AE 502 Project 4

Linyi Hou | linyih2 | github.com/TigerHou2

May 15, 2021

Problem 1

Problem Statement: Given the simulated optical measurements, use the Gauss initial orbit determination (IOD) algorithm to estimate the Keplerian orbit elements for the Earth orbiting satellite. The simulated measurements are time from epoch (seconds), topocentric right ascension (radians), and topocentric declination (radians).

Assumptions:

- The observer (ground based telescope) is located at [lat,lon] = [$\pi/6$,0] radians.
- The radius of the Earth is 6378.1 km, and the gravitational parameter is 398600.44 km^3/s^2.
- The satellite is tracked three times for a duration of 40 seconds each. During this time, the telescope takes 5 images at 10-second intervals.
- The RA/DEC measurements are made in the same Earth-centered inertial (ECI) coordinates as those of the satellite.
- The Earth is a perfect sphere, and the satellite experiences two-body gravity.
- The Earth's spin rate is 7.2936e-5 rad/s.
- At epoch, the rotating Earth-centered Earth-frame (ECEF) is aligned with ECI.

A: The MATLAB functions used to perform Gaussian IOD are provided here. Specifically, `anglesg.m` and its dependencies are used. Since data is collected in three sets, we choose to maximize the time between measurements by choosing the first measurement in the first set, the middle measurement in the second set, and the last measurement in the last set.

To account for the rotation of the Earth, the location of the observer, which is given by its latitude and longitude in the ECEF, must be converted to the ECI frame by rotating the vector along the z-axis. The MATLAB function `rotz` was used to generate the rotation matrix. The conversion from ECEF to ECI is shown below:

$$\mathbf{R}_{ECI} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R\cos(lat)\cos(lon) \\ R\cos(lat)\sin(lon) \\ R\sin(lat) \end{bmatrix} \tag{1}$$

where R is the radius of the Earth. θ is the rotation of the ECEF relative to ECI, calculated as $\theta = \Omega t$, where Ω is the spin rate of the Earth and t is the time since epoch.

The data collected at all three measurement times, along with the corresponding observer positions, were plugged into the `anglesg.m` function, which yielded the position and velocity vectors of the spacecraft at the second measurement time. These values were then converted to the Keplerian orbit elements. The results are shown in Table 1.

Table 1: Keplerian elements for the center measurement using Gaussian IOD.

a	e	i	Ω	ω	f
19473.27 km	0.25935	11.07^o	53.17^o	110.13^o	241.22^o

Problem 2

Problem Statement: Use the results from Problem 1 to initiate a batch processor to more accurately estimate the state of the satellite. Report the ECI coordinates of the observed object at epoch. Propagate the resulting orbit using two-body dynamics and plot the result.

A: Begin by propagating the estimated orbit, obtained from Problem 1, backward in time to epoch using MATLAB's `ode45` function by assuming two-body dynamics. We will use epoch, or $t = 0$, to anchor the state transition matrix and the nominal trajectory. The equation of motion used is then simply

$$\ddot{\mathbf{r}} = -\frac{\mu\mathbf{r}}{r^3} \tag{2}$$

Next, begin the batch processing procedure by constructing the nominal trajectory $\mathbf{X}_0^*$, and the initial state transition matrix $\mathbf{\Phi}(t_0, t_0)$:

$$\mathbf{X}_0^* = [\mathbf{r}, \mathbf{v}]^T \tag{3}$$

$$\mathbf{\Phi}(t_0, t_0) = \mathbf{I}_6 \tag{4}$$

where $\mathbf{r}$ and $\mathbf{v}$ are the position and velocities estimated by back propagation of Gaussian IOD results from Problem 1 to t_0.

We were also provided with the error covariance matrix $\bar{\mathbf{P}}_0$, and the error matrix $\mathbf{R}$:

$$\bar{\mathbf{P}}_0 = \text{diag}(10^{10})_6 \tag{5}$$

$$\mathbf{R} = \text{diag}((1/3600 * \pi/180)^2)_2 \tag{6}$$

The objective of the batch processor is to solve the normal equations of the following form:

$$(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} + \bar{\mathbf{P}}_0^{-1})\,\hat{\mathbf{x}}_0 = \mathbf{H}^T\mathbf{R}^{-1}\mathbf{y} + \bar{\mathbf{P}}_0^{-1}\bar{\mathbf{x}}_0 \tag{7}$$

In the above equation, $\mathbf{H}$ is the mapping matrix, $\bar{\mathbf{x}}_0$ is the difference between the true trajectory and the reference trajectory, while $\hat{\mathbf{x}}_0$ is the best estimate of that difference. The objective of batch processing is to minimize $\hat{\mathbf{x}}_0$.

For each iteration of batch processing, we will iterate through the observations and accumulate observation data to build up the normal equations. The normal equations are solved at the end of each iteration tp find $\hat{\mathbf{x}}_0$, after which the nominal trajectory is updated, and the next iteration begins if $\hat{\mathbf{x}}_0$ has not converged.

To initialize the process, let $\Lambda = \mathbf{0}_{6\times6}$ and $N = \mathbf{0}_{6\times1}$. The iteration begins at epoch, and integrates the spacecraft state from epoch to the next observation time. Simultaneously, the $\mathbf{A}$ matrix is formed, which is the EOM for the state transition matrix $\mathbf{\Phi}$. The $\mathbf{A}$ matrix is defined as

$$\mathbf{A}(t) = \frac{\partial F(\dot{\mathbf{X}}^*(t), t)}{\partial \mathbf{X}^*} \tag{8}$$

where F is the force model for the orbit, which in this case is simply Eq. (2).

$\mathbf{\Phi}$ is converted to a 36×1 vector to be integrated along with $\mathbf{X}^*$ using `ode45`, following the equation

$$\dot{\mathbf{\Phi}}(t, t_0) = \mathbf{A}(t)\mathbf{\Phi}(t_{i-1}, t_0) \tag{9}$$

Following integration, several states are updated as shown below:

$$\tilde{\mathbf{H}}_i = \frac{\partial G(\mathbf{X}^*, t_i)}{\partial \mathbf{X}^*} \tag{10}$$

$$\mathbf{y}_i = \mathbf{Y}_i - G(\mathbf{X}_i^*, t_i) \tag{11}$$

$$\mathbf{H}_i = \tilde{\mathbf{H}}_i\mathbf{\Phi}(t_i, t_0) \tag{12}$$

$$\Lambda = \Lambda + \mathbf{H_i}^T\mathbf{R}_i^{-1}\mathbf{H}_i \tag{13}$$

$$N = N + \mathbf{H_i}^T\mathbf{R}_i^{-1}\mathbf{y}_i \tag{14}$$

where $\mathbf{Y}_i$ is the observation data (right ascension, declination), and G is the right ascension and declination of the observer according to the nominal trajectory:

$$P_x = r_x - R_x \tag{15}$$

$$P_y = r_y - R_y \tag{16}$$

$$P_z = r_z - R_z \tag{17}$$

$$\mathbf{G} = [\text{atan2}\,(P_y, P_x), \arcsin\,(P_z/P)]^T \tag{18}$$

After all observations are processed, $\hat{\mathbf{x}}_0$ is found by solving

$$\Lambda\hat{\mathbf{x}}_0 = N \tag{19}$$

If $\hat{\mathbf{x}}_0$ is not within tolerance, the next iteration begins by setting the following:

$$\mathbf{X}_0^* = mbX_0^* + \hat{\mathbf{x}}_0 \tag{20}$$

$$\bar{\mathbf{x}}_0 = \bar{\mathbf{x}}_0 - \hat{\mathbf{x}}_0 \tag{21}$$

$$\Lambda = \bar{\mathbf{P}}_0^{-1} \tag{22}$$

$$N = \bar{\mathbf{P}}_0^{-1}\bar{\mathbf{x}}_0 \tag{23}$$

Using the above process, the Keplerian elements at epoch were found:

Table 2: Keplerian elements at epoch using batch processing.

a	e	i	Ω	ω	f
19999.994 km	0.300	11.46^o	57.30^o	114.59^o	233.09^o

And the ECI state is

$$r_0 = [15625.22, 15741.83, -941.14] \text{ km}$$
$$v_0 = [-3.439, 1.878, 0.792] \text{ km/s}$$

Finally, the orbit obtained from propagating the ECI state is shown in

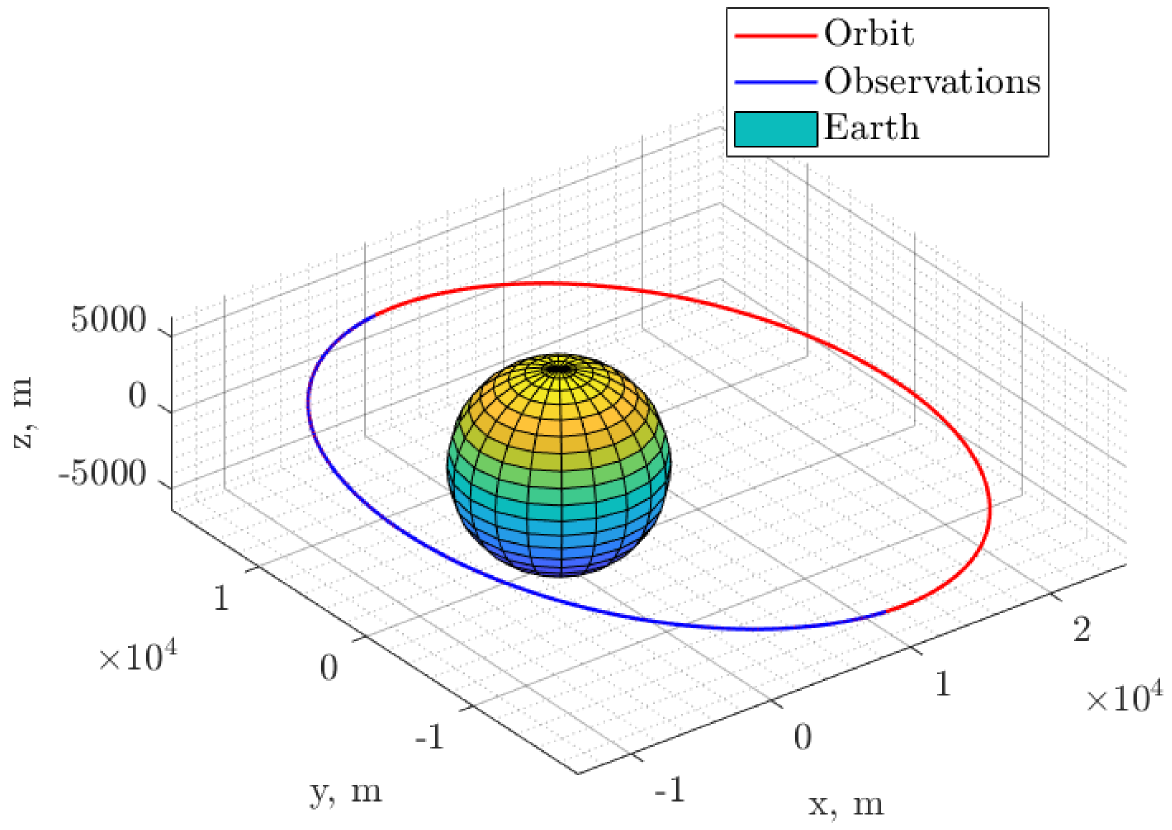

Figure 1: Orbit of the spacecraft estimated by batch processing of observation data.

Appendix A: Project Code

```
close all
clear;clc
addpath('vallado')

%% parse orbit data

dat = importdata('project4_data.txt');

TE = dat(:,1); % seconds
RA = dat(:,2); % radians
DE = dat(:,3); % radians

%% constants and settings

```

```matlab
mu = 398600.44; % km^3/s^2
R = 6378.1; % km
lat = pi/6;
lon = 0;
omega = rad2deg(7.2936e-5);
site = R * [cos(lat)*cos(lon), cos(lat)*sin(lon), sin(lat)]';
options = odeset('RelTol',1e-12,'AbsTol',1e-12);

%% problem 1

% we take the 1st measurement in each track
% indices = 1 + 5*(0:2);
indices = [1, 8, 15];
te_vec = TE(indices);
ra_vec = RA(indices);
de_vec = DE(indices);
site1 = rotz(te_vec(1)*omega) * site;
site2 = rotz(te_vec(2)*omega) * site;
site3 = rotz(te_vec(3)*omega) * site;

% use Gaussian IOD from Vallado
[r,v] = anglesg(de_vec(1),de_vec(2),de_vec(3),...
                ra_vec(1),ra_vec(2),ra_vec(3),...
                te_vec(1),te_vec(2),te_vec(3),...
                site1, site2, site3, R, mu, 1);

clc

[a,e,i,o,w,f] = Get_Orb_Params(r,v,mu);
disp( 'Problem 1:' )
disp(['    a = ' num2str(a) ' km'])
disp(['    e = ' num2str(norm(e))])
disp(['    i = ' num2str(rad2deg(i)) ' deg'])
disp(['    o = ' num2str(rad2deg(o)) ' deg'])
disp(['    w = ' num2str(rad2deg(w)) ' deg'])
disp(['    f = ' num2str(rad2deg(f)) ' deg'])
disp(' ')


%% problem 2
```

```

x0_hat = [r,v]';
x_star = x0_hat;
x0_bar = zeros(6,1);
STM = eye(6);
P0_bar = diag(1e10*ones(6,1));
Ri = diag([(1/3600*pi/180)^2,(1/3600*pi/180)^2]);
L = zeros(6,6);
N = zeros(6,1);

%% define EOMs for ode45

xdot = @(X) [X(4:6);-mu*X(1)/norm(X(1:3))^3; ...
                    -mu*X(2)/norm(X(1:3))^3; ...
                    -mu*X(3)/norm(X(1:3))^3; ];
Q = @(X) [      mu/norm(X(1:3))^5*(3*X(1)^2-norm(X(1:3))^2), ...
                3*mu*X(1)*X(2)/norm(X(1:3))^5, ...
                3*mu*X(1)*X(3)/norm(X(1:3))^5; ...
                ...
                3*mu*X(2)*X(1)/norm(X(1:3))^5, ...
                mu/norm(X(1:3))^5*(3*X(2)^2-norm(X(1:3))^2), ...
                3*mu*X(2)*X(3)/norm(X(1:3))^5; ...
                ...
                3*mu*X(3)*X(1)/norm(X(1:3))^5, ...
                3*mu*X(3)*X(2)/norm(X(1:3))^5, ...
                mu/norm(X(1:3))^5*(3*X(3)^2-norm(X(1:3))^2); ...
            ];
A = @(X) [   zeros(3), eye(3); ...
             Q(X), zeros(3)];
STM_dot = @(X,STM)  reshape( A(X) * reshape(STM,6,6), 36, 1);
combined_dot = @(combined) [xdot(combined(1:6));...
                            STM_dot(combined(1:6),combined(7:42))];

%% back propagate to epoch

[~,out] = ode45(@(t,y) xdot(y), [te_vec(2),0], x_star, options);
combined_orig = out(end,:)';
combined_orig = [combined_orig(1:6); reshape(STM,36,1)];
combined = combined_orig;

```

```
%% perform batch processing

len = length(TE);

while (1)

    tp = 0;

    for i = 1:len
        t = TE(i);
        Y = [RA(i),DE(i)]';

        [timestep,combined] = ...
            ode45(@(t,y) combined_dot(y), [tp,t], combined, options);
        combined = combined(end,:)';
        STM = reshape(combined(7:42),6,6);
        x = combined(1);
        y = combined(2);
        z = combined(3);

        R = rotz(t*omega) * site;
        Px = x - R(1);
        Py = y - R(2);
        Pz = z - R(3);
        P = norm([Px,Py,Pz]);

        Gy = [atan2(Py,Px),asin(Pz/P)]';
        Ht = [  -sin(Gy(1))*cos(Gy(1)) / Px, ...
                cos(Gy(1))^2 / Px, ...
                0, 0, 0, 0; ...
                -tan(Gy(2))*Px/P^2, ...
                -tan(Gy(2))*Py/P^2, ...
                cos(Gy(2))/P, ...
                0, 0, 0];

        yi = Y - Gy;
        Hi = Ht * STM;

        L = L + Hi' * inv(Ri) * Hi;
        N = N + Hi' * inv(Ri) * yi;

        tp = t;

    end

    x0_hat = L \ N;
    metric = norm(x0_hat);
    P0 = inv(L);

    if metric < 1e-6
        break
    end

    L = inv(P0_bar);
    N = inv(P0_bar) * x0_bar;

    combined_orig = combined_orig + [x0_hat;zeros(36,1)];
    combined = combined_orig;
    x0_bar = x0_bar - x0_hat;

end

r = combined(1:3);
v = combined(4:6);

%% propagate and show results

[~,out] = ode45(@(t,y) xdot(y), [TE(1),0], [r;v], options);
u = out(end,:)';

r0 = u(1:3);
v0 = u(4:6);
[a,e,i,o,w,f] = Get_Orb_Params(r0,v0,mu);
disp( 'Problem 2:' )
disp(['    a = ' num2str(a) ' km'])
disp(['    e = ' num2str(norm(e))])
disp(['    i = ' num2str(rad2deg(i)) ' deg'])
disp(['    o = ' num2str(rad2deg(o)) ' deg'])
disp(['    w = ' num2str(rad2deg(w)) ' deg'])
disp(['    f = ' num2str(rad2deg(f)) ' deg'])
```

```

[~,orbit] = ode45(@(t,y) xdot(y), 0:20:50000, [r;v], options);
[~,data] = ode45(@(t,y) xdot(y), 0:20:TE(15), [r;v], options);

% Earth
[X,Y,Z] = sphere;

figure(1)
hold on
plot3(orbit(:,1), orbit(:,2),orbit(:,3),'r','LineWidth',1.4)
plot3(data(:,1), data(:,2),data(:,3),'b','LineWidth',1.4)
% scatter3(0,0,0,10,'k','filled')
surf(X*6371,Y*6371,Z*6371)
hold off
view(3)
xlabel('x, m')
ylabel('y, m')
zlabel('z, m')
legend('Orbit','Observations','Earth','Location','best')
axis equal
setgrid
latexify(16,14,14)
```