

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 :

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

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:

$$\underline{r} = -2436.45\hat{i} - 2436.45\hat{j} + 6891.037\hat{k} \text{ km} \quad (2)$$

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

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:

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

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:

```
1 syms x y z
2 global mu RE J2
3
4 % constants
5 % mu = 398600.4;           % G * M1 * M2
6 % RE = 6378.145;          % Earth radius
7 % J2 = 0.00108248;        % J2
8
9 % for symbolic representation
10 syms mu RE J2
11
12 % radius
13 r = sqrt(x^2 + y^2 + z^2);
14
15 % U point mass
16 Up = mu/r;
17
18 % latitude
19 phi = asin(z/r);
20
21 % U J2
22 UJ2 = -mu/r * J2 * (RE/r)^2 * ( 3/2 * ( sin(phi) )^2 - 1/2 );
23
24 % gradient
25 d_UJ2 = gradient(UJ2, [x y z]);
26 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:

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

Problem 1b - Orbital Elements

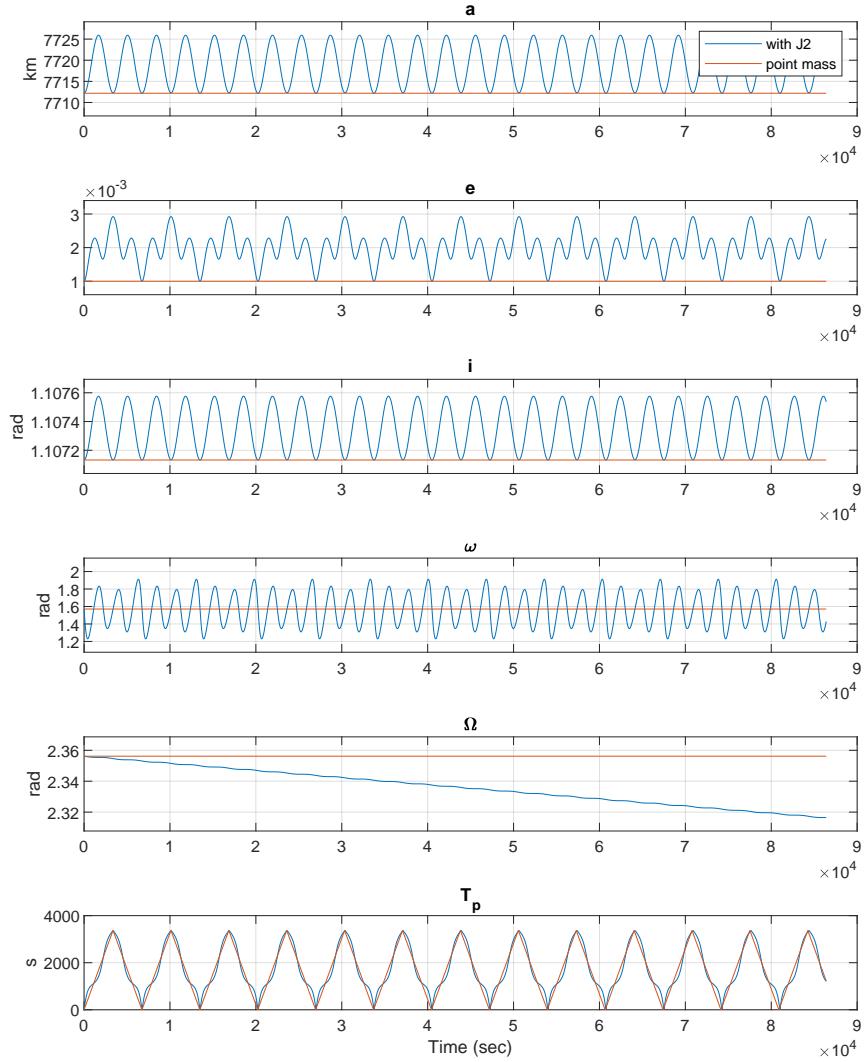


Fig. 1 Orbital elements of point mass and point mass with J2 zonal elements

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 $6.74\text{e}3$ 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.

Problem 1 - 2-Body EOM Orbit

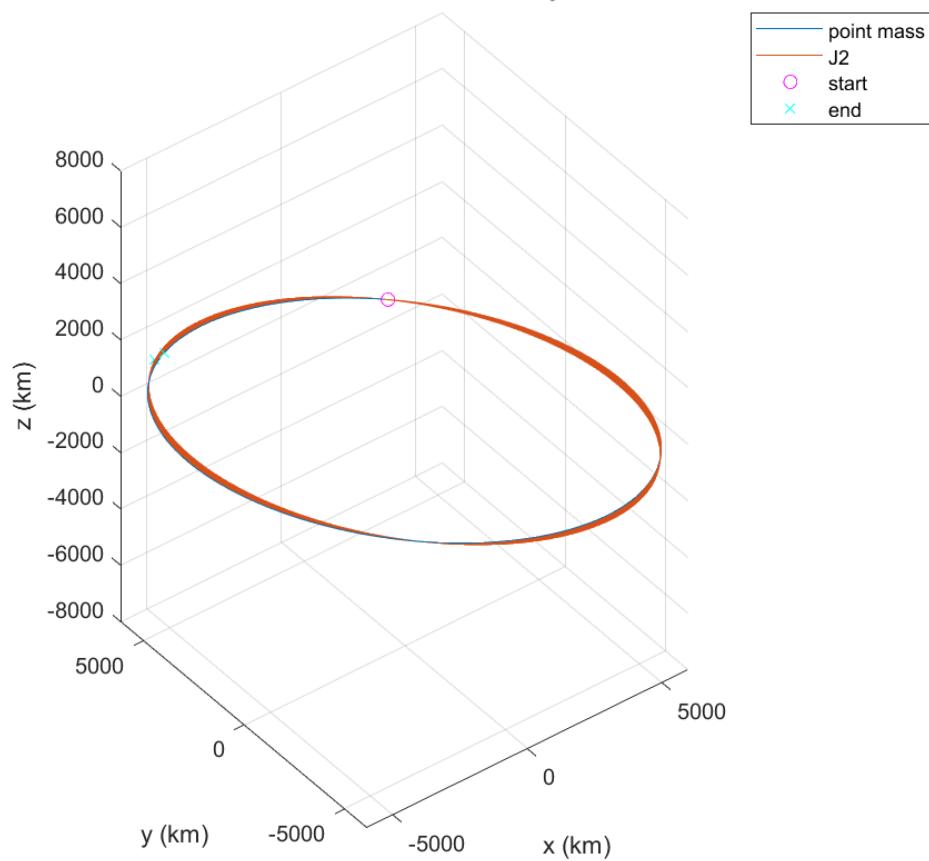


Fig. 2 2-body EOM Orbit of Point Mass with J2 term

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:

$$E = \frac{v^2}{2} - U \quad (4)$$

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.

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

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 $25.815 \text{ km}^2/\text{s}^2$ for the duration of the simulation.

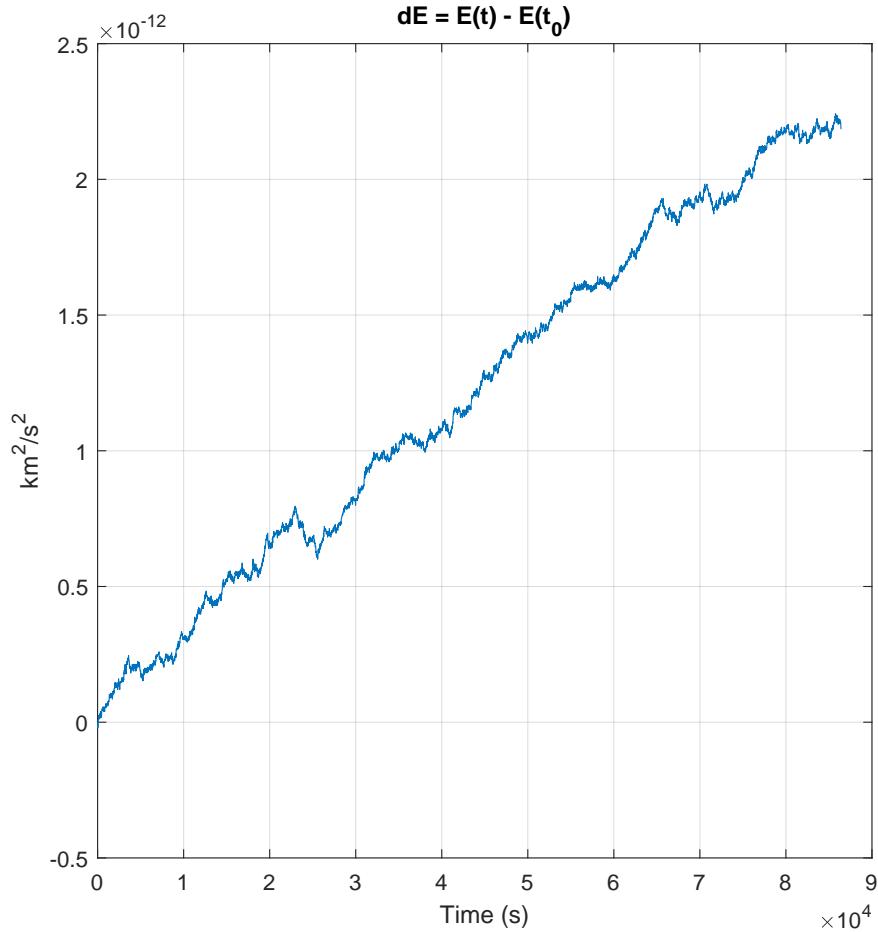


Fig. 3 Delta Specific Energy

Problem 1d

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

$$\underline{h} = \underline{r} \times \underline{v} \quad (5)$$

and plot $d\underline{h}_k = \underline{h}_k(t) - \underline{h}_k(t_0)$ to show that it remains constant.

Solution

The k th-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 k th component remained within an order of magnitude of -10 to its initial value.

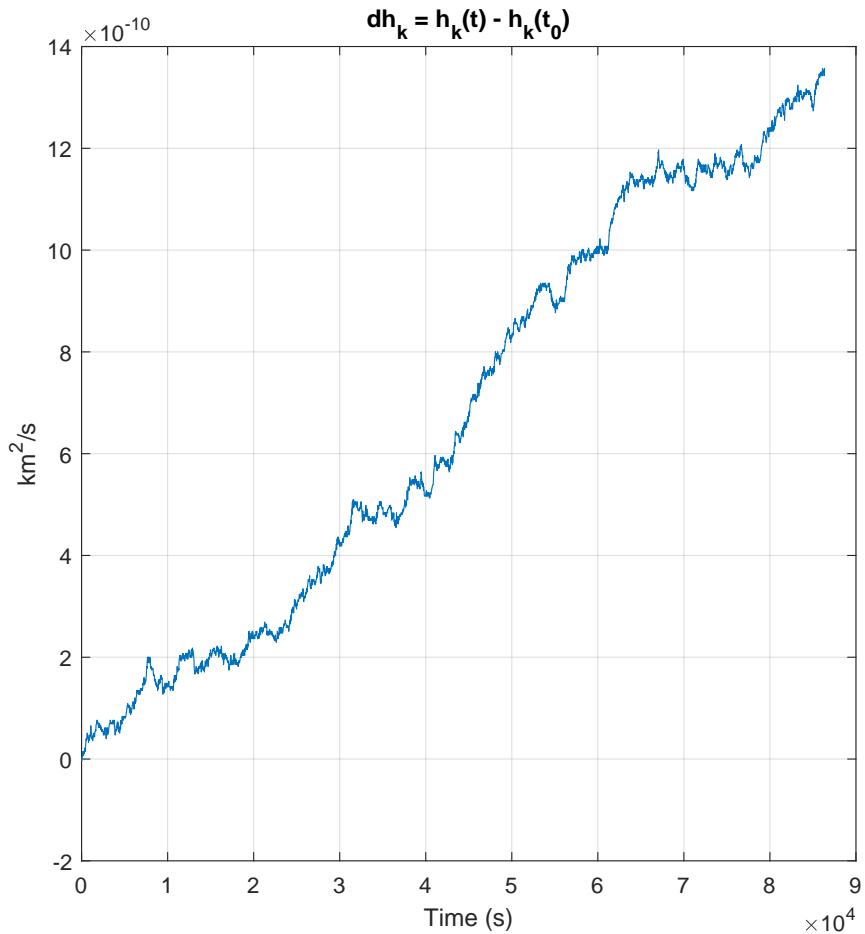


Fig. 4 Delta Angular Momentum

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:

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

where

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

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.

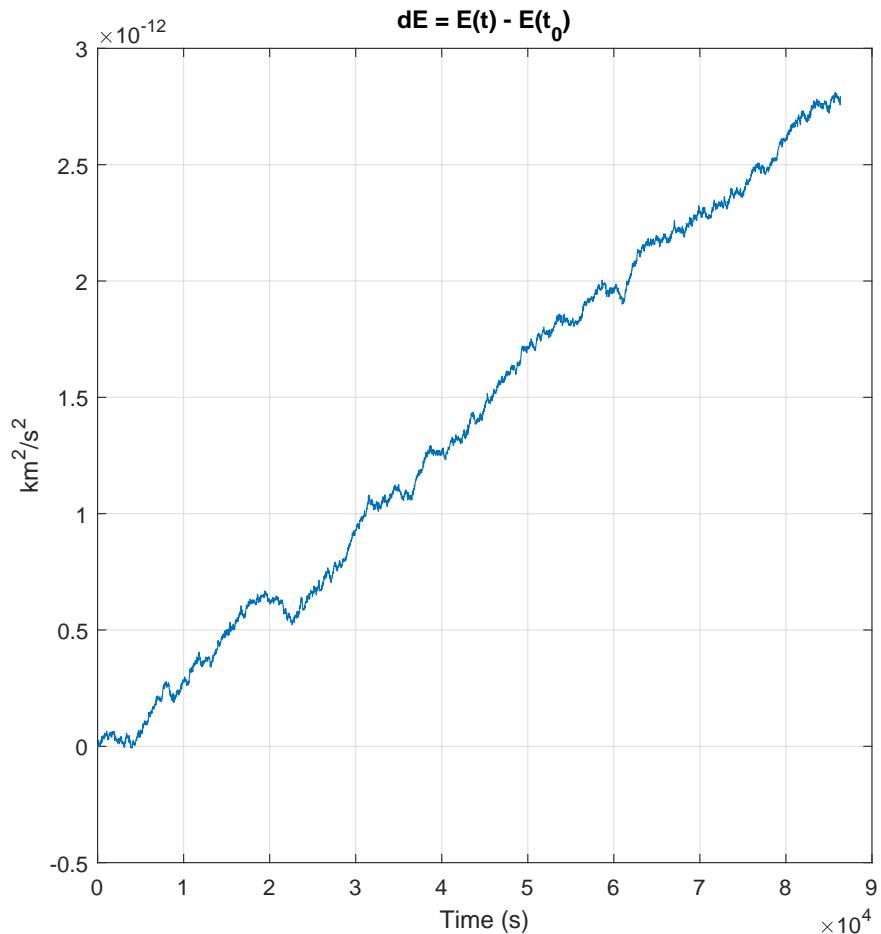


Fig. 5 Delta Specific Energy for Point Mass with J2 Terms and Drag

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

Problem 2b - Orbital Elements

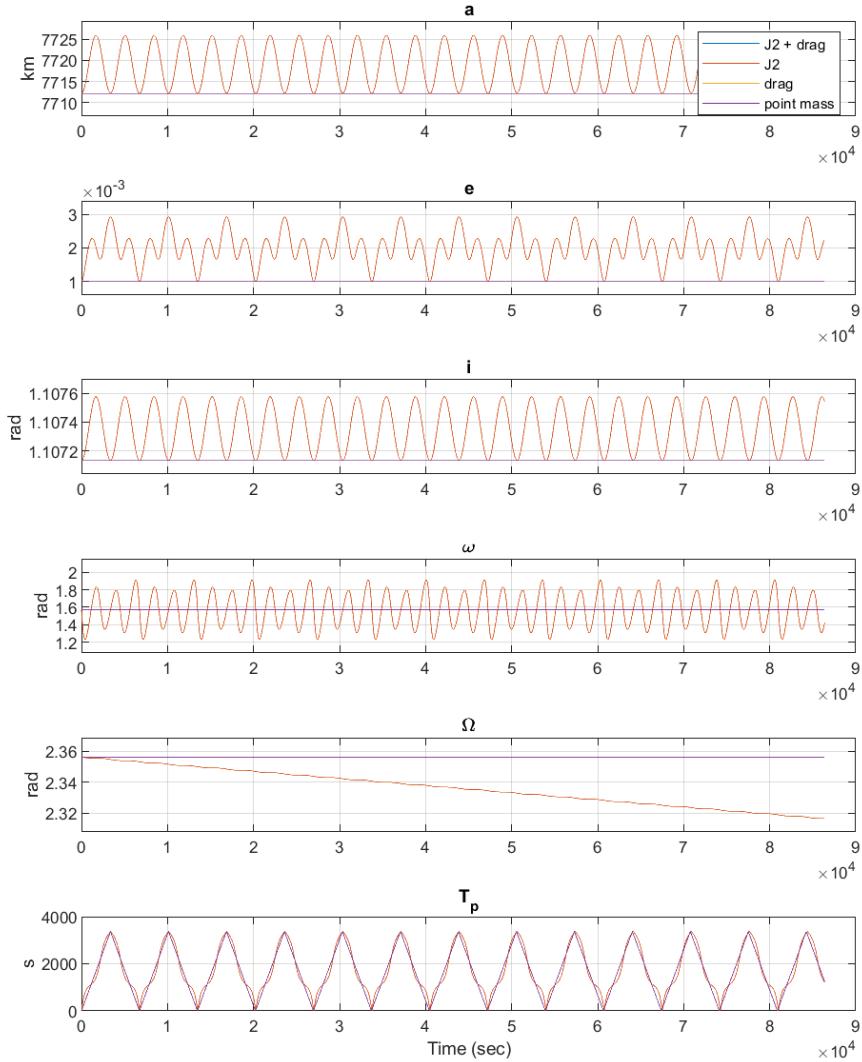


Fig. 6 Orbital Elements for Point Mass with J2 Terms and Drag

Problem 2b - Orbital Elements Point Mass Diff

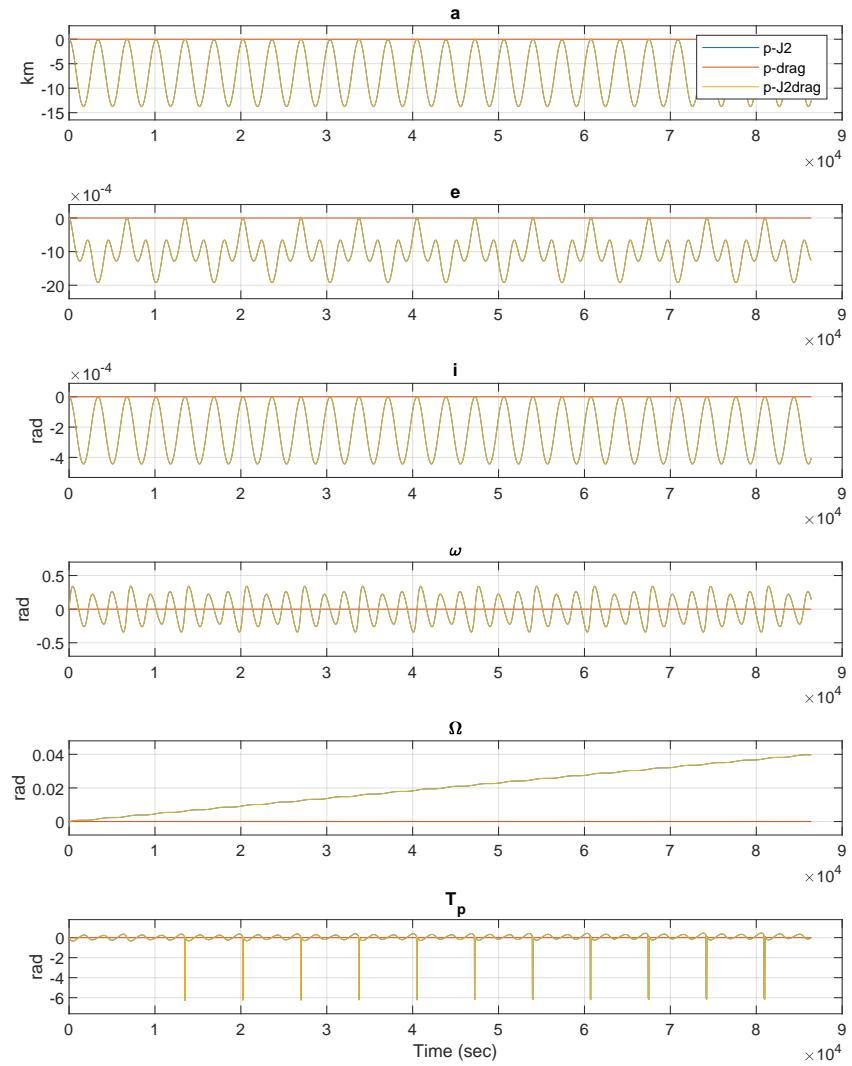


Fig. 7 Orbital Elements Point Mass Difference

Problem 2b - Orbital Elements J2 and Drag Diff

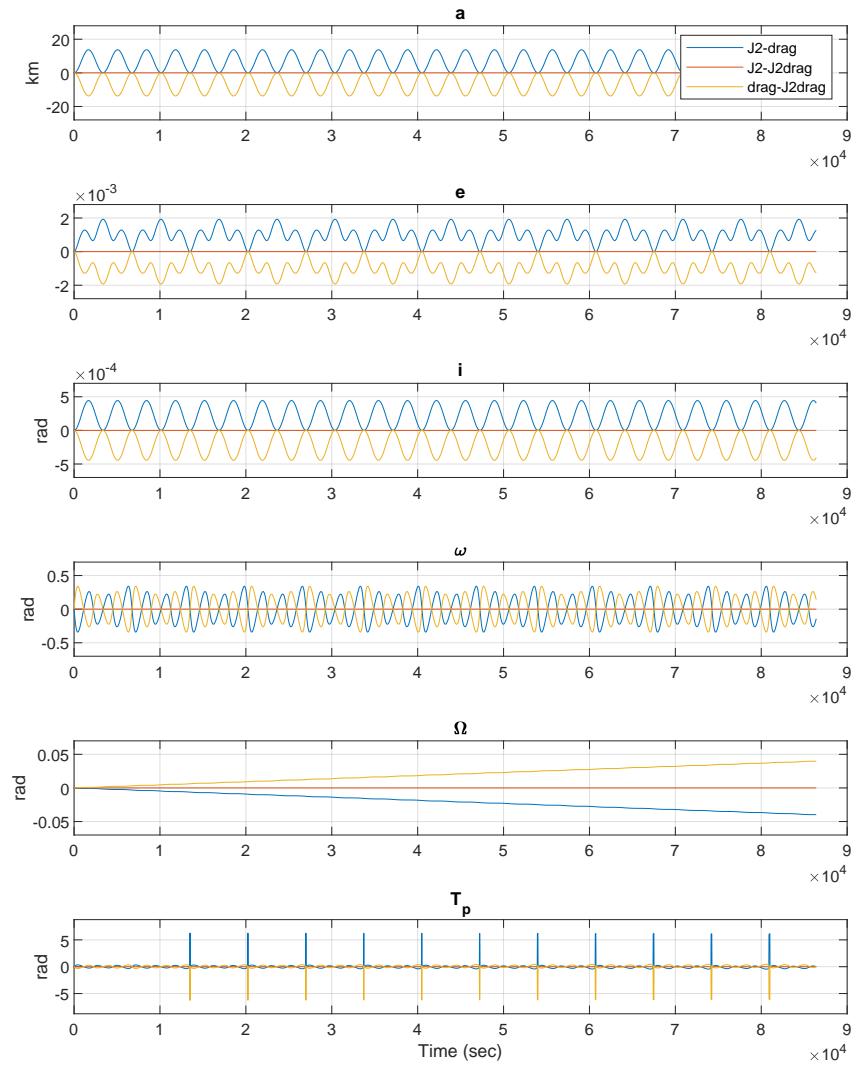


Fig. 8 Orbital Elements J2 and Drag Difference

Appendix

$\partial U / \partial x$ derivation

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

Derivation

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

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

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

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

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

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

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

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

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

(A)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

✓ matches
matlab output

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

HW2 MATLAB code

```

1 % ASE 389 Orbit Determination
2 % HW 2
3 % Junette Hsin
4
5 clear;
6
7 positionA = [100 100 600 600];
8 positionB = [100 100 700 900];
9
10 %&% Problem 1
11
12 syms x y z
13 global mu RE J2
14
15 % constants
16 mu = 398600.4;           % G * M1 * M2
17 RE = 6378.145;           % Earth radius
18 J2 = 0.00108248;         % J2
19
20 % testing
21 % syms mu RE J2
22
23 % radius
24 r = sqrt(x^2 + y^2 + z^2);
25
26 % U point mass
27 Up = mu/r;
28
29 % latitude
30 phi = asin(z/r);
31
32 % U J2
33 UJ2 = -mu/r * J2 * (RE/r)^2 * ( 3/2 * ( sin(phi) )^2 - 1/2 );
34
35 % gradient
36 d_UJ2 = gradient(UJ2, [x y z]);
37 d_UJ2 = simplify(d_UJ2);
38
39 % Initial conditions (km)
40 r0 = [ -2436.45; -2436.45; 6891.037 ];
41 v0 = [ 5.088611; -5.088611; 0 ];
42 rv0 = [r0; v0];
43
44 % orbital elements (and sanity check)
45 oe0 = rv2oe(rv0);
46 rv_check2 = oe2rv(oe0);
47 [oe_check, oe_extra] = rv2orb_OG(rv0);
48 [rv_check] = orb2rv_OG(oe_check, oe_extra);
49
50 % 1 day period
51 % a = oe(1);
52 % T = abs(2 * pi * sqrt(a^3 / mu));           % period
53 T = 60 * 60 * 24;
54 dt = 20;
55
56 % set ode45 params
57 rel_tol = 1e-14;           % 1e-14 accurate; 1e-6 coarse
58 abs_tol = 1e-16;
59 options = odeset('reltol', rel_tol, 'abstol', abs_tol);
60
61 % INTEGRATE! Point mass and J2
62 [t_p, x_p] = ode45(@TwoBod_6states, [0:dt:T], [r0; v0], options);
63 [t_J2, x_J2] = ode45(@TwoBod_UJ2, [0:dt:T], [r0; v0], options);
64
65 % -----
66

```

```

67 name = 'Problem 1 - 2-Body EOM Orbit';
68 h = figure('name', name, 'position', positionA + [50 0 0 0]);
69
70 x = x_p;
71 plot3(x(:,1), x(:,2), x(:,3)); hold on; grid on;
72 x = x_J2;
73 plot3(x(:,1), x(:,2), x(:,3)); hold on; grid on;
74
75 x = x_p;
76 plot3(x(1,1), x(1,2), x(1,3), 'mo')
77 plot3(x(end,1), x(end,2), x(end,3), 'cx')
78
79 x = x_J2;
80 plot3(x(1,1), x(1,2), x(1,3), 'mo')
81 plot3(x(end,1), x(end,2), x(end,3), 'cx')
82
83 legend('point mass', 'J2', 'start', 'end')
84
85 xlabel('x (km)')
86 ylabel('y (km)')
87 zlabel('z (km)')
88 % legend('orbit', 'start', 'end')
89
90 sgtitle(name)
91 save_pdf(h, name);
92
93 % problem 1b
94
95 clear oe oe_check oe_p Tp_pTp_pJ2 oe_pJ2
96
97 for i = 1:length(x_p)
98 [oe_p(i, :) , Tp_p(i,:)] = rv2oe( x_p(i, :) );
99 oe_check(i, :) = rv2orb_OG( x_p(i, :) );
100 % Tp_p(i,:) = perigee_pass(oe_p(i,:), x_p(i,:));
101 end
102
103
104 for i = 1:length(x_J2)
105 oe_J2(i,:) = rv2oe( x_J2(i, :) );
106 oe_check(i,:) = rv2orb_OG( x_J2(i, :) );
107
108 Tp_J2(i,:) = perigee_pass(oe_J2(i,:), x_J2(i,:));
109 end
110
111 % Time of perigee passage
112 % n = mean motion = sqrt(mu/a^3)
113 % E = acos( r/a * cos(nu) + e )
114 % M = mean anomaly = E - e*sin(E)
115 % Tp = t0 - M/n
116
117 % -----
118
119 labels = {'a', 'e', 'i', '\omega', '\Omega', 'T_p'};
120 units = {'km', '', 'rad', 'rad', 'rad', 'rad'};
121 name = 'Problem 1b - Orbital Elements';
122 h = figure('name', name, 'position', positionB + [50 0 0 0]);
123 for i = 1:5
124 subplot(6,1,i)
125 plot(t_J2, oe_J2(:, i)); hold on; grid on;
126 plot(t_p, oe_p(:, i));
127 title(labels{i});
128 ylabel(units{i});
129 increase_ylim;
130 if i == 1
131 legend('with J2', 'point mass');
132 end
133 end
134 subplot(6,1,6)

```

```

135 plot(t_J2, Tp_J2); hold on; grid on;
136 plot(t_p, Tp_p);
137 title('T_p')
138 ylabel('s')
139 s gtitle(name)
140 xlabel('Time (sec)')
141 save_pdf(h, name);
142
143 %% Problem 1c: energy
144
145 clear vnorm
146 clear h
147 clear h_mom
148 clear dE
149 clear dh
150 clear dhnorm
151
152 for i = 1:length(x_J2)
153
154 U(:, i) = comp_U(x_J2(i, 1:3));
155 vnorm(:, i) = sqrt( x_J2(i,4)^2 + x_J2(i,5)^2 + x_J2(i,6)^2 );
156 E(:, i) = vnorm(:, i)^2 / 2 - U(:, i);
157 a = [ x_J2(i,1), x_J2(i,2), x_J2(i,3) ];
158 b = [ x_J2(i,4), x_J2(i,5), x_J2(i,6) ];
159 h_mom(:, i) = cross(a, b);
160
161 dE(:, i) = E(i) - E(1);
162 dh(:, i) = h_mom(:, i) - h_mom(:, 1);
163 dhnorm(:, i) = norm(dh(:, i));
164
165 end
166
167 %% -----
168
169 name = 'Problem 1c - Delta Specific Energy';
170 h = figure('name', name, 'position', positionA + [50 0 0 0]);
171 plot(t_J2, dE);
172 title('dE = E(t) - E(t_0)')
173 xlabel('Time (s)')
174 ylabel('km^2/s^2')
175 save_pdf(h, name);
176
177
178 %% Problem 1d: angular momentum
179
180
181 name = 'Problem 1d - Delta Angular Momentum';
182 h = figure('name', name, 'position', positionA + [50 0 0 0]);
183 plot(t_J2, dh(3, :));
184 title('dh_k = h_k(t) - h_k(t_0)')
185 xlabel('Time (s)')
186 ylabel('km^2/s')
187 save_pdf(h, name);
188
189
190 %% Problem 2: DRAG
191
192 global CD A m p0 r0_drag H dtheta
193
194 CD = 2;
195 A = 3.6e-6; %% --> km^2
196 m = 1350;
197 p0 = 4e-4; %% --> -13 --> -4, multiply 1e9 when going 1/m^3 to 1/km^3
198 r0_drag = 7298.145;
199 H = 200;
200 dtheta = 7.29211585530066e-5;
201
202 %% set 1 day period (again just in case)

```

```

203 T = 60 * 60 * 24;
204
205 % set ode45 params
206 rel_tol = 1e-14;           % 1e-14 accurate; 1e-6 coarse
207 abs_tol = 1e-16;
208 options = odeset('reltol', rel_tol, 'abstol', abs_tol );
209
210 % INTEGRATE! Point mass and J2 and drag
211 [t_J2drag, x_J2drag] = ode45(@TwoBod_UJ2_drag, [0:dt:T], [r0; v0], options);
212
213 % INTEGRATE! Point mass and drag
214 [t_drag, x_drag] = ode45(@TwoBod_drag, [0:dt:T], [r0; v0], options);
215
216 %% Problem 2a: specific energy
217
218
219
220 clear vnorm
221 clear h
222 clear h_mom
223 clear dE
224 clear dh
225 clear dhnorm
226
227 for i = 1:length(x_J2drag)
228
229 U(:,i) = comp_U(x_J2drag(i, 1:3));
230 vnorm(:,i) = sqrt( x_J2drag(i,4)^2 + x_J2drag(i,5)^2 + x_J2drag(i,6)^2 );
231 E(:,i) = vnorm(:,i)^2 / 2 - U(:,i);
232 a = [ x_J2drag(i,1), x_J2drag(i,2), x_J2drag(i,3) ];
233 b = [ x_J2drag(i,4), x_J2drag(i,5), x_J2drag(i,6) ];
234 h_mom(:,i) = cross( a, b );
235
236 dE(:,i) = E(i) - E(1);
237 dh(:,i) = h_mom(:,i) - h_mom(:,1);
238 dhnorm(:,i) = norm( dh(:,i) );
239
240 end
241
242 %% -----
243
244 name = 'Problem 2a - Delta Specific Energy';
245 h = figure('name', name, 'position', positionA + [50 0 0 0]);
246 plot(t_J2drag, dE);
247 title('dE = E(t) - E(t_0)')
248 xlabel('Time (s)')
249 ylabel('km^2/s^2')
250 save_pdf(h, name);
251
252 %% Problem 2b: orbital elements
253
254
255 clear oe_drag Tp_drag
256
257 for i = 1:length(x_drag)
258 [oe_drag(i, :), Tp_drag(i,:)] = rv2oe( x_drag(i, :) );
259 oe_check(i, :) = rv2orb_OG( x_p(i, :) );
260 %%     Tp_p(i,:) = perigee_pass(oe_p(i,:), x_p(i,:));
261 end
262
263 for i = 1:length(x_J2drag)
264 [oe_J2drag(i, :), Tp_J2drag(i,:)] = rv2oe( x_J2drag(i, :) );
265 oe_check(i, :) = rv2orb_OG( x_p(i, :) );
266 end
267
268 %% -----
269
270 labels = {'a', 'e', 'i', '\omega', '\Omega', 'T_p'};

```

```

271 units = {'km', '', 'rad', 'rad', 'rad', 'rad'};
272 name = 'Problem 2b - Orbital Elements';
273 h = figure('name', name, 'position', positionB + [50 0 0 0]);
274 for i = 1:5
275 subplot(6,1,i)
276 plot(t_J2drag, oe_J2drag(:,i)); hold on; grid on;
277 plot(t_J2, oe_J2(:, i));
278 plot(t_drag, oe_drag(:, i));
279 plot(t_p, oe_p(:, i));
280 title(labels{i});
281 ylabel(units{i});
282 increase_ylim;
283 if i == 1
284 legend('J2 + drag', 'J2', 'drag', 'point mass');
285 end
286 end
287 subplot(6,1,6)
288 plot(t_J2drag, Tp_J2drag); hold on; grid on;
289 plot(t_J2, Tp_J2);
290 plot(t_drag, Tp_drag);
291 plot(t_p, Tp_p);
292 title('T_p')
293 ylabel('s')
294 sgttitle(name)
295 xlabel('Time (sec)')
296 save_pdf(h, name);
297
298
299 % -----
300 % differences in orbital elements
301
302 % 6 combinations:
303 % (1) p - J2
304 % (2) p - drag
305 % (3) p - J2drag
306 % (4) J2 - drag
307 % (5) J2 - J2drag
308 % (6) drag - J2drag
309
310 % add Tp, make your life easier
311
312 oe_p = [oe_p, Tp_p];
313 oe_J2 = [oe_J2, Tp_J2];
314 oe_J2drag = [oe_J2drag, Tp_J2drag];
315 oe_drag = [oe_drag, Tp_drag];
316
317 p_J2 = oe_p - oe_J2;
318 p_drag = oe_p - oe_drag;
319 p_J2drag = oe_p - oe_J2drag;
320 J2_drag = oe_J2 - oe_drag;
321 J2_J2drag = oe_J2 - oe_J2drag;
322 drag_J2drag = oe_drag - oe_J2drag;
323
324 labels = {'a', 'e', 'i', '\omega', '\Omega', 'T_p', 'T_p'};
325 units = {'km', '', 'rad', 'rad', 'rad', 'rad', 's'};
326 name = 'Problem 2b - Orbital Elements Point Mass Diff';
327 h = figure('name', name, 'position', positionB + [50 0 0 0]);
328 for i = 1:6
329 subplot(6,1,i)
330 plot(t_J2, p_J2(:,i)); hold on; grid on;
331 plot(t_J2, p_drag(:,i));
332 plot(t_J2, p_J2drag(:,i));
333
334 title(labels{i});
335 ylabel(units{i});
336 increase_ylim;
337 if i == 1

```

```

339 legend('p-J2', 'p-drag', 'p-J2drag');
340 end
341 end
342 s g title(name)
343 xlabel('Time (sec)')
344 save_pdf(h, name);
345
346 labels = {'a', 'e', 'i', '\omega', '\Omega', 'T_p', 'T_p'};
347 units = {'km', '', 'rad', 'rad', 'rad', 'rad', 's'};
348 name = 'Problem 2b - Orbital Elements J2 and Drag Diff';
349 h = figure('name', name, 'position', positionB + [50 0 0 0]);
350 for i = 1:6
351 subplot(6,1,i)
352 plot(t_J2, J2_drag(:,i)); hold on; grid on;
353 plot(t_J2, J2_J2drag(:,i));
354 plot(t_J2, drag_J2drag(:,i));
355
356 title(labels{i});
357 ylabel(units{i});
358 increase_ylim;
359 if i == 1
360 legend('J2-drag', 'J2-J2drag', 'drag-J2drag');
361 end
362 end
363 s g title(name)
364 xlabel('Time (sec)')
365 save_pdf(h, name);
366
367
368
369 %%%
370 %%% subfunctions
371
372 function save_pdf(h, name)
373
374 % save as cropped pdf
375 set(h, 'Units', 'Inches');
376 pos = get(h, 'Position');
377 set(h, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Inches', 'PaperSize', [pos(3), pos(4)]);
378 print(h, name, '-dpdf', '-r0')
379
380 end
381
382 function increase_ylim
383
384 ylims = get(gca, 'ylim');
385 yd = ylims(2) - ylims(1);
386 set(gca, 'ylim', [ylims(1) - 0.2*yd, ylims(2) + 0.2*yd]);
387
388 end
389
390 function U = comp_U(rv)
391
392 global mu J2 RE
393
394 x = rv(1);
395 y = rv(2);
396 z = rv(3);
397
398 % radius
399 r = sqrt(x^2 + y^2 + z^2);
400
401 % U point mass
402 Up = mu/r;
403
404 % latitude
405 phi = asin(z/r);
406
```

```

407 % U J2
408 UJ2 = -mu/r * J2 * (RE/r)^2 * ( 3/2 * ( sin(phi) )^2 - 1/2 );
409
410 % U point mass
411 U = Up + UJ2;
412
413 end
414
415 function Tp = perigee_pass(oe, x)
416
417 global mu
418
419 % perigee passing
420 a = oe(1);
421 e = oe(2);
422 nu = oe(6);
423 % r = norm([ x_pJ2(i,1) x_pJ2(i,2) x_pJ2(i,3) ]);
424 r = sqrt(x(1)^2 + x(2)^2 + x(3)^2);
425
426 n = sqrt(mu/a^3);
427 E = acos(r/a * cos(nu) + e);
428 M = E - e*sin(E);
429 Tp = M/n;
430
431 end

```

rv2oe function

```

1 function [oe, Tp] = rv2oe(rv)
2 % -----
3 % Inputs
4 % rv = [6x1] position and velocity states vector in ECI frame
5 %
6 % Outputs
7 % oe = [6x1] orbital elements: a, e, i, w, Omega, nu
8 % a = semimajor axis
9 % e = eccentricity
10 % i = inclination
11 % w = argument of perigee
12 % O = right ascension of ascending node
13 % M = mean anomaly
14 % Tp = time of perigee passing (optional output)
15 %
16
17 global mu
18
19 r = rv(1:3);
20 v = rv(4:6);
21
22 % angular momentum
23 h = cross(r, v);
24
25 % node vector
26 nhat = cross([0 0 1], h);
27
28 % eccentricity
29 evec = ( (norm(v)^2 - mu/norm(r))*r - dot(r, v)*v ) / mu;
30 e = norm(evec);
31
32 % specific mechanical energy
33 energy = norm(v)^2/2 - mu/norm(r);
34
35 % semi-major axis and p
36 if abs(e-1.0)>eps
37 a = -mu/(2*energy);
38 p = a*(1-e^2);
39 else

```

```

40 p = norm(h)^2/mu;
41 a = inf;
42 end
43
44 % inclination
45 i = acos(h(3)/norm(h));
46
47 % right ascension of ascending node (check for equatorial orbit)
48 if i > 0.000001
49 O = acos( nhat(1)/norm(nhat) );
50 else
51 O = 0;
52 end
53 if isnan(O)
54 O = 0;
55 end
56 if nhat(2)<0
57 O = 2*pi - O;
58 end
59
60 % argument of perigee
61 if e > 0.000001
62 w = acos( dot(nhat ,evec )/( norm(nhat)*e));
63 else
64 w = 0;
65 end
66 if isnan(w)
67 w = 0;
68 end
69 % if e(3)<0
70 % argp = 360-argp
71 % end
72
73 % true anomaly
74 nu = acos( dot(evec ,r) / (e*norm(r)) );
75 % if dot(r,v)<0
76 % nu = 360 - nu
77 % end
78
79 %Apply Quadrant Checks to All Determined Angles
80 % idx = nhat(2) < 0; if any(idx); O(idx) = 2*pi - O(idx); end
81 % idx = evec(3) < 0; if any(idx); w(idx) = 2*pi - w(idx); end
82 idx = dot(r,v) < 0; if any(idx); nu(idx) = 2*pi - nu(idx); end
83
84 oe = [a; e; i; w; O; nu];
85
86 % perigee passing time
87
88 rnorm = norm(r);
89 n = sqrt(mu/a^3);
90 E = acos( rnorm/a * cos(nu) + e );
91 M = E - e*sin(E);
92 Tp = M/n;
93
94 end

```

oe2rv function

```

1 function [rv] = oe2rv(oe)
2 %
3 % Purpose: Convert orbital elements and time past epoch to the classic
4 % Cartesian position and velocity
5 %
6 % Inputs:
7 % oe      = [6x1] or [1x6] orbital elements
8 % delta_t = t - t0 time interval
9 % mu      = Gravity * Mass (of Earth) constant

```

```

10  %
11  % Outputs:
12  % rv      = position and velocity state vector
13  % -----
14
15  % global delta_t
16  global mu
17
18  a      = oe(1);
19  e      = oe(2);
20  i      = oe(3);
21  w      = oe(4);
22  LAN    = oe(5);
23  % M0     = oe(6);
24  nu     = oe(6);
25
26  % nu is TRUE ANOMALY --> use Kepler's to calculate MEAN ANOMALY
27  % E = 2*atan( sqrt( (1-e)/(1+e) ) * tan(nu/2) );
28  % M = M0 + sqrt( mu/a^3 ) * (delta_t);
29  % E = keplerEq(M, e, eps);
30  % E = kepler(M, e);
31  % nu = 2*atan( sqrt( (1+e)/(1-e) ) * tan(E/2) );
32
33  p = a * ( 1 - e^2 );           % intermediate variable
34  r = p / ( 1 + e*cos(nu) );    % r_magnitude, polar coordinates
35
36  % Perifocal position and velocity
37
38  r_pf = [ r * cos(nu); r * sin(nu); 0 ];
39  v_pf = [ -sqrt(mu/p) * sin(nu); sqrt(mu/p) * (e + cos(nu)); 0 ];
40
41  % Perifocal to ECI transformation, 3-1-3 rotation
42  R11 = cos(LAN)*cos(w) - sin(LAN)*sin(w)*cos(i);
43  R12 = -cos(LAN)*sin(w) - sin(LAN)*cos(w)*cos(i);
44  R13 = sin(LAN)*sin(i);
45
46  R21 = sin(LAN)*cos(w) + cos(LAN)*sin(w)*cos(i);
47  R22 = -sin(LAN)*sin(w) + cos(LAN)*cos(w)*cos(i);
48  R23 = -cos(LAN)*sin(i);
49
50  R31 = sin(w)*sin(i);
51  R32 = cos(w)*sin(i);
52  R33 = cos(i);
53
54  R = [R11 R12 R13; R21 R22 R23; R31 R32 R33];
55
56  % Transform perifocal to ECI frame
57  r_vec = R * r_pf;
58  v_vec = R * v_pf;
59
60  % Position and state vector
61  rv = [r_vec; v_vec];
62
63  end
64
65  %% Kepler equation solvers
66
67  function E = keplerEq(M,e,eps)
68  % Function solves Kepler's equation M = E-e*sin(E)
69  % Input - Mean anomaly M [rad], Eccentricity e and Epsilon
70  % Output eccentric anomaly E [rad].
71  En = M;
72  Ens = En - (En-e*sin(En)-M)/(1 - e*cos(En));
73  while ( abs(Ens-En) > eps )
74  En = Ens;
75  Ens = En - (En - e*sin(En) - M)/(1 - e*cos(En));
76  end
77  E = Ens;

```

```

78 end
79
80 function E = kepler(M, e)
81 f = @(E) E - e * sin(E) - M;
82 E = fzero(f, M); %-- I would use M as the initial guess instead of 0
83 end

```

Integrated Equations of Motion

```

1
2 function dx = TwoBod_6states(t, x)
3 %
4 % Inputs
5 % t = [Nx1] time vector (orbit is Keplerian, doesn't matter)
6 % x = [6x1] state vector
7 %
8 % Outputs
9 % dx = [6x1] derivative of state vector
10 %

11
12 global mu
13
14 dx = zeros(6, 1); % force column vector
15
16 dx(1:3) = x(4:6);
17 % r_norm = sqrt( x(1)^2 + x(2)^2 + x(3)^2 );
18 r_norm = norm(x(1:3));
19 dx(4:6) = ( - mu / r_norm^3 ) * x(1:3);
20
21 end
22
23 function dx = TwoBod_UJ2(t, x)
24 %
25 % Inputs
26 % t = [Nx1] time vector (orbit is Keplerian, doesn't matter)
27 % x = [6x1] state vector
28 %
29 % Outputs
30 % dx = [6x1] derivative of state vector
31 %

32
33 global mu J2 RE
34
35 dx = zeros(6, 1); % force column vector
36
37 dx(1:3) = x(4:6);
38 % r_norm = sqrt( x(1)^2 + x(2)^2 + x(3)^2 );
39 r_norm = norm(x(1:3));
40 dx(4:6) = ( - mu / r_norm^3 ) * x(1:3);
41
42 % J2 stuff
43 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)
44 ^^(7/2));
44 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)
45 ^^(7/2));
45 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)
46 ^^(7/2));
46
47 dx(4:6) = dx(4:6) + [dUx; dUy; dUz];
48
49 end
50
51 function dx = TwoBod_UJ2(t, x)
52 %
53 % Inputs
54 % t = [Nx1] time vector (orbit is Keplerian, doesn't matter)
55 % x = [6x1] state vector

```

```

56  %
57  % Outputs
58  %   dx = [6x1] derivative of state vector
59  % -----
60
61  global mu J2 RE
62  global CD A m p0 r0_drag H dtheta
63
64  dx = zeros(6, 1);    % force column vector
65
66  dx(1:3) = x(4:6);
67  % r_norm = sqrt( x(1)^2 + x(2)^2 + x(3)^2 );
68  rnorm = norm(x(1:3));
69  dx(4:6) = ( - mu / rnorm^3 ) * x(1:3);
70
71  % drag stuff
72  pA = p0 * exp( -(rnorm - r0_drag)/H );
73  V_A = [dx(4) + dtheta * dx(2); dx(5) - dtheta * dx(1); dx(6)];
74  VA = sqrt( ( dx(4) + dtheta * dx(2) )^2 + ( dx(5) - dtheta * dx(1) )^2 + dx(6)^2 );
75
76  ddx_drag = - 1/2 * CD * A/m * pA * VA * V_A;
77  dx(4:6) = dx(4:6) + ddx_drag;
78
79 end
80
81 function dx = TwoBod_UJ2(t, x)
82 % -----
83 % Inputs
84 %   t = [Nx1] time vector (orbit is Keplerian, doesn't matter)
85 %   x = [6x1] state vector
86 %
87 % Outputs
88 %   dx = [6x1] derivative of state vector
89 % -----
90
91 global mu J2 RE
92 global CD A m p0 r0_drag H dtheta
93
94 dx = zeros(6, 1);    % force column vector
95
96 dx(1:3) = x(4:6);
97 % r_norm = sqrt( x(1)^2 + x(2)^2 + x(3)^2 );
98 rnorm = norm(x(1:3));
99 dx(4:6) = ( - mu / rnorm^3 ) * x(1:3);
100
101 % J2 stuff
102 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)
103 ^^(7/2));
104 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)
105 ^^(7/2));
106 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)
107 ^^(7/2));
108
109 dx(4:6) = dx(4:6) + [dUx; dUy; dUz];
110
111 % drag stuff
112 pA = p0 * exp( -(rnorm - r0_drag)/H );
113 V_A = [dx(4) + dtheta * dx(2); dx(5) - dtheta * dx(1); dx(6)];
114 VA = sqrt( ( dx(4) + dtheta * dx(2) )^2 + ( dx(5) - dtheta * dx(1) )^2 + dx(6)^2 );
115
116 ddx_drag = - 1/2 * CD * A/m * pA * VA * V_A;
117 dx(4:6) = dx(4:6) + ddx_drag;
118
119 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.