Table of Contents

Aero 351 Orbits Final

Joel Surfleet


```
clear; close all; clc;
mu_sun = 132712440018;
mu_earth = 398600;
mu_venus = 324859;
r_sun = 696000;     % km
r_earth = 6378;     % km
r_venus = 6052;     % km
options = odeset('RelTol',1e-8,'AbsTol',1e-8);
```

Problem 1

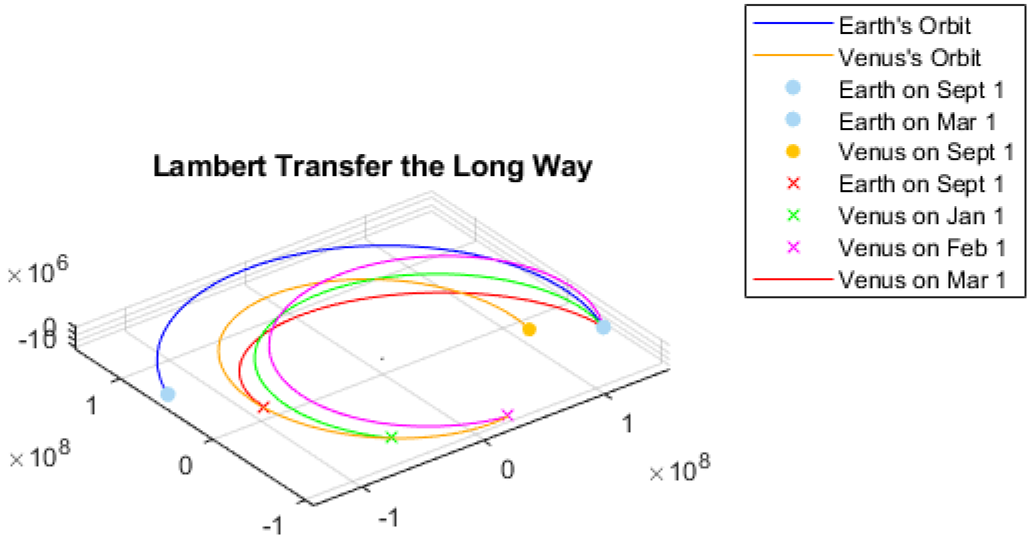
Exit date: 1 september, 2023 Earth > Venus 500 km alt circular orbit @ earth 200 x 10000 km alt orbit @ venus Possible arrival dates: jan 1, feb 1, march 1 2024

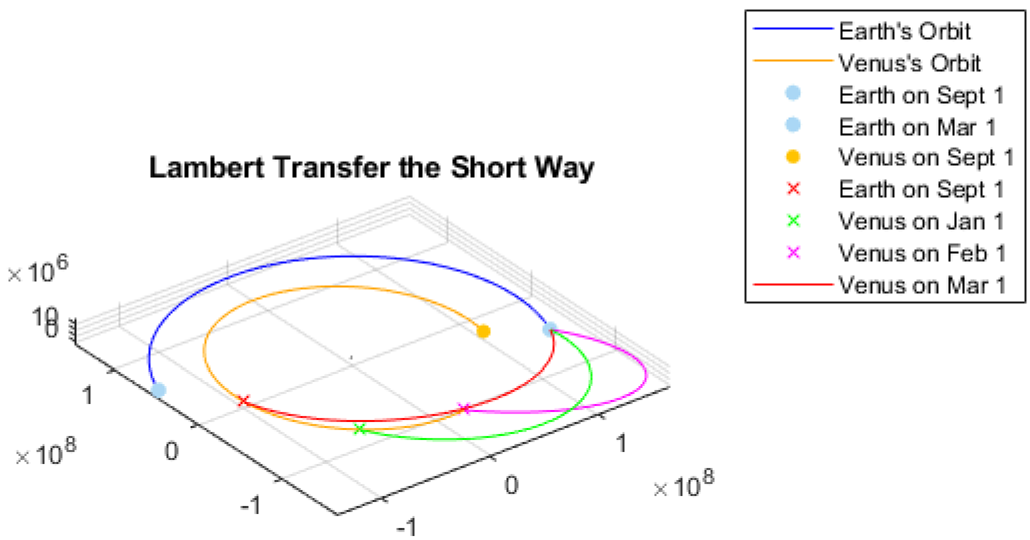
```
% Find Julian Dates of the given dates

T0.J = juldat(1, 9, 2023);
T1.J = juldat(1, 1, 2024);
T2.J = juldat(1, 2, 2024);
T3.J = juldat(1, 3, 2024);

% find the time taken for each transfer

T1.t = (T1.J-T0.J)*86400;
T2.t = (T2.J-T0.J)*86400;
T3.t = (T3.J-T0.J)*86400;

% find the positions of earth and venus on Sept 1

[T0.R,T0.V]   = planetRV(3,T0.J);
[T0.Rv,T0.Vv] = planetRV(2,T0.J);
[T1.R,T1.V]   = planetRV(2,T1.J);
[T2.R,T2.V]   = planetRV(2,T2.J);
[T3.R,T3.V]   = planetRV(2,T3.J);

% find lambert velocitys for the 6 possibilities
[T1.V1,T1.V2] = lamberts(T0.R,T1.R,T1.t,mu_sun,"prograde");
```

```
[T2.V1,T2.V2] = lamberts(T0.R,T2.R,T2.t,mu_sun,"prograde");
[T3.V1,T3.V2] = lamberts(T0.R,T3.R,T3.t,mu_sun,"prograde");
[T1.V3,T1.V4] = lamberts(T0.R,T1.R,T1.t,mu_sun,"retrograde");
[T2.V3,T2.V4] = lamberts(T0.R,T2.R,T2.t,mu_sun,"retrograde");
[T3.V3,T3.V4] = lamberts(T0.R,T3.R,T3.t,mu_sun,"retrograde");

% Propagate Earth and Venus's Trajectory
timespan = [0 T3.t];
initialstate = [T0.R T0.V];
[~,T0.earth] = ode45(@twobody, timespan, initialstate, options, mu_sun);

initialstate = [T0.Rv T0.Vv];
[~,T3.venus] = ode45(@twobody, timespan, initialstate, options, mu_sun);

% Propagate lambert trajectories

timespan = [0 T1.t];
initialstate = [T0.R T1.V1];
[~,T1.pro] = ode45(@twobody, timespan, initialstate, options, mu_sun);
initialstate = [T0.R T1.V3];
[~,T1.retro] = ode45(@twobody, timespan, initialstate, options, mu_sun);

timespan = [0 T2.t];
initialstate = [T0.R T2.V1];
[~,T2.pro] = ode45(@twobody, timespan, initialstate, options, mu_sun);
initialstate = [T0.R T2.V3];
[~,T2.retro] = ode45(@twobody, timespan, initialstate, options, mu_sun);

timespan = [0 T3.t];
initialstate = [T0.R T3.V1];
[~,T3.pro] = ode45(@twobody, timespan, initialstate, options, mu_sun);
initialstate = [T0.R T3.V3];
[~,T3.retro] = ode45(@twobody, timespan, initialstate, options, mu_sun);

% Plot Long Way
figure('name','Long Way','numbertitle','off')

[xsph,ysph,zsph] = sphere;
surf(r_sun*xsph,r_sun*ysph,r_sun*zsph,"LineStyle","none",'FaceColor','k')

hold on; grid on; axis equal;

plot3(T0.earth(:,1),T0.earth(:,2),T0.earth(:,3),'color','b')
plot3(T3.venus(:,1),T3.venus(:,2),T3.venus(:,3),'color','#FFA500')
plot3(T0.R(1),T0.R(2),T0.R(3),'.','markersize',20,'color','#ADD8F6')
plot3(T0.earth(end,1),T0.earth(end,2),T0.earth(end,3),'.','markersize',20,'color','#ADD8F6
plot3(T0.Rv(1),T0.Rv(2),T0.Rv(3),'.','markersize',18,'color','#FFC500')
plot3(T1.R(1),T1.R(2),T1.R(3),'x','color','r')
plot3(T2.R(1),T2.R(2),T2.R(3),'x','color','g')
plot3(T3.R(1),T3.R(2),T3.R(3),'x','color','m')

plot3(T1.pro(:,1),T1.pro(:,2),T1.pro(:,3),'color','r')
plot3(T2.pro(:,1),T2.pro(:,2),T2.pro(:,3),'color','g')
plot3(T3.pro(:,1),T3.pro(:,2),T3.pro(:,3),'color','m')
```

```
title("Lambert Transfer the Long Way")
legend("","Earth's Orbit","Venus's Orbit","Earth on Sept 1","Earth on Mar
 1","Venus on Sept 1","Earth on Sept 1","Venus on Jan 1","Venus on Feb
 1","Venus on Mar 1")

% Plot Short Way
figure('name','Short Way','numbertitle','off')

[xsph,ysph,zsph] = sphere;
surf(r_sun*xsph,r_sun*ysph,r_sun*zsph,"LineStyle","none",'FaceColor','k')

hold on; grid on; axis equal;

plot3(T0.earth(:,1),T0.earth(:,2),T0.earth(:,3),'color','b')
plot3(T3.venus(:,1),T3.venus(:,2),T3.venus(:,3),'color','#FFA500')
plot3(T0.R(1),T0.R(2),T0.R(3),'.','markersize',20,'color','#ADD8F6')
plot3(T0.earth(end,1),T0.earth(end,2),T0.earth(end,3),'.','markersize',20,'color','#ADD8F6
plot3(T0.Rv(1),T0.Rv(2),T0.Rv(3),'.','markersize',18,'color','#FFC500')
plot3(T1.R(1),T1.R(2),T1.R(3),'x','color','r')
plot3(T2.R(1),T2.R(2),T2.R(3),'x','color','g')
plot3(T3.R(1),T3.R(2),T3.R(3),'x','color','m')

plot3(T1.retro(:,1),T1.retro(:,2),T1.retro(:,3),'color','r')
plot3(T2.retro(:,1),T2.retro(:,2),T2.retro(:,3),'color','g')
plot3(T3.retro(:,1),T3.retro(:,2),T3.retro(:,3),'color','m')

title("Lambert Transfer the Short Way")
legend("","Earth's Orbit","Venus's Orbit","Earth on Sept 1","Earth on Mar
 1","Venus on Sept 1","Earth on Sept 1","Venus on Jan 1","Venus on Feb
 1","Venus on Mar 1")

% Calculate the r values for the initial and final circular orbits
r0 = 500 + r_earth;

rp = 200   + r_venus;
ra = 10000 + r_venus;

a = (ra + rp) / 2;
ecc = 1 - (rp / a);

% DeltaV for Transfer 1, Prograde
T1.Vinfpro1 = norm(T1.V1 - T0.V);
T1.deltaV1 = sqrt(T1.Vinfpro1^2 + 2*mu_earth/r0)-sqrt(mu_earth/r0);
T1.Vinfpro2 = norm(T1.V2 - T1.V);
T1.deltaV2 = sqrt(T1.Vinfpro2^2 + 2*mu_venus/rp)-sqrt(mu_venus/rp*(1+ecc));
T1.deltaVpro = T1.deltaV1 + T1.deltaV2;

% DeltaV for Transfer 1, Retrograde
T1.Vinfretro1 = norm(T1.V3 - T0.V);
T1.deltaV3 = sqrt(T1.Vinfretro1^2 + 2*mu_earth/r0)-sqrt(mu_earth/r0);
T1.Vinfretro2 = norm(T1.V4 - T1.V);
T1.deltaV4 = sqrt(T1.Vinfretro2^2 + 2*mu_venus/rp)-sqrt(mu_venus/rp*(1+ecc));
T1.deltaVretro = T1.deltaV3 + T1.deltaV4;
```

```
% DeltaV for Transfer 2, Prograde
T2.Vinfpro1 = norm(T2.V1 - T0.V);
T2.deltaV1 = sqrt(T2.Vinfpro1^2 + 2*mu_earth/r0)-sqrt(mu_earth/r0);
T2.Vinfpro2 = norm(T2.V2 - T2.V);
T2.deltaV2 = sqrt(T2.Vinfpro2^2 + 2*mu_venus/rp)-sqrt(mu_venus/rp*(1+ecc));
T2.deltaVpro = T2.deltaV1 + T2.deltaV2;

% DeltaV for Transfer 2, Retrograde
T2.Vinfretro1 = norm(T2.V3- T0.V);
T2.deltaV3 = sqrt(T2.Vinfretro1^2 + 2*mu_earth/r0)-sqrt(mu_earth/r0);
T2.Vinfretro2 = norm(T2.V4 - T2.V);
T2.deltaV4 = sqrt(T2.Vinfretro2^2 + 2*mu_venus/rp)-sqrt(mu_venus/rp*(1+ecc));
T2.deltaVretro = T2.deltaV3 + T2.deltaV4;

% DeltaV for Transfer 1, Prograde
T3.Vinfpro1 = norm(T3.V1 - T0.V);
T3.deltaV1 = sqrt(T3.Vinfpro1^2 + 2*mu_earth/r0)-sqrt(mu_earth/r0);
T3.Vinfpro2 = norm(T3.V2 - T3.V);
T3.deltaV2 = sqrt(T3.Vinfpro2^2 + 2*mu_venus/rp)-sqrt(mu_venus/rp*(1+ecc));
T3.deltaVpro = T3.deltaV1 + T3.deltaV2;

% DeltaV for Transfer 1, Retrograde
T3.Vinfretro1 = norm(T3.V3 - T0.V);
T3.deltaV3 = sqrt(T1.Vinfretro1^2 + 2*mu_earth/r0)-sqrt(mu_earth/r0);
T3.Vinfretro2 = norm(T3.V4 - T3.V);
T3.deltaV4 = sqrt(T3.Vinfretro2^2 + 2*mu_venus/rp)-sqrt(mu_venus/rp*(1+ecc));
T3.deltaVretro = T3.deltaV3 + T3.deltaV4;

% Disp it
disp("*****************************************************")
disp("**********************Problem 1**********************")
disp("*****************************************************")

disp(" ")
disp("Delta-V for Transfer 1, prograde:   " + T1.deltaVpro)
disp("Delta-V for Transfer 1, retrograde: " + T1.deltaVretro)
disp(" ")
disp("Delta-V for Transfer 2, prograde:   " + T2.deltaVpro)
disp("Delta-V for Transfer 2, retrograde: " + T2.deltaVretro)
disp(" ")
disp("Delta-V for Transfer 3, prograde:   " + T3.deltaVpro)
disp("Delta-V for Transfer 3, retrograde: " + T3.deltaVretro)
disp(" ")
```


Lambert Transfer the Long Way
Earth's Orbit
Venus's Orbit
Earth on Sept 1
Earth on Mar 1
Venus on Sept 1
Earth on Sept 1
Venus on Jan 1
Venus on Feb 1
Venus on Mar 1
×10^6
×10^8
×10^8


Lambert Transfer the Short Way
Earth's Orbit
Venus's Orbit
Earth on Sept 1
Earth on Mar 1
Venus on Sept 1
Earth on Sept 1
Venus on Jan 1
Venus on Feb 1
Venus on Mar 1
×10^6
×10^8
×10^8

Problem 2

heliocentric hohmann transfer ellipse P = 3/4 year r aphelion = 1 au r earth = 1 au circular r perigee = 10000 find deltaV

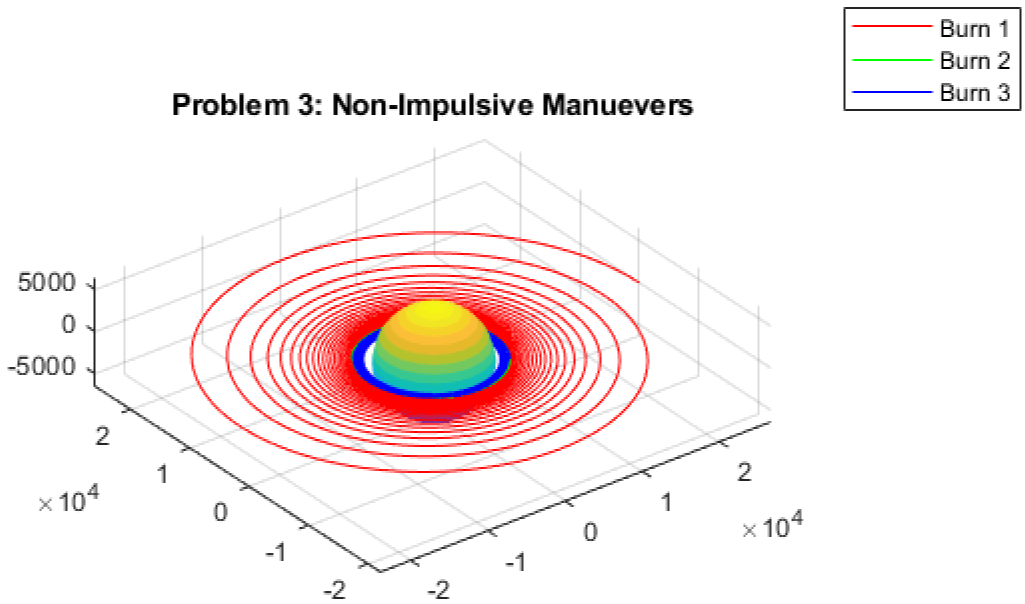
```
perigee = 10000 + r_earth;

AU = 149598000; % Km

year = 365.25 * 86400; % seconds

a = (((3/4)*year*sqrt(mu_sun))/(2*pi))^(2/3);

perihelion = (2*a) - AU;

a = (perihelion+AU)/2;

vt = sqrt(2*mu_sun)*sqrt(1/AU - 1/(2*a));
vearth = sqrt(mu_sun/AU);

% use that velocity to find vinf entering jupiter's SOI

vinf1 = vt - vearth;

% use that vinf to calculate the hyperbolic trajectory

ecc = 1 + perigee*vinf1^2/mu_earth;

delta = -2*asin(1/ecc); %rad

% Use that hyperbolic trajectory to calculate deltaV

Vinf2 = [vinf1*cos(delta) vinf1*sin(delta) 0];

Vearth = [vearth 0 0];

V = Vinf2 + Vearth;

deltaV = norm(V - [vt 0 0]);

disp("*****************************************************")
disp("*********************Problem 2*********************")
disp("*****************************************************")

disp(" ")
disp("Delta-V for Earth Flyby: " + deltaV)
disp(" ")
```

```
*****************************************************
*********************Problem 2*********************
*****************************************************

Delta-V for Earth Flyby: 4.5781
```

Problem 3

```
Isp = 5000;     % s
T   = -0.006;   % kN
m   = 600;      % kg

R = [26578;0;0];
V = [0;3.8726;0];

state0 = [R;V;m];
[~,state1] = ode45(@nonimpulse,[0 3.5*86400],state0,options,mu_earth,T,Isp);
[~,state2] = ode45(@twobody,[0 5.0*86400],state1(end,1:6),options,mu_earth);
[~,state3] = ode45(@nonimpulse,[0 0.6*86400],[state2(end,1:6)
 state1(end,7)],options,mu_earth,T,Isp);

R = state3(end,1:3);
V = state3(end,4:6);

coe = RV2COE(R,V,mu_earth);
a = coe(7);
ecc = coe(2);

Rp = a * (1 - ecc);

figure('name','Constant Thrust plot','numbertitle','off')

[xsph,ysph,zsph] = sphere;
surf(r_earth*xsph,r_earth*ysph,r_earth*zsph,"LineStyle","none")

hold on; grid on; axis equal;

plot3(state1(:,1),state1(:,2),state1(:,3),'color','r')
plot3(state2(:,1),state2(:,2),state2(:,3),'color','g')
plot3(state3(:,1),state3(:,2),state3(:,3),'color','b')

title("Problem 3: Non-Impulsive Manuevers")
legend("","Burn 1","Burn 2","Burn 3")

disp("*****************************************************")
disp("*********************Problem 3*********************")
disp("*****************************************************")

disp(" ")
disp("Altitude of Perigee: " + (Rp - r_earth))
disp("Final Mass: " + state3(end,7))
disp(" ")
```

```
*****************************************************
*********************Problem 3*********************
*****************************************************

Altitude of Perigee: 144.0178
Final Mass: 556.6679
```



Problem 4

```
Rp = 200 + r_earth;
Ra = Rp;
V = sqrt(mu_earth/Rp);
dV = 1.075;
ecc = (Ra-Rp)/(Ra+Rp);
a = (Rp + Ra)/2;

burns = 0;

while ecc < 1
    P(burns+1) = 2*pi/sqrt(mu_earth)*a^(3/2);
    V = V + dV;
    burns = burns + 1;
    coe = RV2COE([Rp 0 0],[0 V 0],mu_earth);
    ecc = coe(2);
    a = coe(7);
end

disp("*****************************************************")
disp("*********************Problem 4**********************")
disp("*****************************************************")
```

```
disp(" ")
disp("Total Burns: " + burns + " burns")
disp("Time from first burn to last burn: " + (sum(P(2)) + sum(P(3))) / 3600
 + " hrs")
disp(" ")
```

```
****************************************************
********************Problem 4*********************
****************************************************

Total Burns: 3 burns
Time from first burn to last burn: 9.011 hrs
```

Problem 5

```
disp("****************************************************")
disp("********************Problem 5*********************")
disp("****************************************************")

disp(" ")
disp("Vp = c*Va" + newline + "Vp = mu/h*(1+ecc)" + newline + ...
    "Va = mu/h*(1-ecc)" + newline + "c = Vp/Va" + newline + ...
    "c = mu/h*(1+ecc) / mu/h*(1-ecc)" + newline + ...
    "c = (1+ecc) / (1-ecc)" + newline + "c * (1 - ecc) = 1 + ecc" ...
    + newline + "c - c * ecc = 1 + ecc" + newline + ...
    "c - c * ecc - ecc = 1" + newline + "c - ecc * (c + 1) = 1" + ...
    newline+"c - 1 = ecc * (c + 1)" + newline + "(c - 1)/(c + 1) = ecc"...
    + newline + "ecc = (c - 1)/(c + 1)")
disp(" ")
```

```
****************************************************
********************Problem 5*********************
****************************************************

Vp = c*Va
Vp = mu/h*(1+ecc)
Va = mu/h*(1-ecc)
c = Vp/Va
c = mu/h*(1+ecc) / mu/h*(1-ecc)
c = (1+ecc) / (1-ecc)
c * (1 - ecc) = 1 + ecc
c - c * ecc = 1 + ecc
c - c * ecc - ecc = 1
c - ecc * (c + 1) = 1
c - 1 = ecc * (c + 1)
(c - 1)/(c + 1) = ecc
ecc = (c - 1)/(c + 1)
```

Fun Stuff

```
function J0 = juldat(D,M,Y)
```

```
J0 = 367*Y - floor((7*(Y+floor((M+9)/12)))/4) + floor((275*M)/9) + D +
 1721013.5;

end

function [planet_coes] = AERO351planetary_elements2(planet_id,T)
% Planetary Ephemerides from Meeus (1991:202-204) and J2000.0
% Output:
% planet_coes
% a = semimajor axis (km)
% ecc = eccentricity
% inc = inclination (degrees)
% raan = right ascension of the ascending node (degrees)
% w_hat = longitude of perihelion (degrees)
% L = mean longitude (degrees)

% Inputs:
% planet_id - planet identifier:
% 1 = Mercury
% 2 = Venus
% 3 = Earth
% 4 = Mars
% 5 = Jupiter
% 6 = Saturn
% 7 = Uranus
% 8 = Neptune

if planet_id == 1
    a = 0.387098310; % AU but in km later
    ecc = 0.20563175 + 0.000020406*T - 0.0000000284*T^2 - 0.00000000017*T^3;
    inc = 7.004986 - 0.0059516*T + 0.00000081*T^2 + 0.000000041*T^3; %degs
    raan = 48.330893 - 0.1254229*T-0.00008833*T^2 - 0.000000196*T^3; %degs
    w_hat = 77.456119 +0.1588643*T -0.00001343*T^2+0.000000039*T^3; %degs
    L = 252.250906+149472.6746358*T-0.00000535*T^2+0.000000002*T^3; %degs
elseif planet_id == 2
    a = 0.723329820; % AU
    ecc = 0.00677188 - 0.000047766*T + 0.000000097*T^2 + 0.00000000044*T^3;
    inc = 3.394662 - 0.0008568*T - 0.00003244*T^2 + 0.000000010*T^3; %degs
    raan = 76.679920 - 0.2780080*T-0.00014256*T^2 - 0.000000198*T^3; %degs
    w_hat = 131.563707 +0.0048646*T -0.00138232*T^2-0.000005332*T^3; %degs
    L = 181.979801+58517.8156760*T+0.00000165*T^2-0.000000002*T^3; %degs
elseif planet_id == 3
    a = 1.000001018; % AU
    ecc = 0.01670862 - 0.000042037*T - 0.0000001236*T^2 + 0.00000000004*T^3;
    inc = 0.0000000 + 0.0130546*T - 0.00000931*T^2 - 0.000000034*T^3; %degs
    raan = 0.0; %degs
    w_hat = 102.937348 + 0.3225557*T + 0.00015026*T^2 + 0.000000478*T^3; %degs
    L = 100.466449 + 35999.372851*T - 0.00000568*T^2 + 0.000000000*T^3; %degs
elseif planet_id == 4
    a = 1.523679342; % AU
    ecc = 0.09340062 + 0.000090483*T - 0.00000000806*T^2 - 0.00000000035*T^3;
    inc = 1.849726 - 0.0081479*T - 0.00002255*T^2 - 0.000000027*T^3; %degs
    raan = 49.558093 - 0.2949846*T-0.00063993*T^2 - 0.000002143*T^3; %degs
```

```
    w_hat = 336.060234 +0.4438898*T -0.00017321*T^2+0.000000300*T^3; %degs
    L = 355.433275+19140.2993313*T+0.00000261*T^2-0.000000003*T^3; %degs
elseif planet_id == 5
    a = 5.202603191 + 0.0000001913*T; % AU
    ecc = 0.04849485+0.000163244*T - 0.0000004719*T^2 + 0.00000000197*T^3;
    inc = 1.303270 - 0.0019872*T + 0.00003318*T^2 + 0.000000092*T^3; %degs
    raan = 100.464441 + 0.1766828*T+0.00090387*T^2 - 0.000007032*T^3; %degs
    w_hat = 14.331309 +0.2155525*T +0.00072252*T^2-0.000004590*T^3; %degs
    L = 34.351484+3034.9056746*T-0.00008501*T^2+0.000000004*T^3; %degs
elseif planet_id == 6
    a = 9.5549009596 - 0.0000021389*T; % AU
    ecc = 0.05550862 - 0.000346818*T -0.0000006456*T^2 + 0.00000000338*T^3;
    inc = 2.488878 + 0.0025515*T - 0.00004903*T^2 + 0.000000018*T^3; %degs
    raan = 113.665524 - 0.2566649*T-0.00018345*T^2 + 0.000000357*T^3; %degs
    w_hat = 93.056787 +0.5665496*T +0.00052809*T^2-0.000004882*T^3; %degs
    L = 50.077471+1222.1137943*T+0.00021004*T^2-0.000000019*T^3; %degs
elseif planet_id == 7
    a = 19.218446062-0.0000000372*T+0.00000000098*T^2; % AU
    ecc = 0.04629590 - 0.000027337*T + 0.0000000790*T^2 + 0.00000000025*T^3;
    inc = 0.773196 - 0.0016869*T + 0.00000349*T^2 + 0.00000000016*T^3; %degs
    raan = 74.005947 + 0.0741461*T+0.00040540*T^2 +0.000000104*T^3; %degs
    w_hat = 173.005159 +0.0893206*T -0.00009470*T^2+0.000000413*T^3; %degs
    L = 314.055005+428.4669983*T-0.00000486*T^2-0.000000006*T^3; %degs
elseif planet_id == 8
    a = 30.110386869-0.0000001663*T+0.00000000069*T^2; % AU
    ecc = 0.00898809 + 0.000006408*T -0.0000000008*T^2;
    inc = 1.769952 +0.0002557*T +0.00000023*T^2 -0.0000000000*T^3; %degs
    raan = 131.784057 - 0.0061651*T-0.00000219*T^2 - 0.000000078*T^3; %degs
    w_hat = 48.123691 +0.0291587*T +0.00007051*T^2-0.000000000*T^3; %degs
    L = 304.348665+218.4862002*T+0.00000059*T^2-0.000000002*T^3; %degs
end

planet_coes = [a;ecc;inc;raan;w_hat;L];
%Convert to km:
au = 149597870;
planet_coes(1) = planet_coes(1)*au;
end

function [R,V,coe] = planetRV(planet_id,J)
mu = 132712440018;

T = (J - 2451545)/36525;

[planet_coes] = AERO351planetary_elements2(planet_id,T);

a = planet_coes(1);
ecc = planet_coes(2);
inc = planet_coes(3);
RAAN = anglecheck(planet_coes(4));
w_hat = anglecheck(planet_coes(5));
L = anglecheck(planet_coes(6));

h = sqrt(mu*a*(1-ecc^2));
w = anglecheck(w_hat - RAAN);
```

```
Me = anglecheck(L - w_hat);
Me = Me * pi / 180;

if Me < pi
    E = Me + ecc/2;
elseif Me > pi
    E = Me - ecc/2;
end

f = @(E) E - ecc*sin(E) - Me;
fp = @(E) 1 - ecc*cos(E);

i = 1;

while f(E)/fp(E) < 10^-8
    E = E - f(E)/fp(E);
    i = i + 1;
    if i >= 100
        break
    end
end

E = anglecheck(E*180/pi);

theta = anglecheck(2*atand(sqrt((1+ecc)/(1-ecc))*tand(E/2)));

coe = [h, ecc, RAAN, inc, w, theta];

[R,V] = COE2RV(coe,mu);

end

function [R,V] = COE2RV(coe,mu)

% Converts the 6 classical orbital elements into a position and velocity
% vector
%
% input:
%   coe: an array with the classical orbital elemnts:
%     (1) h     = Angular Velocity
%     (2) ecc   = Eccentricity
%     (3) RAAN  = Right Ascension of ascending node
%     (4) inc   = Inclination in degrees
%     (5) w     = Argument of Perigee in degrees
%     (6) Theta  = True Anomaly in degrees
%   mu: gravitational parameter of object being orbitted

h     = coe(1);
ecc   = coe(2);
RAAN  = coe(3);
inc   = coe(4);
w     = coe(5);
theta = coe(6);
```

```matlab
% Find r and v in the perifocal frame
R_pf = (h^2/mu) * (1/(1+ecc*cosd(theta))) *
 (cosd(theta)*[1;0;0]+sind(theta)*[0;1;0]);
V_pf = (mu/h) * (-sind(theta)*[1;0;0] + (ecc+cosd(theta))*[0;1;0]);

% Perfom euler sequence to get the rotation matrix from perifocal to ECI
C = (basicrot(3,w)*basicrot(1,inc)*basicrot(3,RAAN))';

R = C*R_pf;
R = R';

V = C*V_pf;
V = V';

end

function [C] = basicrot(a,theta)

if a == 1
    C = [1 0 0 ; 0 cosd(theta) sind(theta); 0 -sind(theta) cosd(theta)];
elseif a == 2
    C = [cosd(theta) 0 -sind(theta); 0 1 0; sind(theta) 0 cosd(theta)];
elseif a == 3
    C = [cosd(theta) sind(theta) 0; -sind(theta) cosd(theta) 0; 0 0 1];
end
end

function [V1,V2,z] = lamberts(R1,R2,t,mu,type)

r1 = norm(R1);
r2 = norm(R2);

% Calculate change in Theta in rad
dtheta = r2trueanomaly(R1, R2, type);

% Calculate A
A = sin(dtheta) * sqrt(r1*r2) / sqrt(1-cos(dtheta));

% Establish y function
y = @(z) r1 + r2 + A * ( (z*stumpS(z) - 1) / sqrt(stumpC(z)) );

% Find z using bisection
F = @(z) (y(z)/stumpC(z))^(1.5) * stumpS(z) + A * sqrt(y(z)) - sqrt(mu)*t;

z0 = -100;
while F(z0) < 0
    z0 = z0 + 0.1;
    if z0 > 100
        z0 = 0;
    end
end

z = bisection(z0-2*pi, z0+2*pi, F, 10^-4);
```

```
% Calculate y
y_final = y(z);

% Calculate f, g , gdot
f = 1 - y_final/r1;
g = A * sqrt(y_final/mu);
gdot = 1 - y_final/r2;

% calculate v1, v2
V1 = (1/g)*(R2-f*R1);

V2 = (1/g)*(gdot*R2-R1);
end

function s = stumpS(z)
    if z > 0
        s = (sqrt(z) - sin(sqrt(z)))/(sqrt(z))^3;
    elseif z < 0
        s = (sinh(sqrt(-z)) - sqrt(-z))/(sqrt(-z))^3;
    else
        s = 1/6;
    end
end

function c = stumpC(z)
    if z > 0
        c = (1 - cos(sqrt(z)))/z;
    elseif z < 0
        c = (cosh(sqrt(-z)) - 1)/(-z);
    else
        c = 1/2;
    end
end

function c = bisection(a, b, f, TOL)
% Bisection Method
% Given initial interval [a, b] such that f(a)*f(b) < 0
while ((b - a)/2 > TOL)
    c = (a + b)/2;
    if f(c) == 0
        break;
    end
    if (f(a)*f(c) < 0)
        b = c;
    else
        a = c;
    end
end
end

function dtheta = r2trueanomaly(R1,R2,type)
% outputs difference in true anomaly between R1 and R2.
% RADIANS
```

```
r1 = norm(R1);
r2 = norm(R2);

dtheta = acos(dot(R1,R2)/r1/r2);

C12 = cross(R1,R2);

if type == "prograde"
    if C12(3) <= 0
        dtheta = 2*pi - dtheta;
    end
elseif type == "retrograde"
    if  C12(3) >= 0
        dtheta = 2*pi - dtheta;
    end
end

end

function dstate = twobody(t, state, mu)

% equation of motion for a two body system

x = state(1);
y = state(2);
z = state(3);
dx = state(4);
dy = state(5);
dz = state(6);

r = norm([x y z]);

ddx = -mu * x / r^3;
ddy = -mu * y / r^3;
ddz = -mu * z / r^3;

dstate = [dx;dy;dz;ddx;ddy;ddz]; % must be a column vector

end

function [theta] = anglecheck(theta)
if (theta >= 360)
    theta = theta - floor(theta/360)*360;
elseif (theta < 0)
    theta = theta - (floor(theta/360) - 1)*360;
end
end

function out = nonimpulse(t,y,mu,T,Isp)

r = norm(y(1:3));
v = norm(y(4:6));

out(1:3,1) = y(4:6);
```

```
out(4,1) = -mu*y(1)/(r^3) + (T/y(7))*(y(4)/v);
out(5,1) = -mu*y(2)/(r^3) + (T/y(7))*(y(5)/v);
out(6,1) = -mu*y(3)/(r^3) + (T/y(7))*(y(6)/v);

out(7,1) = T*1000/Isp/9.81;

end

function coe = RV2COE(R,V,mu)

%Specific Mechanical Energy
E = ((norm(V) ^ 2) / 2) - (mu/norm(R));

%Semi-Major Axis in km
a = -mu / (2 * E);

% angular momentum
H = cross(R,V);
h = norm(H);

%Calculate eccentricity vector
ECC = cross(V,H)/mu - (R/norm(R));

%Eccentrictry angle
ecc = norm(ECC);

%inclination
inc = acosd(H(3)/h);

%n
N = cross([0 0 1], H);

%right ascension of the ascending node
RAAN = acosd(N(1)/norm(N));

%quadrant check
if(N(2) < 0)
    RAAN = 360 - RAAN;
end

%argument of periapsis
w = acosd(dot(N,ECC)/(ecc * norm(N)));

%quadrant check
if(ECC(3) < 0)
    w = 360 - w;
end

%true anomaly
theta = acosd(dot(R,ECC)/(ecc * norm(R)));

%quadrant check
if(dot(R,V) < 0)
```

```
    theta = 360 - theta;
end

coe = [h, ecc, RAAN, inc, w, theta, a];
end
```

```
*****************************************************
*********************Problem 1**********************
*****************************************************

Delta-V for Transfer 1, prograde:   12.0432
Delta-V for Transfer 1, retrograde: 114.9566

Delta-V for Transfer 2, prograde:   9.9867
Delta-V for Transfer 2, retrograde: 107.8251

Delta-V for Transfer 3, prograde:   13.6484
Delta-V for Transfer 3, retrograde: 104.0448
```

Published with MATLAB® R2022a