

# **ASE387P.2 Mission Analysis and Design**

## **Homework 2**

Junette Hsin

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

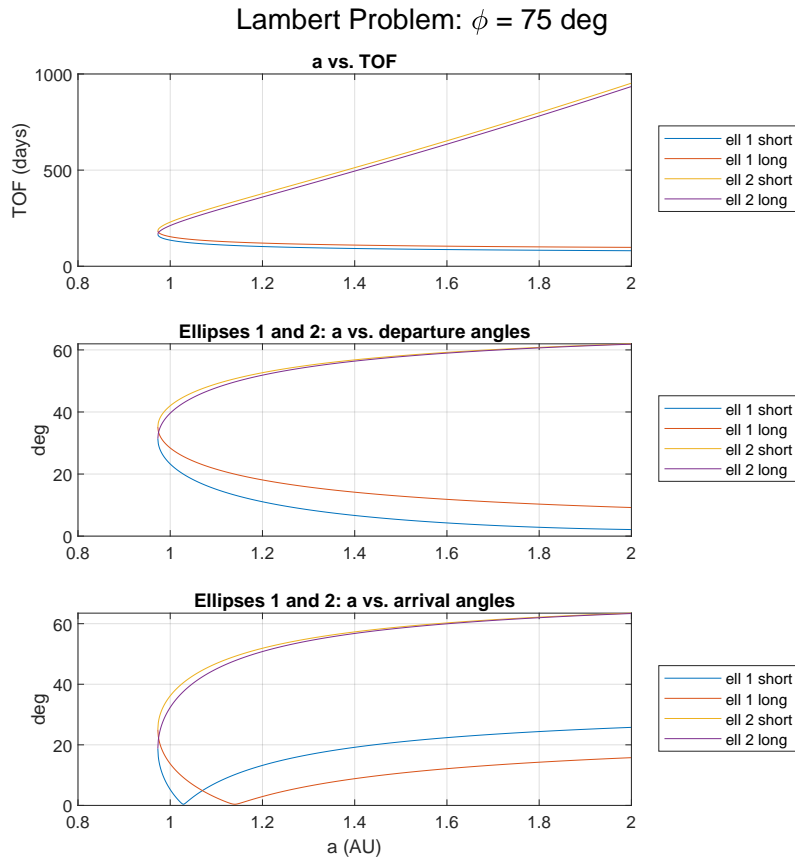
### Problem 1

Consider the possible transfer orbits from Earth to Mars. Assume both planets are in coplanar circular heliocentric orbits. Consider a transfer angle of  $75^\circ$ .

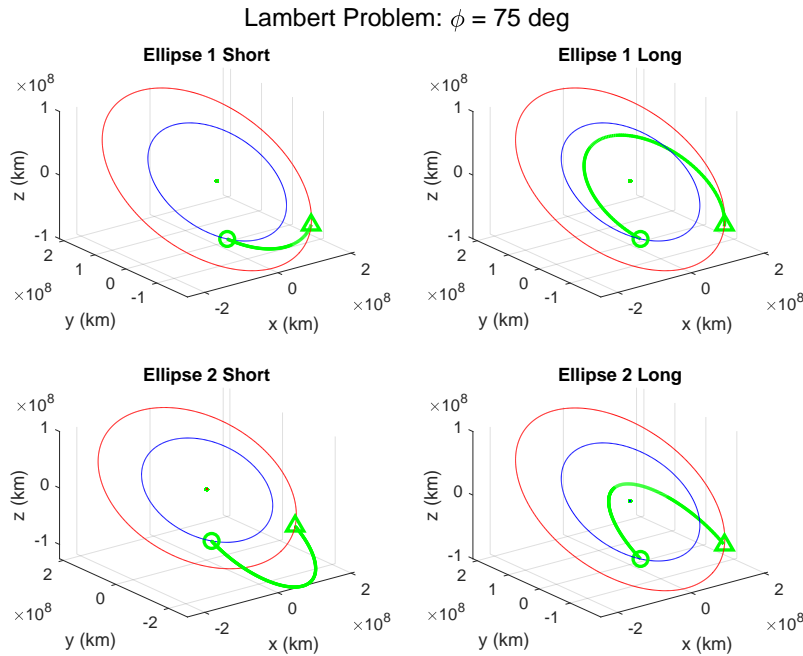
A

Choose a range of semi-major axes up to a distance of 2 A.U. and draw a plot (to scale) of the semi-major axes (X-axis) versus time-of-flight (Y-axis) of the transfer orbit. Note that there are two possible times for flight for each transfer orbit semi-major axis – the short-way and the long-way.

*Solution*



**Fig. 1** TOF and Departure/Arrival Angles vs.  $a$



**Fig. 2 Visualization of Transfer Orbits**

(Bonus) Figure 2 visualizes the transfer orbits for both ellipses. The red orbit represents Mars and the blue orbit represents Earth. The green circles indicate the initial departure position and the green triangles indicate the final arrival position. The green lines map out the trajectory of the probe on its path to Mars.

**B**

What is the value of the semi-major axis and eccentricity for the minimum energy transfer orbit?

*Solution*

$A_{min} = 0.97301$  AU and  $e_{min} = 0.53129$ .

**C**

For the given transfer angle and for any chosen semi-major axis, at the departure and arrival points, calculate the angles between the velocity of the probe and the velocity of the departure and arrival planets, respectively. Let us call these “departure angle” and “arrival angle” – this may be non-standard terminology. We have two of each – one for short-way travel and another for long-way travel.

*Solution*

Please see Figure 1. The Battin Method from Vallado (Fundamentals of Astrodynamics and Applications, ed. 4) was used to calculate the velocity vectors and angles for the departure and arrival positions.

**D**

Plot the departure and arrival angles for this case as a function of the transfer orbit semi-major axis.

*Solution*

Please see Figure 1.

**E**

Discuss the significance of these plots for decision-making on the choice of semimajor axes for the fixed transfer angle.

*Solution*

Lower departure and arrival angles may be desirable for a mission, as the delta-V required to enable capture by the destination planet's gravity would be lower. Balancing the desirability of a spacecraft orbit versus costs such as fuel expenditure needs to be evaluated.

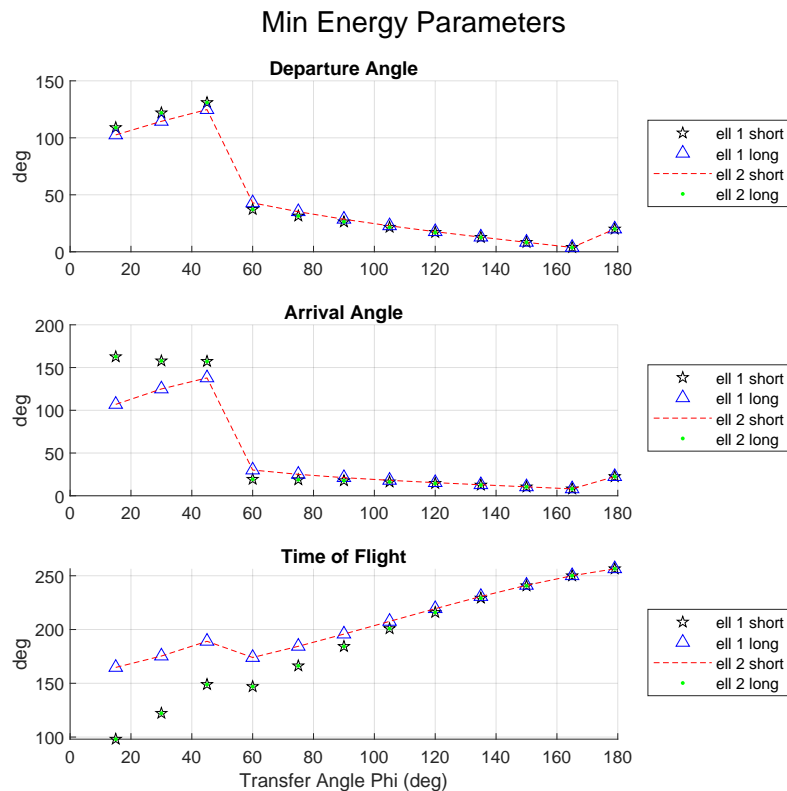
## Problem 2

Repeat Problem 1 for 15° increments of the transfer angle, that is for [15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180] degrees.

**A**

Plot the departure angle, the arrival angle, and the time-of-flight for the minimum energy transfer orbit as a function of the transfer angle.

*Solution*



**Fig. 3** Departure Angle, Arrival Angle, and TOF for Phi = 15 through 180 deg

**B**

Discuss the significance of these plots for decision-making on the choice of transfer angle and the semi-major axes of the transfer orbit.

*Solution*

Lower departure and arrival angles may be desirable for a mission, as the delta-V required to enable capture by the destination planet's gravity would be lower.

## Appendix

### MATLAB code

```
1      %% HW 2
2      % Junette Hsin
3
4      % close all;
5      clear;
6
7      %% transfer angle = 75 deg
8
9      % Define parameters for a state lookup:
10     % t0      = 'Oct 20, 2020 11:00 AM CST';
11     t0      = 'May 22, 1950';
12
13     phi_t_des = 180;
14     [ell_1_min, ell_2_min, amin_AU, emin] = lambert_prob(t0, phi_t_des, 1);
15
16
17     %% phi = 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180 degrees
18
19     phi_d_hist = [];
20     phi_a_hist = [];
21     tof_hist = [];
22     phi_t_hist = [];
23
24     phi0 = 15;
25     for phi = phi0 : 15 : 180
26
27         phi_t_des = phi;
28         [ell_1_min, ell_2_min] = lambert_prob(t0, phi_t_des, 0);
29
30         % departure angle:
31         % (1) ELL 1 SHORT, (2) ELL 1 LONG, (3) ELL 2 SHORT, (4) ELL 2 LONG
32         phi_d_hist = [phi_d_hist; ...
33             ell_1_min.phi_ds, ell_1_min.phi_dl, ell_2_min.phi_ds, ell_2_min.phi_dl];
34
35         % arrival angle
36         % (1) ELL 1 SHORT, (2) ELL 1 LONG, (3) ELL 2 SHORT, (4) ELL 2 LONG
37         phi_a_hist = [phi_a_hist; ...
38             ell_1_min.phi_as, ell_1_min.phi_al, ell_2_min.phi_as, ell_2_min.phi_al];
39
40         % time of flight
41         % (1) ELL 1 SHORT, (2) ELL 1 LONG, (3) ELL 2 SHORT, (4) ELL 2 LONG
42         tof_hist = [tof_hist; ...
43             ell_1_min.dt_s, ell_1_min.dt_l, ell_2_min.dt_s, ell_2_min.dt_l];
44
45         % transfer angle
46         phi_t_hist = [phi_t_hist; phi_t_des];
47
48     end
49
50
51     %%
52
53     colors = {'k', 'b', 'r', 'g'};
54     style = {'p', '^', '—', '.'};
55     lwidth = [3, 1.5, 1, 1];
56
57     figure()
58         subplot(3,1,1)
59             hold on;
60             for i = 1:4
61 %                 scatter(phi_t_hist, phi_d_hist(:,i), 4, colors{i});
62                 h(i) = plot(phi_t_hist, phi_d_hist(:,i), [colors{i} style{i}]);
63             end
64             title('Departure Angle')
```

```

65         ylabel('deg')
66         legend(h, 'ell 1 short', 'ell 1 long', 'ell 2 short', 'ell 2 long', '
        location', 'eastoutside');
67
68     subplot(3,1,2)
69     hold on;
70     for i = 1:4
71         % scatter(phi_t_hist, phi_a_hist(:,i), 4, colors{i});
72         h(i) = plot(phi_t_hist, phi_a_hist(:,i), [colors{i} style{i}]);
73     end
74     title('Arrival Angle')
75     ylabel('deg')
76     legend(h, 'ell 1 short', 'ell 1 long', 'ell 2 short', 'ell 2 long', '
        location', 'eastoutside');
77
78     subplot(3,1,3)
79     hold on;
80     for i = 1:4
81         % scatter(phi_t_hist, tof_hist(:,i), 4, colors{i});
82         h(i) = plot(phi_t_hist, tof_hist(:,i), [colors{i} style{i}]);
83     end
84     title('Time of Flight')
85     ylabel('deg')
86     legend(h, 'ell 1 short', 'ell 1 long', 'ell 2 short', 'ell 2 long', '
        location', 'eastoutside');
87
88     xlabel('Transfer Angle Phi (deg)')
89
90     sgtitle('Min Energy Parameters')
91
92     %% save plots
93
94
95     %% RIGHT HERE
96     function [ell_1_min, ell_2_min, amin_AU, emin] = lambert_prob(t0, phi_des, plot_option)
97
98     addpath(genpath('mice'));
99     addpath(genpath('spice_data'));
100
101     % Load kernel file
102     cspice_furnsh('spice_data/naif0011.tls');
103     cspice_furnsh('spice_data/de421.bsp');
104     cspice_furnsh('spice_data/pck00010.tpc');
105
106     abcorr = 'NONE';
107
108     % Convert the epoch to ephemeris time (secs)
109     et_t0 = cspice_str2et(t0);
110
111     % get states —> Sun to Earth
112     target = 'Earth';
113     frame = 'J2000';
114     observer = 'Sun';
115     abcorr = 'NONE';
116
117     % get sun position
118     et = et_t0; % propagate ephemeris time by 1 day in secs
119     X_sunE = spice_state(et, target, frame, abcorr, observer);
120
121     % get states —> Sun to Mars
122     target = 'Mars';
123     frame = 'J2000';
124     observer = 'Sun';
125     abcorr = 'NONE';
126
127     % get sun position
128     et = et_t0; % propagate ephemeris time by 1 day in secs
129     X_sunM = spice_state(et, target, frame, abcorr, observer);

```

```

130
131 % get angle between Earth and Mars velocities
132 r_E = X_sunE(1:3);
133 r_M = X_sunM(1:3);
134 v_E = X_sunE(4:6);
135 v_M = X_sunM(4:6);
136
137 % angle and velocity angles
138 phi_r = acosd( dot(r_E, r_M) / (norm(r_E)*norm(r_M)) );
139 phi_v = acosd( dot(v_E, v_M) / (norm(v_E)*norm(v_M)) );
140
141 i = 0;
142 while abs(phi_r - phi_des) > 0.01
143
144     % propagate by 0.1 day
145     i = i + 0.01;
146     et = et_t0 + i*86400;
147
148     % get velocity
149     X_sunM = spice_state(et, target, frame, abcorr, observer);
150     r_M = X_sunM(1:3);
151     v_M = X_sunM(4:6);
152
153     % get angle
154     phi_r = acosd( dot(r_E, r_M) / (norm(r_E)*norm(r_M)) );
155     phi_v = acosd( dot(v_E, v_M) / (norm(v_E)*norm(v_M)) );
156
157 end
158
159 % units in km
160 rd_E = r_E ;
161 ra_M = r_M ;
162 vd_E = v_E ;
163 va_M = v_M ;
164
165 % propagate to get full Earth orbit
166 X_sunE_hist = [];
167 target = 'Earth';
168 for i = 0:365+1
169
170     et = et_t0 + i*86400;
171
172     % get state
173     X_sunE = spice_state(et, target, frame, abcorr, observer);
174
175     % save Mars vector
176     X_sunE_hist = [X_sunE_hist; X_sunE];
177 end
178
179 % propagate to get full Mars orbit
180 X_sunM_hist = [];
181 target = 'Mars';
182 for i = 0:687+1
183
184     et = et_t0 + i*86400;
185
186     % get state
187     X_sunM = spice_state(et, target, frame, abcorr, observer);
188
189     % save Mars vector
190     X_sunM_hist = [X_sunM_hist; X_sunM];
191 end
192
193
194 %% Part 1a
195 % Lambert — from VALLADO ed. 4, pgs 467 — 475
196
197 % Arrival (Mars), AU units

```



```

198 ra_mag = norm(ra_M);
199 rd_mag = norm(rd_E);
200
201 % Vallado method ...
202 cos_dv = dot(rd_E, ra_M) / (rd_mag * ra_mag);
203
204 % chord
205 c = sqrt( rd_mag^2 + ra_mag^2 - 2*rd_mag*ra_mag*cos_dv );
206
207 % semiperimeter
208 s = ( ra_mag + rd_mag + c ) / 2;
209
210 % min semimajor axis
211 amin = s/2;
212
213 % initialize
214 a_hist = [];
215
216 % 1 AU = km2AU km
217 km2AU = 149598073;
218
219 % loop
220 for a = amin : 0.001*km2AU : 2*km2AU
221
222     % time of flight
223     [ell_1, ell_2] = a2tof(s, c, a, rd_E, ra_M, vd_E, va_M);
224
225     % if 1st iteration, create ellipse hist
226     if a == amin
227         ell_1_hist = ell_1;
228         ell_2_hist = ell_2;
229
230     % else, build array
231     else
232         fnames = fieldnames(ell_1);
233         for i = 1:length(fnames)
234             ell_1_hist.(fnames{i}) = [ell_1_hist.(fnames{i}); ell_1.(fnames{i})];
235         end
236         fnames = fieldnames(ell_2);
237         for i = 1:length(fnames)
238             ell_2_hist.(fnames{i}) = [ell_2_hist.(fnames{i}); ell_2.(fnames{i})];
239         end
240
241     end
242
243     a_hist = [a_hist; a];
244
245 end
246
247 % dt_hist_years = dt_hist / 365;
248 a_hist_AU = a_hist / km2AU;
249
250 %% plot
251
252 if plot_option > 0
253
254     fname = 'Ellipse 1 and 2 TOF and Angles';
255     pos = [100 100 700 700];
256     figure('name', fname, 'position', pos)
257     subplot(3,1,1)
258     plot(a_hist_AU, ell_1_hist.dt_s); hold on;
259     plot(a_hist_AU, ell_1_hist.dt_l); hold on;
260     plot(a_hist_AU, ell_2_hist.dt_s); hold on;
261     plot(a_hist_AU, ell_2_hist.dt_l); hold on;
262     legend('ell 1 short', 'ell 1 long', 'ell 2 short', 'ell 2 long', 'location', 'eastoutside');
263     ylabel('TOF (days)')
264     title('a vs. TOF')

```

```

265     subplot(3,1,2)
266         plot(a_hist_AU, ell_1_hist.phi_ds); hold on;
267         plot(a_hist_AU, ell_1_hist.phi_dl);
268         plot(a_hist_AU, ell_2_hist.phi_ds);
269         plot(a_hist_AU, ell_2_hist.phi_dl);
270         legend('ell 1 short', 'ell 1 long', 'ell 2 short', 'ell 2 long', 'location', '
                eastoutside')
271         title('Ellipses 1 and 2: a vs. departure angles');
272         ylabel('deg')
273     subplot(3,1,3)
274         plot(a_hist_AU, ell_1_hist.phi_as); hold on;
275         plot(a_hist_AU, ell_1_hist.phi_al);
276         plot(a_hist_AU, ell_2_hist.phi_as);
277         plot(a_hist_AU, ell_2_hist.phi_al);
278         legend('ell 1 short', 'ell 1 long', 'ell 2 short', 'ell 2 long', 'location', '
                eastoutside')
279         title('Ellipses 1 and 2: a vs. arrival angles');
280         ylabel('deg')
281         xlabel('a (AU)');
282
283     sgtitle(['Lambert Problem: \phi = ' num2str(phi_des) ' deg'])
284
285 end
286
287 % [V1, V2] = LAMBERTBATTIN(rd, ra, 'retro', tof);
288
289 %% propagate orbits
290
291 if plot_option > 0
292
293     a_ind = 160;
294
295     fname = 'Ellipse 1 and 2, Short and Long';
296     pos = [100 100 800 600];
297     figure('name', fname, 'position', pos)
298
299     % subplot 1
300     subplot(2,2,1)
301
302     [rv_hist, oe_hist] = prop_probe ...
303         (ell_1_hist, rd_E, ra_M, 'short', a_ind);
304     ftitle = 'Ellipse 1 Short';
305     plot_probe(rv_hist, X_sunE_hist, X_sunM_hist, rd_E, ra_M, ftitle)
306
307     % subplot 2
308     subplot(2,2,2)
309
310     [rv_hist, oe_hist] = prop_probe ...
311         (ell_1_hist, rd_E, ra_M, 'long', a_ind);
312     ftitle = 'Ellipse 1 Long';
313     plot_probe(rv_hist, X_sunE_hist, X_sunM_hist, rd_E, ra_M, ftitle)
314
315
316     % subplot 3
317     subplot(2,2,3)
318
319     [rv_hist, oe_hist] = prop_probe ...
320         (ell_2_hist, rd_E, ra_M, 'short', a_ind);
321     ftitle = 'Ellipse 2 Short';
322     plot_probe(rv_hist, X_sunE_hist, X_sunM_hist, rd_E, ra_M, ftitle)
323
324     % subplot 4
325     subplot(2,2,4)
326
327     [rv_hist, oe_hist] = prop_probe ...
328         (ell_2_hist, rd_E, ra_M, 'long', a_ind);
329     ftitle = 'Ellipse 2 Long';
330     plot_probe(rv_hist, X_sunE_hist, X_sunM_hist, rd_E, ra_M, ftitle)

```

```

331
332     sgtitle(['Lambert Problem: \phi = ' num2str(phi_des) ' deg'])
333
334 end
335
336 %% minimum energy transfer orbit
337
338 pmin = rd_mag*ra_mag/c * (1 - cosd(phi_r));
339 emin = sqrt( 1 - 2*pmin/s );
340 amin_AU = amin / km2AU;
341
342 sprintf('a_min = %.5g AU, e_min = %.5g', amin_AU, emin)
343
344 [ell_1_min, ell_2_min] = a2tof(s, c, amin, rd_E, ra_M, vd_E, va_M);
345
346
347 end
348
349
350 %% subfunctions
351
352 function plot_probe(rv_hist, X_sunE_hist, X_sunM_hist, rd_E, ra_M, ftitle)
353
354     z_rv = zeros(length(rv_hist), 1);
355     z_E = zeros(length(X_sunE_hist), 1);
356     z_M = zeros(length(X_sunM_hist), 1);
357
358     plot3([z_rv rv_hist(:,1)], [z_rv rv_hist(:,2)], [z_rv rv_hist(:,3)], 'g', 'linewidth', 2);
359     hold on;
360     scatter3([rd_E(1)], [rd_E(2)], [rd_E(3)], 80, 'go', 'linewidth', 2);
361     scatter3([ra_M(1)], [ra_M(2)], [ra_M(3)], 80, 'g^', 'linewidth', 2);
362     plot3([z_E X_sunE_hist(:,1)], [z_E X_sunE_hist(:,2)], [z_E X_sunE_hist(:,3)], 'b—')
363     plot3([z_M X_sunM_hist(:,1)], [z_M X_sunM_hist(:,2)], [z_M X_sunM_hist(:,3)], 'r—')
364     title(ftitle)
365     xlabel('x (km)'); ylabel('y (km)'); zlabel('z (km)');
366
367 end
368
369 function [rv_hist, oe_hist] = prop_probe(ell_x_hist, rd, ra, dt_str, a_ind)
370
371     % probe orbit (min energy)
372     if strcmp(dt_str, 'short')
373         vd = ell_x_hist.vd_s(a_ind,:);
374         va = ell_x_hist.va_s(a_ind,:);
375         dt = ell_x_hist.dt_s(a_ind);
376     elseif strcmp(dt_str, 'long')
377         vd = ell_x_hist.vd_l(a_ind,:);
378         va = ell_x_hist.va_l(a_ind,:);
379         dt = ell_x_hist.dt_l(a_ind);
380     else
381         disp('Choose long or short')
382         return
383     end
384
385     % sun mu (m^3/s^2)
386     mu_sun_m = 1.32712440018e20;
387     mu_sun_km = mu_sun_m / (1000^3);
388
389     % probe state in km
390     X_probe0 = [rd'; vd'];
391     oe_probe0 = rvOrb.rv2orb(X_probe0, mu_sun_km);
392     X_probef = [ra'; va'];
393     oe_probef = rvOrb.rv2orb(X_probef, mu_sun_km);
394
395     % delta anomaly
396     dM = oe_probef(6) - oe_probe0(6);
397     if strcmp(dt_str, 'long')
398         dM = 2*pi - dM;

```

```

399     end
400
401     % mean motion (rad/days)
402     n = dM / dt;
403     if strcmp(dt_str, 'long')
404         n = -n;
405     end
406
407     % propagate orbit
408     oe_hist = oe_probe0;
409     rv_hist = [X_probe0'];
410     for i = 1:round(dt)
411
412         oe = oe_probe0;
413
414         % propagate nu by i days
415         nu = oe_probe0(6) + n*i;
416         oe(6) = nu;
417
418         % convert to cartesian
419         rv = rvOrb.orb2rv(oe, mu_sun_kM);
420
421         oe_hist = [oe_hist; oe];
422         rv_hist = [rv_hist; rv'];
423
424     end
425
426 end
427
428 function [ell_1, ell_2] = a2tof(s, c, a, rd, ra, vd_E, va_M)
429 %
430 % Inputs:
431 %   s = semiperimeter (km)
432 %   c = chord (km)
433 %   a = semimajor axis (km)
434 %
435 % Outputs:
436 %   ellipse 1
437 %   ellipse 2
438 %       each containing:
439 %       TOF for short and long
440 %       departure velocities for short and long
441 %       arrival velocities for short and long
442 %       departure angles for short and long
443 %       arrival angles for short and long
444 %
445
446 % time of flight
447 alpha1 = 2 * asin( sqrt( s/(2*a) ) );
448 alpha2 = 2*pi - alpha1;
449 beta1 = 2 * asin( sqrt( (s-c)/(2*a) ) );
450 beta2 = - beta1;
451
452 % sun mu (m^3/s^2)
453 mu_sun_m = 1.32712440018e20;
454 mu_sun_kM = mu_sun_m / (1000^3);
455
456 % time of flight
457 nrev = 0;
458 dt1_s = sqrt( a^3/mu_sun_kM ) * ...
459     (( 2*nrev*pi + alpha1 - sin(alpha1) - (beta1 - sin(beta1)) ));
460 dt1_l = sqrt( a^3/mu_sun_kM ) * ...
461     (( 2*nrev*pi + alpha1 - sin(alpha1) + (beta1 - sin(beta1)) ));
462 dt2_s = sqrt( a^3/mu_sun_kM ) * ...
463     (( 2*nrev*pi + alpha2 - sin(alpha2) - (beta2 - sin(beta2)) ));
464 dt2_l = sqrt( a^3/mu_sun_kM ) * ...
465     (( 2*nrev*pi + alpha2 - sin(alpha2) + (beta2 - sin(beta2)) ));
466

```

```

467 % convert from seconds to days
468 day2sec = 60*60*24;
469
470 ell_1.dt_s = dt1_s / day2sec;
471 ell_1.dt_l = dt1_l / day2sec;
472 ell_2.dt_s = dt2_s / day2sec;
473 ell_2.dt_l = dt2_l / day2sec;
474
475 % -----
476 % VELOCITIES
477
478 % geometry
479 rd_mag = norm(rd);
480 ra_mag = norm(ra);
481 dnu = acos( dot(rd, ra) / ( rd_mag*ra_mag ) );
482 de = alpha1 - beta1;
483
484 % commented out failed velocity code
485
486 % June Battin
487 [vd1_s, va1_s] = LAMBERTBATTIN_km_sun_June(rd, ra, 'pro', dt1_s);
488 [vd1_l, va1_l] = LAMBERTBATTIN_km_sun_June(rd, ra, 'pro', dt1_l);
489 [vd2_s, va2_s] = LAMBERTBATTIN_km_sun_June(rd, ra, 'pro', dt2_s);
490 [vd2_l, va2_l] = LAMBERTBATTIN_km_sun_June(rd, ra, 'pro', dt2_l);
491
492 ell_1.vd_s = vd1_s;
493 ell_1.va_s = va1_s;
494 ell_1.vd_l = vd1_l;
495 ell_1.va_l = va1_l;
496 ell_2.vd_s = vd2_s;
497 ell_2.va_s = va2_s;
498 ell_2.vd_l = vd2_l;
499 ell_2.va_l = va2_l;
500
501 % -----
502 % ANGLES
503
504 % ELLIPSE 1 departure angle, short and long
505 vd_s = ell_1.vd_s;
506 phil_ds = acosd( dot(vd_s, vd_E) / ( norm(vd_s)*norm(vd_E) ) );
507 vd_l = ell_1.vd_l;
508 phil_dl = acosd( dot(vd_l, vd_E) / ( norm(vd_l)*norm(vd_E) ) );
509
510 ell_1.phi_ds = phil_ds;
511 ell_1.phi_dl = phil_dl;
512
513 % ELLIPSE 1 arrival angle, short and long
514 va_s = ell_1.va_s;
515 phil_as = acosd( dot(va_s, va_M) / ( norm(va_s)*norm(va_M) ) );
516 va_l = ell_1.va_l;
517 phil_al = acosd( dot(va_l, va_M) / ( norm(va_l)*norm(va_M) ) );
518
519 ell_1.phi_as = phil_as;
520 ell_1.phi_al = phil_al;
521
522 % ELLIPSE 2 departure angle, short and long
523 vd_s = ell_2.vd_s;
524 phi2_ds = acosd( dot(vd_s, vd_E) / ( norm(vd_s)*norm(vd_E) ) );
525 vd_l = ell_2.vd_l;
526 phi2_dl = acosd( dot(vd_l, vd_E) / ( norm(vd_l)*norm(vd_E) ) );
527
528 ell_2.phi_ds = phi2_ds;
529 ell_2.phi_dl = phi2_dl;
530
531 % ELLIPSE 2 arrival angle, short and long
532 va_s = ell_2.va_s;
533 phi2_as = acosd( dot(va_s, va_M) / ( norm(va_s)*norm(va_M) ) );
534 va_l = ell_2.va_l;

```

```

535 phi2_al = acosd( dot(va_l, va_M) / ( norm(va_l)*norm(va_M) ) );
536
537 ell_2.phi_as = phi2_as;
538 ell_2.phi_al = phi2_al;
539
540 end
541
542 %% Gauss solution
543
544 function [vd, va] = lambert_gauss(rd, ra, dtl_s, mu_sun_km, de)
545
546 rd_mag = norm(rd);
547 ra_mag = norm(ra);
548 dnu = acos( dot(rd, ra) / ( rd_mag*ra_mag ) );
549
550 % Gauss solution
551 L = (rd_mag + ra_mag) / ( 4*sqrt( rd_mag*ra_mag ) * cos(dnu/2) ) - 1/2;
552 m = mu_sun_km * dtl_s^2 / ( 2*sqrt(rd_mag*ra_mag) * cos(dnu/2) )^3;
553
554 yold = 0;
555 ynew = 1;
556
557 while abs(yold - ynew) > 0.001
558     yold = ynew;
559     x1 = m / yold^2 - L;
560     x2 = (de - sin(de)) / ( sin(de/2) )^3;
561     ynew = (L + x1)*x2 + 1;
562 end
563
564 cos_de2 = 1 - 2*x1;
565 p = rd_mag*ra_mag*(1 - cos(dnu)) / ...
566     ( rd_mag + ra_mag - 2*sqrt(rd_mag*ra_mag)*cos(dnu/2)*cos_de2 );
567
568 f = 1 - ra_mag/p * ( 1-cos(dnu) );
569 g = ra_mag*rd_mag * sin(dnu) / sqrt( mu_sun_km * p );
570
571 df = sqrt(1/p) * tan(dnu/2) * ( ( 1-cos(dnu) )/p - 1/ra_mag - 1/rd_mag );
572 dg = 1 - rd_mag/p * (1-cos(dnu));
573
574 vd = (ra - f*rd)/g;
575 va = (dg*ra - rd)/g;
576
577 end
578
579 %% Kepler equation solver
580
581 function E = keplerEq(M,e,eps)
582 % Function solves Kepler's equation M = E-e*sin(E)
583 % Input — Mean anomaly M [rad] , Eccentricity e and Epsilon
584 % Output eccentric anomaly E [DEG].
585     En = M;
586     Ens = En - (En-e*sind(En)-M)/(1 - e*cosd(En));
587     while ( abs(Ens-En) > eps )
588         En = Ens;
589         Ens = En - (En - e*sind(En) - M)/(1 - e*cosd(En));
590     end
591     E = Ens;
592 end
593
594 function E = kepler(M, e)
595     f = @(E) E - e * sind(E) - M;
596     E = fzero(f, M); % <— I would use M as the initial guess instead of 0
597 end
598
599 %% annotation
600
601 function h_text(h2, text1, text2, text3, text4)
602

```

```

603     pos = get(h2, 'position');
604     delete(findall(gcf,'type','annotation'));
605     text = { ''; ''; text1; text2; text3; text4 };
606     annotation('textbox', pos, ...
607         'String', text, ...
608         'edgecolor', 'none');
609     axis off
610
611 end
612
613 %% commented out failed velocity code
614
615 %% Battin solution (definitely works)
616 %% ellipse 1, long and short
617 [vd1_s, val_s] = LAMBERTBATTIN_km_sun(rd, ra, 'pro', dt1_s);
618 [vd1_l, val_l] = LAMBERTBATTIN_km_sun(rd, ra, 'pro', dt1_l);
619 %
620 %% ellipse 2, long and short
621 [vd2_s, va2_s] = LAMBERTBATTIN_km_sun(rd, ra, 'pro', dt2_s);
622 [vd2_l, va2_l] = LAMBERTBATTIN_km_sun(rd, ra, 'pro', dt2_l);
623 %
624 % compute velocities — Bettadpur and Vallado combination
625 n = @(alpha1, beta1, dt1_s) ...
626     (alpha1 - beta1) - 2*cos( (alpha1+beta1)/2 )*sin( (alpha1-beta1)/2 );
627 f = @(alpha1, beta1) ...
628     1 - a/rd_mag * ( 1 - cos(alpha1 - beta1) );
629 g = @(alpha1, beta1, dt1_s) ...
630     dt1_s - 1/n(alpha1, beta1, dt1_s) * ( de - sin(alpha1 - beta1) );
631 %
632 K = 4*a/c^2 * ( 1/2*(rd_mag+ra_mag+c) - rd_mag ) * ( 1/2*(rd_mag+ra_mag+c) - ra_mag );
633 p1 = @(alpha1, beta1) ...
634     K * sin( (alpha1+beta1)/2 )^2;
635 p2 = @(alpha1, beta1) ...
636     K * sin( (alpha1-beta1)/2 )^2;
637 %
638 dg = @(alpha1, beta1, p1) ...
639     1 - rd_mag/p1(alpha1, beta1) * ( 1-cos(dnu) );
640 vd1_s = ( ra - f(alpha1, beta1)*rd ) / ...
641     g(alpha1, beta1, dt1_s);
642 vd1_l = - vd1_s;
643 vd2_s = ( ra - f(alpha2, beta2)*rd ) / ...
644     g(alpha2, beta2, dt2_s);
645 vd2_l = - vd2_s;
646
647
648 va2_s = ( dg(alpha2, beta2, p2)*ra - rd ) / ...
649     g(alpha2, beta2, dt2_s);
650 vd2_l = ( ra - f(alpha2, beta2)*rd ) / ...
651     g(alpha2, beta2, dt2_l);
652 va2_l = ( dg(alpha2, beta2, p2)*ra - rd ) / ...
653     g(alpha2, beta2, dt2_l);
654
655 % Gauss solution
656 [vd1_s_gauss, val_s_gauss] = lambert_gauss(rd, ra, dt1_s, mu_sun_km, de);
657 [vd1_l_gauss, val_l_gauss] = lambert_gauss(rd, ra, dt1_l, mu_sun_km, de);
658 [vd2_s_gauss, va2_s_gauss] = lambert_gauss(rd, ra, dt2_s, mu_sun_km, de);
659 [vd2_l_gauss, va2_l_gauss] = lambert_gauss(rd, ra, dt2_l, mu_sun_km, de);
660
661 % Universal variables
662 A = sqrt(ra_mag*rd_mag)*sin(dnu) / ( sqrt( 1 - cos(dnu) ) );
663 psi_n = 0;
664 c2 = 1/2;
665 c3 = 1/6;
666 psi_up = 4*pi^2;
667 psi_low = -4*pi;
668 %
669 dtn = 1;
670 while abs( dtn - dt1_s ) > 0.01

```

```

671 %
672 %     yn = rd_mag + ra_mag + A*(psi_n*c3 - 1)/sqrt(c2);
673 %     if A > 0 && yn < 0
674 %         psi_low = psi_low + 0.01;
675 %     end
676 %
677 %     xn = sqrt(yn/c2);
678 %     dtn = (xn^3*c3 + A*sqrt(yn)) / (sqrt(mu_sun_k));
679 %
680 %     if dtn < dtl_s
681 %         psi_n = psi_low;
682 %     else
683 %         psi_n = psi_up;
684 %     end
685 %
686 %     psi_npl = (psi_up + psi_low)/2;
687 %
688 %     % find c2 and c3
689 %
690 %     psi_n = psi_npl;
691 %
692 % end
693
694
695 function [vd, va] = LAMBERTBATTIN_km_sun_June(rd, ra, dm, dt)
696 % VALLADO BATTIN METHOD
697
698 % mu = 3.986004418e14; % m3/s2
699 mu_sun_m = 1.32712440018e20;
700 mu_sun_km = mu_sun_m / (1000^3);
701 mu = mu_sun_km;
702
703 % cos and sin dnu
704 ra_mag = norm(ra);
705 rd_mag = norm(rd);
706 cos_dnu = dot(rd, ra)/(rd_mag*ra_mag);
707 dnu = acos(cos_dnu);
708 sin_dnu = sin(dnu);
709
710 % the angle needs to be positive to work for the long way
711 if dnu < 0.0
712     dnu = 2*pi + dnu;
713 end
714
715 % geometry
716 c = sqrt( rd_mag^2 + ra_mag^2 - 2*rd_mag*ra_mag*cos_dnu );
717 s = ( rd_mag + ra_mag + c ) / 2;
718 eps = (ra_mag - rd_mag) / rd_mag;
719
720 % tan2w —> rdp
721 sqrt_rda = sqrt( ra_mag / rd_mag );
722 tan22w = ( eps^2/4 ) / ( sqrt_rda + sqrt_rda^2*( 2 + sqrt_rda ) );
723 rdp = sqrt( rd_mag*ra_mag ) * ( cos( dnu/4 )^2 + tan22w );
724
725 % obtain L
726 if dnu < pi
727     num = sin(dnu/4)^2 + tan22w;
728     den = sin(dnu/4)^2 + tan22w + cos(dnu/2);
729 else
730     num = cos(dnu/4)^2 + tan22w - cos(dnu/2);
731     den = cos(dnu/4)^2 + tan22w;
732 end
733 L = num / den;
734
735 m = mu * dt^2 / ( 8*rdp^3 );
736
737 % orbit is elliptical
738 xn = L;

```



```

739 eta = xn / ( sqrt( 1+xn ) + 1 )^2;
740 x = -xn;
741
742 i = 0;
743 % loop
744 while (1)
745
746     i = i + 1;
747
748     x = xn;
749
750     % omg crazy recursion in fraction denominator
751     tempx = seebatt(x);
752
753     % h1
754     num = (L+x)^2 * ( 1 + 3*x + tempx );
755     den = ( 1 + 2*x + L ) * ( 4*x + tempx*( 3+x ) );
756     h1 = num / den;
757
758     % h2
759     num = m*(x - L + tempx);
760     h2 = num / den;
761
762     % solve cubic y^3 - y^2 - h1*y^2 - h2 = 0
763     rts = roots([1 -1 -h1 -h2]);
764
765     B = 27*h2 / ( 4*( 1 + h1 )^3 );
766     U = B / ( 2*( sqrt( 1+B ) + 1 ) );
767
768     K = seebattk(U);
769     y = (1 + h1)/3 * ( 2 + ( sqrt(1+B) )/( 1 + 2*U*K^2 ) );
770     xn = sqrt( ( (1-L)/2 )^2 + ( m/y^2 ) ) - (1+L)/2;
771
772     if abs(xn - x) < 0.000001 && i > 30
773         break
774     end
775
776 end
777
778 % semi-major axis!!!
779 a = mu_sun_km*dt^2 / ( 16 * rdp^2 * x * y^2 );
780
781 if a > 0
782
783     % obtain f, g, and dg
784     alpha = 2 * asin( sqrt( s/(2*a) ) );
785     beta = 2 * asin( sqrt( (s-c)/(2*a) ) );
786
787     % min
788     amin = s / 2;
789     tmin = sqrt( amin^3/mu_sun_km ) * ( pi - beta + sin(beta) );
790     if dt > tmin
791         alpha = 2*pi - alpha;
792     end
793     de = alpha - beta;
794
795     f = 1 - a/rd_mag * ( 1 - cos(de) );
796     g = dt - sqrt( a^3/mu_sun_km ) * ( de - sin(de) );
797     dg = 1 - a/ra_mag * ( 1 - cos(de) );
798
799 else
800
801     disp('oh no')
802
803 end
804
805 % velocities!!
806 vd = (ra - f*rd) / g;

```

```
807 va = (dg*ra - rd) / g;  
808  
809 end
```