

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.

Introduction

Problem 1

Statement

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 $\ddot{\mathbf{r}} = -\nabla U$, 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:

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

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

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

Solution

Problem 1a

Statement

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

Problem 1b

Statement

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

Problem 1c

Statement

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

Problem 1d

Statement

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

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

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

Solution

Problem 2

Statement

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:

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

where

$$C_D = 2.0$$

$$A = 3.6 \text{ m}^2$$

$$m = 1350 \text{ kg}$$

$$\rho_0 = 4 \times 10^{-13} \text{ kg/m}^3$$

$$r_0 = 7298.145 \text{ km}$$

$$H = 200.0 \text{ km}$$

$$\dot{\theta} = 7.29211585530066 \times 10^{-5} \text{ rad/s}$$

$$\rho_A = \rho_0 e^{-(r-r_0)/H}$$

$$\underline{V}_A = \begin{bmatrix} \dot{x} + \dot{\theta}y \\ \dot{y} - \dot{\theta}x \\ \dot{z} \end{bmatrix}$$

$$V_A = \sqrt{(\dot{x} + \dot{\theta}y)^2 + (\dot{y} - \dot{\theta}x)^2 + \dot{z}^2}$$

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

Statement

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

Problem 2b

Statement

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?

Solution

Conclusion

Appendix

HW1 MATLAB code

```
1  % ASE 389 Orbit Determination
2  % HW 1
3  % Junette Hsin
4
5  %% Problem 1
6
7  global mu
8
9  mu = 398600.5 ;
10 r = [ -2436.45; -2436.45; 6891.037 ];
11 v = [ 5.088611; 5.088611; 0 ];
12
13 rv = [ r; v];
14 oe = rv2oe(rv);
15
16 %% Problem 2
17
18 rv = oe2rv(oe);
19
20 %% Problem 3
21
22 x = -2436.45;
23 y = -2436.45;
24 z = 6891.037;
25 mu = 398600.5;
26
27 dux = -2*mu*x / ( x^2 + y^2 + z^2 )^2;
28 duy = -2*mu*y / ( x^2 + y^2 + z^2 )^2;
29 duz = -2*mu*z / ( x^2 + y^2 + z^2 )^2;
30
31 rnorm = sqrt( x^2 + y^2 + z^2 );
32
33 dux = -mu*x / ( rnorm )^3;
34 duy = -mu*y / ( rnorm )^3;
35 duz = -mu*z / ( rnorm )^3;
36
37 %% Problem 4
38
39 a = oe(1);
40 T = abs(2 * pi * sqrt(a^3 / mu)); % period
41
42 toler = 1e-8; % 1e-14 accurate; 1e-6 coarse
43 options = odeset('reltol', toler, 'abstol', toler );
44 [t,x] = ode45(@TwoBod_6states, [0 2*T], [r; v], options);
45
46 for i = 1:length(t)
47 rnorm(i) = norm(x(i, 1:3));
48 vnorm(i) = norm(x(i, 4:6));
49 H(i, :) = cross(x(i, 1:3), x(i, 4:6));
50 hnorm(i) = norm(H(i, :));
51 end
52
53 anorm = 0;
54 for i = 2:length(t)
55 a = (x(i, 4:6) - x(i-1, 4:6)) / ( t(i) - t(i-1) );
56 anorm(i) = norm(a);
57 end
58
59 % -----
60
61 name = 'Problem 4: 2-Body EOM';
62 h = figure('name', name);
63
64 % position
```

```

65 subplot(3,1,1)
66 plot(t, rnorm); grid on
67 title('r norm')
68 ylabel('km')
69
70 % velocity
71 subplot(3,1,2)
72 plot(t, vnorm); grid on
73 title('v norm')
74 ylabel('km/s')
75
76 % acceleration
77 subplot(3,1,3)
78 plot(t, anorm); grid on
79 title('a norm');
80 ylabel('km/s^2')
81 xlabel('time (sec)')
82
83 sgtitle(name)
84
85 save_pdf(h, 'prob4_2bodeom');
86
87 % -----
88
89 name = 'Problem 4: 2-Body EOM Orbit';
90 h = figure('name', name);
91 plot3(x(:,1), x(:,2), x(:,3)); hold on; grid on;
92 plot3(x(1,1), x(1,2), x(1,3), 'o')
93 plot3(x(end,1), x(end,2), x(end,3), 'x')
94 xlabel('x (km)')
95 ylabel('y (km)')
96 zlabel('z (km)')
97 legend('orbit', 'start', 'end')
98
99 sgtitle(name)
100
101 save_pdf(h, 'prob4_2bodeom_orbit');
102
103 % -----
104
105 clear oe
106 for i = 1:length(t)
107     oe(i,:) = rv2oe(x(i,:));
108 end
109
110 labels = {'a', 'e', 'i', '\omega', '\Omega', '\nu'};
111 units = {'km', '', 'rad', 'rad', 'rad', 'rad'};
112 name = 'Problem 4: 2-Body Orbital Elements';
113 h = figure('name', name, 'position', [100 100 500 600]);
114 for i = 1:6
115     subplot(6,1,i)
116     plot(t, oe(:, i)); grid on
117     title(labels{i});
118     ylabel(units{i});
119 end
120 xlabel('time (sec)')
121 sgtitle(name)
122
123 save_pdf(h, 'prob4_2bdoes');
124
125 % -----
126
127 name = 'Problem 4: 2-Body Specific Angular Momentum';
128 h = figure('name', name);
129 subplot(2,1,1)
130 scatter3(H(:,1), H(:,2), H(:,3)); grid on
131 xlabel('x (km^2/s)')
132 ylabel('y (km^2/s)')

```

```

133 zlabel('z (km^2/s)')
134 title('h (scatter plot)')
135
136 subplot(2,1,2)
137 plot(t, hnorm); grid on
138 xlabel('time (sec)')
139 ylabel('km^2/s')
140 title('h norm vs time')
141
142 sgtitle(name)
143
144 save_pdf(h, 'prob4_angmom')
145
146 %% Problem 5
147
148 % specific kinetic energy
149 for i = 1:length(t)
150 T(i) = 0.5 * vnorm(i)^2;
151 U(i) = mu / rnorm(i);
152 end
153 E = T - U;
154
155 name = 'Problem 4: 2-Body Specific Energy';
156 h = figure('name', name);
157 subplot(2,1,1)
158 plot(t, E); grid on; hold on;
159 plot(t, T);
160 plot(t, U);
161 ylabel('km^2/s^2')
162 legend('Total', 'Kinetic', 'Potential')
163 title('Total Specific Energy: Kinetic - Potential')
164 subplot(2,1,2)
165 plot(t, [0 diff(E)]); grid on
166 title('Change in Total Specific Energy')
167 xlabel('Time (sec)')
168 ylabel('km^2/s^2')
169 sgtitle(name)
170
171 save_pdf(h, 'prob5_energy')
172
173 %% subfunctions
174
175 function save_pdf(h, name)
176
177 % save as cropped pdf
178 set(h,'Units','Inches');
179 pos = get(h,'Position');
180 set(h,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize',[pos(3), pos(4)])
181 print(h,name,'-dpdf','-r0')
182
183 end

```

rv2oe function

```

1 function oe = rv2oe(rv)
2 % -----
3 % Inputs
4 %   rv = [6x1] position and velocity states vector
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 %       Omega  = right ascension of ascending node
13 %       nu     = true anomaly

```

```

14  % -----
15
16  global mu
17
18  r = rv(1:3);
19  v = rv(4:6);
20
21  % angular momentum
22  h = cross(r,v);
23
24  % node vector
25  nhat = cross([0 0 1], h);
26
27  % eccentricity
28  evec = ( (norm(v)^2 - mu/norm(r))*r - dot(r,v)*v ) / mu;
29  e = norm(evec);
30
31  % specific mechanical energy
32  energy = norm(v)^2/2 - mu/norm(r);
33
34  % semi-major axis and p
35  if abs(e-1.0)>eps
36  a = -mu/(2*energy);
37  p = a*(1-e^2);
38  else
39  p = norm(h)^2/mu;
40  a = inf;
41  end
42
43  % inclination
44  i = acos(h(3)/norm(h));
45
46  % right ascension of ascending node (check for equatorial orbit)
47  if i > 0.000001
48  Omega = acos( nhat(1)/norm(nhat) );
49  else
50  Omega = 0;
51  end
52  if isnan(Omega)
53  Omega = 0;
54  end
55  if nhat(2)<0
56  Omega = 2*pi - Omega;
57  end
58
59  % argument of perigee
60  if e > 0.000001
61  w = acos( dot(nhat,evec)/(norm(nhat)*e) );
62  else
63  w = 0;
64  end
65  if isnan(w)
66  w = 0;
67  end
68  % if e(3)<0
69  %   argp = 360-argp
70  % end
71
72  % true anomaly
73  nu = acos( dot(evec,r) / (e*norm(r)) );
74  % if dot(r,v)<0
75  %   nu = 360 - nu
76  % end
77
78  oe = [a; e; i; w; Omega; nu];
79
80  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

```

s

References

- [1] Donald D. Mueller, J. W., and Bate, R. R., *Fundamentals of Astrodynamics*, Dover Publications, Inc., 1971.
- [2] Jah, M. K., “ASE 389P.4 Methods of Orbit Determination Module 3,” , January 2021.
- [3] Jah, M. K., “ASE 389P.4 Methods of Orbit Determination Module 2,” , January 2021.