

ASE387P.2 Mission Analysis and Design

Homework 3

Junette Hsin

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

Problem 1

A

To calculate the synodic period, first, the period of Mars was calculated in years:

$$T_{Mars} = \frac{1}{\dot{L}_{Mars}} \times \frac{100 \text{ years}}{1 \text{ century}} \times \frac{360^\circ}{1 \text{ rev}} \quad (1)$$

The synodic period was then calculated with the following equation:

$$SP_{Mars} = \frac{T_{Mars}}{||T_{Mars} - 1||} \quad (2)$$

The synodic period came out to be 2.13526965089401.

B

All analyses for this section are for a desired 60 degree transfer angle in the clockwise direction from the initial / departure position.

The desired transfer angle of 60 degrees was defined to be the angle between the initial / departure longitude and the final / arrival longitude.

$$\Delta L_{des} = L_{Mf} - L_{Ei} \quad (3)$$

For an outbound transfer, the initial longitude would be Earth and the final longitude would be Mars. For inbound transfer, the initial longitude would be Mars and the final longitude would be Earth. Equation 3 could also be rewritten as:

$$L_{Mf} = \Delta L_{des} + L_{Ei} \quad (4)$$

The final longitude can also be found from the motion of the planets:

$$L_{Mf} = L_{Mi} + \Delta t \dot{L}_M \quad (5)$$

Combining Equations 4 and 5 result in:

$$\Delta t = \frac{\Delta L_{des} + L_{Ei} - L_{Mi}}{\dot{L}_M} \quad (6)$$

The document **aprx_pos_planets.pdf** was used to compute the orbital elements and Cartesian states of Earth and Mars. Once Δt and an initial time are known, the positions of the planets are known. The Lambert solver from HW 2 was used to find the time of flight for the minimum energy transfer.

The above algorithm was iterated until the TOF for the minimum energy transfer aligned with the time for a 60 degree transfer angle between departure and arrival positions to develop. For example, for an outbound trajectory from Earth to Mars, the 60 degree transfer angle is computed between the initial Earth position and the final Mars position. Mars will take some amount of time, which we will call Δt_M , to travel from its initial position at launch date to its final position at the arrival date. The algorithm was iterated until the TOF for the minimum energy transfer aligned with Δt_M .

1st Launch Window

The launch date (T0), arrival date (T1), and the time of flight for the 1st launch window are displayed in the figure below.

The longitude orbital elements are:

E (T0) longitude = 352.527823989418°

M (T0) longitude = 320.871016064409°

M (T1) longitude = 52.5278239893489°

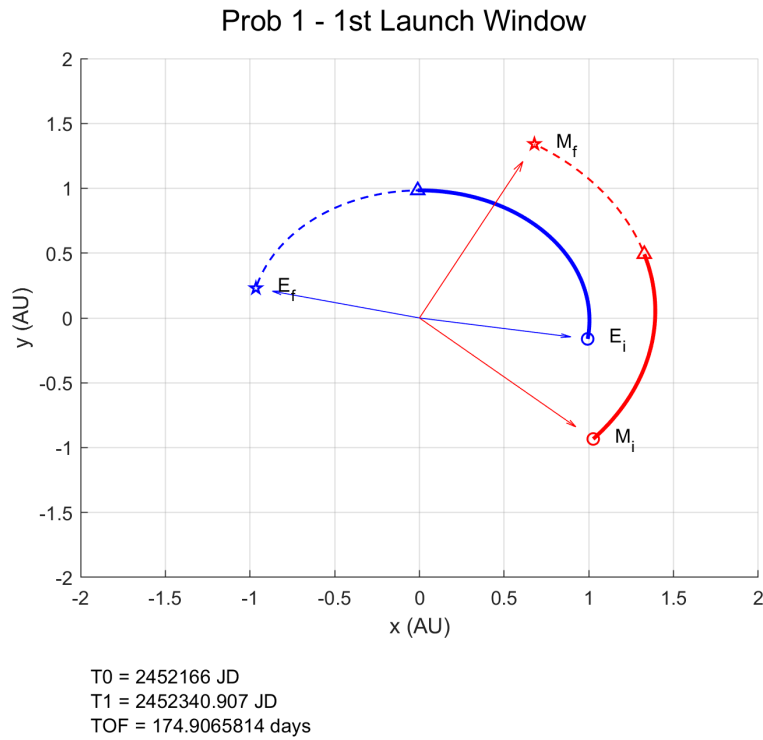


Fig. 1 1st Earth to Mars 60 degrees transfer

2nd Launch Window

The launch date (T0), arrival date (T1), and the time of flight for the 2nd launch window are displayed in the figure below.

E (T0) longitude = 51.0669403240372°

M (T0) longitude = 14.8080745374352 °

M (T1) longitude = 111.066940324018 °

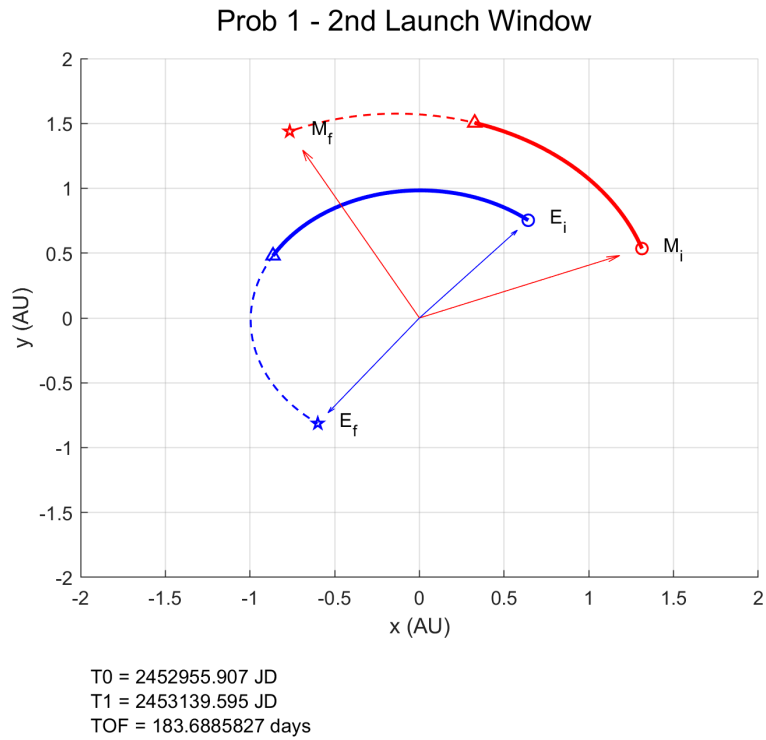


Fig. 2 2nd Earth to Mars 60 degrees transfer

C

The launch date (T0), arrival date (T1), and the time of flight for the inbound conic are displayed in the figure below.

E (T0) longitude = 75.1475543810473 °

E (T1) longitude = 279.017723434249 °

M (T0) longitude = 219.017723434434 °

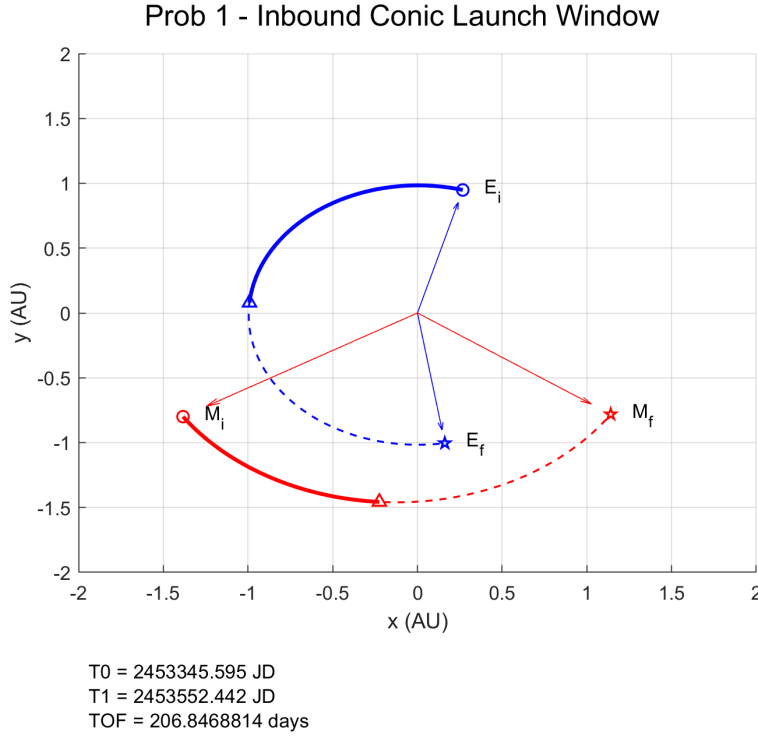


Fig. 3 Inbound Mars to Earth 60 degrees transfer

Problem 2

A

In the previous section, the angle between Earth and Mars was computed using the approximate longitude from **aprx_pos_planets.pdf**. For Problem 2, the 1.85° inclination of Mars' orbit was factored into the angle calculation.

To compute the desired inclination-corrected longitude for Mars, the departure vector, \vec{r}_{dep} was first rotated 60° degrees about the Earth orbit normal. In the ecliptic frame, this is the same as a rotation about the Z axis:

$$\vec{r}_{rot} = R_z(60^\circ) \vec{r}_{dep} \quad (7)$$

Then, the rotated vector, \vec{r}_{rot} was projected onto the Mars orbit normal, \hat{h} , which is inclined 1.85° about the Martian longitude of the ascending node.

$$\vec{r}_{proj,h} = \frac{\vec{r}_{rot} \cdot \hat{h}}{||\vec{r}_{rot}|| ||\hat{h}||} \hat{h} \quad (8)$$

Finally, the projected vector, $\vec{r}_{proj,h}$ was subtracted from the \vec{r}_{rot} to obtain the projection of the rotated vector onto the Martian orbital plane, $r_{proj,plane}$, which is also our desired arrival vector, r_{arr} .

$$\vec{r}_{arr} = \vec{r}_{rot} - \vec{r}_{proj,h} \quad (9)$$

The transfer angle between \vec{r}_{dep} and \vec{r}_{arr} will either be greater than or less than 60 degrees, depending on whether the Martian orbit normal is inclined toward or away from \vec{r}_{dep} . A minimization routine was then used to augment the rotation angle used to create \vec{r}_{rot} until the final transfer angle was within a tolerance of 60 degrees.

The launch date (T0), arrival date (T1), and the time of flight for the 2nd launch window are displayed in the figure below. **The z axis is zoomed in to show the exaggerated inclination of Mars' orbit.**

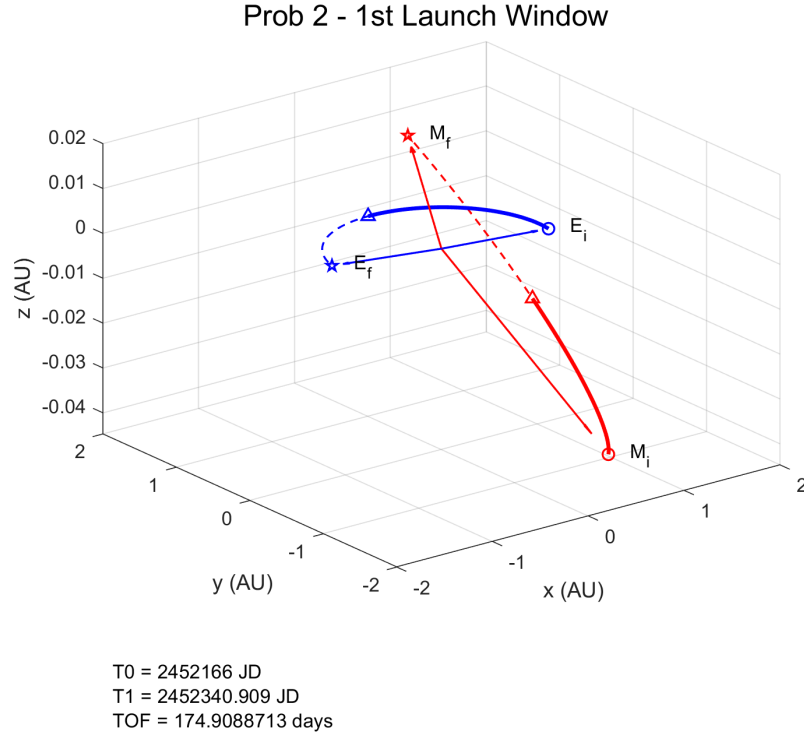
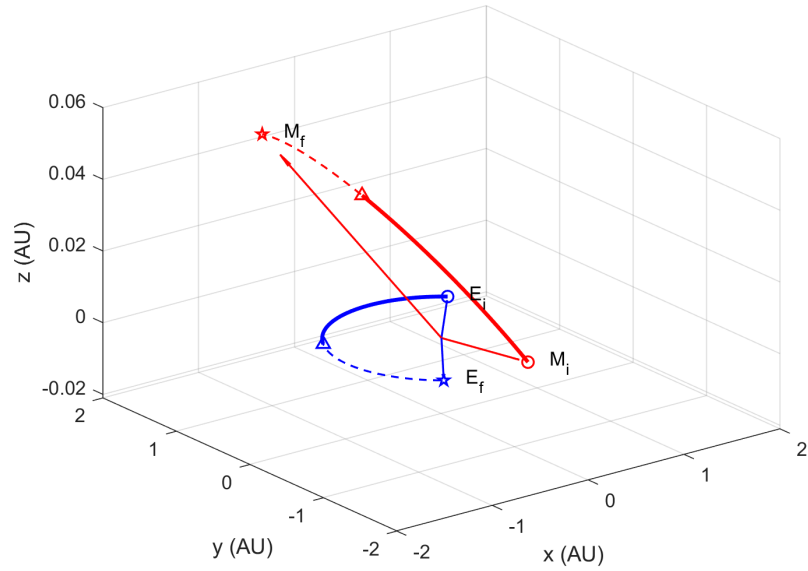


Fig. 4 1st Earth to Mars 60 degrees transfer with Mars inclination

Prob 2 - 2nd Launch Window



$T_0 = 2452955.909$ JD
 $T_1 = 2453139.624$ JD
TOF = 183.7152165 days

Fig. 5 2nd Earth to Mars 60 degrees transfer with Mars inclination

Prob 2 - Inbound Conic Launch Window

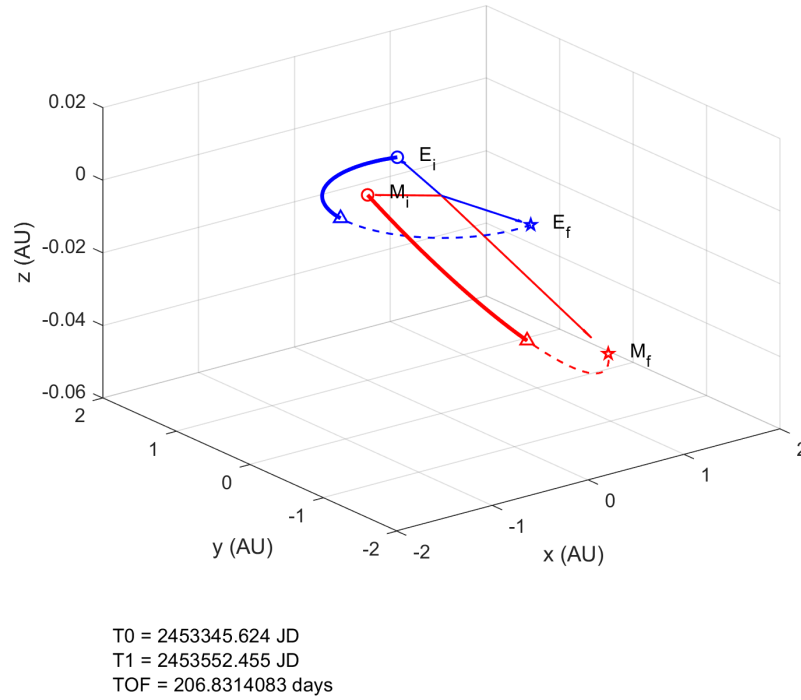


Fig. 6 Inbound Mars to Earth 60 degrees transfer with Mars inclination

B

For the 1st launch window, the Martian longitude at the arrival point is 52.529023989479° . In Problem 1, the longitude is 52.5278239893489° . That is a difference of $0.00120000013009758^\circ$.

For the 2nd launch window, the Martian longitude at the arrival point is 111.082097302582° . In Problem 1, the longitude is 111.06694032401° . That is a difference of 0.0151569785719943° .

For the inbound conic, the Earth longitude at the arrival point is 279.030980413165° . In Problem 1, the longitude is 279.017723434249° . That is a difference of 0.013256978916047° .

Problem 3

A

The Earth SOI was found to be $924644.363478889 \text{ km}$.

The Mars SOI was found to be $577259.24645827 \text{ km}$. Plots for the central and perturbing accelerations for the entire transfer duration as well as in the vicinity of the SOI are given below.

Problem 3 - Gravitational Acceleration

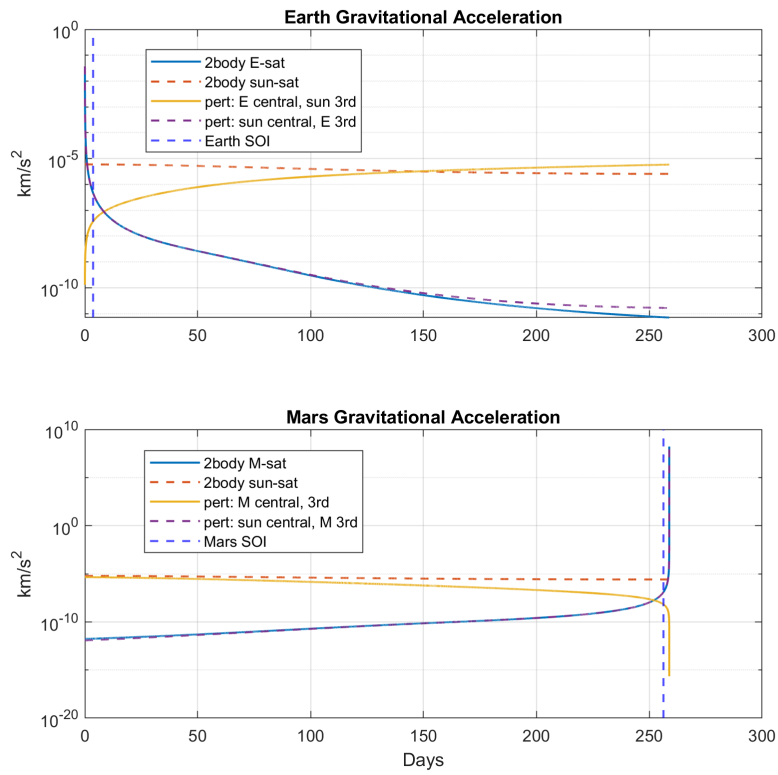


Fig. 7 Gravitational acceleration

Problem 3 - Gravitational Acceleration - SOI Vicinity

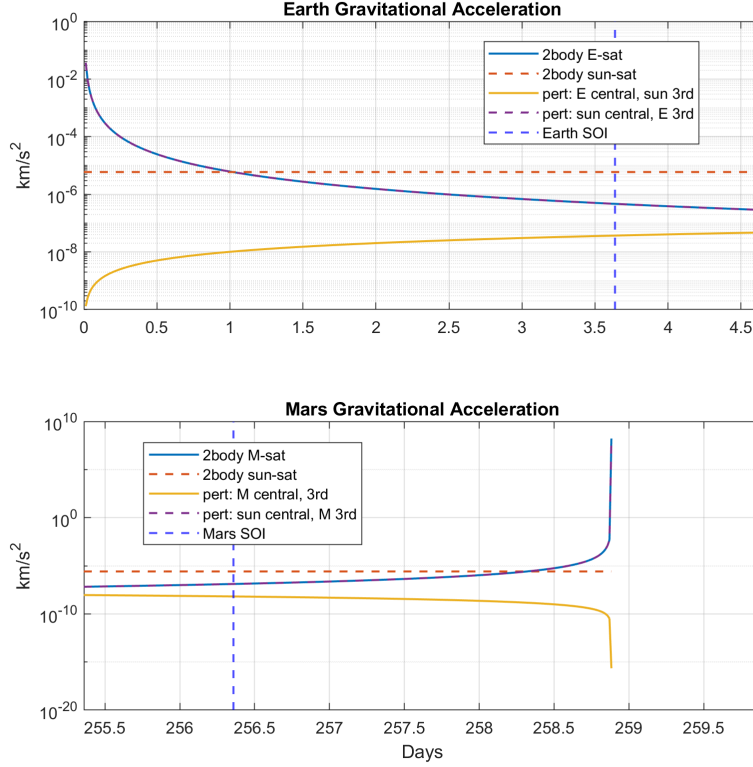


Fig. 8 Gravitational acceleration - SOI Vicinity

B

Plots for the ratios and rate of change of central to perturbing accelerations for the entire transfer duration as well as in the vicinity of the SOI are given below.

Figure 10 shows the ratios in the vicinity of the SOIs. The location to patch the conics should be applied at the edge of the sphere of influence. By considering only the gravitational force between the spacecraft and the body whose sphere of influence is considered, complicated N-body problems can be reduced to 2-body problems. Once the spacecraft is out of the smaller body's sphere of influence, then the gravitational force between the spacecraft and the larger body is used for trajectory calculations.

Problem 3 - Earth, Mars, and Sun acceleration ratios

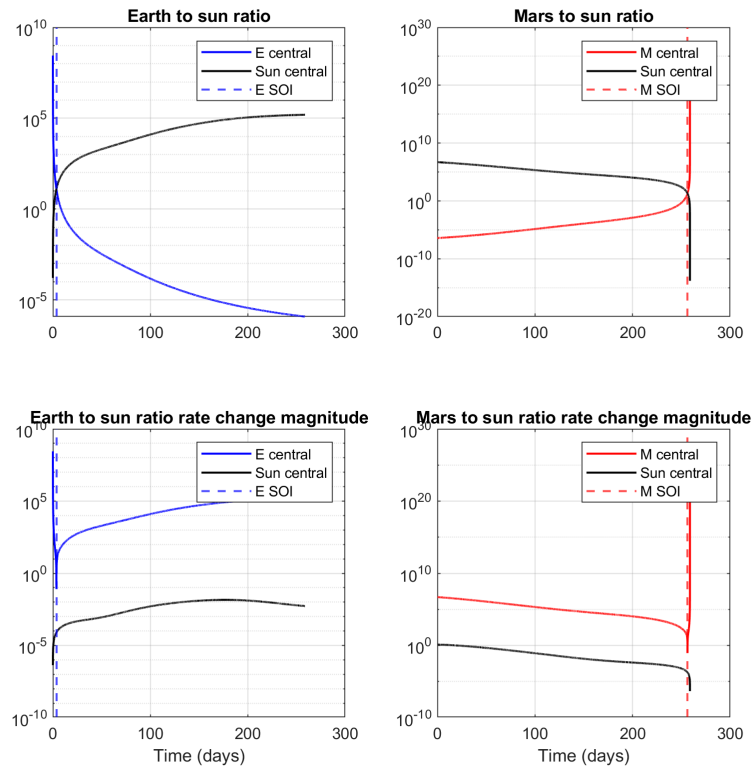


Fig. 9 Gravitational acceleration ratios

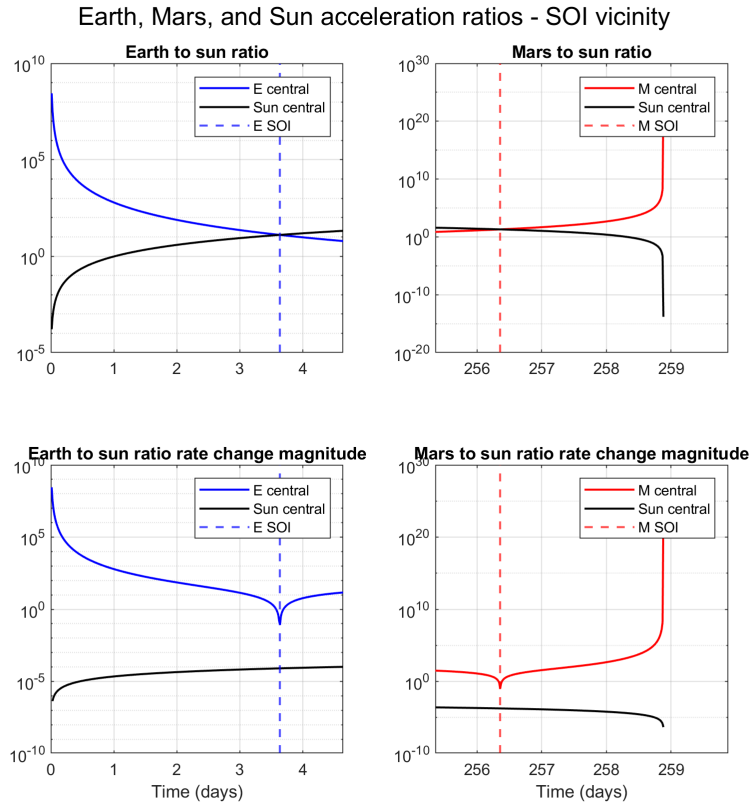


Fig. 10 Gravitational acceleration ratios - SOI Vicinity

C

The flight path angle is needed to compute the conditions for capture upon entering a planet's sphere of influence. The true anomaly and delta-V required for the capture burn can be calculated from the flight path angle.

The flight path angle is also used to compute the conditions for leaving the sphere of influence. The fly-by heliocentric orbit uses the flight path angle to calculate the angular momentum as well as the spacecraft true anomaly in its heliocentric orbit.

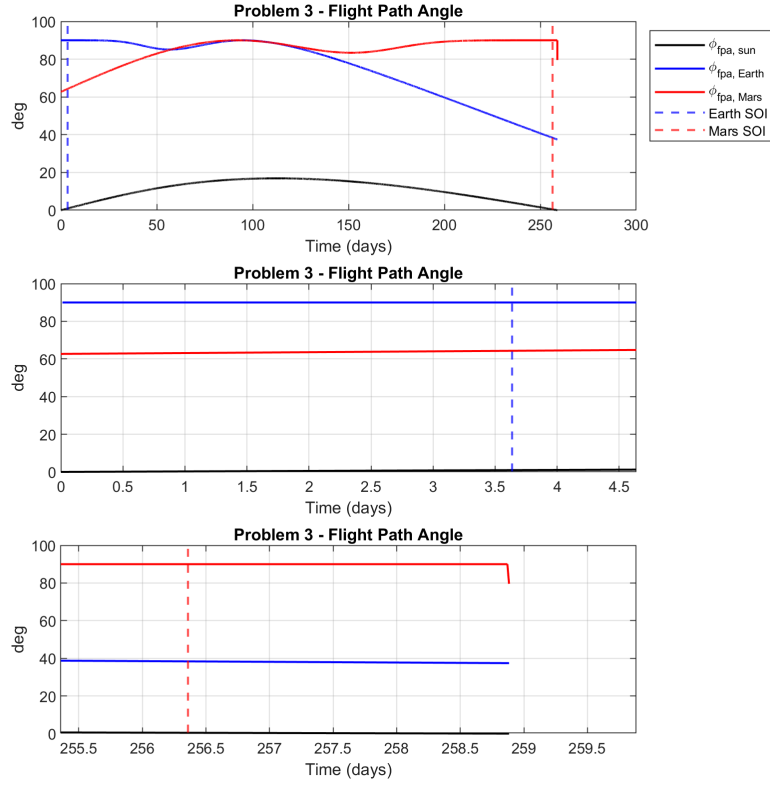


Fig. 11 Flight Path Angle

Problem 4

A

The separation distance is calculated:

$$\Delta = \sqrt{\frac{e+1}{e-1}} \quad (10)$$

The eccentricity is calculated from:

$$e = 1 + \frac{r_p v_{inf}^2}{\mu} \quad (11)$$

The turning angle can also be calculated from eccentricity:

$$\phi = 2 \sin^{-1} \frac{1}{e} \quad (12)$$

$\Delta \vec{v}$ can be calculated from:

$$\Delta \vec{v} = \vec{v}_f - \vec{v}_i \quad (13)$$

\vec{v}_f is a function of the turning angle and planet velocities:

$$\vec{v}_f = R_x(\phi)(\vec{v}_i - \vec{v}_p) + \vec{v}_p \quad (14)$$

B

To alter the transfer conic and phasing from the gravity-assist fly-by, the turning angle ϕ and the Δv must be added to the orbit. If the planet does not intersect with the target planet with the added velocity, then there is a possibility that no valid solution exist at the current phasing with the given entrance velocity. To change the entrance velocity or improve phasing, one could start the interplanetary trajectory design with the lowest possible flyby distance. If a solution is still not reached, then increment the distance or increase the time for a different phasing.

Appendix

MATLAB code

```
1
2 %% HW 2
3 %% Junette Hsin
4
5 % Keplerian elements
6 % a = semi-major axis
7 % e = eccentricity
8 % i = inclination
9 % L = mean longitude
10 % wbar = longitude of perihelion
11 % Omega = longitude of ascending node
12
13 clear; clc
14
15 % Earth
16 Earth.a0 = 1.00000261;
17 Earth.da = 0.00000562;
18 Earth.e0 = 0.01671123;
19 Earth.de = -0.00004392;
20 Earth.I0 = -0.00001531;
21 Earth.dI = -0.01294668;
22 Earth.L0 = 100.46457166;
23 Earth.dL = 35999.37244981;
24 Earth.wbar0 = 102.93768193;
25 Earth.dwbar = 0.32327364;
26 Earth.Omega0 = 0;
27 Earth.dOmega = 0;
28
29 % Mars
30 Mars.a0 = 1.52371034;
31 Mars.da = 0.00001847;
32 Mars.e0 = 0.09339410;
33 Mars.de = 0.00007882;
34 Mars.I0 = 1.84969142;
35 Mars.dI = -0.00813131;
36 Mars.L0 = -4.55343205 + 360;
37 Mars.dL = 19140.30268499;
38 Mars.wbar0 = -23.94362959;
39 Mars.dwbar = 0.44441088;
40 Mars.Omega0 = 49.55953891;
41 Mars.dOmega = -0.29257343;
42
43 % sun mu (m^3/s^2)
44 mu_sun_m = 1.32712440018e20;
45 mu_sun_km = mu_sun_m / (1000^3);
46 mu = mu_sun_km;
47
48
49 %% synodic period
50
51 dL_E = Earth.dL;
52 dL_M = Mars.dL; % degrees per century
53
54 % period of Earth
```

```

55 T_E = 365.25;
56
57 % period of Mars
58 dL_M_degyear = dL_M / 100; % degs per year
59 T_M_years = (1/dL_M_degyear) * 360; % period (days per 360 deg)
60
61 % synodic period
62 SP_M = T_M_years / abs(T_M_years - 1);
63
64 disp('Synodic period:')
65 disp(SP_M)
66
67 %% find launch date
68
69 T0 = 2451545.0; % units = days
70
71 T = T0;
72 [dt_days1, tof1_1, tof1_2] = launch_date(T, Earth, Mars, mu);
73 % find 1st launch date
74 while abs(dt_days1 - tof1_1) > 1 && abs(dt_days1 - tof1_2) > 1
75
76     T = T + 1;
77     [dt_days1, tof1_1, tof1_2] = launch_date(T, Earth, Mars, mu);
78
79 end
80
81 % check
82 [r_E0, ~, oe_E0] = xyz_ecl(T, Earth);
83 [r_M0, ~, oe_M0] = xyz_ecl(T, Mars);
84 [r_E1, ~, oe_E1] = xyz_ecl(T + dt_days1, Earth);
85 [r_M1, ~, oe_M1] = xyz_ecl(T + dt_days1, Mars);
86 phi = acosd(dot(r_E0, r_M1) / (norm(r_E0)*norm(r_M1)))
87 disp('1st launch window: ')
88 disp('E0 longitude = ')
89 L_E0 = oe_E0(4)
90 disp('M0 longitude = ')
91 L_M0 = oe_M0(4)
92 disp('M1 longitude = ')
93 L_M1 = oe_M1(4)
94
95 T0 = T;
96 T1 = T0 + dt_days1;
97
98 % ok let's do a propagation ...
99
100 r_E_hist_T0T1 = [];
101 r_M_hist_T0T1 = [];
102 for i = 0 : 0.1 : dt_days1
103
104     Ti = T0 + i;
105
106     % Earth
107     [r_E, r_p, oe_E] = xyz_ecl(Ti, Earth);
108     r_E_hist_T0T1 = [r_E_hist_T0T1; r_E'];
109     oe_E_hist_T0T1(i+1,:) = oe_E';
110
111     % Mars
112     [r_M, r_p, oe_M] = xyz_ecl(Ti, Mars);
113     r_M_hist_T0T1 = [r_M_hist_T0T1; r_M'];
114     oe_M_hist_T0T1(i+1,:) = oe_M';
115
116 end
117
118 txt = {sprintf('T0 = %.10g JD', T0); ...
119         sprintf('T1 = %.10g JD', T1); ...
120         sprintf('TOF = %.10g days', dt_days1) ...
121         };

```

```

122 plot3_p1p2(r_E_hist_T0T1, r_M_hist_T0T1, r_E0, r_M0, r_E1, r_M1, 'Prob 1 - 1st Launch Window'
123 , txt)
124 fn.save_pdf(gcf)
125
126 %% find 2nd launch date
127
128 T = T1;
129 [dt_days1, tof1_1, tof1_2] = launch_date(T, Earth, Mars, mu);
130 % find 1st launch date
131 while abs(dt_days1 - tof1_1) > 1 && abs(dt_days1 - tof1_2) > 1
132
133     T = T + 1;
134     [dt_days1, tof1_1, tof1_2] = launch_date(T, Earth, Mars, mu);
135
136 end
137
138 % check
139 [r_E0, ~, oe_E0] = xyz_ecl(T, Earth);
140 [r_M0, ~, oe_M0] = xyz_ecl(T, Mars);
141 [r_E1, ~, oe_E1] = xyz_ecl(T + dt_days1, Earth);
142 [r_M1, ~, oe_M1] = xyz_ecl(T + dt_days1, Mars);
143 phi = acosd(dot(r_E0, r_M1) / (norm(r_E0)*norm(r_M1)))
144 disp('2nd launch window: ')
145 disp('E0 longitude = ')
146 L_E0 = oe_E0(4)
147 disp('M0 longitude = ')
148 L_M0 = oe_M0(4)
149 disp('M1 longitude = ')
150 L_M1 = oe_M1(4)
151
152 T0 = T;
153 T1 = T0 + dt_days1;
154
155 % ok let's do a propagation ...
156
157 r_E_hist_T0T1 = [];
158 r_M_hist_T0T1 = [];
159 for i = 0 : 0.1 : dt_days1
160
161     Ti = T0 + i;
162
163     % Earth
164     [r_E, r_p, oe_E] = xyz_ecl(Ti, Earth);
165     r_E_hist_T0T1 = [r_E_hist_T0T1; r_E'];
166     % oe_E_hist_T0T1(i+1,:) = oe_E';
167
168     % Mars
169     [r_M, r_p, oe_M] = xyz_ecl(Ti, Mars);
170     r_M_hist_T0T1 = [r_M_hist_T0T1; r_M'];
171     % oe_M_hist_T0T1(i+1,:) = oe_M';
172
173 end
174
175 txt = {sprintf('T0 = %.10g JD', T0); ...
176        sprintf('T1 = %.10g JD', T1); ...
177        sprintf('TOF = %.10g days', dt_days1) ...
178        };
179 plot3_p1p2(r_E_hist_T0T1, r_M_hist_T0T1, r_E0, r_M0, r_E1, r_M1, 'Prob 1 - 2nd Launch Window'
180 , txt)
181 fn.save_pdf(gcf)
182
183 %% inbound conic
184
185 T = T1;
186 [dt_days1, tof1_1, tof1_2] = launch_date(T, Mars, Earth, mu);
187 % find 1st launch date

```

```

188 while abs(dt_days1 - tof1_1) > 1 && abs(dt_days1 - tof1_2) > 1
189
190     T = T + 1;
191     [dt_days1, tof1_1, tof1_2] = launch_date(T, Mars, Earth, mu);
192
193 end
194
195 % check
196 [r_E0, ~, oe_E0] = xyz_ecl(T, Earth);
197 [r_M0, ~, oe_M0] = xyz_ecl(T, Mars);
198 [r_E1, ~, oe_E1] = xyz_ecl(T + dt_days1, Earth);
199 [r_M1, ~, oe_M1] = xyz_ecl(T + dt_days1, Mars);
200 phi = acosd(dot(r_E0, r_M1) / (norm(r_E0)*norm(r_M1)))
201 disp('Inbound conic launch window: ')
202 disp('E0 longitude = ')
203 L_E0 = oe_E0(4)
204 disp('E1 longitude = ')
205 L_E1 = oe_E1(4)
206 disp('M0 longitude = ')
207 L_M1 = oe_M0(4)
208
209 T0 = T;
210 T1 = T0 + dt_days1;
211
212 % ok let's do a propagation ...
213
214 r_E_hist_T0T1 = [];
215 r_M_hist_T0T1 = [];
216 for i = 0 : 0.1 : dt_days1
217
218     Ti = T0 + i;
219
220     % Earth
221     [r_E, r_p, oe_E] = xyz_ecl(Ti, Earth);
222     r_E_hist_T0T1 = [r_E_hist_T0T1; r_E'];
223     % oe_E_hist_T0T1(i+1,:) = oe_E';
224
225     % Mars
226     [r_M, r_p, oe_M] = xyz_ecl(Ti, Mars);
227     r_M_hist_T0T1 = [r_M_hist_T0T1; r_M'];
228     % oe_M_hist_T0T1(i+1,:) = oe_M';
229
230 end
231
232 txt = {sprintf('T0 = %.10g JD', T0); ...
233        sprintf('T1 = %.10g JD', T1); ...
234        sprintf('TOF = %.10g days', dt_days1) ...
235        };
236 plot3_p1p2(r_E_hist_T0T1, r_M_hist_T0T1, r_E0, r_M0, r_E1, r_M1, 'Prob 1 - Inbound Conic
237 Launch Window', txt)
238 fn.save_pdf(gcf)
239
240 %% subfunctions
241
242 function h = plot3_quiver(r1, r2, style)
243
244     h = quiver3(r1(1), r1(2), r1(3), r2(1), r2(2), r2(3), style);
245
246 end
247
248 function plot3_p1p2(r_E_hist, r_M_hist, r_E0, r_M0, r_E1, r_M1, ftitle, plt_txt)
249
250 if ~exist('plt_txt', 'var')
251     plt_txt = '';
252 end
253
254 figure('name', ftitle, 'position', [100 100 600 700])

```



```

255
256 subplot(3,1,1:2)
257
258     fn.plot3_xyz(r_E_hist, 'b--');
259     hold on; grid on;
260     n = 1000;
261     fn.plot3_xyz(r_E_hist(1:n,:), 'b', 2);
262     fn.plot3_xyz(r_E_hist(n,:), 'b^');
263     fn.plot3_xyz(r_M_hist, 'r--');
264     fn.plot3_xyz(r_M_hist(1:n,:), 'r', 2);
265     fn.plot3_xyz(r_M_hist(n,:), 'r^');
266
267     % E0 point
268     fn.plot3_xyz(r_E0, 'bo');
269     plot3_quiver([0 0 0], r_E0, 'b');
270     txt = 'E_i';
271     text(r_E0(1), r_E0(2), r_E0(3), txt)
272
273     % M0 point
274     fn.plot3_xyz(r_M0, 'ro');
275     plot3_quiver([0 0 0], r_M0, 'r');
276     txt = 'M_i';
277     text(r_M0(1), r_M0(2), r_M0(3), txt)
278
279     % E1 point
280     fn.plot3_xyz(r_E1, 'bp');
281     plot3_quiver([0 0 0], r_E1, 'b');
282     txt = 'E_f';
283     text(r_E1(1), r_E1(2), r_E1(3), txt)
284
285     % M1 point
286     fn.plot3_xyz(r_M1, 'rp');
287     plot3_quiver([0 0 0], r_M1, 'r');
288     txt = 'M_f';
289     text(r_M1(1), r_M1(2), r_M1(3), txt)
290
291     xlim([-2 2])
292     ylim([-2 2])
293     % axis equal
294     xlabel('x (AU)')
295     ylabel('y (AU)')
296     zlabel('z (AU)')
297     view(0,90)
298
299 subplot(3,1,3)
300
301     fn.plt_txt(plt_txt)
302
303     sgtitle(ftitle)
304
305 end
306
307 function [dt_days1, tof1, tof2] = launch_date(T0, dep, arr, mu)
308
309
310     [r_dep0, ~, oe_dep0] = xyz_ecl(T0, dep);
311     [r_arr0, ~, oe_arr0] = xyz_ecl(T0, arr);
312
313     % km to au
314     km2au = 6.6845871226706E-9;
315     au2km = 1/km2au;
316
317     % mean longitude for Departure and Arrival
318     L_dep0 = oe_dep0(4);
319     L_arr0 = oe_arr0(4);
320
321     % desired delta longitude
322     dL_des = 60;

```

```

323
324     % required time (in days)
325
326     % dL = dL_des + L_M0 - L_E0;
327     % if dL < 0
328     %     dL = dL + 360;
329     % elseif dL > 360
330     %     while dL > 360
331     %         dL = dL - 360;
332     %     end
333     % end
334
335     % desired longitude for arrival
336     L_des = L_dep0 + dL_des;
337
338     % delta longitude
339     dL = L_des - L_arr0;
340     if dL < 0
341         dL = dL + 360;
342     elseif dL > 360
343         dL = dL - 360;
344     end
345
346     dL_arr = arr.dL;
347     dt_days1 = dL / dL_arr * 100 * 365.25;
348     % dt_days1 = (dL_des + L_E0 - L_M0) / dL_M * 100 * 365.25;
349     T1 = T0 + dt_days1;
350
351     % new time (60 deg phasing)
352     [r_dep1, ~, oe_dep1] = xyz_ecl(T1, dep);
353     [r_arr1, ~, oe_arr1] = xyz_ecl(T1, arr);
354     L_dep1 = oe_dep1(4);
355     L_arr1 = oe_arr1(4);
356
357     [ell_1, ell_2] = bett_lambert(r_dep0' * au2km, r_arr1' * au2km, mu);
358     tof1 = ell_1.tof / 86400;
359     tof2 = ell_2.tof / 86400;
360
361 end
362
363 %% HW 2
364 % Junette Hsin
365
366 % Keplerian elements
367 % a = semi-major axis
368 % e = eccentricity
369 % i = inclination
370 % L = mean longitude
371 % wbar = longitude of perihelion
372 % Omega = longitude of ascending node
373
374 clear; clc
375
376 % Earth
377 Earth.a0 = 1.00000261;
378 Earth.da = 0.00000562;
379 Earth.e0 = 0.01671123;
380 Earth.de = -0.00004392;
381 Earth.I0 = -0.00001531;
382 Earth.dI = -0.01294668;
383 Earth.L0 = 100.46457166;
384 Earth.dL = 35999.37244981;
385 Earth.wbar0 = 102.93768193;
386 Earth.dwbar = 0.32327364;
387 Earth.Omega0 = 0;
388 Earth.dOmega = 0;
389
390 % Mars

```

```

391 Mars.a0 = 1.52371034;
392 Mars.da = 0.00001847;
393 Mars.e0 = 0.09339410;
394 Mars.de = 0.00007882;
395 Mars.I0 = 1.84969142;
396 Mars.dI = -0.00813131;
397 Mars.L0 = -4.55343205 + 360;
398 Mars.dL = 19140.30268499;
399 Mars.wbar0 = -23.94362959;
400 Mars.dwbar = 0.44441088;
401 Mars.Omega0 = 49.55953891;
402 Mars.dOmega = -0.29257343;
403
404 % sun mu (m^3/s^2)
405 mu_sun_m = 1.32712440018e20;
406 mu_sun_km = mu_sun_m / (1000^3);
407 mu = mu_sun_km;
408
409
410 %% synodic period
411
412 dL_E = Earth.dL;
413 dL_M = Mars.dL; % degrees per century
414
415 % period of Earth
416 T_E = 365.25;
417
418 % period of Mars
419 dL_M_degyear = dL_M / 100; % degs per year
420 T_M_years = (1/dL_M_degyear) * 360; % period (days per 360 deg)
421
422 % synodic period
423 SP_M = T_M_years / abs(T_M_years - 1);
424
425 disp('Synodic period:')
426 disp(SP_M)
427
428 %% find launch date
429
430 T0 = 2451545.0; % units = days
431
432 T = T0;
433 [dt_days1, tof1_1, tof1_2] = launch_date(T, Earth, Mars, mu);
434 % find 1st launch date
435 while abs(dt_days1 - tof1_1) > 1 && abs(dt_days1 - tof1_2) > 1
436
437     T = T + 1;
438     [dt_days1, tof1_1, tof1_2] = launch_date(T, Earth, Mars, mu);
439
440 end
441
442 % check
443 [r_E0, ~, oe_E0] = xyz_ecl(T, Earth);
444 [r_M0, ~, oe_M0] = xyz_ecl(T, Mars);
445 [r_E1, ~, oe_E1] = xyz_ecl(T + dt_days1, Earth);
446 [r_M1, ~, oe_M1] = xyz_ecl(T + dt_days1, Mars);
447 phi = acosd(dot(r_E0, r_M1) / ( norm(r_E0)*norm(r_M1) ))
448 disp('1st launch window: ')
449 disp('E0 longitude = ')
450 L_E0 = oe_E0(4)
451 disp('M0 longitude = ')
452 L_M0 = oe_M0(4)
453 disp('M1 longitude = ')
454 L_M1 = oe_M1(4)
455
456 T0 = T;
457 T1 = T0 + dt_days1;
458

```

```

459 % ok let's do a propagation ...
460 r_E_hist_T0T1 = [];
461 r_M_hist_T0T1 = [];
462 for i = 0 : 0.1 : dt_days1
463
464     Ti = T0 + i;
465
466     % Earth
467     [r_E, r_p, oe_E] = xyz_ecl(Ti, Earth);
468     r_E_hist_T0T1 = [r_E_hist_T0T1; r_E'];
469     % oe_E_hist_T0T1(i+1,:) = oe_E';
470
471     % Mars
472     [r_M, r_p, oe_M] = xyz_ecl(Ti, Mars);
473     r_M_hist_T0T1 = [r_M_hist_T0T1; r_M'];
474     % oe_M_hist_T0T1(i+1,:) = oe_M';
475
476 end
477
478 txt = {sprintf('T0 = %.10g JD', T0); ...
479        sprintf('T1 = %.10g JD', T1); ...
480        sprintf('TOF = %.10g days', dt_days1) ...
481        };
482 plot3_p1p2(r_E_hist_T0T1, r_M_hist_T0T1, r_E0, r_M0, r_E1, r_M1, 'Prob 2 - 1st Launch Window'
483            , txt)
484
485 fn.save_pdf(gcf)
486
487 %% find 2nd launch date
488
489 T = T1;
490 [dt_days1, tof1_1, tof1_2] = launch_date(T, Earth, Mars, mu);
491 % find 1st launch date
492 while abs(dt_days1 - tof1_1) > 1 && abs(dt_days1 - tof1_2) > 1
493
494     T = T + 1;
495     [dt_days1, tof1_1, tof1_2] = launch_date(T, Earth, Mars, mu);
496
497 end
498
499 % check
500 [r_E0, ~, oe_E0] = xyz_ecl(T, Earth);
501 [r_M0, ~, oe_M0] = xyz_ecl(T, Mars);
502 [r_E1, ~, oe_E1] = xyz_ecl(T + dt_days1, Earth);
503 [r_M1, ~, oe_M1] = xyz_ecl(T + dt_days1, Mars);
504 phi = acosd(dot(r_E0, r_M1) / (norm(r_E0)*norm(r_M1)))
505 disp('2nd launch window: ')
506 disp('E0 longitude = ')
507 L_E0 = oe_E0(4)
508 disp('M0 longitude = ')
509 L_M0 = oe_M0(4)
510 disp('M1 longitude = ')
511 L_M1 = oe_M1(4)
512
513 T0 = T;
514 T1 = T0 + dt_days1;
515
516 % ok let's do a propagation ...
517
518 r_E_hist_T0T1 = [];
519 r_M_hist_T0T1 = [];
520 for i = 0 : 0.1 : dt_days1
521
522     Ti = T0 + i;
523
524     % Earth
525     [r_E, r_p, oe_E] = xyz_ecl(Ti, Earth);
526     r_E_hist_T0T1 = [r_E_hist_T0T1; r_E'];

```

```

526     %      oe_E_hist_T0T1(i+1,:) = oe_E';
527
528     % Mars
529     [r_M, r_p, oe_M] = xyz_ecl(Ti, Mars);
530     r_M_hist_T0T1 = [r_M_hist_T0T1; r_M'];
531     %      oe_M_hist_T0T1(i+1,:) = oe_M';
532
533     end
534
535     txt = { sprintf('T0 = %.10g JD', T0); ...
536             sprintf('T1 = %.10g JD', T1); ...
537             sprintf('TOF = %.10g days', dt_days1) ...
538             };
539     plot3_p1p2(r_E_hist_T0T1, r_M_hist_T0T1, r_E0, r_M0, r_E1, r_M1, 'Prob 2 - 2nd Launch Window'
540               , txt)
541     fn.save_pdf(gcf)
542
543     %% inbound conic
544
545     T = T1;
546     [dt_days1, tof1_1, tof1_2] = launch_date(T, Mars, Earth, mu);
547     % find 1st launch date
548     while abs(dt_days1 - tof1_1) > 1 && abs(dt_days1 - tof1_2) > 1
549
550         T = T + 1;
551         [dt_days1, tof1_1, tof1_2] = launch_date(T, Mars, Earth, mu);
552
553     end
554
555     % check
556     [r_E0, ~, oe_E0] = xyz_ecl(T, Earth);
557     [r_M0, ~, oe_M0] = xyz_ecl(T, Mars);
558     [r_E1, ~, oe_E1] = xyz_ecl(T + dt_days1, Earth);
559     [r_M1, ~, oe_M1] = xyz_ecl(T + dt_days1, Mars);
560     phi = acosd(dot(r_E0, r_M1) / ( norm(r_E0)*norm(r_M1) ))
561     disp('Inbound conic launch window: ')
562     disp('E0 longitude = ')
563     L_E0 = oe_E0(4)
564     disp('E1 longitude = ')
565     L_E1 = oe_E1(4)
566     disp('M0 longitude = ')
567     L_M1 = oe_M0(4)
568
569
570     T0 = T;
571     T1 = T0 + dt_days1;
572
573     % ok let's do a propagation ...
574
575     r_E_hist_T0T1 = [];
576     r_M_hist_T0T1 = [];
577     for i = 0 : 0.1 : dt_days1
578
579         Ti = T0 + i;
580
581         % Earth
582         [r_E, r_p, oe_E] = xyz_ecl(Ti, Earth);
583         r_E_hist_T0T1 = [r_E_hist_T0T1; r_E'];
584         %      oe_E_hist_T0T1(i+1,:) = oe_E';
585
586         % Mars
587         [r_M, r_p, oe_M] = xyz_ecl(Ti, Mars);
588         r_M_hist_T0T1 = [r_M_hist_T0T1; r_M'];
589         %      oe_M_hist_T0T1(i+1,:) = oe_M';
590
591     end
592

```

```

593     txt = { sprintf('T0 = %.10g JD', T0); ...
594             sprintf('T1 = %.10g JD', T1); ...
595             sprintf('TOF = %.10g days', dt_days1) ...
596             };
597     plot3_plp2(r_E_hist_T0T1, r_M_hist_T0T1, r_E0, r_M0, r_E1, r_M1, 'Prob 2 - Inbound Conic
    Launch Window', txt)
598     fn.save_pdf(gcf)
599
600 %% subfunctions
601
602 function plot3_plp2(r_E_hist, r_M_hist, r_E0, r_M0, r_E1, r_M1, ftitle, plt_txt)
603
604 if ~exist('plt_txt', 'var')
605     plt_txt = '';
606 end
607
608 figure('name', ftitle, 'position', [100 100 600 700])
609
610 subplot(3,1,1:2)
611
612     fn.plot3_xyz(r_E_hist, 'b--');
613     hold on; grid on;
614     n = 1000;
615     fn.plot3_xyz(r_E_hist(1:n,:), 'b', 2);
616     fn.plot3_xyz(r_E_hist(n,:), 'b^');
617     fn.plot3_xyz(r_M_hist, 'r--');
618     fn.plot3_xyz(r_M_hist(1:n,:), 'r', 2);
619     fn.plot3_xyz(r_M_hist(n,:), 'r^');
620
621 % E0 point
622     fn.plot3_xyz(r_E0, 'bo');
623     fn.plot3_quiver([0 0 0], r_E0, 'b');
624     txt = '      E_i';
625     text(r_E0(1), r_E0(2), r_E0(3), txt)
626
627 % M0 point
628     fn.plot3_xyz(r_M0, 'ro');
629     fn.plot3_quiver([0 0 0], r_M0, 'r');
630     txt = '      M_i';
631     text(r_M0(1), r_M0(2), r_M0(3), txt)
632
633 % E1 point
634     fn.plot3_xyz(r_E1, 'bp');
635     fn.plot3_quiver([0 0 0], r_E1, 'b');
636     txt = '      E_f';
637     text(r_E1(1), r_E1(2), r_E1(3), txt)
638
639 % M1 point
640     fn.plot3_xyz(r_M1, 'rp');
641     fn.plot3_quiver([0 0 0], r_M1, 'r');
642     txt = '      M_f';
643     text(r_M1(1), r_M1(2), r_M1(3), txt)
644
645     xlim([-2 2])
646     ylim([-2 2])
647     % axis equal
648     xlabel('x (AU)')
649     ylabel('y (AU)')
650     zlabel('z (AU)')
651     % view(0,90)
652
653 subplot(3,1,3)
654
655     fn.plt_txt(plt_txt)
656
657     sgtitle(ftitle)
658
659 end

```

```

660
661 function [dt_days1, tof1, tof2] = launch_date(T0, dep, arr, mu)
662
663
664 [r_dep0, ~, oe_dep0] = xyz_ecl(T0, dep);
665 [r_arr0, ~, oe_arr0] = xyz_ecl(T0, arr);
666
667 % km to au
668 km2au = 6.6845871226706E-9;
669 au2km = 1/km2au;
670
671 % mean longitude for Departure and Arrival
672 L_dep0 = oe_dep0(4);
673 L_arr0 = oe_arr0(4);
674
675 %% omg figure it out
676
677 h = [0 0 1]';
678 ri = r_dep0 / norm(r_dep0);
679 rf = r_arr0 / norm(r_arr0);
680
681 % arrival planet frame - X axis at right ascension node
682 arr_x = fn.rotate_xyz([1 0 0], arr.Omega0*pi/180, 3);
683
684 % arrival planet orbit normal
685 arr_z = cross(arr_x, rf);
686 arr_z = arr_z / norm(arr_z);
687 if arr_z(3) < 0
688     arr_z = -arr_z;
689 end
690
691 % arrival planet "y" axis
692 arr_y = cross(arr_z, arr_x);
693
694 % frame DCM
695 arr_C_dep = [arr_x'; arr_y'; arr_z'];
696 dep_C_arr = arr_C_dep';
697
698 % find phi
699 dL_des = 60;
700 dL = dL_des;
701 err = 1e-4;
702 [phi, rf, ri_proj_h, ri_rot] = calc_phi(dL_des, err, ri, arr_z);
703
704 if phi > dL
705     while abs(phi - dL) > err
706
707         dL_des = dL_des - err;
708         [phi, rf, ri_proj_h, ri_rot] = calc_phi(dL_des, err, ri, arr_z);
709
710     end
711 else
712     while abs(phi - dL) > err
713
714         dL_des = dL_des + err;
715         [phi, rf, ri_proj_h, ri_rot] = calc_phi(dL_des, err, ri, arr_z);
716
717     end
718 end
719
720
721
722
723 %%
724
725 % desired longitude for arrival
726 L_des = L_dep0 + dL_des;
727

```

```

728     % delta longitude
729     dL = L_des - L_arr0;
730     if dL < 0
731         dL = dL + 360;
732     elseif dL > 360
733         dL = dL - 360;
734     end
735
736     dL_arr = arr.dL;
737     dt_days1 = dL / dL_arr * 100 * 365.25;
738     % dt_days1 = (dL_des + L_E0 - L_M0) / dL_M * 100 * 365.25;
739     T1 = T0 + dt_days1;
740
741     % new time (60 deg phasing)
742     [r_dep1, ~, oe_dep1] = xyz_ecl(T1, dep);
743     [r_arr1, ~, oe_arr1] = xyz_ecl(T1, arr);
744     L_dep1 = oe_dep1(4);
745     L_arr1 = oe_arr1(4);
746
747     [ell_1, ell_2] = bett_lambert(r_dep0' * au2km, r_arr1' * au2km, mu);
748     tof1 = ell_1.tof / 86400;
749     tof2 = ell_2.tof / 86400;
750
751 end
752
753 function [phi, rf, ri_proj_h, ri_rot] = calc_phi(dL_des, err, ri, arr_z)
754
755     ri_rot = fn.rotate_xyz(ri, dL_des*pi/180, 3);
756
757     % now project ri_rot onto arrival orbit normal
758     ri_proj_h = dot(ri_rot, arr_z) * arr_z;
759
760     % obtain projection of ri_rot onto orbit plane
761     rf = ri_rot - ri_proj_h;
762     rf = rf / norm(rf);
763
764     phi = acosd(dot(ri, rf));
765
766 end
767
768 function [r_Mi, Ti] = find_trans_theta(dL_des, T0, Mars, Earth)
769
770     % get departure state
771     [r_E0, ~, oe_E0] = xyz_ecl(T0, Earth);
772     L_E0 = oe_E0(4);
773     r_dep = r_E0;
774
775     % get arrival state
776     [r_M0, ~, oe_M0] = xyz_ecl(T0, Mars);
777     L_Mi = oe_M0(4);
778     r_arr = r_M0;
779
780     r_dep = r_dep / norm(r_dep);
781     r_arr = r_arr / norm(r_arr);
782
783     theta = acosd(dot(r_arr, r_dep));
784     err = abs(theta - dL_des);
785
786     i = 0;
787     while err > 0.01 || L_E0 > L_Mi
788
789         % if Mars "ahead" of earth
790         if L_Mi > L_E0
791             % increment time "smart"
792             if err > 10
793                 di = 1;
794             elseif err > 1
795                 di = 0.1;

```



```

796         elseif err > 0.1
797             di = 0.01;
798         else
799             di = 0.001;
800         end
801
802         % if Earth "ahead" of Mars
803         else
804             di = 1;
805         end
806         i = i + di;
807         Ti = T0 + i;
808
809         % get arrival state
810         [r_Mi, ~, oe_Mi] = xyz_ecl(Ti, Mars);
811         r_arr = r_Mi;
812         L_Mi = oe_Mi(4);
813
814         % calc transfer angle
815         r_arr = r_arr / norm(r_arr);
816         theta = acosd(dot(r_arr, r_dep));
817
818         % transfer angle error
819         err = abs(theta - dL_des);
820
821     end
822
823 end
824
825 %%% HW 3
826 % Junette Hsin
827
828 % sun mu (m^3/s^2)
829 mu_sun_m = 1.32712440018e20;
830 mu_sun_km = mu_sun_m / (1000^3);
831 mu = mu_sun_km;
832
833 pos = [100 100 700 700];
834
835 %%% simplest case possible
836
837 % % Arrival (Mars), AU units
838 % r_norm_i = norm(X_sunE_hist(1:3));
839 % r_norm_f = norm(X_sunM_hist(1:3));
840
841 % semimajor axis
842 a_E = 1.4959787e11/1000;
843 a_M = 227.956e6;
844 r_norm_i = a_E;
845 r_norm_f = a_M;
846
847 % velocities
848 v_norm_i = sqrt(mu/r_norm_i);
849 v_norm_f = sqrt(mu/r_norm_f);
850
851 % initial vector
852 r_dep = [1 0 0] * r_norm_i;
853 v_dep = [0 1 0] * v_norm_i;
854 X_dep = [r_dep v_dep];
855
856 % final vector
857 r_arr = [-1 0 0] * r_norm_f;
858 v_arr = [0 -1 0] * v_norm_f;
859 X_arr = [r_arr v_arr];
860
861
862 %%% HOHMANN TRANSFER
863

```

```

864 % Arrival (Mars), AU units
865 r_norm_i = norm(X_dep(1:3));
866 r_norm_f = norm(X_arr(1:3));
867
868 a_trans = (r_norm_f + r_norm_i)/2;
869
870 v_init = sqrt( mu/r_norm_i );
871 v_fin = sqrt( mu/r_norm_f );
872
873 % delta v magnitude
874 v_trans_a = sqrt( 2*mu/r_norm_i - mu/a_trans );
875 v_trans_b = sqrt( 2*mu/r_norm_f - mu/a_trans );
876
877 dv_a = v_trans_a - v_init;
878 dv_b = v_fin - v_trans_b;
879 dv = norm(dv_a) + norm(dv_b);
880
881 % transfer time
882 tau_trans = pi * sqrt( a_trans^3 / mu );
883
884 % delta v direction
885 dv_init = X_dep(4:6) / norm(X_dep(4:6)) * v_trans_a;
886 dv_fin = X_arr(4:6) / norm(X_arr(4:6)) * v_trans_b;
887
888 % initial satellite state for Hohmann transfer
889 rv0_sat = X_dep;
890 rv0_sat(4:6) = dv_init;
891
892
893 %% propagate
894
895 % set ode45 params
896 rel_tol = 1e-10; % 1e-14 accurate; 1e-6 coarse
897 abs_tol = 1e-10;
898 options = odeset('reltol', rel_tol, 'abstol', abs_tol );
899
900 % delta time
901 dt = tau_trans / 20000;
902 % dt = 100;
903
904 % propagate satellite orbit
905 [t, X_sunsat_hist] = ode45(@fn.EOM, [0 : dt : tau_trans], rv0_sat, options);
906
907 % back-propagate Mars from arrival
908 [t, X_sunM_hist] = ode45(@fn.EOM, [0 : -dt : -tau_trans], X_arr, options);
909 X_sunM_hist = flip(X_sunM_hist);
910
911 % forward propagate Earth
912 [t, X_sunE_hist] = ode45(@fn.EOM, [0 : dt : tau_trans], X_dep, options);
913
914 % a useful vector
915 X_Esat_hist = -X_sunE_hist + X_sunsat_hist;
916 X_satE_hist = -X_Esat_hist;
917 X_Msat_hist = -X_sunM_hist + X_sunsat_hist;
918 X_satM_hist = -X_Msat_hist;
919
920 test_plot = 0;
921 % test if sat to Earth/Mars is correct
922 if test_plot == 1
923
924     figure()
925
926     plot3_xyz(X_sunsat_hist, 'g', 1.2)
927     hold on; grid on;
928     plot3_xyz(X_sunsat_hist + X_satE_hist, 'b--', 1.2);
929     plot3_xyz(X_sunsat_hist + X_satM_hist, 'r--', 1.2);
930
931     plot3( [X_sunsat_hist(1,1), X_sunsat_hist(1,1) + X_satM_hist(1,1)], ...

```

```

932         [X_sunsat_hist(1,2), X_sunsat_hist(1,2) + X_satM_hist(1,2)], ...
933         [X_sunsat_hist(1,3), X_sunsat_hist(1,3) + X_satM_hist(1,3)] );
934
935     plot3( [X_sunsat_hist(end,1), X_sunsat_hist(end,1) + X_satE_hist(end,1)], ...
936           [X_sunsat_hist(end,2), X_sunsat_hist(end,2) + X_satE_hist(end,2)], ...
937           [X_sunsat_hist(end,3), X_sunsat_hist(end,3) + X_satE_hist(end,3)] );
938
939     end
940
941     %% Hohmann plot
942
943     leg_hist = [];
944     ftitle = 'Problem 3 - Hohmann Transfer';
945     figure('name', ftitle, 'position', [50 50 700 500])
946
947     % departure
948     leg = quiver3(0, 0, 0, X_dep(1), X_dep(2), X_dep(3)); hold on; grid on;
949     leg_hist = [leg_hist; leg];
950
951     % arrival
952     leg = quiver3(0, 0, 0, X_arr(1), X_arr(2), X_arr(3));
953     leg_hist = [leg_hist; leg];
954
955     % Mars traj
956     leg = plot3_xyz(X_sunM_hist, 'r', 1);
957     leg_hist = [leg_hist; leg];
958     plot3_xyz(X_sunM_hist, 'ro', 1, 1);
959     plot3_xyz(X_sunM_hist, 'r^', 1, 'end');
960
961     % Earth traj
962     leg = plot3_xyz(X_sunE_hist, 'b', 1);
963     leg_hist = [leg_hist; leg];
964     plot3_xyz(X_sunE_hist, 'bo', 1, 1);
965     plot3_xyz(X_sunE_hist, 'b^', 1, 'end');
966
967     % sun
968     leg = scatter3(0, 0, 0, 'filled');
969     leg_hist = [leg_hist; leg];
970
971     % Hohmann
972     leg = plot3_xyz(X_sunsat_hist, 'g', 2);
973     leg_hist = [leg_hist; leg];
974     plot3_xyz(X_sunsat_hist, 'go', 2, 1);
975     plot3_xyz(X_sunsat_hist, 'g^', 2, 'end');
976
977     legend(leg_hist, 'Earth_{init}', 'Mars_{fin}', 'Mars traj', 'Earth traj', 'sun', 'Hohmann
978           ',
979           xlabel('x (km)')
980           ylabel('y (km)')
981           zlabel('z (km)')
982
983           view(0, 90)
984
985           title(ftitle)
986
987           axis equal
988
989     %% gravity
990
991     % GMs
992     mu_E = 3.986004418e5;
993     mu_sun = mu;
994     mu_M = 0.042828e6;
995
996     % mass
997     m_E = 5.9724e24;
998     m_sun = 1988500e24;

```

```

999     m_M = 0.64169e24;
1000
1001     % SOI
1002     r_SOI_Esun_ana = ( m_E/m_sun )^(2/5)*a_E;
1003     r_SOI_Msun_ana = ( m_M/m_sun )^(2/5)*a_M;
1004     disp('Analytical r_SOI_Esun norm: ')
1005     disp(norm(r_SOI_Esun_ana))
1006     disp('Analytical r_SOI_Msun norm: ')
1007     disp(norm(r_SOI_Msun_ana))
1008
1009     % initialize
1010     a_Esat_hist = [];
1011     a_Msat_hist = [];
1012     a_sunsat_hist = [];
1013     a_pert_Esun_hist = [];
1014     a_pert_sunE_hist = [];
1015     a_pert_Msun_hist = [];
1016     a_pert_sunM_hist = [];
1017
1018     const_vec = 0;
1019
1020     % determine gravity
1021     for i = 1:length(X_sunsat_hist)
1022
1023         % Get current states and positions
1024         X_sunsat = X_sunsat_hist(i,:);
1025         X_sunE   = X_sunE_hist(i,:);
1026         X_sunM   = X_sunM_hist(i,:);
1027
1028         r_sunsat = X_sunsat(1:3);
1029         r_satsun = -r_sunsat;
1030         r_sunE   = X_sunE(1:3);
1031         r_sunM   = X_sunM(1:3);
1032
1033         % Earth to satellite vector
1034         r_Esun = -r_sunE;
1035         r_Esat = r_Esun + r_sunsat;
1036         r_satE = -r_Esat;
1037
1038         % Mars to satellite vector
1039         r_Msun = -r_sunM;
1040         r_Msat = r_Msun + r_sunsat;
1041         r_satM = -r_Msat;
1042
1043         % central body accel Earth-satellite
1044         a_Esat = - mu_E * r_Esat / norm(r_Esat)^3;
1045         a_Esat_hist = [a_Esat_hist; norm(a_Esat)];
1046
1047         % central body accel sun-satellite
1048         a_sunsat = - mu_sun * r_sunsat / norm(r_sunsat)^3;
1049         a_sunsat_hist = [a_sunsat_hist; norm(a_sunsat)];
1050
1051         % disturbance (third body) Earth-sun
1052         % a_pert = - mu_E * r_Esat / norm(r_Esat)^3 - ...
1053         % mu_sun * ( r_satsun/norm(r_satsun)^3 + r_Esun/norm(r_Esun)^3);
1054         % Vallado disturbance
1055         % 3rd body pert: Earth-centered, sun pert
1056         a_pert_Esun = - mu_sun * ( r_satsun/norm(r_satsun)^3 - r_Esun/norm(r_Esun)^3 );
1057         a_pert_Esun_hist = [a_pert_Esun_hist; norm(a_pert_Esun)];
1058
1059         % 3rd body pert: sun-centered, Earth pert
1060         a_pert_sunE = - mu_E * ( r_satE/norm(r_satE)^3 - r_sunE/norm(r_sunE)^3 );
1061         a_pert_sunE_hist = [a_pert_sunE_hist; norm(a_pert_sunE)];
1062
1063         % Mars to satellite vector
1064         r_Msun = -r_sunM;
1065         r_Msat = r_Msun + r_sunsat;
1066

```

```

1067 % central body accel Mars-satellite
1068 a_Msat = - mu_M * r_Msat / norm(r_Msat)^3;
1069 a_Msat_hist = [a_Msat_hist; norm(a_Msat)];
1070
1071 % 3rd-body pert: Mars-centered, sun pert
1072 a_pert_Msun = - mu_sun * ( r_satsun/norm(r_satsun)^3 - r_Msun/norm(r_Msun)^3 );
1073 a_pert_Msun_hist = [a_pert_Msun_hist; norm(a_pert_Msun)];
1074
1075 % 3rd-body pert: sun-centered, Mars pert
1076 a_pert_sunM = - mu_M * ( r_satM/norm(r_satM)^3 - r_sunM/norm(r_sunM)^3 );
1077 a_pert_sunM_hist = [a_pert_sunM_hist; norm(a_pert_sunM)];
1078
1079
1080 end
1081
1082
1083 %% SOI crossings
1084
1085 dt = t(2) - t(1);
1086 for i = 1:length(a_Msat_hist)
1087
1088     % Earth central-perturbation body
1089     ratio_Esun(i,:) = a_Esat_hist(i) / a_pert_Esun_hist(i);
1090     ratio_sunE(i,:) = a_sunsat_hist(i) / a_pert_sunE_hist(i);
1091
1092     % Earth-sat norm
1093     r_Esat_norm(i,:) = norm(X_Esat_hist(i, 1:3));
1094
1095     % Mars central-perturbation body
1096     ratio_Msun(i,:) = a_Msat_hist(i) / a_pert_Msun_hist(i);
1097     ratio_sunM(i,:) = a_sunsat_hist(i) / a_pert_sunM_hist(i);
1098
1099     % Mars-sat norm
1100     r_Msat_norm(i,:) = norm(X_Msat_hist(i, 1:3));
1101
1102     if i > 1
1103         dratio_Esun(i,:) = norm(ratio_Esun(i,:) - ratio_Esun(i-1,:))/dt;
1104         dratio_sunE(i,:) = norm(ratio_sunE(i,:) - ratio_sunE(i-1,:))/dt;
1105         dratio_Msun(i,:) = norm(ratio_Msun(i,:) - ratio_Msun(i-1,:))/dt;
1106         dratio_sunM(i,:) = norm(ratio_sunM(i,:) - ratio_sunM(i-1,:))/dt;
1107     else
1108         dratio_Esun(i,:) = 0;
1109         dratio_sunE(i,:) = 0;
1110         dratio_Msun(i,:) = 0;
1111         dratio_sunM(i,:) = 0;
1112     end
1113
1114 end
1115
1116 dratio_Esun = abs(ratio_Esun - ratio_sunE);
1117 i_min = find(dratio_Esun == min(dratio_Esun));
1118 t_i_min_Esun = t(i_min);
1119
1120 dratio_Msun = abs(ratio_Msun - ratio_sunM);
1121 i_min = find(dratio_Msun == min(dratio_Msun));
1122 t_i_min_Msun = t(i_min);
1123
1124
1125 t_days = t/86400;
1126
1127 ftitle = 'Problem 3 - Gravitational Acceleration';
1128 % plot accelerations
1129 pos = pos + [50 0 0 0];
1130 figure('name', ftitle, 'position', pos)
1131
1132 subplot(2,1,1)
1133
1134 semilogy(t_days, a_Esat_hist, 'linewidth', 1.2);

```

```

1135     hold on; grid on;
1136     semilogy(t_days, a_sunsat_hist, '--','linewidth', 1.2);
1137     semilogy(t_days, a_pert_Esun_hist, 'linewidth', 1.2);
1138     semilogy(t_days, a_pert_sunE_hist, '--', 'linewidth', 1.2);
1139     % semilogy((et-et_t0)/86400, a_dist_Esun_P_hist, '-^');
1140     xline(t_i_min_Esun / 86400, 'b--', 'linewidth', 1.2);
1141
1142     legend('2body E-sat', '2body sun-sat', 'pert: E central, sun 3rd', ...
1143           'pert: sun central, E 3rd', 'Earth SOI', 'location', 'best')
1144     ylabel('km/s^2')
1145     title('Earth Gravitational Acceleration')
1146
1147     % Earth SOI
1148     % xlim([0, t_i_min_Esun/86400 + 1])
1149
1150     subplot(2,1,2)
1151     semilogy(t_days, a_Msat_hist, 'linewidth', 1.2);
1152     hold on; grid on;
1153     semilogy(t_days, a_sunsat_hist, '--','linewidth', 1.2);
1154     semilogy(t_days, a_pert_Msun_hist, 'linewidth', 1.2);
1155     semilogy(t_days, a_pert_sunM_hist, '--', 'linewidth', 1.2);
1156     xline(t_i_min_Msun / 86400, 'b--', 'linewidth', 1.2);
1157
1158     legend('2body M-sat', '2body sun-sat', 'pert: M central, 3rd ', ...
1159           'pert: sun central, M 3rd', 'Mars SOI', 'location', 'best')
1160     ylabel('km/s^2')
1161     title('Mars Gravitational Acceleration');
1162
1163     % Mars SOI
1164     % xlim([t_i_min_Msun/86400 - 1, t_days(end) + 1])
1165
1166     sgtitle(ftitle);
1167
1168     xlabel('Days')
1169
1170
1171     ftitle = 'Problem 3 - Gravitational Acceleration - SOI Vicinity';
1172     % plot accelerations
1173     pos = pos + [50 0 0 0];
1174     figure('name', ftitle, 'position', pos)
1175
1176     subplot(2,1,1)
1177
1178     semilogy(t_days, a_Esat_hist, 'linewidth', 1.2);
1179     hold on; grid on;
1180     semilogy(t_days, a_sunsat_hist, '--','linewidth', 1.2);
1181     semilogy(t_days, a_pert_Esun_hist, 'linewidth', 1.2);
1182     semilogy(t_days, a_pert_sunE_hist, '--', 'linewidth', 1.2);
1183     % semilogy((et-et_t0)/86400, a_dist_Esun_P_hist, '-^');
1184     xline(t_i_min_Esun / 86400, 'b--', 'linewidth', 1.2);
1185
1186     legend('2body E-sat', '2body sun-sat', 'pert: E central, sun 3rd', ...
1187           'pert: sun central, E 3rd', 'Earth SOI', 'location', 'best')
1188     ylabel('km/s^2')
1189     title('Earth Gravitational Acceleration')
1190
1191     % Earth SOI
1192     xlim([0, t_i_min_Esun/86400 + 1])
1193
1194     subplot(2,1,2)
1195     semilogy(t_days, a_Msat_hist, 'linewidth', 1.2);
1196     hold on; grid on;
1197     semilogy(t_days, a_sunsat_hist, '--','linewidth', 1.2);
1198     semilogy(t_days, a_pert_Msun_hist, 'linewidth', 1.2);
1199     semilogy(t_days, a_pert_sunM_hist, '--', 'linewidth', 1.2);
1200     xline(t_i_min_Msun / 86400, 'b--', 'linewidth', 1.2);
1201
1202     legend('2body M-sat', '2body sun-sat', 'pert: M central, 3rd ', ...

```

```

1203         'pert: sun central, M 3rd', 'Mars SOI', 'location', 'best')
1204 ylabel('km/s^2')
1205 title('Mars Gravitational Acceleration');
1206
1207 % Mars SOI
1208 xlim([t_i_min_Msun/86400 - 1, t_days(end) + 1])
1209
1210 sgtitle(ftitle);
1211
1212 xlabel('Days')
1213
1214 % -----
1215 ftitle = 'Problem 3 - Earth, Mars, and Sun acceleration ratios';
1216 % plot rate of ratio change
1217 pos = pos + [50 0 0 0];
1218 figure('name', ftitle, 'position', pos)
1219
1220 % EARTH-SUN RATIO
1221 subplot(2,2,1)
1222 semilogy(t_days, ratio_Esun, 'b', 'linewidth', 1.2);
1223 hold on; grid on;
1224 semilogy(t_days, ratio_sunE, 'k', 'linewidth', 1.2);
1225
1226 xline(t_i_min_Esun / 86400, 'b--', 'linewidth', 1.2);
1227
1228 ylim('auto')
1229 legend('E central', 'Sun central', 'E SOI')
1230 title('Earth to sun ratio')
1231
1232 % Earth SOI
1233 % xlim([0, t_i_min_Esun/86400 + 1])
1234
1235 % MARS-SUN RATIO
1236 subplot(2,2,2)
1237 semilogy(t_days, ratio_Msun, 'r', 'linewidth', 1.2);
1238 hold on; grid on;
1239 semilogy(t_days, ratio_sunM, 'k', 'linewidth', 1.2);
1240
1241 xline(t_i_min_Msun / 86400, 'r--', 'linewidth', 1.2);
1242
1243 legend('M central', 'Sun central', 'M SOI')
1244 title('Mars to sun ratio')
1245
1246 % Mars SOI
1247 % xlim([t_i_min_Msun/86400 - 1, t_days(end) + 1])
1248
1249 % EARTH-SUN RATIO CHANGE
1250 subplot(2,2,3)
1251 semilogy(t_days, dratio_Esun, 'b', 'linewidth', 1.2);
1252 hold on; grid on;
1253 semilogy(t_days, dratio_sunE, 'k', 'linewidth', 1.2);
1254 ylim('auto')
1255
1256 xline(t_i_min_Esun / 86400, 'b--', 'linewidth', 1.2);
1257
1258 legend('E central', 'Sun central', 'E SOI')
1259 title('Earth to sun ratio rate change magnitude')
1260 xlabel('Time (days)')
1261
1262 % Earth SOI
1263 % xlim([0, t_i_min_Esun/86400 + 1])
1264
1265 % MARS-SUN RATIO CHANGE
1266 subplot(2,2,4)
1267 semilogy(t_days, dratio_Msun, 'r', 'linewidth', 1.2);
1268 hold on; grid on;
1269 semilogy(t_days, dratio_sunM, 'k', 'linewidth', 1.2);
1270 xline(t_i_min_Msun / 86400, 'r--', 'linewidth', 1.2);

```

```

1271
1272         legend('M central', 'Sun central', 'M SOI')
1273         title('Mars to sun ratio rate change magnitude')
1274
1275
1276         % Mars SOI
1277         xlim([t_i_min_Msun/86400 - 1, t_days(end) + 1])
1278
1279         sgtitle(ftitle);
1280
1281         xlabel('Time (days)')
1282
1283         % -----
1284         ftitle = {'Problem 3 - Earth, Mars, and Sun acceleration ratios - SOI vicinity'};
1285         % plot rate of ratio change
1286         pos = pos + [50 0 0 0];
1287         figure('name', ftitle{1}, 'position', pos)
1288
1289         % EARTH-SUN RATIO
1290         subplot(2,2,1)
1291         semilogy(t_days, ratio_Esun, 'b', 'linewidth', 1.2);
1292         hold on; grid on;
1293         semilogy(t_days, ratio_sunE, 'k', 'linewidth', 1.2);
1294
1295         xline(t_i_min_Esun / 86400, 'b--', 'linewidth', 1.2);
1296
1297         ylim('auto')
1298         legend('E central', 'Sun central', 'E SOI')
1299         title('Earth to sun ratio')
1300
1301         % Earth SOI
1302         xlim([0, t_i_min_Esun/86400 + 1])
1303
1304         % MARS-SUN RATIO
1305         subplot(2,2,2)
1306         semilogy(t_days, ratio_Msun, 'r', 'linewidth', 1.2);
1307         hold on; grid on;
1308         semilogy(t_days, ratio_sunM, 'k', 'linewidth', 1.2);
1309
1310         xline(t_i_min_Msun / 86400, 'r--', 'linewidth', 1.2);
1311
1312         legend('M central', 'Sun central', 'M SOI')
1313         title('Mars to sun ratio')
1314
1315         % Mars SOI
1316         xlim([t_i_min_Msun/86400 - 1, t_days(end) + 1])
1317
1318         % EARTH-SUN RATIO CHANGE
1319         subplot(2,2,3)
1320         semilogy(t_days, dratio_Esun, 'b', 'linewidth', 1.2);
1321         hold on; grid on;
1322         semilogy(t_days, dratio_sunE, 'k', 'linewidth', 1.2);
1323         ylim('auto')
1324
1325         xline(t_i_min_Esun / 86400, 'b--', 'linewidth', 1.2);
1326
1327         legend('E central', 'Sun central', 'E SOI')
1328         title('Earth to sun ratio rate change magnitude')
1329         xlabel('Time (days)')
1330
1331         % Earth SOI
1332         xlim([0, t_i_min_Esun/86400 + 1])
1333
1334         % MARS-SUN RATIO CHANGE
1335         subplot(2,2,4)
1336         semilogy(t_days, dratio_Msun, 'r', 'linewidth', 1.2);
1337         hold on; grid on;
1338         semilogy(t_days, dratio_sunM, 'k', 'linewidth', 1.2);

```



```

1339         xline(t_i_min_Msun / 86400, 'r--', 'linewidth', 1.2);
1340
1341         legend('M central', 'Sun central', 'M SOI')
1342         title('Mars to sun ratio rate change magnitude')
1343
1344
1345         % Mars SOI
1346         xlim([t_i_min_Msun/86400 - 1, t_days(end) + 1])
1347
1348         sgtitle(ftitle);
1349
1350         xlabel('Time (days)')
1351
1352     %% flight path angle
1353
1354     for i = 1:length(X_sunsat_hist)
1355
1356         % FLIGHT PATH ANGLE - SUN
1357         phi_fpa_sunsat(i,:) = find_fpa(X_sunsat_hist, i);
1358
1359         % FLIGHT PATH ANGLE - EARTH
1360         phi_fpa_Esat(i,:) = find_fpa(X_Esat_hist, i);
1361
1362         % FLIGHT PATH ANGLE - MARS
1363         phi_fpa_Msat(i,:) = find_fpa(X_Msat_hist, i);
1364
1365         % test stuff
1366         phi_fpa_sunM(i,:) = find_fpa(X_sunM_hist, i);
1367         phi_fpa_sunE(i,:) = find_fpa(X_sunE_hist, i);
1368
1369     end
1370
1371     ftitle = 'Problem 3 - Flight Path Angle';
1372     pos = pos + [50 0 0 0];
1373     figure('name', ftitle, 'position', pos)
1374
1375     subplot(3,1,1)
1376     plot(t_days, phi_fpa_sunsat, 'k', 'linewidth', 1.2)
1377     hold on; grid on;
1378     plot(t_days, phi_fpa_Esat, 'b', 'linewidth', 1.2)
1379     plot(t_days, phi_fpa_Msat, 'r', 'linewidth', 1.2)
1380     xline(t_i_min_Esun / 86400, 'b--', 'linewidth', 1.2);
1381     xline(t_i_min_Msun / 86400, 'r--', 'linewidth', 1.2);
1382
1383     legend('\phi_{fpa, sun}', '\phi_{fpa, Earth}', '\phi_{fpa, Mars}', ...
1384           'Earth SOI', 'Mars SOI')
1385
1386     xlabel('Time (days)')
1387     ylabel('deg')
1388     title(ftitle)
1389
1390     subplot(3,1,2)
1391     plot(t_days, phi_fpa_sunsat, 'k', 'linewidth', 1.2)
1392     hold on; grid on;
1393     plot(t_days, phi_fpa_Esat, 'b', 'linewidth', 1.2)
1394     plot(t_days, phi_fpa_Msat, 'r', 'linewidth', 1.2)
1395     xline(t_i_min_Esun / 86400, 'b--', 'linewidth', 1.2);
1396     xline(t_i_min_Msun / 86400, 'r--', 'linewidth', 1.2);
1397
1398     % Earth SOI
1399     xlim([0, t_i_min_Esun/86400 + 1])
1400
1401     xlabel('Time (days)')
1402     ylabel('deg')
1403     title(ftitle)
1404
1405     subplot(3,1,3)
1406     plot(t_days, phi_fpa_sunsat, 'k', 'linewidth', 1.2)

```

```

1407         hold on; grid on;
1408         plot(t_days, phi_fpa_Esat, 'b', 'linewidth', 1.2)
1409         plot(t_days, phi_fpa_Msat, 'r', 'linewidth', 1.2)
1410         xline(t_i_min_Esun / 86400, 'b--', 'linewidth', 1.2);
1411         xline(t_i_min_Msun / 86400, 'r--', 'linewidth', 1.2);
1412
1413         % Mars SOI
1414         xlim([t_i_min_Msun/86400 - 1, t_days(end) + 1])
1415
1416         fn.move_legend
1417         xlabel('Time (days)')
1418         ylabel('deg')
1419         title(ftitle)
1420
1421 function phi_fpa = find_fpa(X_bodsat_hist, i)
1422
1423     % position unit vector
1424     r_sunsat = X_bodsat_hist(i, 1:3);
1425     r_sunsat = r_sunsat / norm(r_sunsat);
1426     % velocity unit vector
1427     v_sunsat = X_bodsat_hist(i, 4:6);
1428     v_sunsat = v_sunsat / norm(v_sunsat);
1429     % orbit normal
1430     h = cross(r_sunsat, v_sunsat);
1431     % calculate transverse direction
1432     v_transv = cross(h, r_sunsat);
1433     % velocity unit vector
1434     v_sunsat = X_bodsat_hist(i, 4:6);
1435     v_sunsat = v_sunsat / norm(v_sunsat);
1436     % fpa
1437     phi_fpa = acosd(dot(v_transv, v_sunsat));
1438
1439 end
1440
1441
1442 function h = plot3_xyz(X, style, linewidth, i)
1443
1444 if ~exist('style', 'var')
1445     style = '';
1446 end
1447
1448 if ~exist('linewidth', 'var')
1449     linewidth = 1;
1450 end
1451
1452 if ~exist('i', 'var')
1453     h = plot3(X(:,1), X(:,2), X(:,3), style, 'linewidth', linewidth);
1454 elseif strcmp(i, 'end')
1455     h = plot3(X(end,1), X(end,2), X(end,3), style, 'linewidth', linewidth);
1456 else
1457     h = plot3(X(i,1), X(i,2), X(i,3), style, 'linewidth', linewidth);
1458 end
1459
1460 end

```