

ASE387P.2 Mission Analysis and Design

Homework 4: Orbit Design

Junette Hsin

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

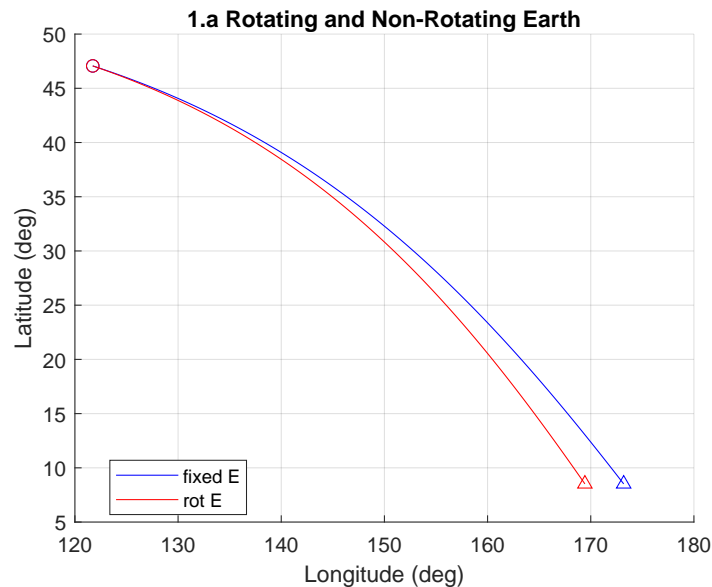
Problem 1: ISS Ground Tracks

A

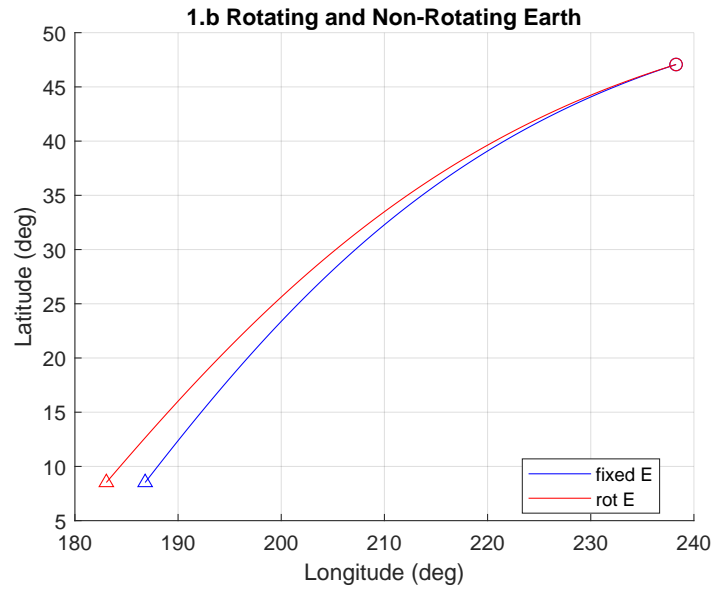
The following parameters were used to initialize the ISS orbit, which were taken from https://in-the-sky.org/spacecraft_elements.php?id=25544 on April 19:

- $a = 6794.588$ km
- $e = 0.00049$
- $i = 51.627$ deg
- $\omega = 40.1116$ deg
- $\Omega = 0$ deg
- $M = 70.88$

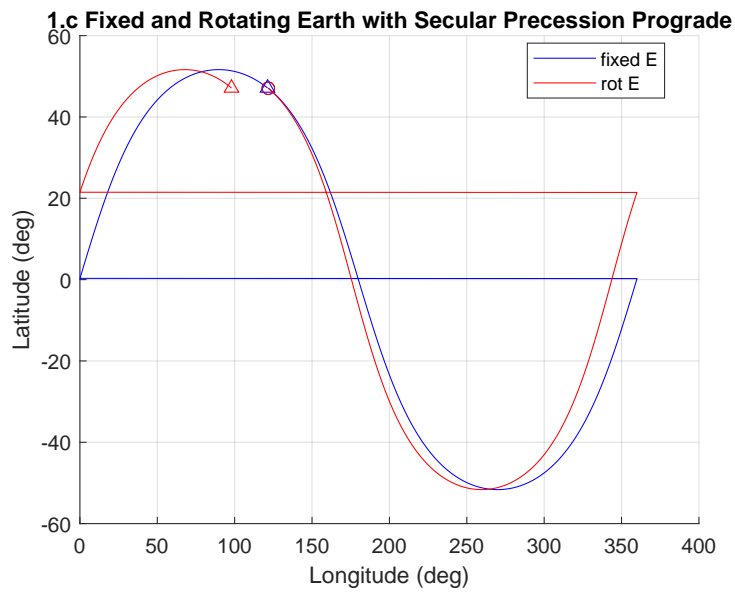
The right ascension of the ascending node, Ω , was set to 0 as stated in the problem statement.



B



C



To derive a formula for predicting the westward drift, first calculate the Draconitic period:

$$T_d = \frac{2\pi}{\dot{\omega} + \dot{M}} \quad (1)$$

Then calculate the rate of secular variation for the ascending node due to J2:

$$\dot{\Omega} = -\frac{3}{2}\bar{n}J_2\left(\frac{a_e}{\bar{a}}\right)^2\frac{1}{(1-\bar{e}^2)^2}\cos\bar{I} \quad (2)$$

\bar{a} , \bar{e} , and \bar{I} can be taken from the initial ISS orbit parameters, and \bar{n} is the mean motion of the orbit:

$$\bar{n} = \sqrt{\frac{\mu}{\bar{a}^3}} \quad (3)$$

Finally, the change in lambda was calculated as:

$$\Delta\lambda = -(w_E - \dot{\Omega})T_d \quad (4)$$

where w_E is the rotation rate of the Earth. $\Delta\lambda$ came out to **-22.870°**.

The calculated change in drift of the ascending node was taken from the difference between the initial longitude and the final longitude after one revolution, which came out to be **-22.790°**. The calculated drift was within 0.1° of the analytical prediction.

Problem 2: Orbit Design

Continuing from Equation 4, after m revolutions,

$$m\Delta\lambda = -(w_E - \dot{\Omega})T_d m \quad (5)$$

A condition for ground track repeat is that $m\Delta\lambda$ needs to be a multiple (let's say k) of 2π , or

$$k = \frac{m\Delta\lambda}{2\pi} = \frac{m(w_E - \dot{\Omega})T_d}{2\pi} = m \frac{T_d}{D_n} \quad (6)$$

For a ground track repeat orbit, the conditions for which the sub-satellite point repeatedly passes a geographical location (ϕ, λ) at regular intervals are given by:

$$(\omega_e - \dot{\Omega})D_n = k2\pi \quad (7)$$

$$(\dot{\omega} + \dot{M})T_d = m2\pi \quad (8)$$

As for frozen orbits, the largest perturbation on gravitational acceleration is due to J2, which is on the order of 10^{-3} . The next-largest perturbation is J3, which is on the order of 10^{-6} . Equation 2 gives the scalar secular variation on $\dot{\Omega}$ due to J2 which grows indefinitely. Equations 9 and 10 give the secular variations due to J2 for $\dot{\omega}$ and \dot{M} :

$$\dot{\omega} = -\frac{3}{4}\bar{n}J_2\left(\frac{a_e}{\bar{a}}\right)^2\frac{1}{(1-\bar{e}^2)^2}(1-5\cos^2\bar{I}) \quad (9)$$

$$\dot{M} = \bar{n}\left[1 - \frac{3}{4}\left(\frac{a_e}{\bar{a}}\right)^2J_2\frac{1}{(1-\bar{e}^2)^{\frac{3}{2}}}(1-3\cos^2\bar{I})\right] \quad (10)$$

J2 also leads to a short-period perturbation on all elements, but only semi-major axis is of interest:

$$\Delta a_{SP}(t) = \bar{a}J_2\left(\frac{a_e}{\bar{a}}\right)^2\left[\left(1 - \frac{3}{2}\sin^2\bar{I}\right)\left(\left(\frac{\bar{a}}{r}\right)^3 - \frac{1}{(1-\bar{e}^2)^{\frac{3}{2}}}\right) + \frac{3}{2}\left(\frac{\bar{a}}{r}\right)^3\sin^2\bar{I}\cos 2(\bar{w} + \bar{f})\right] \quad (11)$$

J3 exerts a long-period perturbation on all elements other than the semi-major axis, but only the eccentricity and perigee are of interest:

$$\Delta e_{LP}(t) = -\frac{1}{2}\frac{J_3}{J_2}\frac{a_e}{\bar{a}}\sin\bar{I}\sin\bar{w}(t) \quad (12)$$

$$\bar{e}\Delta\omega_{LP}(t) = -\frac{1}{2}\frac{J_3}{J_2}\frac{a_e}{\bar{a}}\frac{1}{(1-\bar{e}^2)}\sin\bar{I}\cos\bar{w}(t) \quad (13)$$

For frozen orbits, the average variation rates of e and w are set to 0. Thus, the mean argument of perigee should be around 90° .

$$w = 90^\circ \quad (14)$$

An orbit is sun-synchronous when the precession rate equals the mean motion of the Earth around the Sun. the Ω nodal rate needs to match the average rate of the Sun's motion projected onto the Earth's equator:

$$\frac{d\Omega}{dt} = \dot{\Omega} = \frac{360^\circ}{365.242 \text{ days/year}} = 0.9856^\circ/\text{day} \quad (15)$$

The angular precession for an Earth orbiting satellite is given by Equation 2. One can reform Equation 2 as a formula for inclination:

$$\bar{I} = \cos^{-1} \left[-\frac{2}{3} \frac{d\Omega}{dt} \frac{1}{J_2 \bar{n}} \left(\frac{\bar{a}(1 - \bar{e}^2)}{R_E} \right)^2 \right] \quad (16)$$

The process for determining the mean elements \bar{a} , \bar{e} , and \bar{I} involves making initial guesses and then minimizing the misclosure rate through an optimization routine:

$$\epsilon = m(w_E - \dot{\Omega}) - k(\dot{\omega} - \dot{M}) \quad (17)$$

Initial computed states for position and velocity will be integrated and iterated until a sun-synchronous, frozen, and repeated ground track orbit is found.

B

Problem 3

The calculations for this section were taken from the paper **Five Special Types of Orbits Around Mars** (2010 Liu, Baoyin, and Ma). A frozen orbit is possible on Mars. Mars' $J_2 = 1.95545\text{e-}3$ and $J_3 = 3.14498\text{e-}5$. In comparison, the Earth $J_2 = 1.08263\text{e-}3$ and $J_3 = -2.53266\text{e-}6$. The same positive sign between J_2 and J_3 must be accounted for to prevent the desired eccentricity from becoming negative. For Earth, ω is set to 90° . For Mars, ω must be set to around 270° .

The average variation rate of e from Equation 12 is given by:

$$\dot{e} = \frac{3nJ_3R_p^3 \sin \bar{I}}{4a^3(1 - e^2)^2} \left(\frac{5}{2} \sin^2 \bar{I} - 2 \right) \cos \omega \quad (18)$$

And from Equation 13, the average variation rate of ω is given by:

$$\dot{\omega} = \frac{3\bar{n}J_2R_p^2}{2a^2(1 - e^2)^2} \left[\left(2 - \frac{5}{2} \sin^2 \bar{I} \right) \left(1 + \frac{J_3R_p}{2J_2a(1 - e^2)} \left(\frac{\sin^2 \bar{I} - \bar{e}^2 \cos^2 \bar{I}}{\sin \bar{I}} \right) \frac{\sin \omega}{\bar{e}} \right) + \frac{3J_2R_p^2}{2\bar{a}^2(1 - e^2)^2} D \right] \quad (19)$$

where

$$D = (4 + \frac{7}{12} \bar{e}^2 + 2\sqrt{1 - \bar{e}^2}) - \sin^2 \bar{I} \left(\frac{103}{102} + \frac{3}{8} \bar{e}^2 + \frac{11}{2} \sqrt{1 - \bar{e}^2} \right) + \sin^4 \bar{I} \left(\frac{215}{418} - \frac{15}{32} \bar{e}^2 + \frac{15}{4} \sqrt{1 - \bar{e}^2} \right) + \dots H.O.T. \quad (20)$$

H.O.T. stands for higher order terms than J_3 . As stated in Problem 2, for frozen orbits, the average variation rates of e and ω are set to 0.

Solving the above equations yields the following for e and ω :

$$e = \frac{\frac{J_3R_p}{2J_2\bar{a}} \sin \bar{I} \sin \omega}{1 - \frac{3J_2R_p^2E}{\bar{a}^2(5\sin^2 \bar{I} - 4)}} \quad (21)$$

$$\omega = 270^\circ \quad (22)$$

Appendix

MATLAB code

```
1
2      %% Junette Hsin
3
4  % 0 = no plot. 1 = plot animation. 2 = plot all at once
5  plot_option = 2;
6
7  % Earth
8  Earth.a0 = 1.00000261;
9  Earth.da = 0.00000562;
10 Earth.e0 = 0.01671123;
11 Earth.de = -0.00004392;
12 Earth.i0 = -0.00001531;
13 Earth.dI = -0.01294668;
14 Earth.L0 = 100.46457166;
15 Earth.dL = 35999.37244981;
16 Earth.wbar0 = 102.93768193;
17 Earth.dwbar = 0.32327364;
18 Earth.Omega0 = 0;
19 Earth.dOmega = 0;
20
21
22 %% Problem 1
23
24 % (Ground Tracks) Consider the International Space Station (look up its orbital elements)
25 % in a Keplerian orbit. Assume that at  $t = 0$ , the ISS, its orbital ascending node, and the
26 % Greenwich Meridian are coincident. Note that all references to node crossings and ground
27 % tracks are from a viewpoint attached to the surface of the Earth.
28
29 % Draw and compare the ground tracks for 15-minute duration in two cases:
30 % i. A non-rotating Earth
31 % ii. A uniformly rotating Earth.
32
33 %  $t = 0$ , ascending node and Greenwich Meridian coincident
34
35 % ISS OEs (from https://in-the-sky.org/spacecraft\_elements.php?id=25544)
36 % e0 = 0.00048;
37 % i0 = 51.644 * pi/180; % deg --> rad
38 % w0 = 30.4757 * pi/180; % deg --> rad
39 % w0 = 0;
40 % O0 = 0 * pi/180; % deg --> rad
41 % M0 = 39.7178 * pi/180; % deg --> rad (should be true anomaly)
42 % M0 = (2*pi - w0); % rad (should be true anomaly)
43 % M0 = 0;
44
45 e0 = 0.00049;
46 i0 = 51.6427 * pi/180;
47 w0 = 40.1116 * pi/180;
48 O0 = 0;
49 M0 = 70.88 * pi/180;
50
51 % mean motion --> semimajor axis
52 n = 15.50094 / 86400 * (2*pi); % rev/day --> rad/s
53 mu_E_m3 = 3.986004418e14; % m^3/s^2
54 mu_E_km3 = mu_E_m3 * (1e-3)^3;
55 a0 = (mu_E_km3 / n^2)^(1/3);
56
57 R_E = 6378.1370; % km
58 w_E = 7.292115e-5; % rad/s
59
60 oe0 = [a0; e0; i0; w0; O0; M0];
61 rv0 = rvOrb.orb2rv(oe0, mu_E_km3);
62
63 % set ode45 params
64 rel_tol = 1e-10; % 1e-14 accurate; 1e-6 coarse
```

```

65 abs_tol = 1e-10;
66 options = odeset('reltol', rel_tol, 'abstol', abs_tol );
67
68 % propagate orbit
69 [t, rv_a] = ode45(@fn.EOM, [0 : 15*60], rv0, options);
70
71
72 % -----
73 % problem 1.a.i
74
75 lla = [];
76 lla_1_a = [];
77 for i = 1:length(rv_a)
78
79     lla(i,:) = ecef2lla(rv(i,1:3));
80     lla_1_a(i,:) = ecef2lla_1(rv_a(i,:), R_E, mu_E_km3);
81     lla_1_a(i, 1:2) = lla_1_a(i, 1:2) * 180/pi;
82
83 end
84 % fn.plot3_xyz(rv);
85
86 % -----
87 % problem 1.a.ii
88
89 % Axis 3 rotation matrix
90 [lla_rot_a, rv_rot_a] = lla_rv_rot(t, rv_a, w_E, R_E, mu_E_km3);
91
92
93 % -----
94 % PLOT
95 fname = '1.a Rotating and Non-Rotating Earth';
96 plot_option = 2;
97 plot_gt(plot_option, fname, lla_1_a, lla_rot_a)
98 fn.savePDF(gcf)
99
100
101 %%% problem 1.b
102
103 i0 = (180 - 51.644) * pi/180;    % deg --> rad
104
105 % mean motion --> semimajor axis
106 n = 15.50094 / 86400 * (2*pi);    % rev/day --> rad/s
107 mu_E_m3 = 3.986004418e14;    % m^3/s^2
108 mu_E_km3 = mu_E_m3 * ( 1e-3 )^3 ;
109 a0 = (mu_E_km3 / n^2)^(1/3);
110
111 R_E = 6378.1370;    % km
112 w_E = 7.292115e-5;    % rad/s
113
114 oe0 = [a0; e0; i0; w0; O0; M0];
115 rv0 = rvOrb.orb2rv(oe0, mu_E_km3);
116
117 % set ode45 params
118 rel_tol = 1e-10;    % 1e-14 accurate; 1e-6 coarse
119 abs_tol = 1e-10;
120 options = odeset('reltol', rel_tol, 'abstol', abs_tol );
121
122 % propagate orbit
123 [t, rv_b] = ode45(@fn.EOM, [0 : 15*60], rv0, options);
124
125
126 % -----
127 % Problem 1.b.i
128
129 lla = [];
130 lla_1_b = [];
131 for i = 1:length(rv_b)
132

```

```

133 % lla_1_b(i,:) = ecef2lla(rv(i,1:3));
134 lla_1_b(i,:) = ecef2lla_1(rv_b(i,:), R_E, mu_E_km3);
135 lla_1_b(i, 1:2) = lla_1_b(i, 1:2) * 180/pi;
136
137 end
138 % fn.plot3_xyz(rv);
139
140
141 % -----
142 % problem 1.b.ii
143
144 % Axis 3 rotation matrix
145 [lla_rot_b, rv_rot_b] = lla_rv_rot(t, rv_b, w_E, R_E, mu_E_km3);
146
147
148 % -----
149 % PLOT
150 fname = '1.b Rotating and Non-Rotating Earth';
151 plot_option = 2;
152 plot_gt(plot_option, fname, lla_1_b, lla_rot_b)
153 fn.savePDF(gcf)
154
155
156 %% problem 1.c.i
157
158 % PROGRADE
159
160 % Constants
161 mu = mu_E_km3;
162 J2 = 1.082e-3;
163
164 % O Precession Calcs
165 Odot = -(3/2)*n*(R_E / a0)^2 * J2 * (1/(1-e0^2)^(1/2)) * cos(i0); % O precession
166 sprintf('Odot precession: %.3f', Odot)
167
168 % ISS OEs (from https://in-the-sky.org/spacecraft_elements.php?id=25544)
169 i0 = 51.644 * pi/180; % deg --> rad
170 % M0 = 0;
171
172 oe0 = [a0; e0; i0; w0; O0; M0];
173 rv0 = rvOrb.orb2rv(oe0, mu_E_km3);
174
175 % set ode45 params
176 rel_tol = 1e-10; % 1e-14 accurate; 1e-6 coarse
177 abs_tol = 1e-10;
178 options = odeset('reltol', rel_tol, 'abstol', abs_tol);
179
180 % propagate orbit
181 T = 2*pi*sqrt(a0^3/mu_E_km3);
182 [t, rv_c] = ode45(@fn.EOM_J2, [0 : T], rv0, options);
183
184 % final oe
185 oef = rvOrb.rv2orb(rv_c(end,:), mu_E_km3);
186 Odot = oe0(5) - oef(5);
187
188 lla = [];
189 lla_1_c = [];
190 for i = 1:length(rv_c)
191
192 % lla(i,:) = ecef2lla(rv(i,1:3));
193 lla_1_c(i,:) = ecef2lla_1(rv_c(i,:), R_E, mu_E_km3);
194 lla_1_c(i, 1:2) = lla_1_c(i, 1:2) * 180/pi;
195
196 end
197 % fn.plot3_xyz(rv);
198
199
200 % -----

```

```

201 % Axis 3 rotation matrix
202 [lla_rot_c , rv_rot_c] = lla_rv_rot(t, rv_c, w_E, R_E, mu_E_km3);
203
204
205 % -----
206 % PLOT
207
208 fname = '1.c Fixed and Rotating Earth with Secular Precession Prograde';
209 plot_option = 2;
210 plot_gt(plot_option, fname, lla_1_c, lla_rot_c)
211 fn.savePDF(gcf)
212
213 % -----
214 % RETROGRADE
215
216 % Constants
217 mu = mu_E_km3;
218 J2 = 1.082e-3;
219
220 % O Precession Calcs
221 Odot = -(3/2)*n*(R_E / a0)^2 * J2 * (1/(1-e0^2)^(1/2)) * cos(i0); % O precession
222 sprintf('Odot precession: %.3f', Odot)
223
224 % ISS OEs (from https://in-the-sky.org/spacecraft_elements.php?id=25544)
225 i0 = (180 - 51.644) * pi/180; % deg --> rad
226
227 oe0 = [a0; e0; i0; w0; O0; M0];
228 rv0 = rvOrb.orb2rv(oe0, mu_E_km3);
229
230 % set ode45 params
231 rel_tol = 1e-10; % 1e-14 accurate; 1e-6 coarse
232 abs_tol = 1e-10;
233 options = odeset('reltol', rel_tol, 'abstol', abs_tol);
234
235 % propagate orbit
236 T = 2*pi*sqrt(a0^3/mu_E_km3);
237 [t, rv_c] = ode45(@fn.EOM_J2, [0 : T], rv0, options);
238
239 % final oe
240 oef = rvOrb.rv2orb(rv_c(end,:), mu_E_km3);
241 Odot = oe0(5) - oef(5);
242
243 lla = [];
244 lla_1_c = [];
245 for i = 1:length(rv_c)
246
247     lla(i,:) = ecef2lla(rv(i,1:3));
248     lla_1_c(i,:) = ecef2lla_1(rv_c(i,:), R_E, mu_E_km3);
249     lla_1_c(i, 1:2) = lla_1_c(i, 1:2) * 180/pi;
250
251 end
252 % fn.plot3_xyz(rv);
253
254
255 % -----
256 % Axis 3 rotation matrix
257 [lla_rot_c , rv_rot_c] = lla_rv_rot(t, rv_c, w_E, R_E, mu_E_km3);
258
259
260 % -----
261 % PLOT
262
263 fname = '1.c Fixed and Rotating Earth with Secular Precession Retrograde';
264 plot_option = 2;
265 plot_gt(plot_option, fname, lla_1_c, lla_rot_c)
266 fn.savePDF(gcf)
267
268 %%% Problem 1.c.ii

```



```

269
270 oe_c0 = rvOrb.rv2orb(rv0, mu_E_km3);
271 dt = 1;
272
273 Odot = -3/2 * n * J2 * ( R_E / norm(rv0(1:3)) )^2 * 1/( 1-e0^2 )^2 * cos(i0);
274 wdot = -3/4 * n * (R_E / norm(rv0(1:3)))^2 * J2 * (1 - 5*cosd(i0)^2) / (1-e0^2)^2;
275 Mdot = n * (1-3/4*(R_E / norm(rv0(1:3)))^2 * J2 * (1 - 3*cosd(i0)^2) / (1-e0^2)^(3/2));
276
277 oe_c = oe_c0;
278 rv_c2 = rv0';
279 for i = 1 : T
280
281     oe = oe_c0;
282
283     % augment mean anomaly
284     M0 = oe_c0(6);
285     M = M0 + Mdot * (dt*i);
286     M = mod(M, 2*pi);
287     oe(6) = M;
288
289     % augment RAAN
290     O0 = oe_c0(5);
291     O = O0 + Odot * (dt*i);
292     O = mod(O, 2*pi);
293     oe(5) = O;
294
295     % augment perigee
296     w0 = oe_c0(4);
297     w = w0 + wdot * (dt*i);
298     w = mod(w, 2*pi);
299     oe(4) = w;
300
301     oe_c(i+1,:) = oe;
302     rv_c2(i+1,:) = rvOrb.orb2rv(oe, mu_E_km3);
303
304 end
305
306 lla = [];
307 lla_1_c2 = [];
308 for i = 1:length(rv_c2)
309
310     lla(i,:) = ecef2lla(rv(i,1:3));
311     lla_1_c2(i,:) = ecef2lla_1(rv_c2(i,:), R_E, mu_E_km3);
312     lla_1_c2(i, 1:2) = lla_1_c2(i, 1:2) * 180/pi;
313
314 end
315
316
317
318 % -----
319 % Axis 3 rotation matrix
320 [lla_rot_c2, rv_rot_c2] = lla_rv_rot(t, rv_c2, w_E, R_E, mu_E_km3);
321
322 for i = 1:length(rv_rot_c2)
323     % lla(i,:) = ecef2lla(rv(i,1:3));
324     lla_rot_c2(i,:) = ecef2lla_1(rv_rot_c2(i,:), R_E, mu_E_km3);
325     lla_rot_c2(i, 1:2) = lla_rot_c2(i, 1:2) * 180/pi;
326
327 end
328
329 % fn.plot3_xyz(rv);
330 % -----
331 % PLOT
332
333 fname = '1.c Fixed and Rotating Earth with Secular Precession';
334 plot_option = 2;
335 plot_gt(plot_option, fname, lla_1_c2, lla_rot_c2)
336 fn.savePDF(gcf)

```

```

337
338 %% Problem 1.c.ii check westward drift of ascending crossing of ground track
339
340 Td = 2*pi / (wdot + Mdot);
341 Dn = 2*pi / ( w_E - Odot );
342
343 u0 = w0 + M0;
344
345 u = u0 + udot * T + 2*pi/udot;
346
347 dlambd = -(w_E - Odot) * Td * 180/pi;
348
349 pred = lla_rot_c2(end,2) - lla_rot_c2(1,2);
350 % pred = pred / T;
351
352
353
354 %% PROBLEM 3 - MARS!!
355
356 clc
357
358 mu_M_km3 = 0.042828e6;
359
360 a0 = 3897;
361 i0 = 60 * pi/180;
362 e0 = 0.0063414;
363 w0 = 270 * pi/180;
364 O0 = 0;
365 M0 = 0;
366
367 oe0 = [a0; e0; i0; w0; O0; M0];
368 rv0 = rvOrb.orb2rv(oe0, mu_M_km3);
369
370
371 % set ode45 params
372 rel_tol = 1e-10; % 1e-14 accurate; 1e-6 coarse
373 abs_tol = 1e-10;
374 options = odeset('reltol', rel_tol, 'abstol', abs_tol );
375
376 test = [3983.88414289582
377         52.4285652190091
378        -0.943967564657958
379         0.0696070532710085
380         1.63206285335835
381         3.08662446538281] ;
382
383 % propagate orbit
384 T = 2*pi*sqrt(a0^3/mu_M_km3);
385 [t, rv_M] = ode45(@fn.EOM_Mars_J2_J3, [0 : 0.01 : T*10], test, options);
386
387 figure()
388 fn.plot3_xyz(rv_M);
389
390
391 %% subfunctions
392
393 function [lla_rot, rv_rot] = lla_rv_rot(t, rv_a, w_E, R_E, mu_E_km3)
394
395     theta0 = 0;
396     dt = t(2) - t(1);
397
398     lla_rot = ecef2lla_1(rv_a(1,:) ', R_E, mu_E_km3)';
399     lla_rot(1, 1:2) = lla_rot(1, 1:2) * 180/pi;
400     rv_rot = rv_a(1,:);
401
402     for i = 2:length(rv_a)
403
404         theta = -w_E * dt * (i-1) + theta0;

```

```

405     r_rot = fn.rotate_xyz(rv_a(i,1:3), theta, 3);
406     v_rot = fn.rotate_xyz(rv_a(i,4:6), theta, 3);
407     rv_rot(i,:) = [r_rot; v_rot];
408
409     lla_rot(i,:) = ecef2lla_1(rv_rot(i,:) ', R_E, mu_E_km3);
410     lla_rot(i, 1:2) = lla_rot(i, 1:2) * 180/pi;
411
412
413 end
414
415 end
416
417 % -----
418 function plot_gt(plot_option, fname, lla_1, lla_rot)
419
420 %     fname = '1.a Rotating Earth';
421 if plot_option == 1
422
423     figure('name', fname)
424     plot(lla_1(:, 2), lla_1(:, 1), 'b');
425     hold on; grid on;
426     lh = plot(lla_rot(:, 2), lla_rot(:, 1), 'r');
427     xl = xlim; yl = ylim;
428
429     cla;
430     xlim(xl); ylim(yl);
431     for i = 1:length(lla_rot)
432
433         cla
434
435         % fixed Earth
436         plot(lla_1(1,2), lla_1(1,1), 'bo');
437         lh1 = plot(lla_1(1:i, 2), lla_1(1:i, 1), 'b');
438         plot(lla_1(i, 2), lla_1(i, 1), 'b^');
439
440         % rotating Earth
441         plot(lla_rot(1,2), lla_rot(1,1), 'ro');
442         lh2 = plot(lla_rot(1:i, 2), lla_rot(1:i, 1), 'r');
443         lh = plot(lla_rot(i, 2), lla_rot(i, 1), 'r^');
444
445         legend([lh1 lh2], 'fixed E', 'rot E', 'location', 'best')
446
447         title(sprintf(' %d / %d', i, length(lla_rot) ));
448         pause(0.001)
449
450     end
451
452     title(fname)
453     xlabel('Longitude (deg)');
454     ylabel('Latitude (deg)');
455
456 elseif plot_option == 2
457
458     figure('name', fname)
459     hold on; grid on;
460
461     % fixed Earth
462     plot(lla_1(1,2), lla_1(1,1), 'bo');
463     lh1 = plot(lla_1(:, 2), lla_1(:, 1), 'b');
464     plot(lla_1(end, 2), lla_1(end, 1), 'b^');
465
466     % rotating Earth
467     plot(lla_rot(1,2), lla_rot(1,1), 'ro');
468     lh2 = plot(lla_rot(:, 2), lla_rot(:, 1), 'r');
469     lh = plot(lla_rot(end, 2), lla_rot(end, 1), 'r^');
470
471     legend([lh1 lh2], 'fixed E', 'rot E', 'location', 'best')
472

```

```

473     title(fname)
474     xlabel('Longitude (deg)');
475     ylabel('Latitude (deg)');
476
477 end
478
479 end
480
481 % -----
482 % ECEF to geodetic (lat, lon)
483 function lla = ecef2lla_1(rv_ecef, R_E, mu)
484
485 % extract data
486 r_x = rv_ecef(1); r_y = rv_ecef(2); r_z = rv_ecef(3);
487 r_norm = norm(rv_ecef);
488
489 % obtain OEs
490 oe = rvOrb.rv2orb(rv_ecef, mu);
491 a = oe(1); e = oe(2); i = oe(3);
492 w = oe(4); O = oe(5); nu = oe(6);
493
494 % solve for longitude
495 r_delta = sqrt(r_x^2 + r_y^2);
496 a_sin = asin(r_y / r_delta);
497 a_cos = acos(r_x / r_delta);
498
499 % find quadrant
500 if a_sin > 0
501     if a_cos > 0
502         a = abs(a_cos);
503     else
504         a = pi - abs(a_cos);
505     end
506 else
507     if a_cos > 0
508         a = 2*pi - abs(a_cos);
509     else
510         a = pi + abs(a_cos);
511     end
512 end
513
514 lon = mod(a, 2*pi);
515
516 % iterate for latitude
517 delta = asin(r_z / r_norm);
518
519 % assume first guess
520 lat0 = delta;
521 lat = lat0;
522
523 C = R_E / sqrt(1 - e^2 * (sin(lat))^2);
524 S = R_E * (1 - e^2) / sqrt(1 - e^2 * (sin(lat))^2);
525
526 tan_lat = (r_z + C * e^2 * sin(delta)) / r_delta;
527 lat = atan(tan_lat);
528 h = r_z / (sin(delta)) - S;
529
530 % loop
531 err = 1e-6;
532 while abs(lat - lat0) > 1e-4
533
534     C = R_E / sqrt(1 - e^2 * (sin(lat))^2);
535     S = R_E * (1 - e^2) / sqrt(1 - e^2 * (sin(lat))^2);
536
537     tan_lat = (r_z + C * e^2 * sin(delta)) / r_delta;
538     lat = atan(tan_lat);
539     if abs(90 - i) > 1
540         h = r_z / (cos(delta)) - C;

```

```

541         else
542             h = r_z / ( sin(delta) ) - S;
543         end
544     end
545 end
546
547 % h = h + R_E;
548 lla = [lat; lon; h];
549
550 end
551
552
553
554 % -----
555 % ECEF2LLA - convert earth-centered earth-fixed (ECEF)
556 %             cartesian coordinates to latitude, longitude,
557 %             and altitude
558 %
559 % USAGE:
560 % [lat,lon,alt] = ecef2lla(x,y,z)
561 %
562 % lat = geodetic latitude (radians)
563 % lon = longitude (radians)
564 % alt = height above WGS84 ellipsoid (m)
565 % x = ECEF X-coordinate (m)
566 % y = ECEF Y-coordinate (m)
567 % z = ECEF Z-coordinate (m)
568 %
569 % Notes: (1) This function assumes the WGS84 model.
570 %         (2) Latitude is customary geodetic (not geocentric).
571 %         (3) Inputs may be scalars, vectors, or matrices of the same
572 %             size and shape. Outputs will have that same size and shape.
573 %         (4) Tested but no warranty; use at your own risk.
574 %         (5) Michael Kleder, April 2006
575 function [lla] = ecef2lla_2(ecef)
576
577 x = ecef(1); y = ecef(2); z = ecef(3);
578
579 % WGS84 ellipsoid constants:
580 a = 6378137;
581 e = 8.1819190842622e-2;
582 % calculations:
583 b = sqrt(a^2*(1-e^2));
584 ep = sqrt((a^2-b^2)/b^2);
585 p = sqrt(x.^2+y.^2);
586 th = atan2(a*z,b*p);
587 lon = atan2(y,x);
588 lat = atan2((z+ep^2.*b.*sin(th).^3),(p-e^2.*a.*cos(th).^3));
589 N = a./sqrt(1-e^2.*sin(lat).^2);
590 alt = p./cos(lat)-N;
591 % return lon in range [0,2*pi)
592 lon = mod(lon,2*pi);
593 % correct for numerical instability in altitude near exact poles:
594 % (after this correction, error is about 2 millimeters, which is about
595 % the same as the numerical precision of the overall function)
596 k=abs(x)<1 & abs(y)<1;
597 alt(k) = abs(z(k))-b;
598
599 lla = [lat; lon; alt];
600
601 end

```