Table of Contents


```
function Final351()
```

351 Final

Liam Hood

```
clear; close all; clc;
global mue ;
global mus ;
global muven ;
global re ;
global rven ;
global opts ;
global d2s ;
mue  = 398600 ;
muven = 324900 ;
re = 6378 ;
rven = 6052 ;
mus = 1.32712e11 ;
opts = odeset( 'AbsTol' , 1e-8 , 'RelTol' , 1e-8 ) ;
d2s = 24*60*60 ;
```

1

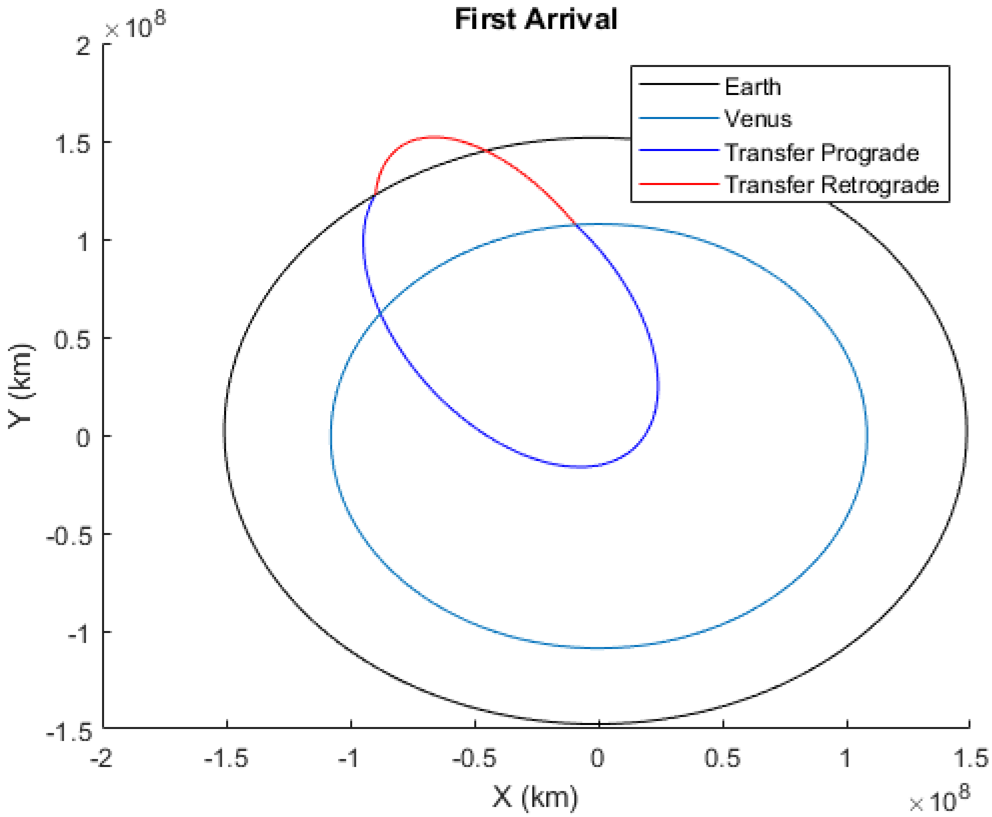
```
Problem1()
```

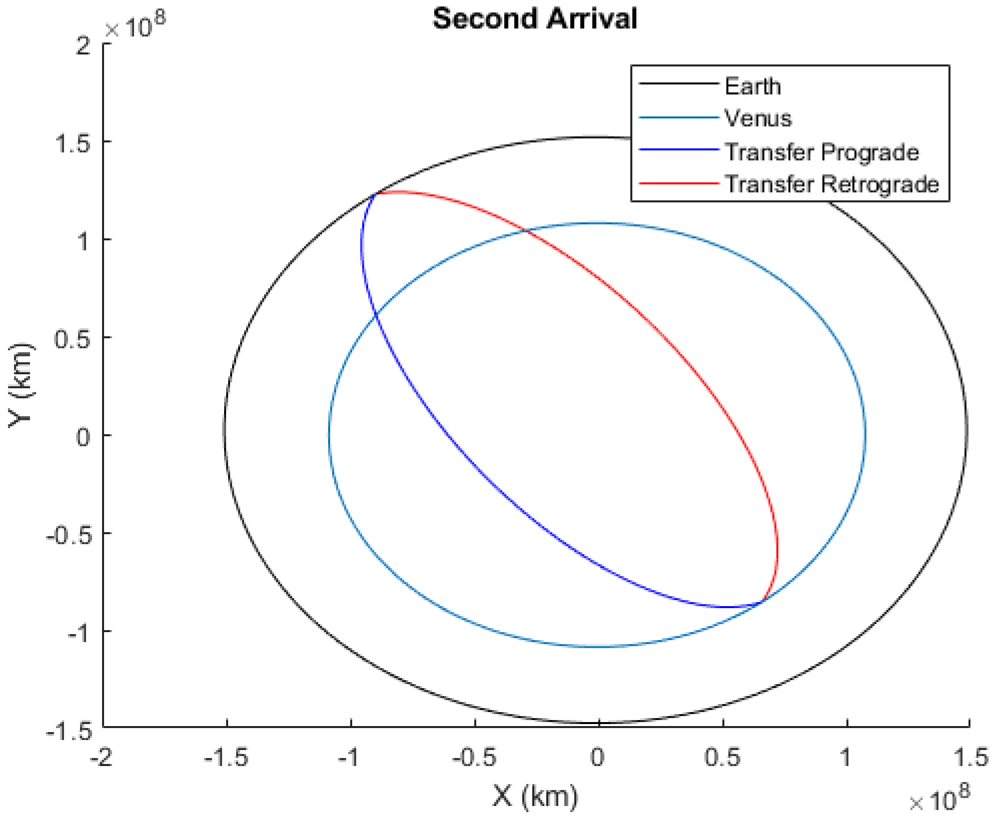
```
1
The s/c will need 3 burns to reach escape velocity
The time from the first burn to reaching escape velocity is
9.011 hours
This time seems correct because it only takes a few periods
to reach escape velocity and the period begins short in LEO.
The number of burns makes sense because the delta v is quite
high relative to how much the delta v needs to increase
```

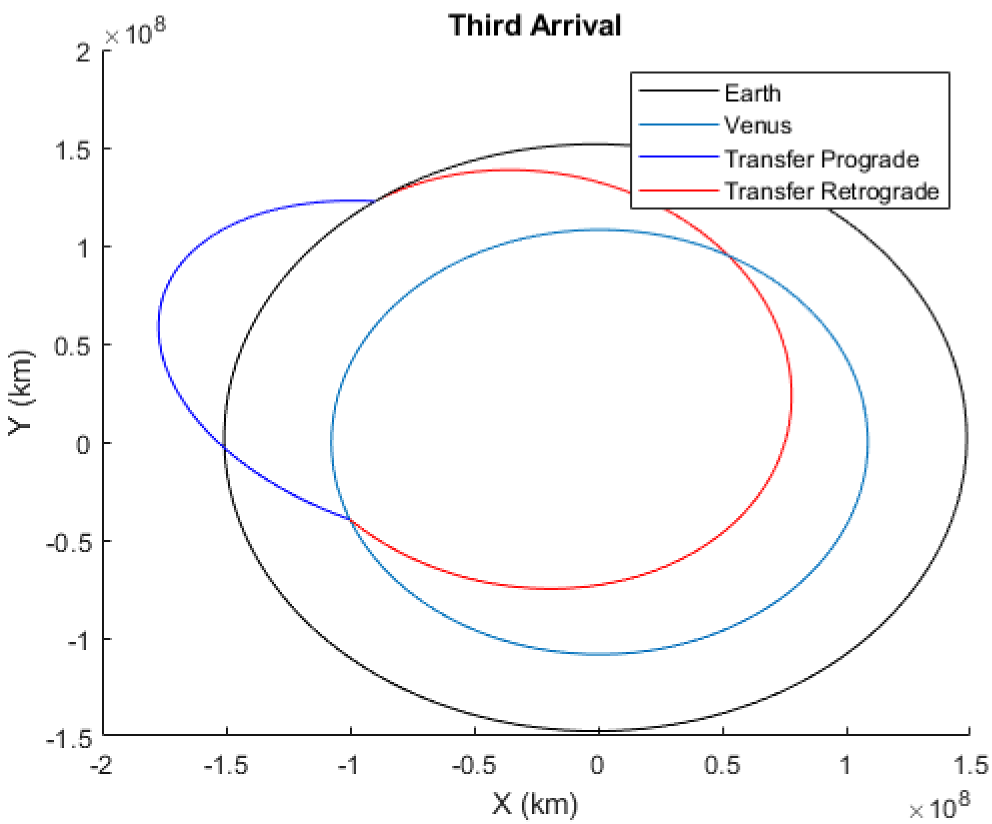
2

```
Problem2()
```

```
2
The best arrival date is 1 May 2018 with a delta v of 33.4781 km/s
This seems like a very high delta v but it is clearly not
an efficient transfer orbit so that makes sense
```




Second Arrival
Earth
Venus
Transfer Prograde
Transfer Retrograde
Y (km)
X (km)
×10^8
×10^8
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5


Third Arrival
Earth
Venus
Transfer Prograde
Transfer Retrograde
Y (km)
X (km)
×10^8
×10^8
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5

3

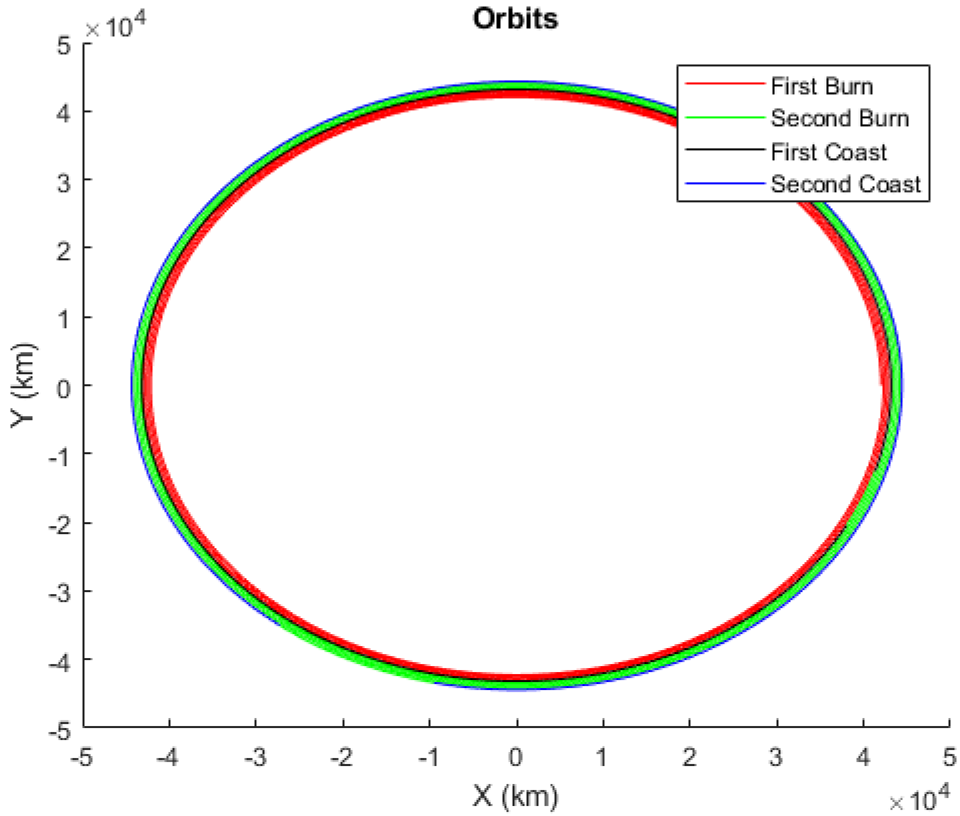
```
Problem3()
```

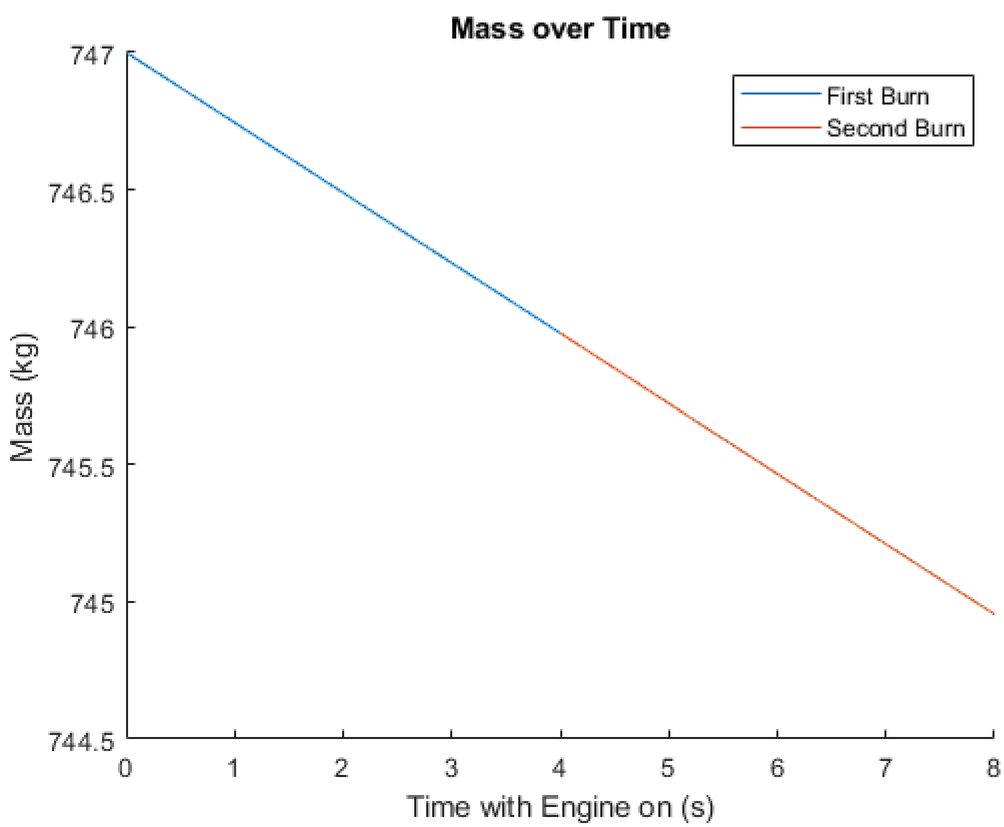
3
The heliocentric speed after fly by is 26.4849 km/s
This is a increase of 1.8892 km/s from initial speed of 26.4492 km/s
This seems correct because it there is a large v inf so there is not much time for Earth's gravity to rotate the vector and so the final delta v is small relative to the velocities
I had the s/c be retrograde because I think that it has to retrograde in order for a s/c going slower than a planet to peform a trailing edge fly by

4

```
Problem4()
```

4
The radius after the second coast is 44387.3014 km
This seems correct because the thrust is very low so the radius shouldn't increase dramatically
The mass after the second burn is 744.9544 kg
This seems correct since the Isp high and the mass does not change very much


Orbits
First Burn
Second Burn
First Coast
Second Coast
Y (km)
X (km)
$\times 10^4$
5
4
3
2
1
0
-1
-2
-3
-4
-5


Mass over Time
First Burn
Second Burn
Mass (kg)
Time with Engine on (s)
747
746.5
746
745.5
745
744.5
0
1
2
3
4
5
6
7
8

Code for Problems

```
function Problem1()
    disp( '1' )
    % Starting orbit
    z = 200 ; % initial altitude km
    rcirc = z + re ; % initial radius of orbit km
    vcirc = sqrt( mue/rcirc ) ; % initial speed of orbit km/s
    dv_Orbit = 1.075 ; % delta v imparted each orbit in km/s
    vesc = sqrt(2)*vcirc ; % parabolic escape velocity
    dv_Tot = vesc - vcirc ; % total change in velocity to reach
escape velocity
    nBurns = ceil( dv_Tot/dv_Orbit ) ; % number of burns to reach
vesc
    vPer = vcirc ;
    for ii = 1:nBurns-1
        vPer = vPer + dv_Orbit ; % New velocity at perigee
        h = vPer * rcirc ; % new h
        ecc = 1/((rcirc*mue)/h^2) -1 ;
        a = (h^2/mue)*(1/(1-ecc^2)) ; % semi-major axis
        T(ii) = ((2*pi)/sqrt(mue))*a^1.5 ; % Period of orbit which
is time between each burn since
    end
    tsec = sum(T) ; % total time in seconds
    t = tsec/(60*60) ; % hours
    disp([ 'The s/c will need ' , num2str( nBurns ) , ' burns to
reach escape velocity' ])
    disp( 'The time from the first burn to reaching escape
velocity is ' )
    disp([ num2str( t ) , ' hours' ])
    disp( 'This time seems correct because it only takes a few
periods ' )
    disp( 'to reach escape velocity and the period begins short in
LEO.' )
    disp( 'The number of burns makes sense because the delta v is
quite ' )
    disp( 'high relative to how much the delta v needs to increase
' )
end

function Problem2()
    disp( ' ' )
    disp( '2' )
    % Planetary states
        dateDeparture = [ 1 , 12 , 2018 ] ;
        [ ~ , ~ , ~ , JdDeparture ] = Julian( [ 12 , 0 , 0 ] ,
dateDeparture ) ;
        tDeparture = JdDeparture/365.25 ;
        coesEarth = planetary_elements( 3 , tDeparture ) ;
        [ rE , vE ] = Pcoes2state( coesEarth , mus ) ;

        dateArr1 = [ 1 , 3 , 2019 ] ;
```

```
            [ ~ , ~ , ~ , JdArr1 ] = Julian( [ 12 , 0 , 0 ] ,
dateArr1 ) ;
            tArr1 = JdArr1/365.25 ;
            coesVenus1 = planetary_elements( 2 , tArr1 ) ;
            [ rV1 , vV1 ] = Pcoes2state( coesVenus1 , mus ) ;

            dateArr2 = [ 1 , 4 , 2019 ] ;
            [ ~ , ~ , ~ , JdArr2 ] = Julian( [ 12 , 0 , 0 ] ,
dateArr2 ) ;
            tArr2 = JdArr2/365.25 ;
            coesVenus2 = planetary_elements( 2 , tArr2 ) ;
            [ rV2 , vV2 ] = Pcoes2state( coesVenus2 , mus ) ;

            dateArr3 = [ 1 , 5 , 2019 ] ;
            [ ~ , ~ , ~ , JdArr3 ] = Julian( [ 12 , 0 , 0 ] ,
dateArr3 ) ;
            tArr3 = JdArr3/365.25 ;
            coesVenus3 = planetary_elements( 2 , tArr3 ) ;
            [ rV3 , vV3 ] = Pcoes2state( coesVenus3 , mus ) ;

        % Interplanetary
            dt1 = ( -JdDeparture + JdArr1 )*d2s;
            dt2 = ( -JdDeparture + JdArr2 )*d2s ;
            dt3 = ( -JdDeparture + JdArr3 )*d2s;
            [ vDep1 , vArr1 ] = Lamberts2( rE , rV1 , dt1 , mus ,
1e-5 , 1 ) ;
            [ vDep1r , vArr1r ] = Lamberts2( rE , rV1 , dt1 , mus ,
1e-5 , 0 ) ;
            [ vDep2 , vArr2 ] = Lamberts2( rE , rV2 , dt2 , mus ,
1e-5 , 1 ) ;
            [ vDep2r , vArr2r ] = Lamberts2( rE , rV2 , dt2 , mus ,
1e-5 , 0 ) ;
            [ vDep3 , vArr3 ] = Lamberts2( rE , rV3 , dt3 , mus ,
1e-5 , 1 ) ;
            [ vDep3r , vArr3r ] = Lamberts2( rE , rV3 , dt3 , mus ,
1e-5 , 0 ) ;

            % Departure
            zDPark = 180 ;
            rDPark = re + zDPark ; % radius of departure park
            vDPark = sqrt( mue / rDPark ) ; % velocity of parking
orbit of departure
            RsoiE = 925000 ;
            vinfD(1) = norm( vDep1 - vE ) ;
            vinfD(2) = norm( vDep2 - vE ) ;
            vinfD(3) = norm( vDep3 - vE ) ;
            for ii = 1:3
                aHD = (mue/2)*((vinfD(ii)^2/2)-(mue/RsoiE))^(-1) ;
                eccHD = rDPark/aHD + 1 ;
                hHD = sqrt( rDPark*mue*(1+eccHD) ) ;
                vpHD(ii) = (mue/hHD)*(1+eccHD) ;
                dvD(ii) = norm( vpHD(ii) - vDPark ) ;
            end
            vinfDr(1) = norm( vDep1r - vE ) ;
```

```
            vinfDr(2) = norm( vDep2r - vE ) ;
            vinfDr(3) = norm( vDep3r - vE ) ;
            for ii = 1:3
                aHD = (mue/2)*((vinfDr(ii)^2/2)-(mue/RsoiE))^(-1) ;
                eccHD = rDPark/aHD + 1 ;
                hHD = sqrt( rDPark*mue*(1+eccHD) ) ;
                vpHDr(ii) = (mue/hHD)*(1+eccHD) ;
                dvDr(ii) = norm( vpHDr(ii) - vDPark ) ;
            end



            % Arrival
            zAa = 10000 ;
            rAa = zAa + rven ;
            zAp = 200 ;
            rAp = zAp + rven ;
            eccA = ( rAa - rAp )/( rAp + rAp ) ;
            hA = sqrt( rAp*muven*(1+eccA) ) ;
            vpA = (muven/hA)*(1+eccA) ;
            RsoiV = 616000 ;
            vinfA(1) = norm( vArr1 - vV1 ) ;
            vinfA(2) = norm( vArr2 - vV2 ) ;
            vinfA(3) = norm( vArr3 - vV3 ) ;
            for ii = 1:3
                aHA = (muven/2)*((vinfA(ii)^2/2)-(muven/RsoiV))^(-1) ;
                eccHA = rAp/aHA + 1 ;
                hHA = sqrt( rAp*muven*(1+eccHA) ) ;
                vpHA(ii) = (muven/hHA)*(1+eccHA) ;
                dvA(ii) = norm( vpHA(ii) - vpA ) ;
            end
            vinfAr(1) = norm( vArr1r - vV1 ) ;
            vinfAr(2) = norm( vArr2r - vV2 ) ;
            vinfAr(3) = norm( vArr3r - vV3 ) ;
            for ii = 1:3
                aHA = (muven/2)*((vinfA(ii)^2/2)-(muven/RsoiV))^(-1) ;
                eccHA = rAp/aHA + 1 ;
                hHA = sqrt( rAp*muven*(1+eccHA) ) ;
                vpHAr(ii) = (muven/hHA)*(1+eccHA) ;
                dvAr(ii) = norm( vpHAr(ii) - vpA ) ;
            end

            dv = dvA + dvD ;
            dvr = dvAr + dvDr ;
            dvr_best = min( dvr ) ;
            dv_best = min( dv ) ;
            disp([ 'The best arrival date is 1 May 2018 with a delta v
of ' , num2str( dv_best ) , ' km/s' ])
            disp( 'This seems like a very high delta v but it is
clearly not' )
            disp( 'an efficient transfer orbit so that makes sense' )

        [ tE , stateE ] = ode45( @TwoBodyMotion , [ 0 dt3 ].*4 ,
[ rE ; vE ] , opts , mus ) ;
```

```
        [ tV1 , stateV1 ] = ode45( @TwoBodyMotion , [ 0 dt1 ].*4 ,
[ rV1 ; vV1 ] , opts , mus ) ;
        [ tV2 , stateV2 ] = ode45( @TwoBodyMotion , [ 0 dt2 ].*4 ,
[ rV2 ; vV2 ] , opts , mus ) ;
        [ tV3 , stateV3 ] = ode45( @TwoBodyMotion , [ 0 dt3 ].*4 ,
[ rV3 ; vV3 ] , opts , mus ) ;
        [ t , stateT1 ] = ode45( @TwoBodyMotion , [ 0 dt1 ] , [ rE ;
vDep1 ] , opts , mus ) ;
        [ t , stateT2 ] = ode45( @TwoBodyMotion , [ 0 dt2 ] , [ rE ;
vDep2 ] , opts , mus ) ;
        [ t , stateT3 ] = ode45( @TwoBodyMotion , [ 0 dt3 ] , [ rE ;
vDep3 ] , opts , mus ) ;
        [ t , stateT1r ] = ode45( @TwoBodyMotion , [ 0 dt1 ] , [ rE ;
vDep1r ] , opts , mus ) ;
        [ t , stateT2r ] = ode45( @TwoBodyMotion , [ 0 dt2 ] , [ rE ;
vDep2r ] , opts , mus ) ;
        [ t , stateT3r ] = ode45( @TwoBodyMotion , [ 0 dt3 ] , [ rE ;
vDep3r ] , opts , mus ) ;

        figure
        hold on
        plot3( stateE(:,1) , stateE(:,2) , stateE(:,3) , 'k' )
        plot3( stateV1(:,1) , stateV1(:,2) , stateV1(:,3) )
        plot3( stateT1(:,1) , stateT1(:,2) , stateT1(:,3) , 'b' )
        plot3( stateT1r(:,1) , stateT1r(:,2) , stateT1r(:,3) , 'r' )
        title( 'First Arrival' )
        xlabel( 'X (km)' )
        ylabel( 'Y (km)' )
        zlabel( 'Z (km)' )
        legend( 'Earth' , 'Venus' , 'Transfer Prograde' , 'Transfer
Retrograde' )
        hold off

        figure
        hold on
        plot3( stateE(:,1) , stateE(:,2) , stateE(:,3) , 'k' )
        plot3( stateV2(:,1) , stateV2(:,2) , stateV2(:,3) )
        plot3( stateT2(:,1) , stateT2(:,2) , stateT2(:,3) , 'b' )
        plot3( stateT2r(:,1) , stateT2r(:,2) , stateT2r(:,3) , 'r' )
        title( 'Second Arrival' )
        xlabel( 'X (km)' )
        ylabel( 'Y (km)' )
        zlabel( 'Z (km)' )
        legend( 'Earth' , 'Venus' , 'Transfer Prograde' , 'Transfer
Retrograde' )
        hold off

        figure
        hold on
        plot3( stateE(:,1) , stateE(:,2) , stateE(:,3) , 'k' )
        plot3( stateV3(:,1) , stateV3(:,2) , stateV3(:,3) )
        plot3( stateT3(:,1) , stateT3(:,2) , stateT3(:,3) , 'b' )
        plot3( stateT3r(:,1) , stateT3r(:,2) , stateT3r(:,3) , 'r' )
        title( 'Third Arrival' )
```

```
        xlabel( 'X (km)' )
        ylabel( 'Y (km)' )
        zlabel( 'Z (km)' )
        legend( 'Earth' , 'Venus' , 'Transfer Prograde' , 'Transfer
Retrograde' )
        hold off

    end

    function Problem3()
        disp( ' ' )
        disp( '3' )
        aEarth = 149.598e6 ; % semi-major axis of Earth
        vEarth = sqrt( mus / aEarth ) ; % velocity of Earth in a
circular orbit
        Tellip = 365.25*(24*60*60)*.75 ; % Period of ellipse
        aEllip = ((Tellip*sqrt(mus))/(2*pi))^(2/3) ;
        rp = 2*aEllip - aEarth ;
        ecc = ( aEarth - rp )/( aEarth + rp ) ;
        h = sqrt( aEarth*mus*(1-ecc) ) ;
        vaellip = mus/h*(1-ecc) ;

        vinf = -vEarth - vaellip ;
        rph = 1000+re ;
        ecch = 1 + (rph*vinf^2)/mue ;
        delta = 2*asin(1/ecch) ;
        initialHelioV = -[ vaellip , 0 ] ;
        vinf2 = vinf*[ cos(-delta) , sin(-delta) ] ;
        finalHelioV = [ vEarth , 0 ] +  vinf2 ;

        dHelioV = norm( initialHelioV - finalHelioV ) ;
        initialHelioS = norm( initialHelioV ) ;
        finalHelioS = norm( finalHelioV ) ;

        disp([ 'The heliocentric speed after fly by is ' ,
num2str( finalHelioS ) , ' km/s' ])
        disp([ 'This is a increase of ' , num2str( dHelioV ) , ' km/s
from initial speed of' ])
        disp([ num2str( initialHelioS ) , ' km/s' ])
        disp( 'This seems correct because it there is a large v inf so
there' )
        disp( 'is not much time for Earth''s gravity to rotate the
vector and' )
        disp( 'so the final delta v is small relative to the
velocities' )
        disp( 'I had the s/c be retrograde because I think that it has
to ' )
        disp( 'retrograde in order for a s/c going slower than a
planet to ' )
        disp( 'peform a trailing edge fly by' )
    end

    function Problem4()
        disp( ' ' )
```

```
        disp( '4' )
        g0 = 9.81 ;
        tBurnd = 4 ; % days
        tCoastd = 1 ; % days
        tBurn = tBurnd * d2s ; % seconds
        tCoast = tCoastd * d2s ; % seconds
        ISP = 3100 ; % seconds
        thrust = 90e-3 ; % newtons
        mass = 747 ; % kg
        r0 = [ 42000 ; 0 ; 0 ] ;
        v0 = [ 0 ; 3.0807 ; 0 ] ;
        [ tb1 , stateb1 ] = ode45( @NonImpulsive , [ 0 , tBurn ] ,
[ r0' , v0' , mass ] , opts ,  mue , thrust , ISP , g0 ) ;
        [ tc1 , statec1 ] = ode45( @TwoBodyMotion , [ 0 , tCoast ] ,
stateb1( length(tb1) , 1:6 ) , opts , mue ) ;
        [ tb2 , stateb2 ] = ode45( @NonImpulsive , [ 0 , tBurn ] ,
[ statec1( length(tc1) , 1:6 ) , stateb1( length(tb1) , 7 ) ] ,
opts ,  mue , thrust , ISP , g0 ) ;
        [ tc2 , statec2 ] = ode45( @TwoBodyMotion , [ 0 , tCoast ] ,
stateb2( length(tb2) , 1:6 ) , opts , mue ) ;

        figure
        hold on
        plot3( stateb1(:,1) , stateb1(:,2) , stateb1(:,3) , '-r' )
        plot3( stateb2(:,1) , stateb2(:,2) , stateb2(:,3) , '-g' )
        plot3( statec1(:,1) , statec1(:,2) , statec1(:,3) , '-k' )
        plot3( statec2(:,1) , statec2(:,2) , statec2(:,3) , '-b' )
        title( 'Orbits' )
        xlabel( 'X (km)' )
        ylabel( 'Y (km)' )
        zlabel( 'Z (km)' )
        legend( 'First Burn' , 'Second Burn' , 'First Coast' , 'Second
Coast' )
        hold off
        true_tb2 = ( tb1( length( tb1 ) ) + tb2 )/d2s ;
        true_tb1 = tb1/d2s ;
        figure
        hold on
        plot( true_tb1 , stateb1(:,7) )
        plot( true_tb2 , stateb2(:,7) )
        title( 'Mass over Time' )
        xlabel( 'Time with Engine on (s)' )
        ylabel( 'Mass (kg)' )
        legend( 'First Burn' , 'Second Burn' )
        hold off

        final_radius = norm( statec2(length(statec2),1:3) ) ;
        disp([ 'The radius after the second coast is  ' ,
num2str( final_radius ) , ' km' ])
        disp( 'This seems correct because the thrust is very low so
the ' )
        disp( 'radius shouldn''t increase dramatically' )

        final_mass = norm( stateb2(length(stateb2),7) ) ;
```

```
        disp([ 'The mass after the second burn is ' ,
num2str( final_mass ) , ' kg' ])
        disp( 'This seems correct since the Isp high and the mass does
not ' )
        disp( 'change very much' )
   end
```

Functions

```
   function [ Jd , Jo , UT , J2000 ] = Julian( time , date )
   %Calculates the Julian Date from a date and time
   %   Uses an input of date in form [dd,mm,yyyy] and time in UT
[hour,minute,second] to find Julian date. BCE years should be
   %   negative

   % Julian date without time
       Jo = 367*date(3) -
floor(( 7*( date(3)+floor(( date(2)+9 )/12 )) )/4)+floor((275*date(2))/9)
+ date(1) + 1721013.5 ;
   % Time
       hour = time(1) ; %hours past noon as fraction of a day
       minute = time(2)/(60) ; %minutes as fraction of a day
       second = time(3)/(60*60) ; %seconds as fraction of a day
       UT = hour + minute + second ; % add time together
       Jd = Jo + UT/24 ; % Full Julian date

   % J2000 date
       J2000 = Jd - 2451545 ;
   end

   function [planet_coes] = planetary_elements(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)
   % [ a , ecc , inc , raan , w_hat , L ]

   % 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 ] = Pcoes2state( p_coes , mu )
            d2r = pi/180 ;
            h = sqrt( p_coes(1)*mu*(1-p_coes(2)^2) ) ;
            omega = p_coes(6) - p_coes(4) ;
            M = p_coes(6)*d2r - p_coes(5)*d2r ;
```

```
            T = (2*pi)/sqrt(mu)*p_coes(1) ;
            t = M*T/(2*pi) ;
            [ theta ] = time2theta( t , T , p_coes(2) ) ;
            [ r , v ] = coes2state(  h , p_coes(2) , theta ,
p_coes(4) , omega , p_coes(3) , mu ) ;

        function [ theta ] = time2theta( t , T , ecc )
        % Find true anomaly at a time

        n = 2*pi/T ; % mean motion
        Me = n*t ;

        % Guess of E
        if Me < pi
            E0 = Me + ecc/2 ;
        else
            E0 = Me - ecc/2 ;
        end

        % Use Newtons to find E
            tol = 10^-8 ; % Tolerance
            lim = 1000 ; % Maximum iteration
            f = @(E) E - ecc*sin(E) - Me ; % Function handle for E
            fprime = @(E) 1 - ecc*cos(E) ; % function handle for
derivative of E
            [ E ] = newton( E0 , f , fprime , tol , lim ) ; % Apply
Newtons

        theta = 2*atan(tan(E/2)*sqrt((1+ecc)/(1-ecc))) ; % find true
anomaly
        % correction to make it positive
            if theta < 0
                theta = theta + 2*pi ;
            end
        theta = theta*(180/pi) ;
        end
        function [ r , v ] = coes2state(  h , ecc , theta , RAAN ,
omega , inc , mu )
            r_peri = (h^2/mu) * ( 1/( 1 + ecc*cosd(theta) ) ) *
[ cosd( theta ) ; sind( theta ) ; 0 ] ;
            v_peri = (mu/h) * [ -sind( theta ) ; ecc+cosd(theta) ;
0 ] ;

            d2r = pi/180 ;
            RAAN = d2r*RAAN ;
            omega = d2r*omega ;
            inc = d2r*inc ;
            Q(1,1) = -sin(RAAN)*cos(inc)*sin(omega) +
cos(RAAN)*cos(omega) ;
            Q(1,2) = -sin(RAAN)*cos(inc)*cos(omega) -
cos(RAAN)*sin(omega) ;
            Q(1,3) = sin(RAAN)*sin(inc) ;
            Q(2,1) = cos(RAAN)*cos(inc)*sin(omega) +
sin(RAAN)*cos(omega) ;
```

```
            Q(2,2) = cos(RAAN)*cos(inc)*cos(omega) -
sin(RAAN)*sin(omega) ;
            Q(2,3) = -cos(RAAN)*sin(inc) ;
            Q(3,1) = sin(inc)*sin(omega) ;
            Q(3,2) = sin(inc)*cos(omega) ;
            Q(3,3) = cos(inc) ;

            r = Q*r_peri ;
            v = Q*v_peri ;
        end
    end

    function [ v1 , v2 ] = Lamberts2( r1 , r2 , dt , mu , tol , pro )
    % pro is 1 or 0 for prograde or retrograde respectively

        r1mag = norm( r1 ) ;
        r2mag = norm( r2 ) ;
        rcross = cross( r1 , r2 ) ;

        % Find delta theta
            if pro == 1
                if rcross(3) >= 0
                    dtheta = acos( dot(r1,r2)/(r1mag*r2mag) ) ;
                else
                    dtheta = 2*pi - acos( dot(r1,r2)/(r1mag*r2mag) ) ;
                end
            else
                if rcross(3) < 0
                    dtheta = acos( dot(r1,r2)/(r1mag*r2mag) ) ;
                else
                    dtheta = 2*pi - acos( dot(r1,r2)/(r1mag*r2mag) ) ;
                end
            end

        A = sin( dtheta )*sqrt( r1mag*r2mag/(1-cos(dtheta)) ) ;
            z = 0 ;
            C = 1/2 ;
            S = 1/6 ;
            zup = 4*pi^2 ;
            zlow = -4*pi^2 ;
            y = r1mag + r2mag + (A*(z*S-1))/sqrt(C) ;
            chi = sqrt(y/C) ;
            dtloop = (chi^3*S)/sqrt(mu) + (A*sqrt(y))/sqrt(mu) ;
            while abs( dtloop - dt ) > tol
                    if dtloop <= dt
                        zlow = z ;
                    else
                        zup = z ;
                    end
                 z = ( zup + zlow ) / 2 ;
                [ S , C ] = Stumpf( z ) ;
                y = r1mag + r2mag + (A*(z*S-1))/sqrt(C) ;
                chi = sqrt(y/C) ;
                dtloop = (chi^3*S)/sqrt(mu) + (A*sqrt(y))/sqrt(mu) ;
```

```
            end
            f = 1 - y/r1mag ;
            g = A*sqrt(y/mu) ;
            gdot = 1 - y/r2mag ;

            v1 = ( 1/g )*( r2 - f*r1 ) ;
            v2 = ( 1/g )*( gdot*r2 - r1 ) ;
    end

    function dstate_dt = NonImpulsive( t , state , mu , thrust , isp ,
g0 )
        % Finds change in state with respect to time. Input time, t,
in seconds and
        % state as position vector followed by velocity vector as well
as mu

        rad = norm( [ state(1) state(2) state(3) ] ) ; % radius
        vel = norm( [ state(4) state(5) state(6) ] ) ; % velocity
        m = state(7) ; % mass

        dx = state(4) ; % velocity in x
        dy = state(5) ; % velocity in y
        dz = state(6) ; % velocity in z
        ddx = -(mu*state(1)/rad^3)+(thrust*state(4))/(1e3*m*vel) ; %
acceleration in x
        ddy = -(mu*state(2)/rad^3)+(thrust*state(5))/(1e3*m*vel) ; %
acceleration in y
        ddz = -(mu*state(3)/rad^3)+(thrust*state(6))/(1e3*m*vel) ; %
acceleration in z
        mdot = -thrust/(g0*isp) ;

        dstate_dt = [ dx ; dy ; dz ; ddx ; ddy ; ddz ; mdot ] ;

    end

    function dstate_dt = TwoBodyMotion( t , state , mu )
    % Finds change in state with respect to time. Input time, t, in
seconds and
    % state as position vector followed by velocity vector as well as
mu

    rad = norm( [ state(1) state(2) state(3) ] ) ; %radius

    dx = state(4) ; % velocity in x
    dy = state(5) ; % velocity in y
    dz = state(6) ; % velocity in z
    ddx = -mu*state(1)/rad^3 ; % acceleration in x
    ddy = -mu*state(2)/rad^3 ; % acceleration in y
    ddz = -mu*state(3)/rad^3 ; % acceleration in z

    dstate_dt = [ dx ; dy ; dz ; ddx ; ddy ; ddz ] ;

    end
```

```
end
```

Published with MATLAB® R2018b