ASE 389P.4 Methods of Orbit Determination
Homework 2: Orbit Propagation with Perturbations

Junette Hsin
Masters Student, Aerospace Engineering and Engineering Mechanics, University of Texas, Austin, TX 78712

New force models have been added to the orbit propagator created in Homework 1. The effects that these new forces have on the Keplerian orbit elements and the total specific energy were analyzed.

Problem 1

Integrate the equations of motion for one day using the same initial conditions as in Homework #1. However, now include the Earth's oblateness, i.e., the J_2 term. The equations of motion are still $\ddot{\underline{r}} = \nabla U$, but now U includes J_2:

$$U = U_{\text{point mass}} + U_{J_2} = \frac{\mu}{r}\left[1 - J_2\left(\frac{R_{\text{Earth}}}{r}\right)^2\left(\frac{3}{2}\sin^2(\phi) - \frac{1}{2}\right)\right] \tag{1}$$

Use the following (note that some of these values are different from Homework #1!):

$$\begin{aligned} J_2 &= 0.00108248 \\ \mu &= 398600.4 \text{ km}^3/\text{s}^2 \\ R_{\text{Earth}} &= 6378.145 \text{ km} \\ \phi &= \text{Latitude} \rightarrow \sin(\phi) = \frac{z}{r} \\ r &= \sqrt{x^2 + y^2 + z^2} \end{aligned}$$

Use the initial conditions from Homework #1, which are:

$$\underline{r} = -2436.45\hat{i} - 2436.45\hat{j} + 6891.037\hat{k}\ km \tag{2}$$

$$\underline{v} = \dot{\underline{r}} = 5.088611\hat{i} - 5.088611\hat{j} + 0.0\hat{k}\ km/s \tag{3}$$

The state at the final time is posted below to verify that your integrator is working properly.

Solution

Problem 1a

> Use the MATLAB symbolic toolbox to compute the Cartesian partial derivatives of U. Compute $\partial U/\partial x$ by hand (include the derivation in the write-up) and compare your results with MATLAB. The full derivation may be scanned and turned in as an appendix, but include the final equation in the typed write-up.

Solution

The derivation by hand is included in the appendix. The hand-derived and MATLAB results were the same. The final equation for $\partial U/\partial x$ is:

$$\partial U/\partial x = -J_2 \mu R_E^2 \left(\frac{3x}{2}\right) \frac{x^2 + y^2 - 4z^2}{(x^2 + y^2 + z^2)^{7/2}} \tag{1}$$

The MATLAB code used for the entire homework, including calculating the partial derivatives is included in the Appendix, but the snippet used to solve Problem 1a is shown here:

```
syms x y z
global mu RE J2

% constants
% mu = 398600.4;       % G * M1 * M2
% RE = 6378.145;       % Earth radius
% J2 = 0.00108248;     % J2

% for symbolic representation
syms mu RE J2

% radius
r = sqrt(x^2 + y^2 + z^2);

% U point mass
Up = mu/r;

% latitude
phi = asin(z/r);

% U J2
UJ2 = -mu/r * J2 * (RE/r)^2 * ( 3/2 * ( sin(phi) )^2 - 1/2 );

% gradient
d_UJ2 = gradient(UJ2, [x y z]);
d_UJ2 = simplify(d_UJ2);
```

Problem 1b

Plot the orbital elements a, e, i, Ω, ω, and T_p for one day at 20 second intervals, where T_p is the time of perigee passage and include the figure in your write-up. Be sure to label your axes and ensure that everything in each figure is readable. Using your insight from the two-body model, what conclusions can you draw about the J_2 effect on Keplerian orbital elements? Compute the period of the orbit ($P = 2\pi\sqrt{a^3/\mu}$) based on the initial state (Eqs (2)-(3)). How does the trend in the plots compare to the period?

Solution

Based on the initial state, the period of the orbit is:

$$P = 2\pi\sqrt{a^3/\mu} = 6.740270469266463e + 03s \tag{2}$$

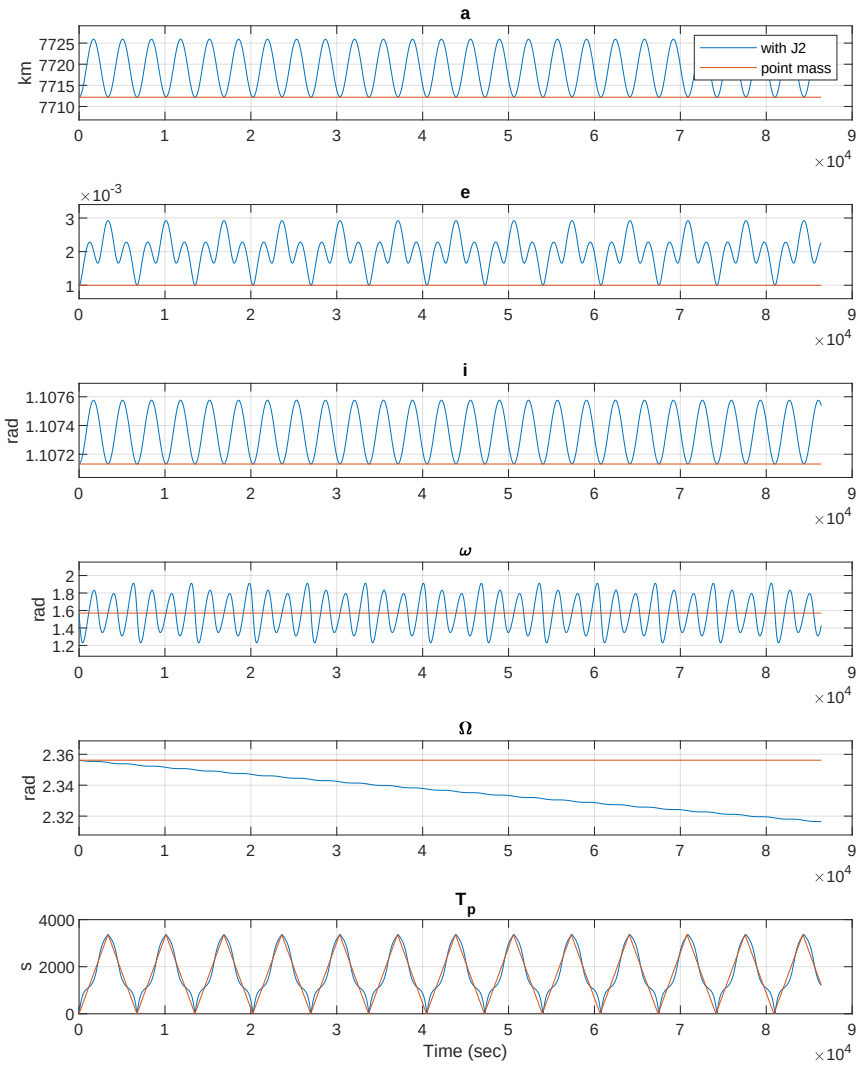

Fig. 1 Orbital elements of point mass and point mass with J2 zonal elements

Figure 1 shows that the Keplerian orbital elements are remarkably stable over time. Once J2 terms were introduced, the orbit elements exhibited periodic oscillating behavior for most of the orbital elements with the exception of right ascension of the ascending node. The motion of the orbiting body in the orbital plane is planar, therefore the body will move through lines of nodes at the ascending node, where it moves from -Z to +Z in the orbit frame (in other words, the body enters the northern hemisphere from the southern hemisphere at the crossing of the ascending node) [1]. The linearly decreasing ramp indicates that the J2 terms perturb a constant rate effect on the node.

The time of perigee passing has a period of approximately 6.74e3 seconds, which matches the calculation of the orbit period. The other orbit elements contain higher frequency content; perhaps these elements are likely sensitive to the J2 terms in geometric ways. In Figure 2, the orbits of the point mass and point mass with modeled J2 terms is shown. The point mass orbit is very stable while the J2 orbit perturbed more and more as the day went by.

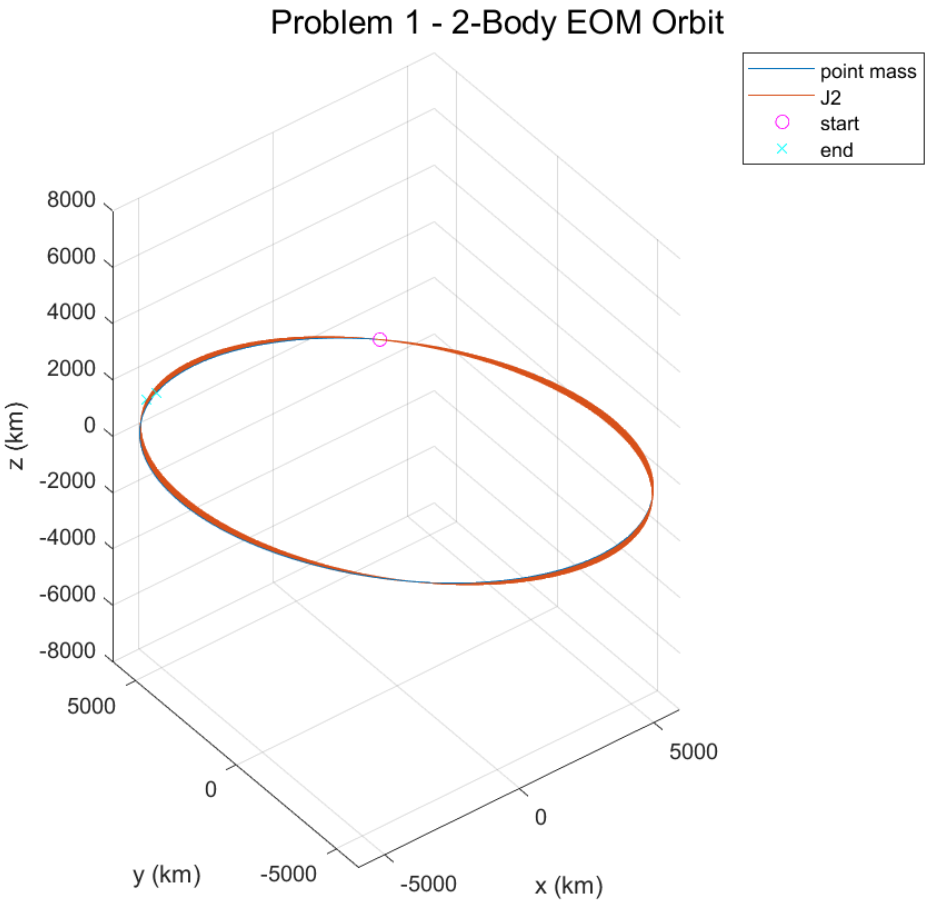

Fig. 2 2-body EOM Orbit of Point Mass with J2 term

Problem 1c

Compute the specific energy (energy/mass) and show that it is conserved around the orbit by plotting $dE = E(t) - E(t_0)$. Use the following equation with U from above:

$$E = \frac{v^2}{2} - U \tag{4}$$

Solution

The gravitational potential function (below) was added to the magnitude of the velocity squared over 2 for each timestep of the simulation, which was specified at 20 seconds, to compute the specific energy.

$$U = U_{pointmass} + U_{J_2} = \frac{\mu}{r}\left[1 - J_2\left(\frac{R_{Earth}}{r}^2\right)\left(\frac{3}{2}sin^2(\phi) - \frac{1}{2}\right)\right] \tag{3}$$

Then, to show that the specific energy is conserved, the specific energy at each timestep was subtracted with its initial value. The order of the magnitude on the plot is -12, which is a tiny change. The specific energy was around 25.815 km^2/s^2 for the duration of the simulation.

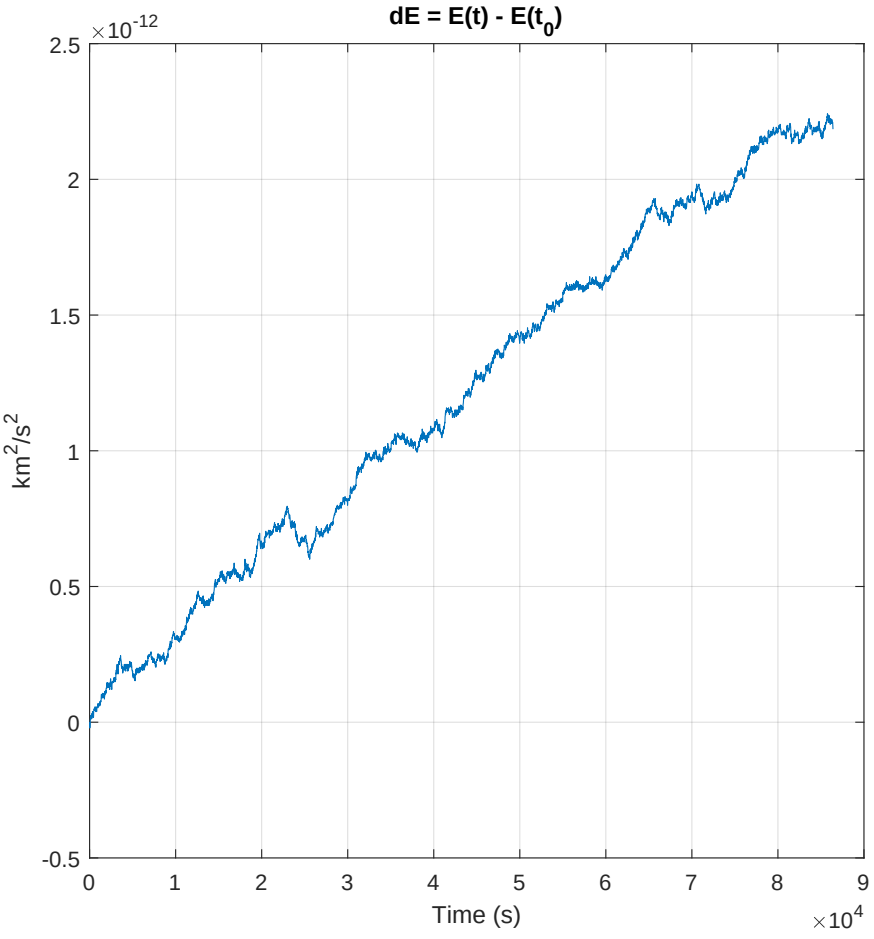

Fig. 3 Delta Specific Energy

Problem 1d

Compute h_k, the k-component of the angular momentum vector

$$\underline{h} = \underline{r} \times \underline{v} \tag{5}$$

and plot $dh_k = h_k(t) - h_k(t_0)$ to show that it remains constant.

Solution

The kth-component of the (specific) angular momentum vector was also incredibly stable for the duration of the simulation. The angular momentum vector is normal to the orbit plane and is a fundamental property of the orbit frame and the conservation of orbital energy. Changing orbital energy would lead to a changing specific angular momentum vector. Although the other components of the angular momentum vector experienced periodic behavior, the kth component remained within an order of magnitude of -10 to its initial value.

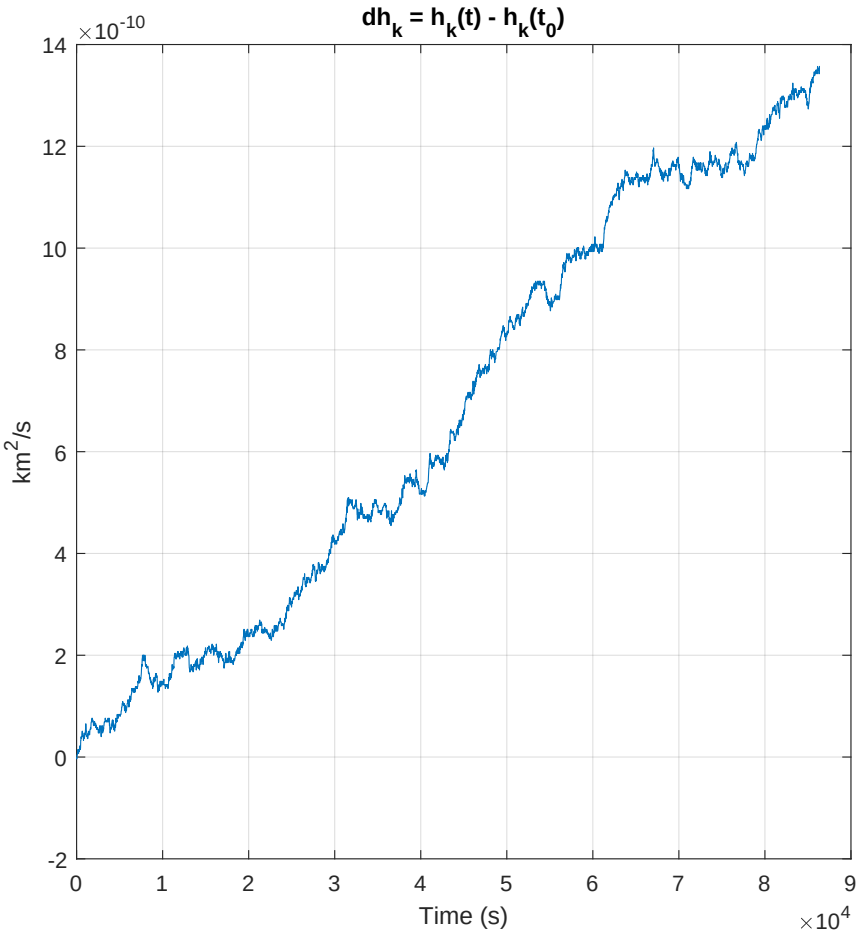

Fig. 4 Delta Angular Momentum

The plot for the k component of the angular momentum vector was computed and then differenced.

Problem 2

Integrate the equations of motion with the conditions given in Problem 1 that include the Earth point-mass, J_2, and now also drag. Use the following relationship for the acceleration due to drag:

$$\underline{\ddot{r}} = -\frac{1}{2} C_D \frac{A}{m} \rho_A V_A \underline{V}_A \tag{6}$$

where

$$\begin{aligned}
C_D &= 2.0 \\
A &= 3.6 \text{ m}^2 \\
m &= 1350 \text{ kg} \\
\rho_0 &= 4 \times 10^{-13} \text{ kg/m}^3 \\
r_0 &= 7298.145 \text{ km} \\
H &= 200.0 \text{ km} \\
\dot{\theta} &= 7.29211585530066 \times 10^{-5} \text{ rad/s} \\
\rho_A &= \rho_0 e^{-(r-r_0)/H} \\
\underline{V}_A &= \begin{bmatrix} \dot{x} + \dot{\theta} y \\ \dot{y} - \dot{\theta} x \\ \dot{z} \end{bmatrix} \\
V_A &= \sqrt{\left(\dot{x} + \dot{\theta} y\right)^2 + \left(\dot{y} - \dot{\theta} x\right)^2 + \dot{z}^2}
\end{aligned}$$

Make sure you have agreement in all of your units when computing the drag acceleration! Compare your result to the online values to ensure that your integrator is working properly.

Solution

Problem 2a

> Compute the specific energy at 20 second intervals and plot $dE = E(t) - E(t_0)$ (using Eq. (4)). Include the plot in your write-up. What can you infer from the plot? Is the total energy conserved? Why or why not?

Solution

From the plot shown below, the inference that total energy is conserved can confidently be made. Specific energy is related to the angular momentum of the system, which is conserved even with the presence of drag [2]. Drag is not a conservative force, and thus does not have a significant impact on the system energy.

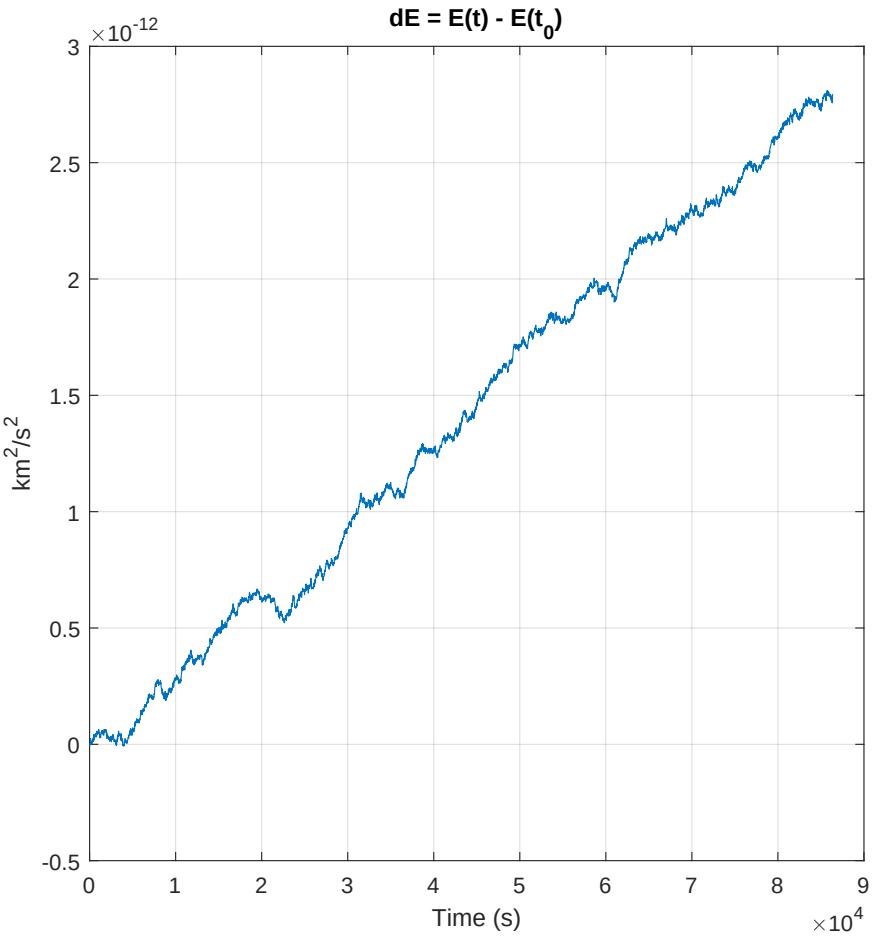

Fig. 5 Delta Specific Energy for Point Mass with J2 Terms and Drag

Problem 2b

> Compute the same Keplerian orbital elements generated in Problem 1b at 20 second intervals (a, e, i, Ω, ω, and T_p) and plot the differences in these elements from those computed in Problem 1. That is, generate time histories of each orbital element with and without drag and plot the differences, e.g., $a_{2B+J2+Drag} - a_{2B+J2}$. Include the results in your write-up. What can you observe in these plots? Which orbital elements are impacted by drag and how are they affected?

Results for the four cases (point mass, J2, drag, and J2 + drag) and their differences are shown below. To compare differences, there ended up being 6 delta cases analyzed:

1) point mass - J2
2) point mass - drag
3) point mass - (J2 + drag)
4) J2 - drag
5) J2 - (J2 + drag)
6) drag - (J2 + drag)

The J2 terms are extremely dominant and make it hard to discern the error between J2 and J2 + drag. The perturbation that drag ended up having on the final state of point mass is shown below (p-drag):

position	-0.009913492249325 $\hat{i}$	0.547834133612923 $\hat{j}$	-0.776935848989524 $\hat{k}$
velocity	-0.000683365863896 $\hat{i}$	0.000451816362101 $\hat{j}$	0.000329869465077 $\hat{k}$

$* 10^{-8}$

Drag seems to have a near-negligible effect on the motion of the body. The plot of the difference between the point mass and drag is almost zero for the duration of the simulation. Drag does seem to have a slightly larger influence on right ascension of the ascending node than the other elements for this batch of runs, but the effects become far more interesting in the orbit elements plot between J2 and drag. The time of perigee passing appears to be the most sensitive to drag; slight perturbations can be observed in the plots below. Overall, the J2 effects are clearly dominant in Figure 8.

Solution

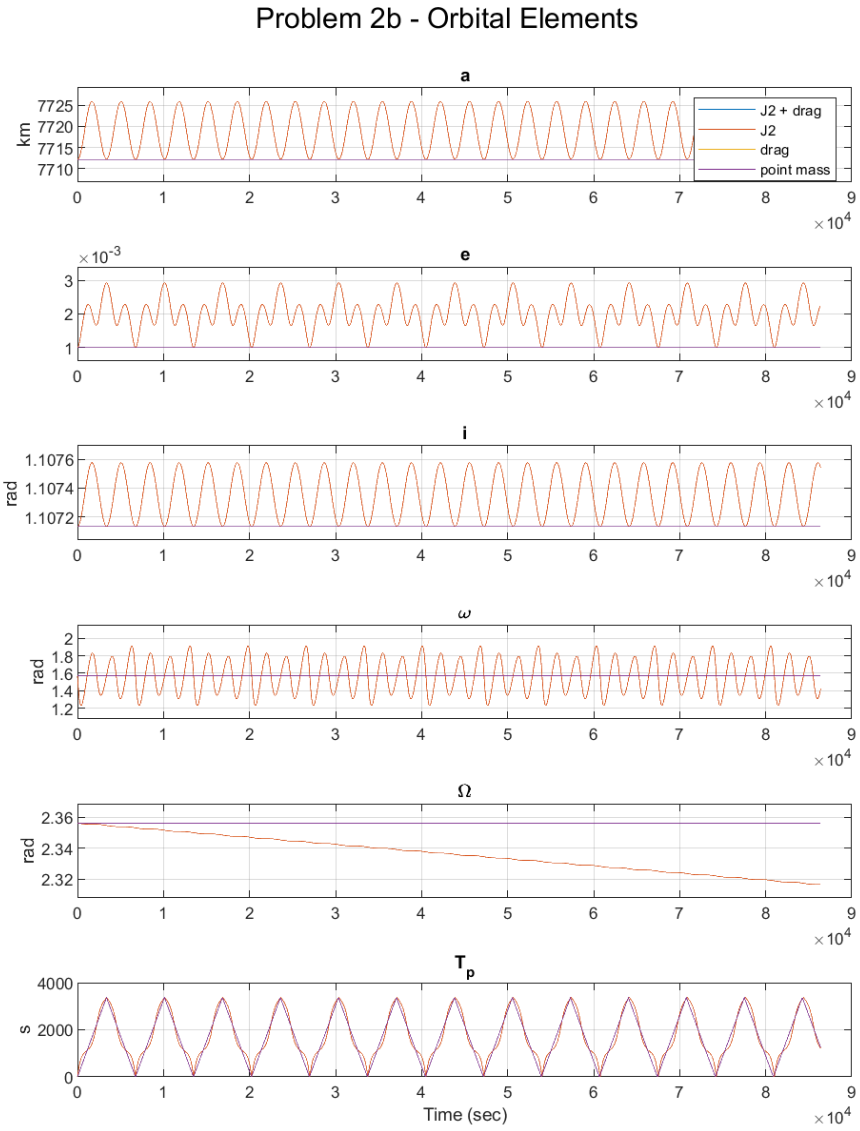

Fig. 6 Orbital Elements for Point Mass with J2 Terms and Drag

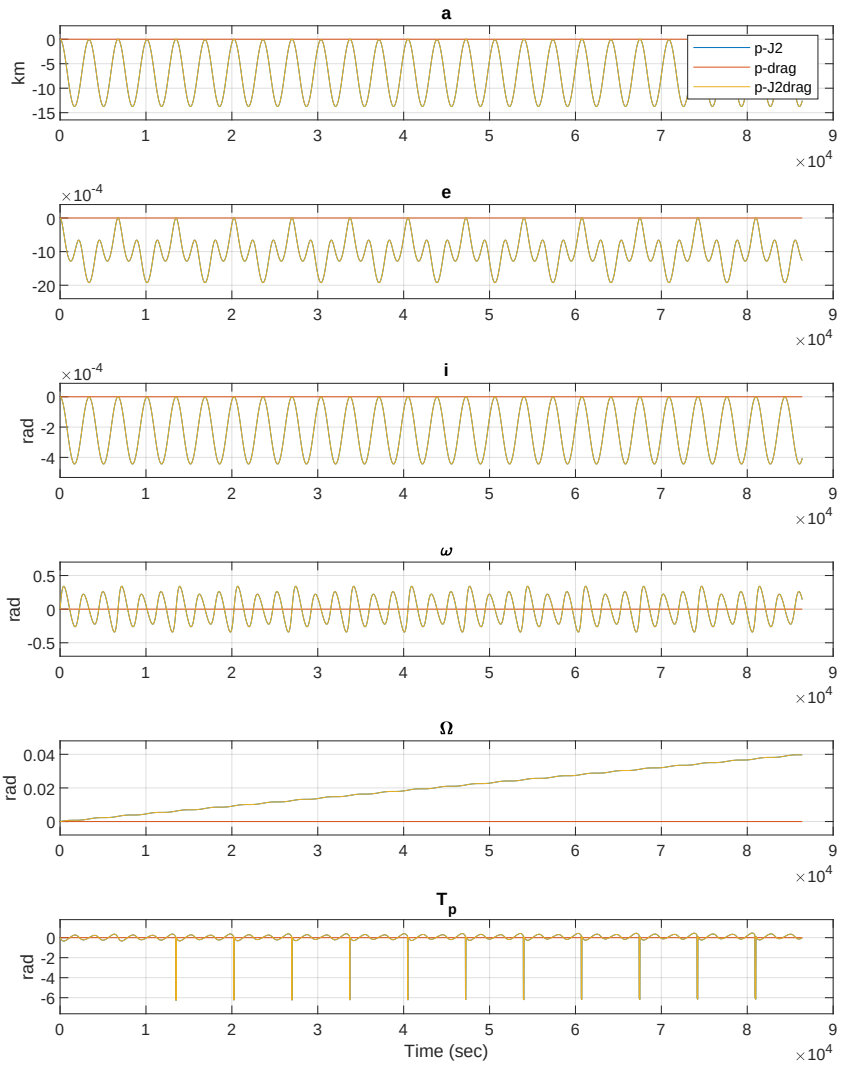

Fig. 7 Orbital Elements Point Mass Difference

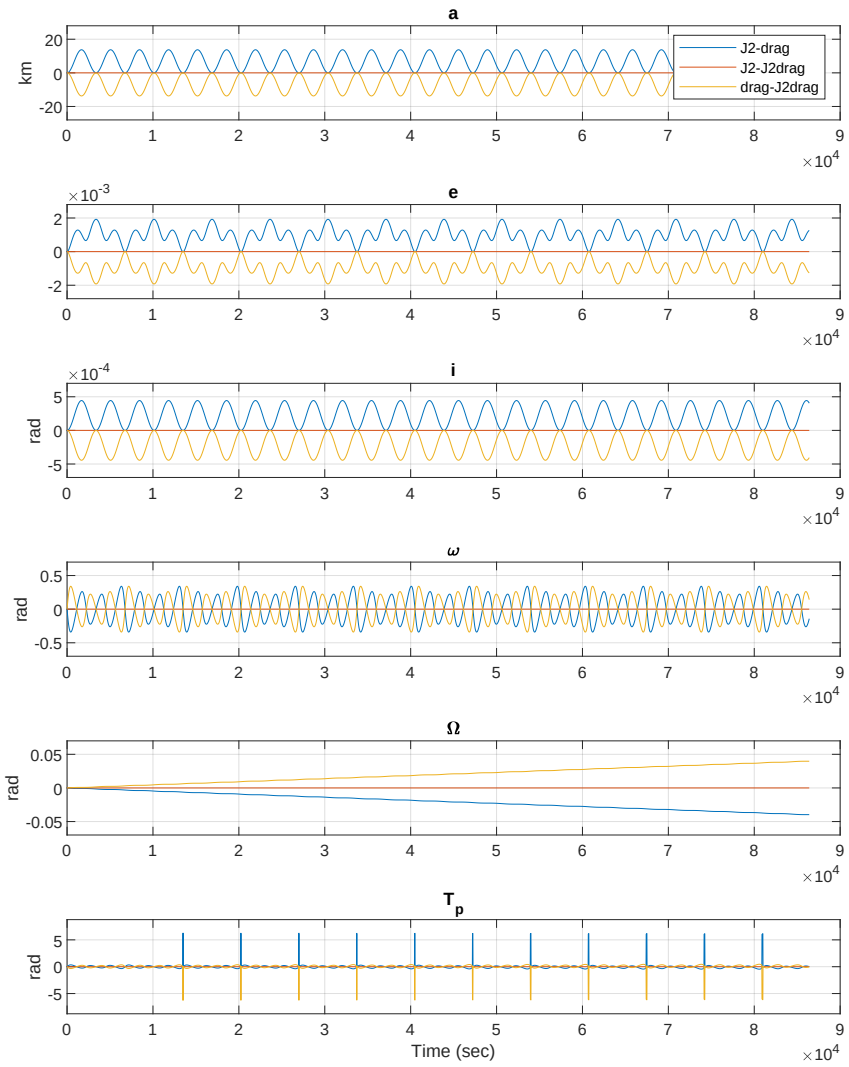

Fig. 8 Orbital Elements J2 and Drag Difference

Appendix

$\partial U/\partial x$ derivation

$$U = U_{\text{point mass}} + U_{J_2} = \frac{\mu}{r}\left[1 - J_2\left(\frac{R_{\text{Earth}}}{r}\right)^2\left(\frac{3}{2}\sin^2(\phi) - \frac{1}{2}\right)\right] \tag{1}$$

Derivation

$U_p = \frac{\mu}{r} \rightarrow \boxed{\nabla U_p = -\frac{\mu}{r^3}\underline{r}}$ (done in previous homework)

$$\boxed{\frac{\partial U_p}{\partial x} = -\frac{\mu x}{(x^2+y^2+z^2)^{3/2}}}$$

$\phi = \arcsin\left(\frac{z}{r}\right)$

$$U_{J_2} = -\frac{\mu}{r} J_2 \left(\frac{R_E}{r}\right)^2 \left(\frac{3}{2}\sin^2(\phi) - \frac{1}{2}\right)$$

$$= \left[-J_2\right]\left[\mu r^{-1}\right]\left[R_E^2 r^{-2}\right]\left[\frac{3}{2}\sin^2\left(\arcsin\left(\frac{z}{r}\right)\right) - \frac{1}{2}\right]$$

$$= \left[-J_2 \mu R_E^2\right]\left[r^{-3}\right]\left[\frac{3}{2}\sin^2\left(\arcsin\left(\frac{z}{r}\right)\right) - \frac{1}{2}\right]$$

$\left(\sin(\arcsin(\frac{z}{r}))\right)^2 = \frac{z^2}{r^2}$

$$= \left[-J_2 \mu R_E^2\right]\left[r^{-3}\right]\left[\frac{3}{2}z^2 r^{-2} - \frac{1}{2}\right]$$

$$= \left[-J_2 \mu R_E^2\right]\left[\frac{3}{2}z^2 r^{-5} - \frac{1}{2}r^{-3}\right]$$

(A)

$$\nabla U_{J_2} = \frac{\partial U_{J_2}}{\partial x}\hat{i} + \frac{\partial U_{J_2}}{\partial y}\hat{j} + \frac{\partial U_{J_2}}{\partial z}\hat{k}$$

$$(A) = \frac{3}{2}z^2 r^{-5} - \frac{1}{2}r^{-3} = \frac{3}{2}z^2\left(x^2+y^2+z^2\right)^{-5/2} - \frac{1}{2}\left(x^2+y^2+z^2\right)^{-3/2}$$

$$\frac{\partial A}{\partial x} = \left(\frac{3}{2}z^2\right)\left(-\frac{5}{2}\right)\left(x^2+y^2+z^2\right)^{-7/2}(2x) + \left(\frac{1}{2}\right)\left(+\frac{3}{2}\right)\left(x^2+y^2+z^2\right)^{-5/2}(2x)$$

$$= -\frac{15xz^2}{2}\left(x^2+y^2+z^2\right)^{-7/2} + \frac{3x}{2}\left(x^2+y^2+z^2\right)^{-5/2}$$

$$\frac{\partial A}{\partial x} = \left(\frac{3x}{2}\right)\frac{\left(x^2+y^2+z^2\right)-5z^2}{\left(x^2+y^2+z^2\right)^{7/2}} \quad \hat{\imath} \rightarrow \frac{3x}{2}\,\frac{r^2-5z^2}{r^7}$$

$$\frac{\partial A}{\partial y} = \left(\frac{3y}{2}\right)\frac{\left(x^2+y^2+z^2\right)-5z^2}{\left(x^2+y^2+z^2\right)^{7/2}} \quad \hat{\jmath} \rightarrow \frac{3y}{2}\,\frac{r^2-5z^2}{r^7}$$

$$\frac{\partial A}{\partial z} = \frac{3}{2}z^2\left(-\frac{5}{2}\right)\left(x^2+y^2+z^2\right)^{-7/2}(2z) + 3z\left(x^2+y^2+z^2\right)^{-5/2}$$

$$+\left(\frac{1}{2}\right)\left(\frac{3}{2}\right)\left(x^2+y^2+z^2\right)^{-5/2}(2z)$$

$$= -\frac{15z^3}{2}\left(x^2+y^2+z^2\right)^{-7/2} + \frac{6}{2}z\left(x^2+y^2+z^2\right)^{-5/2}$$

$$+\frac{3z}{2}\left(x^2+y^2+z^2\right)^{-5/2}$$

$$\frac{\partial A}{\partial z} = \frac{3z}{2}\left[\frac{3\left(x^2+y^2+z^2\right)-5z^2}{\left(x^2+y^2+z^2\right)^{7/2}}\right] \quad \hat{k} \rightarrow \frac{3z}{2}\,\frac{3r^2-5z^2}{r^7}$$

$$\nabla A = \frac{\partial A}{\partial x}\hat{\imath} + \frac{\partial A}{\partial y}\hat{\jmath} + \frac{\partial A}{\partial z}\hat{k}$$

$$\frac{\partial U_{J_2}}{\partial x} = \left[-J_2 \mu R_E^2\right]\left(\frac{3x}{2}\,\frac{r^2-5z^2}{r^7}\right)$$

$$\frac{\partial U_{J_2}}{\partial x} = \left[-J_2 \mu R_E^2\right]\left(\frac{3y}{2}\,\frac{r^2-5z^2}{r^7}\right)$$

$$\frac{\partial U_{J_2}}{\partial x} = \left[-J_2 \mu R_E^2\right]\left(\frac{3z}{2}\,\frac{3r^2-5z^2}{r^7}\right)$$

$$\frac{\partial U_p}{\partial x} = -\frac{\mu x}{(x^2+y^2+z^2)^{3/2}}$$

$$\frac{\partial U_{J_2}}{\partial x} = \left[-J_2 \mu R_E^2\right]\left(\frac{3x}{2}\,\frac{r^2 - 5z^2}{r^7}\right)$$

$$= \left[-J_2 \mu R_E^2\right]\left[\frac{3x}{2}\,\frac{x^2+y^2-4z^2}{(x^2+y^2+z^2)^{7/2}}\right]$$ ✓ matches Matlab output

$$\frac{\partial U}{\partial x} = \frac{\partial U_p}{\partial x} + \frac{\partial U_{J_2}}{\partial x}$$

HW2 MATLAB code

```
% ASE 389 Orbit Determination
% HW 2
% Junette Hsin

clear;

positionA = [100 100 600 600];
positionB = [100 100 700 900];

%% Problem 1

syms x y z
global mu RE J2

% constants
mu = 398600.4;      % G * M1 * M2
RE = 6378.145;      % Earth radius
J2 = 0.00108248;    % J2

% testing
% syms mu RE J2

% radius
r = sqrt(x^2 + y^2 + z^2);

% U point mass
Up = mu/r;

% latitude
phi = asin(z/r);

% U J2
UJ2 = -mu/r * J2 * (RE/r)^2 * ( 3/2 * ( sin(phi) )^2 - 1/2 );

% gradient
d_UJ2 = gradient(UJ2, [x y z]);
d_UJ2 = simplify(d_UJ2);

% Initial conditions (km)
r0  = [ -2436.45; -2436.45; 6891.037 ];
v0  = [ 5.088611; -5.088611; 0 ];
rv0 = [r0; v0];

% orbital elements (and sanity check)
oe0 = rv2oe(rv0);
rv_check2 = oe2rv(oe0);
[oe_check, oe_extra] = rv2orb_OG(rv0);
[rv_check] = orb2rv_OG(oe_check, oe_extra);

% 1 day period
% a = oe(1);
% T = abs(2 * pi * sqrt(a^3 / mu));        % period
T = 60 * 60 * 24;
dt = 20;

% set ode45 params
rel_tol = 1e-14;          % 1e-14 accurate; 1e-6 coarse
abs_tol = 1e-16;
options = odeset('reltol', rel_tol, 'abstol', abs_tol );

% INTEGRATE! Point mass and J2
[t_p, x_p] = ode45(@TwoBod_6states, [0:dt:T], [r0; v0], options);
[t_J2, x_J2] = ode45(@TwoBod_UJ2, [0:dt:T], [r0; v0], options);

% ----------------------------------------------------------------------

```

```
name = 'Problem 1 - 2-Body EOM Orbit';
h = figure('name', name, 'position', positionA + [50 0 0 0]);

x = x_p;
plot3(x(:,1), x(:,2), x(:,3)); hold on; grid on;
x = x_J2;
plot3(x(:,1), x(:,2), x(:,3)); hold on; grid on;

x = x_p;
plot3(x(1,1), x(1,2), x(1,3), 'mo')
plot3(x(end,1), x(end,2), x(end,3), 'cx')

x = x_J2;
plot3(x(1,1), x(1,2), x(1,3), 'mo')
plot3(x(end,1), x(end,2), x(end,3), 'cx')

legend('point mass', 'J2', 'start', 'end')

xlabel('x (km)')
ylabel('y (km)')
zlabel('z (km)')
%     legend('orbit', 'start', 'end')

sgtitle(name)
save_pdf(h, name);

%% problem 1b

clear oe oe_check oe_p Tp_p Tp_pJ2 oe_pJ2

for i = 1:length(x_p)
[oe_p(i, :), Tp_p(i,:)] = rv2oe( x_p(i, :) );
oe_check(i, :) = rv2orb_OG( x_p(i, :) );
%     Tp_p(i,:) = perigee_pass(oe_p(i,:), x_p(i,:));
end


for i = 1:length(x_J2)
oe_J2(i,:) = rv2oe( x_J2(i, :) );
oe_check(i,:) = rv2orb_OG( x_J2(i, :) );

Tp_J2(i,:) = perigee_pass(oe_J2(i,:), x_J2(i,:));
end

% Time of perigee passage
% n = mean motion = sqrt(mu/a^3)
% E = acos( r/a * cos(nu) + e )
% M = mean anomaly = E - e*sin(E)
% Tp = t0 - M/n

% ------------------------------------------------------------------------

labels = {'a', 'e', 'i', '\omega', '\Omega', 'T_p'};
units = {'km', '', 'rad', 'rad', 'rad', 'rad'};
name = 'Problem 1b - Orbital Elements';
h = figure('name', name, 'position', positionB + [50 0 0 0]);
for i = 1:5
subplot(6,1,i)
plot(t_J2, oe_J2(:, i)); hold on; grid on;
plot(t_p, oe_p(:, i));
title(labels{i});
ylabel(units{i});
increase_ylim;
if i == 1
legend('with J2', 'point mass');
end
end
subplot(6,1,6)
```

```
plot(t_J2, Tp_J2); hold on; grid on;
plot(t_p, Tp_p);
title('T_p')
ylabel('s')
sgtitle(name)
xlabel('Time (sec)')
save_pdf(h, name);

%% Problem 1c: energy

clear vnorm
clear h
clear h_mom
clear dE
clear dh
clear dhnorm

for i = 1:length(x_J2)

U(:,i) = comp_U(x_J2(i, 1:3));
vnorm(:,i) = sqrt( x_J2(i,4)^2 + x_J2(i,5)^2 + x_J2(i,6)^2 );
E(:,i) = vnorm(:,i)^2 / 2 - U(:,i);
a = [ x_J2(i,1), x_J2(i,2), x_J2(i,3) ];
b = [ x_J2(i,4), x_J2(i,5), x_J2(i,6) ];
h_mom(:,i) = cross( a , b );

dE(:,i) = E(i) - E(1);
dh(:,i) = h_mom(:,i) - h_mom(:,1);
dhnorm(:,i) = norm( dh(:,i) );

end

% --------------------------------------------------------------------------

name = 'Problem 1c - Delta Specific Energy';
h = figure('name', name, 'position', positionA + [50 0 0 0]);
plot(t_J2, dE);
title( 'dE = E(t) - E(t_0)' )
xlabel('Time (s)')
ylabel('km^2/s^2')
save_pdf(h, name);


%% Problem 1d: angular momentum


name = 'Problem 1d - Delta Angular Momentum';
h = figure('name', name, 'position', positionA + [50 0 0 0]);
plot(t_J2, dh(3, :));
title( 'dh_k = h_k(t) - h_k(t_0)' )
xlabel('Time (s)')
ylabel('km^2/s')
save_pdf(h, name);


%% Problem 2: DRAG

global CD A m p0 r0_drag H dtheta

CD = 2;
A = 3.6e-6; % --> km^2
m = 1350;
p0 = 4e-4; % --> -13 --> -4, multiply 1e9 when going 1/m^3 to 1/km^3
r0_drag = 7298.145;
H = 200;
dtheta = 7.29211585530066e-5;

% set 1 day period (again just in case)
```

```
T = 60 * 60 * 24;

% set ode45 params
rel_tol = 1e-14;         % 1e-14 accurate; 1e-6 coarse
abs_tol = 1e-16;
options = odeset('reltol', rel_tol, 'abstol', abs_tol );

% INTEGRATE! Point mass and J2 and drag
[t_J2drag, x_J2drag] = ode45(@TwoBod_UJ2_drag, [0:dt:T], [r0; v0], options);

% INTEGRATE! Point mass and drag
[t_drag, x_drag] = ode45(@TwoBod_drag, [0:dt:T], [r0; v0], options);


%% Problem 2a: specific energy


clear vnorm
clear h
clear h_mom
clear dE
clear dh
clear dhnorm

for i = 1:length(x_J2drag)

U(:,i) = comp_U(x_J2drag(i, 1:3));
vnorm(:,i) = sqrt( x_J2drag(i,4)^2 + x_J2drag(i,5)^2 + x_J2drag(i,6)^2 );
E(:,i) = vnorm(:,i)^2 / 2 - U(:,i);
a = [ x_J2drag(i,1), x_J2drag(i,2), x_J2drag(i,3) ];
b = [ x_J2drag(i,4), x_J2drag(i,5), x_J2drag(i,6) ];
h_mom(:,i) = cross( a , b );

dE(:,i) = E(i) - E(1);
dh(:,i) = h_mom(:,i) - h_mom(:,1);
dhnorm(:,i) = norm( dh(:,i) );

end

% ------------------------------------------------------------------------

name = 'Problem 2a - Delta Specific Energy';
h = figure('name', name, 'position', positionA + [50 0 0 0]);
plot(t_J2drag, dE);
title( 'dE = E(t) - E(t_0)' )
xlabel('Time (s)')
ylabel('km^2/s^2')
save_pdf(h, name);

%% Problem 2b: orbital elements


clear oe_drag Tp_drag

for i = 1:length(x_drag)
[oe_drag(i, :), Tp_drag(i,:)] = rv2oe( x_drag(i, :) );
oe_check(i, :) = rv2orb_OG( x_p(i, :) );
%     Tp_p(i,:) = perigee_pass(oe_p(i,:), x_p(i,:));
end

for i = 1:length(x_J2drag)
[oe_J2drag(i, :), Tp_J2drag(i,:)] = rv2oe( x_J2drag(i, :) );
oe_check(i, :) = rv2orb_OG( x_p(i, :) );
end

% ------------------------------------------------------------------------

labels = {'a', 'e', 'i', '\omega', '\Omega', 'T_p'};
```

```
units = {'km', '', 'rad', 'rad', 'rad', 'rad'};
name = 'Problem 2b - Orbital Elements';
h = figure('name', name, 'position', positionB + [50 0 0 0]);
for i = 1:5
subplot(6,1,i)
plot(t_J2drag, oe_J2drag(:,i)); hold on; grid on;
plot(t_J2, oe_J2(:, i));
plot(t_drag, oe_drag(:, i));
plot(t_p, oe_p(:, i));
title(labels{i});
ylabel(units{i});
increase_ylim;
if i == 1
legend('J2 + drag', 'J2', 'drag', 'point mass');
end
end
subplot(6,1,6)
plot(t_J2drag, Tp_J2drag); hold on; grid on;
plot(t_J2, Tp_J2);
plot(t_drag, Tp_drag);
plot(t_p, Tp_p);
title('T_p')
ylabel('s')
sgtitle(name)
xlabel('Time (sec)')
save_pdf(h, name);



% ------------------------------------------------------------------------
% differences in orbital elements

% 6 combinations:
% (1) p - J2
% (2) p - drag
% (3) p - J2drag
% (4) J2 - drag
% (5) J2 - J2drag
% (6) drag - J2drag

% add Tp, make your life easier

oe_p = [oe_p, Tp_p];
oe_J2 = [oe_J2, Tp_J2];
oe_J2drag = [oe_J2drag, Tp_J2drag];
oe_drag = [oe_drag, Tp_drag];

p_J2 = oe_p - oe_J2;
p_drag = oe_p - oe_drag;
p_J2drag = oe_p - oe_J2drag;
J2_drag = oe_J2 - oe_drag;
J2_J2drag = oe_J2 - oe_J2drag;
drag_J2drag = oe_drag - oe_J2drag;

labels = {'a', 'e', 'i', '\omega', '\Omega', 'T_p', 'T_p'};
units = {'km', '', 'rad', 'rad', 'rad', 'rad', 's'};
name = 'Problem 2b - Orbital Elements Point Mass Diff';
h = figure('name', name, 'position', positionB + [50 0 0 0]);
for i = 1:6
subplot(6,1,i)
plot(t_J2, p_J2(:,i)); hold on; grid on;
plot(t_J2, p_drag(:,i));
plot(t_J2, p_J2drag(:,i));

title(labels{i});
ylabel(units{i});
increase_ylim;
if i == 1
```

```
legend('p-J2', 'p-drag', 'p-J2drag');
end
end
sgtitle(name)
xlabel('Time (sec)')
save_pdf(h, name);

labels = {'a', 'e', 'i', '\omega', '\Omega', 'T_p', 'T_p'};
units = {'km', '', 'rad', 'rad', 'rad', 'rad', 's'};
name = 'Problem 2b - Orbital Elements J2 and Drag Diff';
h = figure('name', name, 'position', positionB + [50 0 0 0]);
for i = 1:6
subplot(6,1,i)
plot(t_J2, J2_drag(:,i)); hold on; grid on;
plot(t_J2, J2_J2drag(:,i));
plot(t_J2, drag_J2drag(:,i));

title(labels{i});
ylabel(units{i});
increase_ylim;
if i == 1
legend('J2-drag', 'J2-J2drag', 'drag-J2drag');
end
end
sgtitle(name)
xlabel('Time (sec)')
save_pdf(h, name);



%%
%% subfunctions

function save_pdf(h, name)

% save as cropped pdf
set(h,'Units','Inches');
pos = get(h,'Position');
set(h,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize',[pos(3), pos(4)])
print(h,name,'-dpdf','-r0')

end

function increase_ylim

ylims  = get(gca, 'ylim');
yd     = ylims(2) - ylims(1);
set(gca, 'ylim', [ylims(1) - 0.2*yd, ylims(2) + 0.2*yd  ]);

end

function U = comp_U(rv)

global mu J2 RE

x = rv(1);
y = rv(2);
z = rv(3);

% radius
r = sqrt(x^2 + y^2 + z^2);

% U point mass
Up = mu/r;

% latitude
phi = asin(z/r);

```

```
% U J2
UJ2 = -mu/r * J2 * (RE/r)^2 * ( 3/2 * ( sin(phi) )^2 - 1/2 );

% U point mass
U = Up + UJ2;

end

function Tp = perigee_pass(oe, x)

global mu

% perigee passing
a = oe(1);
e = oe(2);
nu = oe(6);
%     r = norm([ x_pJ2(i,1) x_pJ2(i,2) x_pJ2(i,3) ]);
r = sqrt( x(1)^2 + x(2)^2 + x(3)^2 );

n = sqrt(mu/a^3);
E = acos( r/a * cos(nu) + e );
M = E - e*sin(E);
Tp = M/n;

end
```

rv2oe function

```
function [oe, Tp] = rv2oe(rv)
% ------------------------------------------------------------------------
% Inputs
%   rv = [6x1] position and velocity states vector in ECI frame
%
% Outputs
%   oe = [6x1] orbital elements: a, e, i, w, Omega, nu
%           a   = semimajor axis
%           e   = eccentricity
%           i   = inclination
%           w   = argument of perigee
%           O   = right ascension of ascending node
%           M   = mean anomaly
%           Tp  = time of perigee passing  (optional output)
% ------------------------------------------------------------------------

global mu

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

% angular momentum
h       = cross(r,v);

% node vector
nhat    = cross( [0 0 1], h );

% eccentricity
evec    = ( (norm(v)^2 - mu/norm(r))*r - dot(r,v)*v ) / mu;
e       = norm(evec);

% specific mechanical energy
energy  = norm(v)^2/2 - mu/norm(r);

% semi-major axis and p
if abs(e-1.0)>eps
a = -mu/(2*energy);
p = a*(1-e^2);
else
```

```
p = norm(h)^2/mu;
a = inf;
end

% inclination
i = acos(h(3)/norm(h));

% right ascension of ascending node (check for equatorial orbit)
if i > 0.000001
O = acos( nhat(1)/norm(nhat) );
else
O = 0;
end
if isnan(O)
O = 0;
end
if nhat(2)<0
O = 2*pi - O;
end

% argument of perigee
if e > 0.000001
w = acos(dot(nhat,evec)/(norm(nhat)*e));
else
w = 0;
end
if isnan(w)
w = 0;
end
% if e(3)<0
%     argp = 360-argp
% end

% true anomaly
nu = acos( dot(evec,r) / (e*norm(r)) );
% if dot(r,v)<0
%     nu = 360 - nu
% end

%Apply Quadrant Checks to All Determined Angles
% idx = nhat(2) < 0; if any(idx); O(idx) = 2*pi - O(idx);  end
% idx = evec(3) < 0; if any(idx); w(idx) = 2*pi - w(idx); end
idx = dot(r,v) < 0; if any(idx); nu(idx) = 2*pi - nu(idx); end

oe = [a; e; i; w; O; nu];

% perigee passing time

rnorm = norm(r);
n = sqrt(mu/a^3);
E = acos( rnorm/a * cos(nu) + e );
M = E - e*sin(E);
Tp = M/n;

end
```

oe2rv function

```
function [rv] = oe2rv(oe)
% ------------------------------------------------------------------------
% Purpose: Convert orbital elements and time past epoch to the classic
% Cartesian position and velocity
%
% Inputs:
%   oe       = [6x1] or [1x6] orbital elements
%   delta_t  = t - t0 time interval
%   mu       = Gravity * Mass (of Earth) constant
```

```
%
% Outputs:
%   rv      = position and velocity state vector
% -------------------------------------------------------------------------

% global delta_t
global mu

a       = oe(1);
e       = oe(2);
i       = oe(3);
w       = oe(4);
LAN     = oe(5);
% M0      = oe(6);
nu      = oe(6);

% nu is TRUE ANOMALY --> use Kepler's to calculate MEAN ANOMALY
% E = 2*atan( sqrt( (1-e)/(1+e) ) * tan(nu/2) );
% M = M0 + sqrt( mu/a^3 ) * (delta_t);
% E = keplerEq(M, e, eps);
% E = kepler(M, e);
% nu = 2*atan( sqrt( (1+e)/(1-e) ) * tan(E/2) );

p = a * ( 1 - e^2 );            % intermediate variable
r = p / ( 1 + e*cos(nu) );      % r_magnitude, polar coordinates

% Perifocal position and velocity

r_pf = [ r * cos(nu); r * sin(nu); 0 ];
v_pf = [ -sqrt(mu/p) * sin(nu); sqrt(mu/p) * (e + cos(nu)); 0 ];

% Perifocal to ECI transformation, 3-1-3 rotation
R11 = cos(LAN)*cos(w) - sin(LAN)*sin(w)*cos(i);
R12 = -cos(LAN)*sin(w) - sin(LAN)*cos(w)*cos(i);
R13 = sin(LAN)*sin(i);

R21 = sin(LAN)*cos(w) + cos(LAN)*sin(w)*cos(i);
R22 = -sin(LAN)*sin(w) + cos(LAN)*cos(w)*cos(i);
R23 = -cos(LAN)*sin(i);

R31 = sin(w)*sin(i);
R32 = cos(w)*sin(i);
R33 = cos(i);

R = [R11 R12 R13; R21 R22 R23; R31 R32 R33];

% Transform perifocal to ECI frame
r_vec = R * r_pf;
v_vec = R * v_pf;

% Position and state vector
rv = [r_vec; v_vec];

end

%% Kepler equation solvers

function E = keplerEq(M,e,eps)
% Function solves Kepler's equation M = E-e*sin(E)
% Input - Mean anomaly M [rad] , Eccentricity e and Epsilon
% Output  eccentric anomaly E [rad].
En  = M;
Ens = En - (En-e*sin(En)- M)/(1 - e*cos(En));
while ( abs(Ens-En) > eps )
En = Ens;
Ens = En - (En - e*sin(En) - M)/(1 - e*cos(En));
end
E = Ens;
```

```
end

function E = kepler(M, e)
f = @(E) E - e * sin(E) - M;
E = fzero(f, M);  % <-- I would use M as the initial guess instead of 0
end
```

Integrated Equations of Motion

```

function dx = TwoBod_6states(t, x)
% ------------------------------------------------------------------------
% Inputs
%   t = [Nx1] time vector (orbit is Keplerian, doesn't matter)
%   x = [6x1] state vector
%
% Outputs
%   dx = [6x1] derivative of state vector
% ------------------------------------------------------------------------

global mu

dx = zeros(6, 1);   % force column vector

dx(1:3) = x(4:6);
% r_norm  = sqrt( x(1)^2 + x(2)^2 + x(3)^2 );
r_norm  = norm(x(1:3));
dx(4:6) = ( - mu / r_norm^3 ) * x(1:3);

end

function dx = TwoBod_UJ2(t, x)
% ------------------------------------------------------------------------
% Inputs
%   t = [Nx1] time vector (orbit is Keplerian, doesn't matter)
%   x = [6x1] state vector
%
% Outputs
%   dx = [6x1] derivative of state vector
% ------------------------------------------------------------------------

global mu J2 RE

dx = zeros(6, 1);   % force column vector

dx(1:3) = x(4:6);
% r_norm  = sqrt( x(1)^2 + x(2)^2 + x(3)^2 );
r_norm  = norm(x(1:3));
dx(4:6) = ( - mu / r_norm^3 ) * x(1:3);

% J2 stuff
dUx =       -(3*J2*RE^2*mu*x(1)*(x(1)^2 + x(2)^2 - 4*x(3)^2)) / (2*(x(1)^2 + x(2)^2 + x(3)^2)
    ^(7/2));
dUy =       -(3*J2*RE^2*mu*x(2)*(x(1)^2 + x(2)^2 - 4*x(3)^2)) / (2*(x(1)^2 + x(2)^2 + x(3)^2)
    ^(7/2));
dUz =  -(3*J2*RE^2*mu*x(3)*(3*x(1)^2 + 3*x(2)^2 - 2*x(3)^2)) / (2*(x(1)^2 + x(2)^2 + x(3)^2)
    ^(7/2));

dx(4:6) = dx(4:6) + [dUx; dUy; dUz];

end

function dx = TwoBod_UJ2(t, x)
% ------------------------------------------------------------------------
% Inputs
%   t = [Nx1] time vector (orbit is Keplerian, doesn't matter)
%   x = [6x1] state vector
```

```
%
% Outputs
%   dx = [6x1] derivative of state vector
% ------------------------------------------------------------------------

global mu J2 RE
global CD A m p0 r0_drag H dtheta

dx = zeros(6, 1);   % force column vector

dx(1:3) = x(4:6);
% r_norm  = sqrt( x(1)^2 + x(2)^2 + x(3)^2 );
rnorm  = norm(x(1:3));
dx(4:6) = ( - mu / rnorm^3 ) * x(1:3);

% drag stuff
pA = p0 * exp( -(rnorm - r0_drag)/H );
V_A = [dx(4) + dtheta * dx(2); dx(5) - dtheta * dx(1); dx(6)];
VA = sqrt( ( dx(4) + dtheta * dx(2) )^2 + ( dx(5) - dtheta * dx(1) )^2 + dx(6)^2 );

ddx_drag = - 1/2 * CD * A/m * pA * VA * V_A;
dx(4:6) = dx(4:6) + ddx_drag;

end

function dx = TwoBod_UJ2(t, x)
% ------------------------------------------------------------------------
% Inputs
%   t = [Nx1] time vector (orbit is Keplerian, doesn't matter)
%   x = [6x1] state vector
%
% Outputs
%   dx = [6x1] derivative of state vector
% ------------------------------------------------------------------------

global mu J2 RE
global CD A m p0 r0_drag H dtheta

dx = zeros(6, 1);   % force column vector

dx(1:3) = x(4:6);
% r_norm  = sqrt( x(1)^2 + x(2)^2 + x(3)^2 );
rnorm  = norm(x(1:3));
dx(4:6) = ( - mu / rnorm^3 ) * x(1:3);

% J2 stuff
dUx =       -(3*J2*RE^2*mu*x(1)*(x(1)^2 + x(2)^2 - 4*x(3)^2)) / (2*(x(1)^2 + x(2)^2 + x(3)^2)
    ^(7/2));
dUy =       -(3*J2*RE^2*mu*x(2)*(x(1)^2 + x(2)^2 - 4*x(3)^2)) / (2*(x(1)^2 + x(2)^2 + x(3)^2)
    ^(7/2));
dUz =  -(3*J2*RE^2*mu*x(3)*(3*x(1)^2 + 3*x(2)^2 - 2*x(3)^2)) / (2*(x(1)^2 + x(2)^2 + x(3)^2)
    ^(7/2));

dx(4:6) = dx(4:6) + [dUx; dUy; dUz];

% drag stuff
pA = p0 * exp( -(rnorm - r0_drag)/H );
V_A = [dx(4) + dtheta * dx(2); dx(5) - dtheta * dx(1); dx(6)];
VA = sqrt( ( dx(4) + dtheta * dx(2) )^2 + ( dx(5) - dtheta * dx(1) )^2 + dx(6)^2 );

ddx_drag = - 1/2 * CD * A/m * pA * VA * V_A;
dx(4:6) = dx(4:6) + ddx_drag;

end
```

s

References

[1] Bob Schutz, G. H. B., Byron Tapley, *Statistical Orbit Determination*, Academic Press, 2004.

[2] Jah, M. K., "ASE 389P.4 Methods of Orbit Determination Module 3," , January 2021.