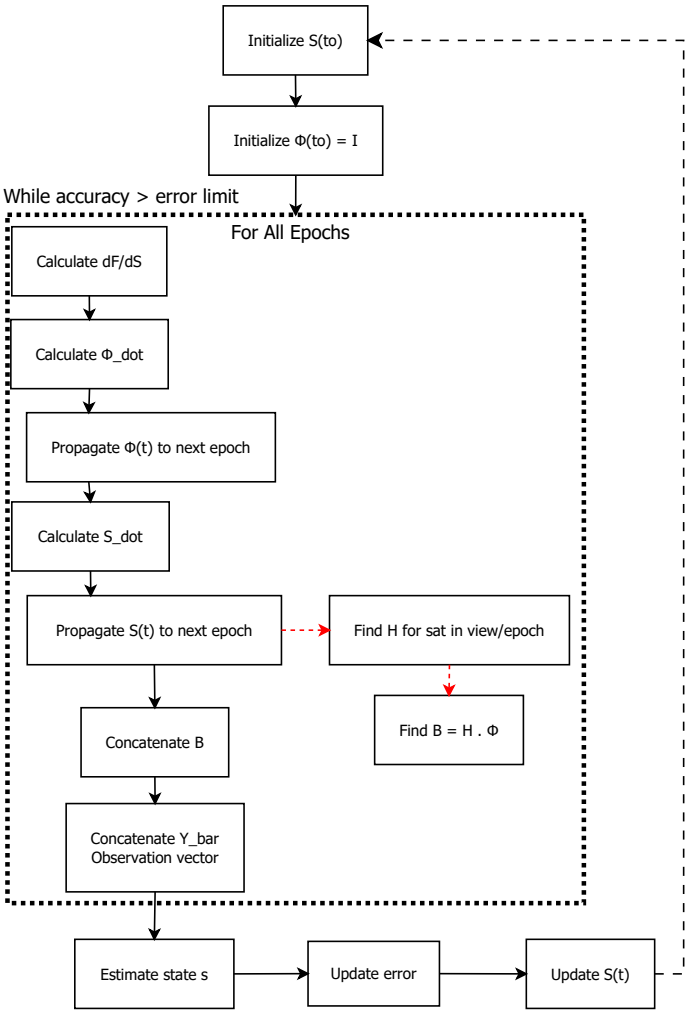


Figure 1.1: Flowchart illustrating steps undertaken to conduct LSQ for a given initial state vector.

As a first guess of initialisation of $\bar{S}$, the positions and velocities at the first epoch are taken from previous assignment. The velocity part is obtained via simple Euler scheme as indicated in the Matlab script in Appendix. Slight deviation is added to the initial vector to observe convergence. Equation 1.11, outlines the initial state vector $\bar{S}$ along with modified $\bar{S}$ to visualise convergence.

$$\bar{S}_0 = \begin{bmatrix} 1.94e6 \\ -5.84e6 \\ 2.93e6 \\ 0.001e6 \\ -0.0031e6 \\ -0.0069e6 \end{bmatrix}$$

Adding the vector [1000m 1000m 1000m 100 m/s 100 m/s 100m/s]' for visualising convergence. (1.11)

$$\bar{S}_0 = \begin{bmatrix} 1.93e6 \\ -5.81e6 \\ 3.0e6 \\ 0.001e6 \\ -0.0032e6 \\ -0.0068e6 \end{bmatrix}$$

1.2. Results

This section presents the results for the algorithm outlined in Figure 1.1. The algorithm initiates using the initialisation vector $\bar{S}$ defined in Equation 1.11. ϕ is chosen to be identity of size 6x6. The error limit is defined by the norm of x,y,z components of $\bar{S}$. A limit of 10^{-3} is used for convergence of iteration. $\frac{dF}{dS}$ is calculated with the aid of Equation 1.10. This is used to calculate $\dot{\phi}$. With the aid of Runge-Kutta integration, this is used to propagate $\phi(t)$ to next epoch. Which is further used to estimate $\dot{\bar{S}}$, later used to propagate $\bar{S}$ to next epoch. Updated information matrix H is calculated for all satellites in the view. The observation vector is noted for all epochs. The updated information matrix H is multiplied with the state transition matrix ϕ, and stored in B for all epochs. After looping through the epochs $H \cdot \phi$ for all epochs is appended in a combined information matrix of size 87x6. The observation vector is obtained with the aid of Equation 1.1 as indicated earlier. Now that the linear LSQ format indicated in Equation 1.1 is obtained, SVD method illustrated in Assignment 2 [2] is used to estimate s. Interested reader is referred to the script presented in the Appendix. Once s is estimated, it is added to $\bar{S}(t_0)$ to update the initialisation vector for next iteration. Since, SVD guided LSQ was implemented no singularity warnings were presented by Matlab. The results of algorithm is presented in Figure 1.2. Figure 1.2 illustrates the evolution of the epoch wise residual norm for increasing iteration numbers. The residual norm obtained using $\bar{Y}$ - $\hat{Y}$, where $\bar{Y}$ represents the vector observations and $\hat{Y}$ represents the vector of estimations. The error is defined as the norm of positional values of $\bar{S}$.

[2] Assignment 2: https://github.com/Alixir/Satellite-Orbit-Determination

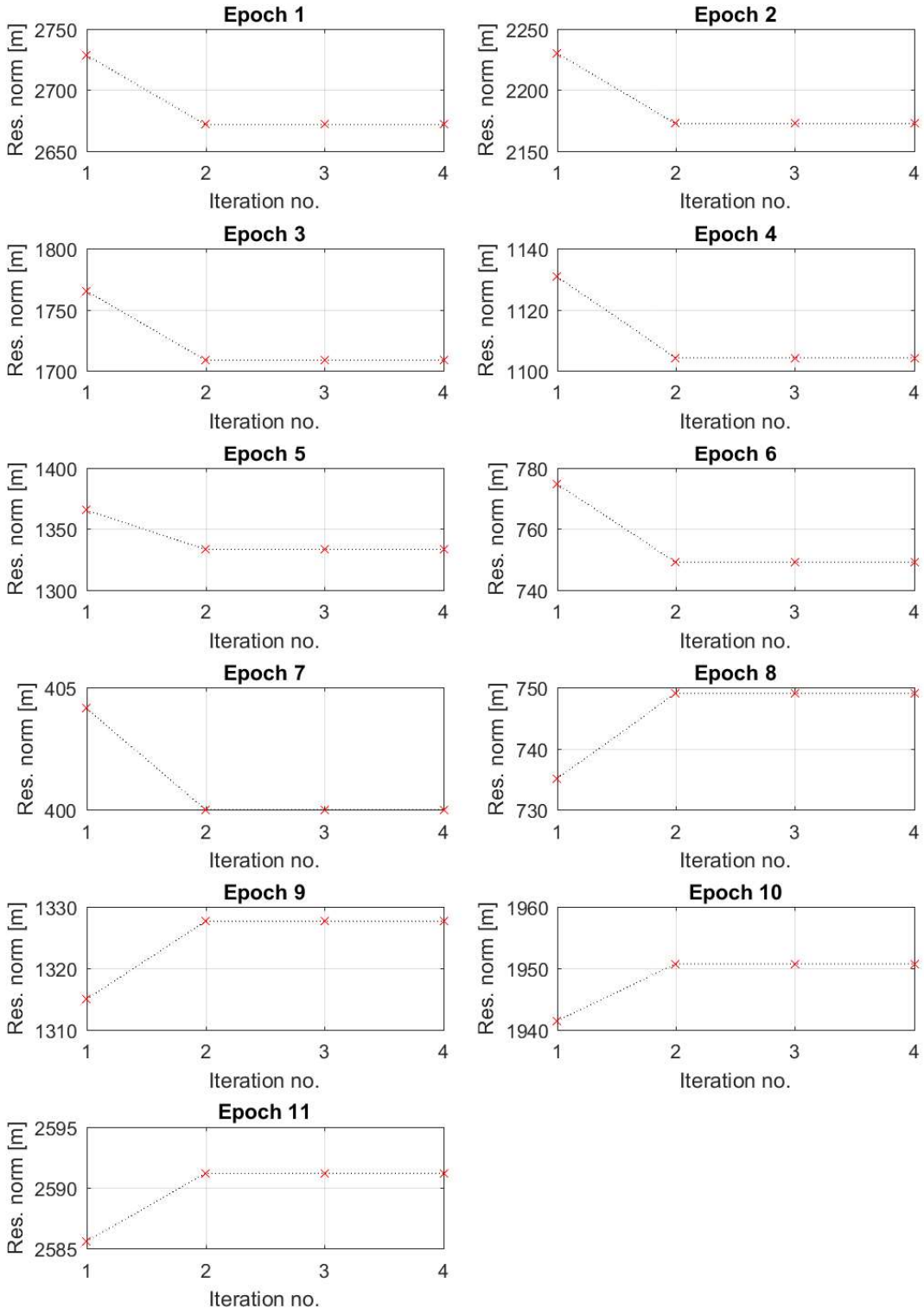


Figure 1.2: Evolution of epoch wise residual norm.

As seen in Figure 1.2, best results are obtained after 4 iterations. Note this depends on the definition of the error limit used. The error margin used here is 10^{-3} as indicated in the above discussion.

2

Estimating gravitational parameter

The aim of this chapter is to include a new dynamical parameter in the state vector $\bar{S}$ defined earlier in Equation 1.3. The dynamical parameter is the gravitational constant μ. An estimate for μ is obtained. Results of the satellite position are compared with the results obtained in Chapter 1.

2.1. Problem Definition

The new state vector after inclusion of μ is represented in Equation 2.1. The dimension of the initial state transition matrix ϕ is now updated to an identity matrix of size 7x7. The gravitational parameter μ defined in Equation 2.2.

$$\bar{S} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \mu \end{bmatrix} \tag{2.1}$$

$$V(r) = -\frac{\mu}{r} \tag{2.2}$$

Based on this definition of μ, the definition of information matrix H, F($\bar{S}$,t) and $\frac{dF}{dS}$ requires to be updated. Gravitational parameter μ, is no longer a constant for all calculations. It is rather updated every iteration. An initial value of $3 \cdot 10^{14}$ is chosen for μ. However, the choice of initial μ does not impact the solution, it always converges to the same value. This will be discussed later. For other elements of $\bar{S}$, same initialisation values from Equation 1.11 is used. The updated F is presented in Equation 2.3 and the corresponding update of $\frac{dF}{dS}$ due to the influence of $\frac{dF}{d\mu}$ is expressed in Equation 2.4. Finally, the updated matrix H is presented in Equation 2.5.

$$F(\bar{S}, t) = \dot{\bar{S}}(t) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ -\frac{\mu \cdot x}{r^3} \\ -\frac{\mu \cdot y}{r^3} \\ -\frac{\mu \cdot z}{r^3} \\ \dot{\mu} \end{bmatrix} \tag{2.3}$$

$$\frac{dF}{d\mu} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{-x}{r^3} \\ \frac{-y}{r^3} \\ \frac{-z}{r^3} \\ 0 \end{bmatrix}$$

$$\frac{dF(S,t)}{dS} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{\mu}{r^3}\left(\frac{3 \cdot x^2}{r^2} - 1\right) & \frac{3 \cdot \mu \cdot x \cdot y}{r^5} & \frac{3 \cdot \mu \cdot x \cdot z}{r^5} & 0 & 0 & 0 & -\frac{x}{r^3} \\ \frac{3 \cdot \mu \cdot x \cdot y}{r^5} & \frac{\mu}{r^3}\left(\frac{3 \cdot y^2}{r^2} - 1\right) & \frac{3 \cdot \mu \cdot x \cdot y}{r^5} & 0 & 0 & 0 & -\frac{y}{r^3} \\ \frac{3 \cdot \mu \cdot x \cdot z}{r^5} & \frac{3 \cdot \mu \cdot y \cdot z}{r^5} & \frac{\mu}{r^3}\left(\frac{3 \cdot z^2}{r^2} - 1\right) & 0 & 0 & 0 & -\frac{z}{r^3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.4}$$

$$H(t_{i+1}) = \begin{bmatrix} \frac{x - x_G}{\rho} & \frac{y - y_G}{\rho} & \frac{z - z_G}{\rho} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.5}$$

The algorithm is indicated with the aid of Figure 2.1. Note how the gravitational parameter μ is updated every iteration.

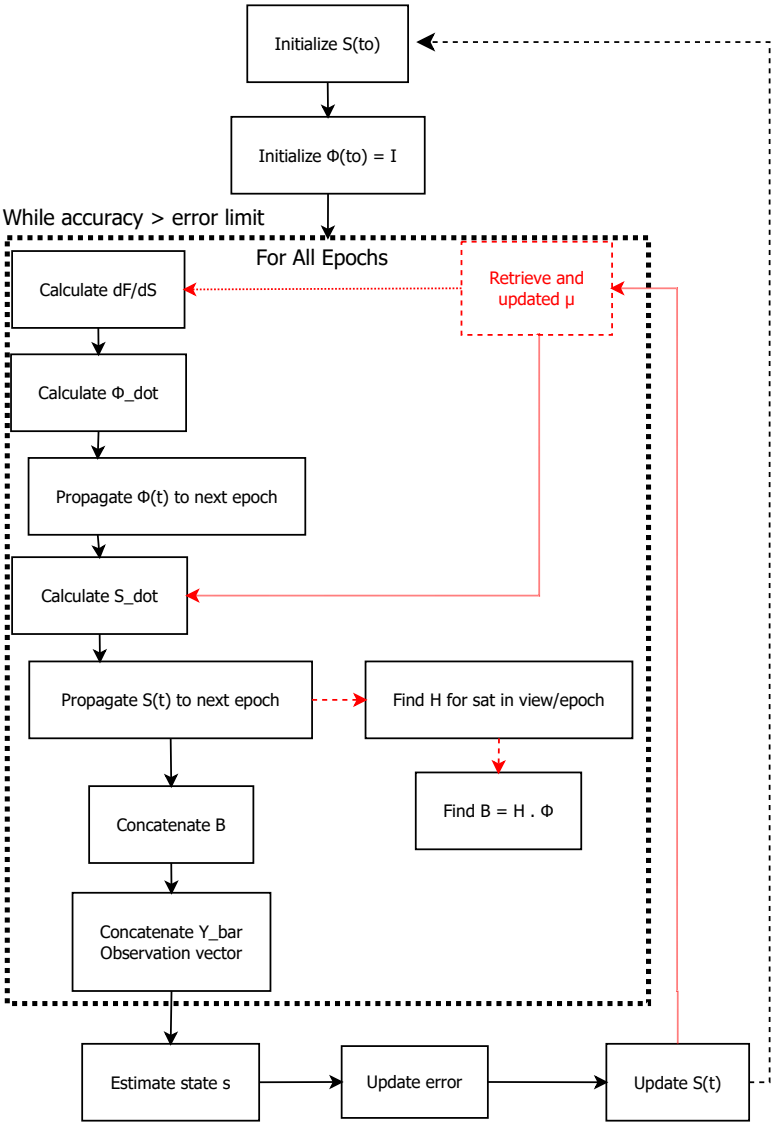


Figure 2.1: Flowchart illustrating steps undertaken to conduct LSQ for updated initial state vector including μ.

2.2. Results and comparison

The results of running the algorithm presented in Figure 2.1 is illustrated with the aid of Figure 2.2. Similar error limit of 10^{-3} is used, and the error is measured as the norm of the positional values of $\bar{S}$. The residual is again taken as the difference between observation vector $\bar{Y}$ and the estimation $\hat{Y}$. Again SVD guided LSQ is used for parameter estimation. Interest reader is referred to follow the script presented in Appendix.

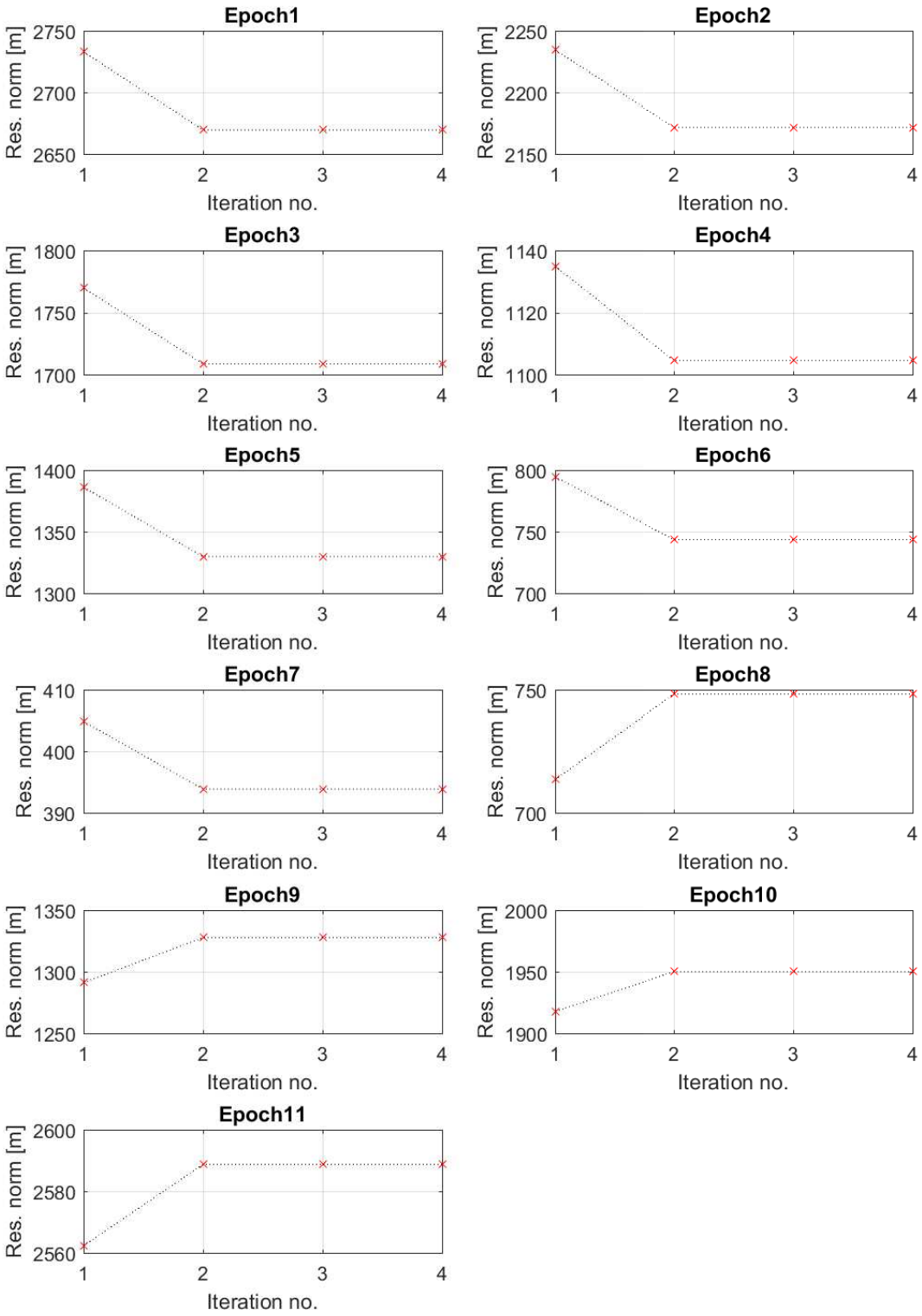


Figure 2.2: Evolution of epoch wise residual norm.

Comparing the results presented in Figures 1.1 and 2.1, both of the residual norms are seen to converge to the provided limit within 4 iterations. Table 2.1 illustrates the final residual norms for the case of 6 unknowns and 7 unknowns in the state vector. Overall, the final values for all epochs are quite close. Overall the residual

norms for state vector with 7 unknowns are lesser in magnitude compared to that of the 6 unknowns. But the difference is not significant.

Table 2.1: Residual behaviour at the end of iterations for unknown with 6 and 7 states.

Epoch	**Residual for 6 unknowns [m]**	**Residual for 7 unknowns [m]**
1	2671	2669
2	2172	2172
3	1709	1709
4	1104	1105
5	1333	1330
6	749.1	743.5
7	400	393.9
8	749.1	748.3
9	1328	1328
10	1951	1950
11	2591	2589

Table 2.2 is used to facilitate the comparison between the positional values from Assignment 2 and for different state vectors. Table 2.2 also illustrates the value of μ parameter obtained for state with 7 unknown parameters. The real value of μ for Earth is: 3.9860e14 (up to 4 decimal places) [1]. Comparing this with the value of μ = 3.9451e14 presented in Table 2.2, the difference from the real value is 1.03%.
Comparing the final position of the satellite from assignment 2 with the final positions obtained in this assignment, it can be observed that, the x, y and z position are fairly close for the case of 6 and 7 unknown state parameters. However, they are different from the values obtained in Assignment 2. Norm of the [x, y, z] for Assignment 2 positional values is: 6.8150e06m while that for states with 6 and 7 unknowns are: 6.8214e06m and 6.8216e06m consecutively. The differences in norm is quite significant (when measured in units of meter, 6400m and 6600m), this could be due to the fundamental differences in which the parameters are estimated. One conducts parameter estimation of $[\Delta x, \Delta y, \Delta z]$, while makes use of state transition matrix relations to conduct parameter estimation of $[\Delta x, \Delta y, \Delta z]$, but the primary initialiser is [x, y, z] along with $[\dot{x}, \dot{y}, \dot{z}]$ which are derived using based Euler integration technique with 10 time steps. Which might be considered to be quite significant.

Table 2.2: Initial and final states showing the change in position and velocities.

	Initial state	**6 unknown parameters**	**7 unknown parameters**
x [m]	1.939e06	2.0251e06	2.0251e06
y [m]	-5.838e06	-6.1198e06	-6.1199e06
z [m]	2.933e06	2.2314e06	2.2315e06
$\dot{x}$ **[m/s]**	984.2	718.2100	719.6624
$\dot{y}$ **[m/s]**	-3148.6	-2.4084e03	-2.4127e03
$\dot{z}$ **[m/s]**	-6859.2	-7.1933e03	-7.1915e03
μ [$m^3 s^{-2}$]	3e14	-	**3.9451e14**

[1] Gravitational Parameter: https://en.wikipedia.org/wiki/Standard_gravitational_parameter

3

Influence of initial state vector on results

The aim of this chapter is to discuss the influence of initial state vector on the results and final estimated values. First the case of 6 unknown states is considered in Section 3.1 followed by the case of 7 unknown state in Section 3.2. The effect is observed in blocks of position, velocity and gravitational parameters.

3.1. Initial state vector with position and velocity.

Table 3.1 outlines the initial state vector along with initial state vectors of increased/decreased position and velocity components. For increased position, the x,y and z positions are increased by 1e06. Since until 1e05 no difference was observed neither in the no. iterations nor the final convergence values. Similarly, for decreased positions, the x,y, and z component are decreased by 1e06. The final values are outlined Table 3.2. Increasing/decreasing the position by 1e06 only impacts the no. of iteration, the rest stays unchanged. Quite surprising how this has no impact of the final values. For increased/decreased velocity, the velocity components are increased/decreased by 1e04 until any consequence is observed. The final values for this is also outline in Table 3.2. Once again, there is no consequence on the final value. Only consequence is in the number of iterations required for convergence to a desired accuracy.

Table 3.1: Different values of states for initial state vector.

	Initial state vector, s_1	**Increased position**, s_2	**Decreased position**, s_3	**Increased velocity**, s_4	**Decreased velocity**, s_5
x [m]	1.9393e06	2.9393e06	9.3925e05	1.9393e06	1.9393e06
y [m]	-5.8377e06	-4.8377e06	-6.8377e06	-5.8377e06	-5.8377e06
z [m]	2.9332e06	3.9332e06	1.9332e06	2.9332e06	2.9332e06
$\dot{x}$ [m/s]	984.1881	984.1881	984.1881	1.0984e04	-9.0158e03
$\dot{y}$ [m/s]	-3.1486e03	-3.1486e03	-3.1486e03	6.8514e03	-1.3149e04
$\dot{z}$ [m/s]	-6.8592e03	-6.8592e03	-6.8592e03	3.1408e03	-1.6859e04

Table 3.2: Final values for different initial state vectors.

Final values	s_1	s_2	s_3	s_4	s_5
x [m]	2.0251e06	2.0251e06	2.0251e06	2.0251e06	2.0251e06
y [m]	-6.1198e06	-6.1198e06	-6.1198e06	-6.1198e06	6.1198e06
z [m]	2.2314e06	2.2314e06	2.2314e06	2.2314e06	2.2314e06
$\dot{x}$ [m/s]	718.2100	718.2100	718.2100	718.2100	718.2100
$\dot{y}$ [m/s]	-2.4084e03	-2.4084e03	-2.4084e03	-2.4084e03	-2.4084e03
$\dot{z}$ [m/s]	-7.1933e03	-7.1933e03	-7.1933e03	-7.1933e03	-7.1933e03
No. of iterations	4	5	5	5	5

3.2. Initial state vector with position, velocity and gravitational parameter

Analysis similar to section 3.1 is carried out here, with the exception that now the state vector is of dimension 7 rather 6. Which is the result of embedding a gravitational parameter term μ. As mentioned earlier in Section 2.1, initial choice of μ alone has no consequence on the final results. It always converges to the same value

within the same number of iterations. Thus for this section, we still analyse the results of varying position and velocity components in the state vector. Interested reader can run the script provided in the Appendix or Git repository. Initial state vectors are similar to that presented in Table 3.1 with the exception of an additional state, μ = 3e14. Initial conditions are tabulated in Table 3.3 while the final values are presented in Table 3.4.

Table 3.3: Different values of states for initial state vector.

	Initial state vector, s_1	**Increased position**, s_2	**Decreased position**, s_3	**Increased velocity**, s_4	**Decreased velocity**, s_5
x [m]	1.9393e06	2.9393e06	9.3925e05	1.9393e06	1.9393e06
y [m]	-5.8377e06	-4.8377e06	-6.8377e06	-5.8377e06	-5.8377e06
z [m]	2.9332e06	3.9332e06	1.9332e06	2.9332e06	2.9332e06
$\dot{x}$ [m/s]	984.1881	984.1881	984.1881	1.0984e04	-9.0158e03
$\dot{y}$ [m/s]	-3.1486e03	-3.1486e03	-3.1486e03	6.8514e03	-1.3149e04
$\dot{z}$ [m/s]	-6.8592e03	-6.8592e03	-6.8592e03	3.1408e03	-1.6859e04
μ [m^3s^{-2}]	3e14	3e14	3e14	3e14	3e14

Table 3.4: Final values for different initial state vectors.

Final values	s_1	s_2	s_3	s_4	s_5
x [m]	2.0251e06	2.0251e06	2.0251e06	2.0251e06	2.0251e06
y [m]	-6.1199e06	-6.1199e06	-6.1199e06	-6.1199e06	-6.1199e06
z [m]	2.2315e06	2.2315e06	2.2315e06	2.2315e06	2.2315e06
$\dot{x}$ [m/s]	719.6224	719.6224	719.6224	719.6224	719.6224
$\dot{y}$ [m/s]	-2.4127e03	-2.4127e03	-2.4127e03	-2.4127e03	-2.4127e03
$\dot{z}$ [m/s]	-7.1915e03	-7.1915e03	-7.1915e03	-7.1915e03	-7.1915e03
μ[m^3s^{-2}]	3.9451e14	3.9451e14	3.9451e14	3.9451e14	3.9451e14
No. of iterations	4	5	5	5	5

To conclude, increasing the order of magnitude of increments added to position/velocity parts for both cases, increases the no. of iteration required to reach the same final value. However, increasing it higher than a limit causes the results to not converge anymore.

4

Standard deviation of gravitational parameter

This chapter focuses on calculating the standard deviation σ, of μ parameter. It is assumed that the observation is uncorrelated and that they have a standard deviation of 5m. Thus co-variance matrix of observation vector is identity matrix of size 87x87, with 5^2 on the diagonal. Once the information matrix is obtained the covariance of state vector can be related to the covariance matrix of the observation vector using the expression outlined in Equation 4.1.

$P_{xx} = (A^T P_{yy}^{-1} A)^{-1}$

Where the term on the left indicates the covariance matrix of the states.

Since P_{yy} is a diagonal matrix, the above expression be simplified to:

$P_{xx} = 5^2 \cdot (A^T A)^{-1}$

Where, A is the concatenated information matrix with full rank.

However, $A^T A$ results in loss of rank.

The loss of rank induces singularity warnings in Matlab and might result in numerical errors.

Unlike many other things in life, there is a quick fix for it.

Commutative law of matrix multiplication dictates the following iff both A and B are invertible:

$(AB)^{-1} = B^{-1} A^{-1}$

Since A is not a square matrix, it cannot be inverted. However, Moore–Penrose inverse can still be applied.

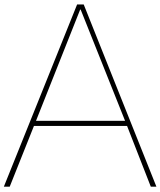
This results in the following relation:

$P_{xx} = 5^2 \cdot A^{+} (A^T)^{+}$

Where the superscript ” + ” denotes it being a pseudo-inverse rather than an inverse.

(4.1)

Based on Equation 2.1 the standard deviation of the gravitation parameter σ_μ is obtained taking the square root of final diagonal element of the matrix P_{xx}. The final diagonal element of P_{xx} is 8.0273e+21. Taking the square root of this gives, σ_μ = 8.9595e+10. Adding this standard deviation to the value of μ obtained via parameter estimation results in μ 3.9460e+14. The updated value of μ after estimation and inclusion of σ_μ is still 1.0045% off compared to the real value. However, it is definitely better than previous estimation which was 1.03% off. It is acceptable but accurate. Esp, since the assumption that observations are uncorrelated and have a standard deviation of 5m, is not entirely true and can easily be observed by checking the covariance matrix of P_{yy}



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 %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 3: Least square estimation of an initial state vector
% and a dynamic parameter
%%=================================================
%% 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;

%% Part 1a
% 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 last 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;
% s0 = [ 0.1 0.1 0.1 0.1 0.1 0.1];
% Standard Gravitational parameter Earth
mu = 3.986004418e14; % [ m^3 s^-2]
% Loop initializations
diff4 = norm(s0); % Current error resolution
iter4 = 0; % Current iteration step
diff_iter = [];
Svect = [];
res = [];
while diff4 >= 10^-3
% while iter4 <= 10
    A4 = [];
    y_bar4 = [];
    iter4 = iter4 +1;
    n = 0;
    phi = eye(6,6); % Initializing state transition matrix phi
    S = s0;
    Svect = [Svect; S];
    % For all epochs
    for e=1:epoch
        clear H dx4 dy4 dz4 drho4;
        r = norm(S(1,1:3)); % 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;
        % 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 ]';
        % Updating S
        S = S + S_dot.*dt;

        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
        B = H*phi; % New information matrix.
        % Accumalating primary information matrix from all the epochs into
        % one matrix
%         A4 = blkdiag(A4,B);
        A4 = vertcat(A4,B);
        % Accumulating observation for all epochs
        % y_bar4 = [y_bar4;transpose(C1(e,1:no_sat(e)) + ccGPS(e,1:no_sat(e)) - drho_rcc) ] ;
        % With inclusion of receiver clock correction from previous assignment
        y_bar4 = [y_bar4;transpose(C1(e,1:no_sat(e)) + (ccGPS(e,1:no_sat(e))- final_clock_error*c) - drho4) ] ;
        n= n+3;
    end

    % rank check to check for information loss in the information matrix
    rank4(iter4) = rank(A4);

    % LSQ with the aid of SVD
    [U4,S4,V4] = svd(A4,'econ'); % Singular value decomposition of A, to avoid singularity errors or solution manifold due to rank deficit
```

```
    S_inv_diag4 = [];
    for nn=1:length(S4)
        S_inv_diag4 = [S_inv_diag4,inv(S4(nn,nn))];
    end
    S_inv4 = diag(S_inv_diag4);

    % Estimation of small letter case, s. s = x_hat4
    x_hat4 = V4*S_inv4*U4'*y_bar4; % Parameter estimation via SVD aided LSQ
    y_hat4 = A4*x_hat4; % Measured vector

    res = [res, norm(y_bar4-y_hat4)];

    nn = 0;
    buf = [];
    for ee = 1:epoch
        buf = [buf,nn+1,sum(no_sat(1:ee))];
        res4 = y_bar4(nn+1:sum(no_sat(1:ee))) - y_hat4(nn+1:sum(no_sat(1:ee))); % residual in
    meters
        res_norm4(ee,iter4) = norm(res4);% Epoch wise norm of the residual vector of all satellites
        nn = sum(no_sat(1:ee));
    end
    % Update error and state
    diff4 = norm(x_hat4(1:3));
    diff_iter = [diff_iter;diff4];
    s0 =s0 + x_hat4';
    if iter4>=100
        disp('Iteration limit of 100 exceeds');
        break
    end
end

%% Plotting the behaviour of epochwise residual, inclusion of clock correction before batch LSQ for
     all epochs
for ee = 1:epoch
        figure(31)
        subplot(6,2,ee)
        plot(1:length(res_norm4(ee,:)),res_norm4(ee,:),'rx');
        hold on
        plot(1:length(res_norm4(ee,:)),res_norm4(ee,:),'k:');
        hold off
        title(['Epoch ' num2str(ee)]);
        ylabel('Res. norm [m]');
        xlabel('Iteration no.');
        grid on
end

nn = 0;
for ee = 1:epoch
        yb =  y_bar4(nn+1:sum(no_sat(1:ee)),1)';
        yh = y_hat4(nn+1:sum(no_sat(1:ee)),1)';
        nn = sum(no_sat(1:ee));
        figure(32)
        subplot(6,2,ee)
        plot(1:length(yb),yb,'r-x');
        hold on
        plot(1:length(yh),yh,'b-*');
        hold off
        title(['Epoch ' num2str(ee)]);
        ylabel('Range');
        xlabel('Sat no.');
        grid on
end
%% Part b
% 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 last 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, 3e14] + [ 0 0 0 0 0 0 0 ];
% Shifting initial vector slightly to visualize convergence.
s0 = s0_buf ;
% s0 = [ 0.1 0.1 0.1 0.1 0.1 0.1];
% Standard Gravitational parameter Earth
% mu = 3.986004418e14; % [ m^3 s^-2]
% Loop initializations
diff4 = norm(s0); % Current error resolution
iter4 = 0; % Current iteration step
diff_iter = [];
Svect = [];
res = [];
while diff4 >= 10^-3
% while iter4 <= 10
    A4 = [];
    y_bar4 = [];
    iter4 = iter4 +1;
    phi = eye(7,7); % Initializing state transition matrix phi
    S = s0;
    Svect = [Svect; S];
    % For all epochs
    mu = S(7);
    for e=1:epoch
        clear H dx4 dy4 dz4 drho4;
        r = norm(S(1,1:3)); % 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 0 1 0 0 ; 0 0 0 0 0 1 0;...
                  (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, -S(1)/r^3;...
                  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, -S(2)/r^3   ;...
                  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, -S(3)/r^3; ...
                  0, 0, 0, 0, 0, 0, 0];

        % Calculating phi_dot
        phi_dot = dF_dS*phi;
        % Updating phi
        phi = phi + phi_dot*dt;
        % 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; 0 ]';
        % Updating S
        S = S + S_dot.*dt;

        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 0]; % Assembling
    epochwise information matrix.
        end
        B = H*phi; % New information matrix.
        % Accumalating primary information matrix from all the epochs into
        % one matrix
        A4 = vertcat(A4,B);
        % Accumulating observation for all epochs
        % y_bar4 = [y_bar4;transpose(C1(e,1:no_sat(e)) + ccGPS(e,1:no_sat(e)) - drho_rcc) ] ;
        % With inclusion of receiver clock correction from previous assignment
        y_bar4 = [y_bar4;transpose(C1(e,1:no_sat(e)) + (ccGPS(e,1:no_sat(e))- final_clock_error*c)
    - drho4) ] ;
    end

    % rank check to check for information loss in the information matrix
```

```
326     rank4(iter4) = rank(A4);
327 
328     % LSQ with the aid of SVD
329     [U4,S4,V4] = svd(A4,'econ'); % Singular value decomposition of A, to avoid singularity errors
        or solution manifold due to rank deficit
330     S_inv_diag4 = [];
331     for nn=1:length(S4)
332         S_inv_diag4 = [S_inv_diag4,inv(S4(nn,nn))];
333     end
334     S_inv4 = diag(S_inv_diag4);
335 
336     % Estimation of small letter case, s. s = x_hat4
337     x_hat4 = V4*S_inv4*U4'*y_bar4; % Parameter estimation via SVD aided LSQ
338     y_hat4 = A4*x_hat4; % Measured vector
339 
340     res = [res, norm(y_bar4-y_hat4)];
341 
342     nn = 0;
343     buf = [];
344     for ee = 1:epoch
345         buf = [buf,nn+1,sum(no_sat(1:ee))];
346         res4 = y_bar4(nn+1:sum(no_sat(1:ee))) - y_hat4(nn+1:sum(no_sat(1:ee))); % residual in
        meters
347         res_norm4(ee,iter4) = norm(res4);% Epoch wise norm of the residual vector of all satellites
348         nn = sum(no_sat(1:ee));
349     end
350     % Update error and state
351     diff4 = norm(x_hat4(1:3));
352     diff_iter = [diff_iter;diff4];
353     s0 =s0 + x_hat4';
354     if iter4>=100
355         disp('Iteration limit of 100 exceeds');
356         break
357     end
358 end
359 
360 %% Plotting the behaviour of epochwise residual, inclusion of clock correction before batch LSQ for
        all epochs
361 for ee = 1:epoch
362         figure(33)
363         subplot(6,2,ee)
364         plot(1:length(res_norm4(ee,:)),res_norm4(ee,:),'rx');
365         hold on
366         plot(1:length(res_norm4(ee,:)),res_norm4(ee,:),'k:');
367         hold off
368         title(['Epoch' num2str(ee)]);
369         ylabel('Res. norm [m]');
370         xlabel('Iteration no.');
371         grid on
372 end
373 
374 nn = 0;
375 for ee = 1:epoch
376         yb =  y_bar4(nn+1:sum(no_sat(1:ee)),1)';
377         yh = y_hat4(nn+1:sum(no_sat(1:ee)),1)';
378         nn = sum(no_sat(1:ee));
379         figure(34)
380         subplot(6,2,ee)
381         plot(1:length(yb),yb,'r-x');
382         hold on
383         plot(1:length(yh),yh,'b-*');
384         hold off
385         title(['Epoch ' num2str(ee)]);
386         ylabel('Range');
387         xlabel('Sat no.');
388         grid on
389 end
390 
391 %% Part C Standard deviation estimation.
392 % Assumed standard deviation of observation vector
393 std_y = 5;
394 % Observation vector co-variance matrix
395 Pyy = (std_y^2)*eye(87,87);
396 % State co-variance matrix
397 Pxx = pinv(A4)*Pyy*pinv(A4');
```

```
398 % Standard deviation of gravitational parameter
399 std_mu = sqrt(Pxx(end,end));
400 % mu combined with its standard deviation
401 mu_with_std = s0(7)+std_mu;
402 % Percent difference between estimated and actual mu.
403 mu_percent_diff = ((3.986004419*10^(14) - mu_with_std)/(3.986004419*10^(14)) )*100;
```