Astrodynamics

University of Florida
Mechanical and Aerospace Engineering

HW 5 Solution

5-1

(a) For a Hohmann transfer, determine expressions for the magnitude of the two impulses, Δv_1 and Δv_2. Nondimensionalize the two impulses by determining the ratios $\Delta v_1/v_{c1}$ and $\Delta v_2/v_{c1}$ and as functions of the quantity R $= r_2/r_1$.

$$\Delta v_1 = \| \Delta^I V_1 \| = v_1^+ - v_1^-$$
$$\Delta v_2 = \| \Delta^I V_2 \| = v_2^+ - v_2^-$$

$$v_1^- = \sqrt{\frac{\mu}{r_1}} \qquad v_2^- = \sqrt{\frac{2\mu}{r_2} - \frac{\mu}{a}}$$
$$v_1^+ = \sqrt{\frac{2\mu}{r_1} - \frac{\mu}{a}} \qquad v_2^+ = \sqrt{\frac{\mu}{r_2}}$$

$$\Delta v_1 = \sqrt{\frac{2\mu}{r_1} - \frac{\mu}{a}} - \sqrt{\frac{\mu}{r_1}} \qquad \Delta v_2 = \sqrt{\frac{\mu}{r_2}} - \sqrt{\frac{2\mu}{r_2} - \frac{\mu}{a}}$$

Nondimensionalizing the above equations using the ratios $\Delta v_1/v_{c1}$, $\Delta v_2/v_{c1}$ and R $= r_2/r_1$:

$$\boxed{\frac{\Delta v_1}{v_{c1}} = \sqrt{\frac{2R}{1+R}} - 1} \qquad \boxed{\frac{\Delta v_2}{v_{c1}} = \sqrt{\frac{1}{R}}(1 - \sqrt{\frac{2}{1+R}})}$$

(b) For a bi-elliptic transfer, determine expressions for the magnitude of the three impulses, Δv1, Δv2, and Δv3. Nondimensionalize the three impulses by determining the ratios Δv1/vc1, $\Delta v2/vc1$, and $\Delta v3/vc1$ and as functions of the quantities R $= r2/r1$ and S $= ri/r2$ (where ri is the apoapsis of the intermediate transfer orbit used in the bi-elliptic transfer).

$$\Delta v_i = \| \Delta^I V_i \| = v_i^+ - v_i^- \qquad \text{for } i = 1, 2, 3$$

$$v_1^- = \sqrt{\frac{\mu}{r_1}} \qquad v_2^- = \sqrt{\frac{2\mu}{r_1} - \frac{\mu}{a_1}} \qquad v_3^- = \sqrt{\frac{2\mu}{r_2} - \frac{\mu}{a_2}}$$
$$v_1^+ = \sqrt{\frac{2\mu}{r_1} - \frac{\mu}{a_1}} \qquad v_2^+ = \sqrt{\frac{2\mu}{r_1} - \frac{\mu}{a_2}} \qquad v_3^+ = \sqrt{\frac{\mu}{r_2}}$$

$$\Delta v_1 = \sqrt{\frac{2\mu}{r_1} - \frac{\mu}{a_1}} - \sqrt{\frac{\mu}{r_1}} \qquad \Delta v_2 = \sqrt{\frac{2\mu}{r_1} - \frac{\mu}{a_2}} - \sqrt{\frac{2\mu}{r_1} - \frac{\mu}{a_1}} \qquad \Delta v_3 = \sqrt{\frac{2\mu}{r_2} - \frac{\mu}{a_2}} - \sqrt{\frac{\mu}{r_2}}$$

Nondimensionalizing the above equations using the ratios $\Delta v_1/v_{c1}$, $\Delta v_2/v_{c1}$, $\Delta v_3/v_{c1}$, R $= r_2/r_1$, and S $= r_i/r_2$:

$$\boxed{\frac{\Delta v_1}{v_{c1}} = \sqrt{\frac{2RS}{1+RS}} - 1} \qquad \boxed{\frac{\Delta v_2}{v_{c1}} = \sqrt{\frac{1}{RS}}(\sqrt{\frac{2}{1+S}} - \frac{2}{1+RS})} \qquad \boxed{\frac{\Delta v_3}{v_{c1}} = \sqrt{\frac{2S}{R+RS}} - \sqrt{\frac{1}{R}}}$$

(c) For a bi-parabolic transfer, determine expressions for the magnitude of the two impulses, Δv_1 and Δv_2. Nondimensionalize the two impulses by determining the ratios $\Delta v_1/v_(c1)$ and $\Delta v_2/v_(c1)$ and as functions of the quantity $\Delta R = r_2/r_1$.

$$\Delta v_i = || \Delta^I V_i || = v_i^+ - v_i^- \qquad \text{for } i = 1, 2, 3$$

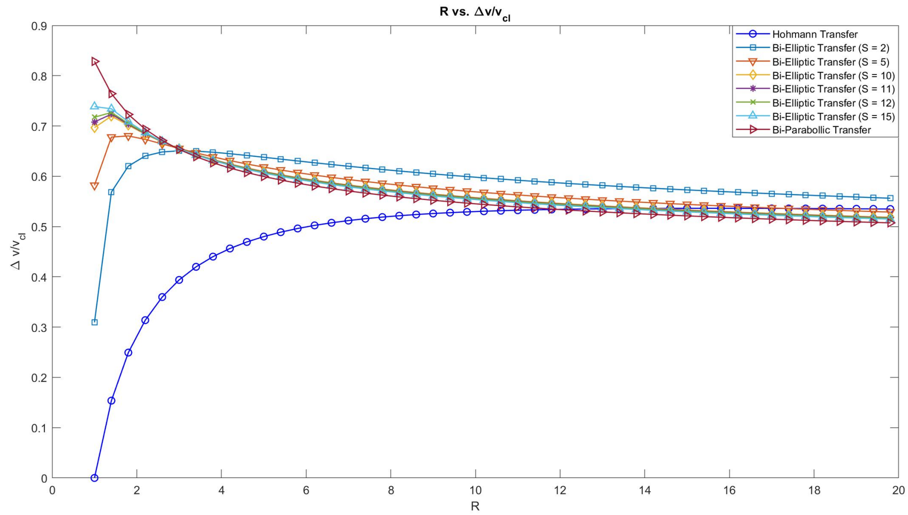
The bi-parabolic transfer is really a limiting case of the bi-elliptic transfer, and is obtained by letting $r_i \to \infty$, which implies S $\to \infty$. Because of that, $\frac{\Delta v_2}{v_{c1}}$ goes to 0. Then, the other two impulses look like this:

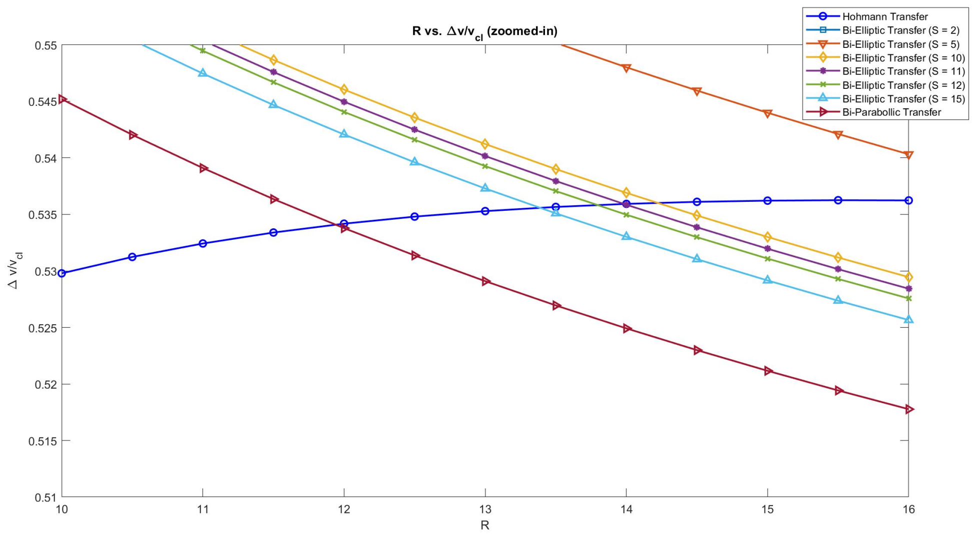
$$v_1^- = \sqrt{\frac{\mu}{r_1}} \qquad v_3^- = \sqrt{\frac{2\mu}{r_2}}$$
$$v_1^+ = \sqrt{\frac{2\mu}{r_1}} \qquad v_3^+ = \sqrt{\frac{\mu}{r_2}}$$
$$\Delta v_1 = \sqrt{\frac{2\mu}{r_1} - \sqrt{\frac{\mu}{r_1}}} \qquad \Delta v_3 = \sqrt{\frac{2\mu}{r_2}} - \sqrt{\frac{\mu}{r_2}}$$

Nondimensionalizing the above equations using the ratios $\Delta v_1/v_{c1}$, $\Delta v_2/v_{c1}$ and R $= r_2/r_1$:

$$\boxed{\frac{\Delta v_1}{v_{c1}} = \sqrt{2} - 1} \qquad \boxed{\frac{\Delta v_2}{v_{c1}} = \sqrt{\frac{1}{R}}(\sqrt{2} - 1)}$$

(d) Make the following two plots in MATLAB of the normalized impulse, $\frac{\Delta v_1}{v_{c1}}$, for each transfer as a function of R, where R is the "x"–axis and $\frac{\Delta v_1}{v_{c1}}$ is the "y"–axis. For use R ϵ [1, 20] and do not change the default settings for the "y"–axis in MATLAB. For the second plot, change the range for R to be such that Rϵ [10, 16] and change the range for $\frac{\Delta v_1}{v_{c1}}$ to be $\frac{\Delta v_1}{v_{c1}} \epsilon$ [0.51, 0.55]. When making both plots, use the values S = (2, 5, 10, 11, 12, 15) for the bi-elliptic transfer. For each plot place all of the lines on the same plot (that is, put the Hohmann transfer, all bi-elliptic transfers, and the bi-parabolic transfer on the same plot).





Code for question 1 (d):

```
%Astro HW#5 Problem 1 (d)
clc; clear;

delv = zeros(3,1);
%% Hohmann Transfer
delv_h = @(R) sqrt((2.*R)./(1+R))-1 + sqrt(1./R).*(1 - sqrt(2./(1+R)));
%% Bi-elliptic Transfer
delv_e = @(R,S) sqrt((2.*R.*S)./(1+R.*S))-1 + sqrt(1./(R.*S)).*(sqrt(2/(1+S)...
    - 2./(1+R.*S))) + sqrt(2*S./(R+R.*S)) - sqrt(1./R);
%% Bi-parabolic Transfer
delv_p = @(R) sqrt(2)-1 + sqrt(1./R).*(sqrt(2)-1);

%% Plotting
S = [2 5 10 11 12 15];
markers = ['s','v','d','*','x','^','o'];
colors = ['#0072BD' '#D95319' '#EDB120' '#7E2F8E' '#77AC30' '#4DBEEE' '#A2142F'];

figure(1)
R = 1:.4:20;
plot(R,delv_h(R),'bo-','LineWidth',1)
hold on
for ii = 1:length(S)
plot(R,delv_e(R,S(ii)),'Marker',markers(ii),'LineWidth',1)
end
plot(R,delv_p(R),'>-','LineWidth',1)
legend('Hohmann Transfer','Bi-Elliptic Transfer (S = 2)','Bi-Elliptic Transfer (S = 5)',...
    'Bi-Elliptic Transfer (S = 10)', 'Bi-Elliptic Transfer (S = 11)', ...
    'Bi-Elliptic Transfer (S = 12)', 'Bi-Elliptic Transfer (S = 15)',...
    'Bi-Parabollic Transfer')
xlabel('R')
ylabel('\Delta v/v_{cl}')
title('R vs. \Deltav/v_{cl}')
hold off

figure(2)
R = 10:.5:16;
plot(R,delv_h(R),'bo-','LineWidth',1.5)
hold on
for ii = 1:length(S)
plot(R,delv_e(R,S(ii)),'Marker',markers(ii),'LineWidth',1.5)
end
plot(R,delv_p(R),'>-','LineWidth',1.5)
legend('Hohmann Transfer','Bi-Elliptic Transfer (S = 2)','Bi-Elliptic Transfer (S = 5)',...
    'Bi-Elliptic Transfer (S = 10)', 'Bi-Elliptic Transfer (S = 11)', ...
```

```
    'Bi-Elliptic Transfer (S = 12)', 'Bi-Elliptic Transfer (S = 15)',...
    'Bi-Parabollic Transfer')
xlabel('R')
ylabel('\Delta v/v_{cl}')
title('R vs. \Deltav/v_{cl} (zoomed-in)')
ylim([.51,.55])
hold off
```

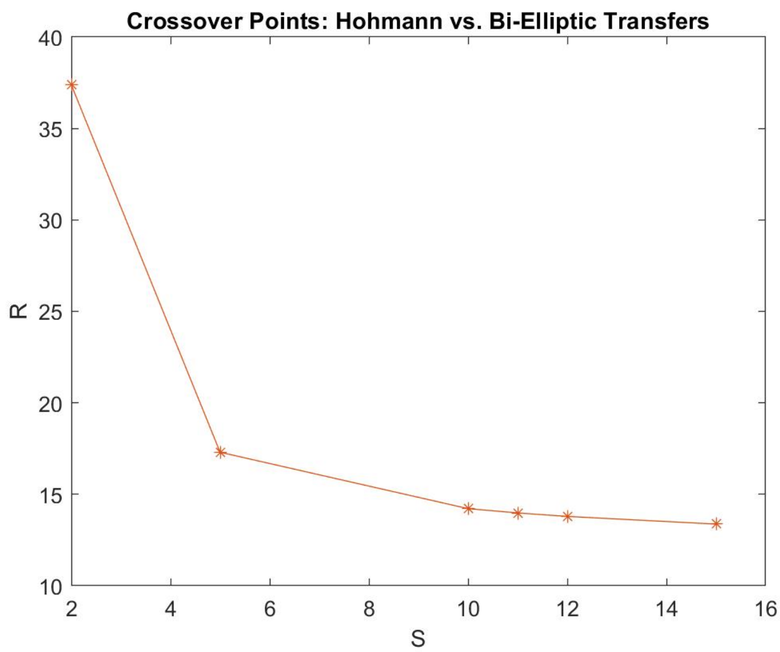
5-2 Suppose now it is desired to determine the values of the ratio R = r_2/r_1 that determines crossover points where the Hohmann transfer becomes less economical that either a bi-elliptic transfer or the bi-parabolic transfer. Using the results of Question 1, solve the following root-finding problems using either the MATLAB root-finder fsolve or your own root-finder:

(a) the value of R where the total impulse of the Hohmann transfer is the same as the total impulse of the bi-parabolic transfer.

The desired value of R in this case is $R = 11.9384$

(b) the values of R where the total impulse of the Hohmann transfer is the same as the total impulse of the bi-elliptic transfers for S = r_i/r_2 = (2, 5, 10, 11, 12, 15) (where r_i is the apoapsis of the intermediate transfer orbit used in the bi-elliptic transfer as given in Question 1)

The desired values of R in this case are $R = 37.382, 17.299, 14.2172, 13.9826, 13.7916, 13.3726$

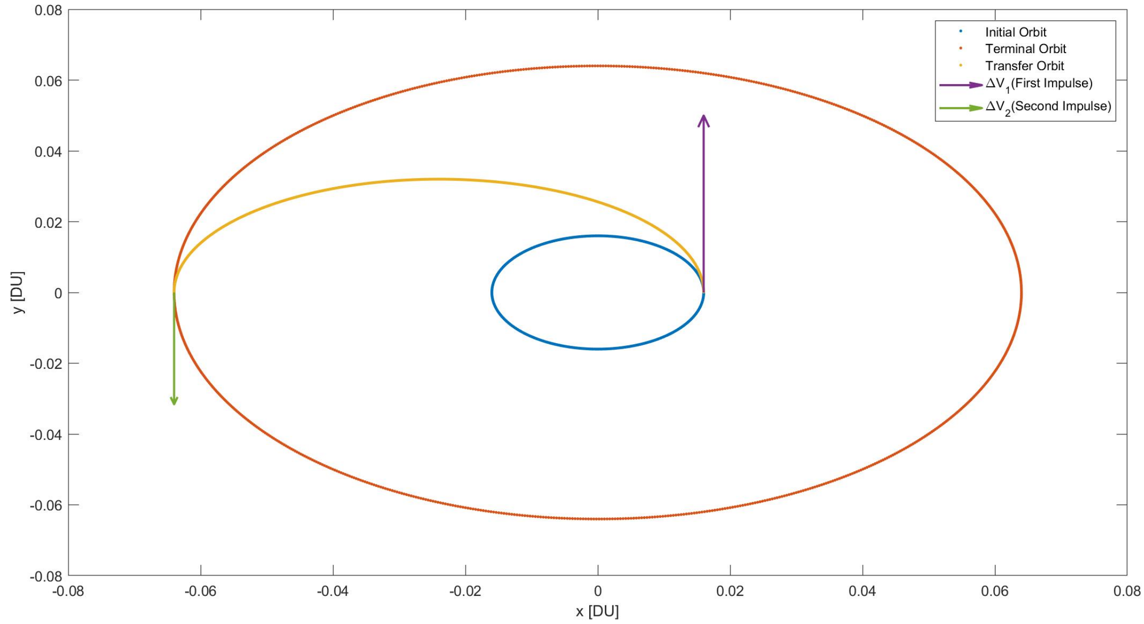


Code for question 2:

```
%Astro HW#5 Problem 2
clc; clear;

%% (a)
F = @(R) (sqrt(2)-1 + sqrt(1./R).*(sqrt(2)-1)) -( sqrt((2.*R)./(1+R))-1 + sqrt(1./R).*(1 - sqrt(2./(1+R))));
x0 = 1;
[R_a,fval_a] = fsolve(F,x0);

%% (b)

S = [2 5 10 11 12 15];
x0 = 12;
R_b = zeros(length(S),1);
fval_b = zeros(length(S),1);
for ii = 1:length(S)
    G = @(R) sqrt((2.*R.*S(ii)./(1+R.*S(ii))))-1 + sqrt(1./(R.*S(ii))).*(sqrt(2/(1+S(ii))...
    - 2./(1+R.*S(ii)))) + sqrt(2*S(ii)./(R+R.*S(ii))) - sqrt(1./R) - (sqrt((2.*R)./...
    (1+R))-1 + sqrt(1./R).*(1 - sqrt(2./(1+R))));
[R_b(ii),fval_b(ii)] = fsolve(G,x0);
end

%% (c)
figure(1)
plot(S,R_b,'Marker','*','Color',[0.8500 0.3250 0.0980])
xlabel('S')
ylabel('R')
title('Crossover Points: Hohmann vs. Bi-Elliptic Transfers')
```

5-3 A spacecraft is in a circular orbit that has a speed of unity in canonical units (that is, $\mu = 1$). From this starting orbit the goal is to rendezvous with a spacecraft that in another co-planat circular orbit with a speed of 0.5 canonical units. Determined which of the Hohmann, bi-elliptic, or bi-parabolic transfer accomplishes the orbit transfer with the lowest impulse. Using MATLAB, plot the initial orbit, the terminal orbit, and the transfer orbit on the same two-dimensional plot. Include the impulses required to accomplish the orbit transfer on your plot using the MATLAB command quiver

Best transfer would be a Hohmann Transfer



Code for question 3:

```
%Astro HW#5 Problem 3
clc; clear;

%% GIVEN
mu              = 1;
speed_orbit1    = 7.905366149846; %[km/s]
r1              = 1/(speed_orbit1)^2;
speed_orbit2    = speed_orbit1/2; %[km/s]
r2              = 1/(speed_orbit2)^2;

%% Orbit 1
a1              = r1;
tau1            = 2*pi*sqrt(a1^3/mu);
tspan1          = [0 tau1];
e               = 0; %0 for circular orbits
```

```
Omega           = 0; %Make the same for proper transfer
inc             = 0; %Coplanar, inc doesn't matter
omega           = 0; %UNDEFINED FOR CIRCULAR ORBIT
nu              = 0; %UNDEFINED FOR CIRCULAR ORBIT
oe1             = [a1,e,Omega,inc,omega,nu];
[r1_in,v1_in]   = oe2rv_Gusman_Lucas(oe1,mu);
% Propagating the Orbit
t0                          = 0;
timev                       = linspace(t0,tau1,1000);
rPCIv                       = zeros(length(timev),3);
vPCIv                       = zeros(length(timev),3);
%Entering the initial conditions into final vector
rPCI1(1,:)                   = r1_in;
vPCI1(1,:)                   = v1_in;
for ii = 2:length(timev) %for loop to iterate over time span
    [rPCIf,vPCIf,E0,nu0,E,nu] = propagateKepler_Gusman_Lucas(t0,timev(ii),
        r1_in,v1_in,mu);
    rPCI1(ii,:)              = rPCIf';
    vPCI1(ii,:)              = vPCIf';
end

%% Orbit 2
a2              = r2;
tau2            = 2*pi*sqrt(a2^3/mu);
tspan2          = [0 tau2];
oe2             = [a2,e,Omega,inc,omega,nu];
[r2_in,v2_in]   = oe2rv_Gusman_Lucas(oe2,mu);
% Propagating the Orbit
timev2                      = linspace(t0,tau2,1000);
rPCIv2                       = zeros(length(timev2),3);
vPCIv2                       = zeros(length(timev2),3);
%Entering the initial conditions into final vector
rPCI2(1,:)                   = r2_in;
vPCI2(1,:)                   = v2_in;
for ii = 2:length(timev2) %for loop to iterate over time span
    [rPCIf,vPCIf,E0,nu0,E,nu] = propagateKepler_Gusman_Lucas(t0,timev2(ii),
        r2_in,v2_in,mu);
    rPCI2(ii,:)              = rPCIf';
    vPCI2(ii,:)              = vPCIf';
end


%% Transfer Orbit
R = a2/a1;
choice = 0;
if R < 12
    fprintf('Most economical transfer: Hohmann Transfer')
    choice = 1;
end
```

```
if choice == 1
    at = (a1+a2)/2;
    taut = 2*pi*sqrt(at^3/mu);
    et  = (a2-a1)/(a1+a2);
    oet = [at,et,Omega,inc,omega,nu];
    [rt_in,vt_in] = oe2rv_Gusman_Lucas(oet,mu);
    speed_test = norm(vt_in,2);
    speed1_plus  = sqrt(2/a1 - 1/at);
    speed1v_plus = [0;speed1_plus;0];
    delv1        = speed1_plus - speed_orbit1;
    speed2_minus = sqrt(2/a2 - 1/at);
    speed2_plus  = speed_orbit2;
    speed2v_plus = [0;-speed2_plus;0];
    delv2        = speed2_plus - speed2_minus;
    % Propagating the Orbit
timevt                      = linspace(t0,taut/2,2000);
rPCIvt                      = zeros(length(timevt),3);
vPCIvt                      = zeros(length(timevt),3);
%Entering the initial conditions into final vector
rPCIt(1,:)                  = rt_in; %rt_in;
vPCIt(1,:)                  = vt_in; %vt_in;
for ii = 2:length(timevt) %for loop to iterate over time span
   [rPCIf,vPCIf,E0,nu0,E,nu] = propagateKepler_Gusman_Lucas(t0,timevt(ii),r1_in,speed1v_plus,mu);
   rPCIt(ii,:)              = rPCIf';
   vPCIt(ii,:)              = vPCIf';
end
end

%% Plotting
figure(1)
plot(rPCI1(:,1),rPCI1(:,2),'.',rPCI2(:,1),rPCI2(:,2),'.',rPCIt(:,1),rPCIt(:,2),'.')
hold on
quiver(r1_in(1),r1_in(2),speed1v_plus(1),speed1v_plus(2),.005,'LineWidth',1.5)
quiver(rPCIt(ii,1),rPCIt(ii,2),speed2v_plus(1),speed2v_plus(2),.008,'LineWidth',1.5)
legend('Initial Orbit','Terminal Orbit','Transfer Orbit',...
    '\DeltaV_1(First Impulse)','\DeltaV_2(Second Impulse)')
xlabel('x [DU]')
ylabel('y [DU]')
hold off
```

5-4

(a) Find the magnitude of each impulse that contributes to the total Δv (in $km * s^{-1}$) required to complete the transfer, where the inclination change is performed at the apoapsis of the transfer orbit (that is, the second impulse both circularizes the final orbit and accomplishes the inclination change).

$$\Delta V_1 = 2.4257 km/s \qquad \Delta V_2 = 2.5795 km/s$$

(b) Find the total $\mathbf{\Delta V}$ required to complete the transfer.

$$\Delta \text{V} = \Delta V_1 + \Delta V_2 = 5.0052 \text{ km/s}$$

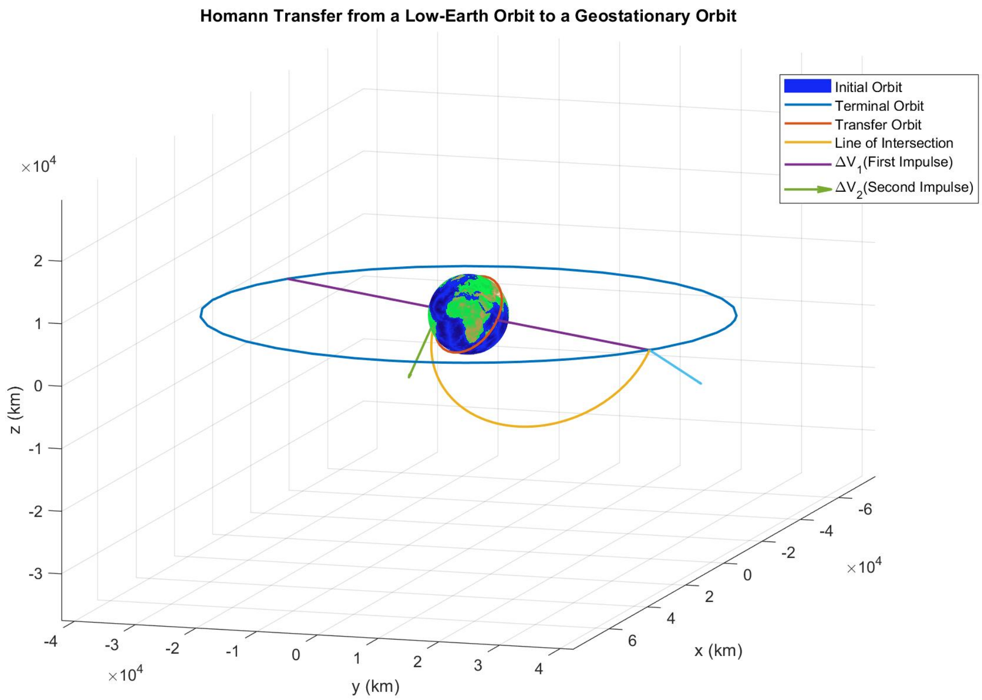
(c) Find the time (in hours) required to complete the transfer.

Time required for transfer: 5.275 hrs.

(d) Assuming that the rocket engine has a specific impulse of 320 s, determine the ratio of the initial and terminal masses due to the magnitude of each impulse obtained in part (a).

Mass ratio: 1.0016

(e) Using MATLAB, plot on the same three-dimensional plot the initial orbit, the terminal orbit, the transfer orbit, and the line of intersection between the initial and terminal orbits. Include the impulses required to accomplish the orbit transfer on your plot using the MATLAB command quiver3.



Code for question 4:

```
%Astro HW#5 Problem 4
% Non-planar Hohmann Transfer with crank impulse at apoapsis of
% transfer orbit (impulse 2)

clc; clear;

%% Constants
mu              = 398600;
earthr          = 6378.145;

%% Orbit 1
a1              = 300 + earthr;
tau1            = 2*pi*sqrt(a1^3/mu);
tspan1          = [0,tau1];
inc1            = 57; %deg
inc1            = inc1*pi/180; %rad
Omega1          = 60; %deg
Omega1          = Omega1*pi/180;%rad
e               = 0; %0 FOR CIRCULAR ORBIT
omega           = 0; %UNDEFINED FOR CIRCULAR ORBIT
nu              = 0; %UNDEFINED FOR CIRCULAR ORBIT
oe1             = [a1,e,Omega1,inc1,omega,nu];
[r10,v10]       = oe2rv_Gusman_Lucas(oe1,mu);
hv1             = cross(r10,v10);
uhv1            = hv1/norm(hv1,2);
[tout1,pout1]   = orbit_path_main (r10,v10,mu,tspan1);

%% Orbit 2 (Geostationary Orbit)
tau2            = 23.934; %hrs sidereal
tau2            = tau2*3600;
tspan2          = [0 tau2];
a2              = ((tau2/(2*pi))^2 * mu)^(1/3);
Omega2          = 0; %UNDEFINED FOR EQUATORIAL ORBIT
inc2            = 0; %0 FOR EQUATORIAL ORBIT
oe2             = [a2,e,Omega2,inc2,omega,nu];
[r20,v20]       = oe2rv_Gusman_Lucas(oe2,mu);
hv2             = cross(r20,v20);
uhv2            = hv2/norm(hv2,2);
[tout2,pout2]   = orbit_path_main (r20,v20,mu,tspan2);

%% Transfer Orbit
lineint         = cross(hv1,hv2)/norm(cross(hv1,hv2),2);
at              = (a1+a2)/2;
rt0             = a1;
rvt0            = rt0*lineint;
taut            = 2*pi*sqrt(at^3/mu);
tspant          = [0,taut/2];
ut0             = cross(uhv1,rvt0)/norm(cross(uhv1,rvt0),2);
```

```
et            = abs(a1-a2)/(a1+a2);
speedt0       = sqrt(mu/rt0);
vt0_minus     = speedt0*ut0;
vt0_plus      = sqrt(2*mu/a1 - mu/at)*ut0;

rtf           = a2;
rvtf          = rtf*lineint;
utf           = cross(uhv2,rvtf)/norm(cross(uhv2,rvtf),2);
speedtf       = sqrt(2*mu/rtf - mu/at);
vtf_minus     = speedtf*ut0;
vtf_plus      = sqrt(mu/rtf)*utf;
[toutt,poutt] = orbit_path_main (rvt0,vt0_plus,mu,tspant);

%% (a)
delv1         = sqrt(2*mu/a1 - mu/at) - speedt0
delvv1        = delv1*ut0;
delvv2        = vtf_plus - vtf_minus;
delv2         = norm(delvv2,2)

%% (b)
delv_total    = delv1+delv2

%% (c)
transfert     = taut/(2*3600) %time to complete transfer [hrs]

%% (d)
g0            = 9.80665;
Isp           = 320;
mratio        = exp(delv_total/(g0*Isp))

%% (e)
% PLOTS

% Earth
earthSphere
hold on

% Final and Initial Orbits
plot3(pout2(:,1),pout2(:,2),pout2(:,3),pout1(:,1),pout1(:,2),pout1(:,3),'
    LineWidth',1.5)

% Transfer Orbit
plot3(poutt(:,1),poutt(:,2),poutt(:,3),'LineWidth',1.5)

% Line of Intersection
span          = [-rvtf' ;rvtf'];
plot3(span(:,1),span(:,2),span(:,3),'LineWidth',1.5)

% Impulse vectors
```

```
impulses        = [delvv1' ; -delvv2'];
positions       = [rvt0' ; -rvtf'];
quiver3(positions(1,1),positions(1,2),positions(1,3),impulses(1,1),...
    impulses(1,2),impulses(1,3),5000,'LineWidth',1.5)
quiver3(positions(2,1),positions(2,2),positions(2,3),impulses(2,1),...
    impulses(2,2),impulses(2,3),5000,'LineWidth',1.5)
% quiver3(r10(1),r10(2),r10(3),v10(1),v10(2),v10(3),5000)
% quiver3(r20(1),r20(2),r20(3),v20(1),v20(2),v20(3),5000)
xlabel('x (km)')
ylabel('y (km)')
zlabel('z (km)')
title('Homann Transfer from a Low-Earth Orbit to a Geostationary Orbit')
legend('Initial Orbit','Terminal Orbit', 'Transfer Orbit',...
    'Line of Intersection','\DeltaV_1(First Impulse)','\DeltaV_2(Second
       Impulse)')
hold off
```

5-5 A Global Positioning System (GPS) spacecraft is launched from the Eastern Test Range (ETR) at the Cape Canaveral Air Force Station and initially inserted into a circular low-Earth orbit (LEO) at an altitude of 350 km with an orbital inclination of 28 deg. From this initial orbit the goal is to transfer the spacecraft to a final circular GPS orbit of radius 26558 km with an orbital inclination of 55 deg using a two-impulse transfer such that the impulses are applied along the line of intersection between the two orbits. Determine the following information:

(a) The line of intersection in Earth-centered inertial (ECI) coordinates.

$$\textit{Line of Intersection (ECI)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \text{ km}$$

(b) The positions of the spacecraft r1 and r2 that define the locations where the two impulses $\Delta^I V_1$ and $\Delta^I V_2$ are applied.

$$[r_1] = \begin{bmatrix} 6728.1 \\ 0 \\ 0 \end{bmatrix} \text{ km }, [r_2] = \begin{bmatrix} 26558 \\ 0 \\ 0 \end{bmatrix} \text{ km}$$

(c) The total ΔV (in $km\ s^{-1}$) if the required inclination change is performed purely at the apoapsis of the transfer orbit (that is, the second impulse both circularizes the final orbit and accomplishes the inclination change).

$$\Delta V = \Delta V_1 + \Delta V_2 = \boxed{4.0437 \text{ km } s^{-1}}$$

(d) The time of flight (in hours) of the transfer ellipse.

Time of flight for transfer: 2.9678 hours

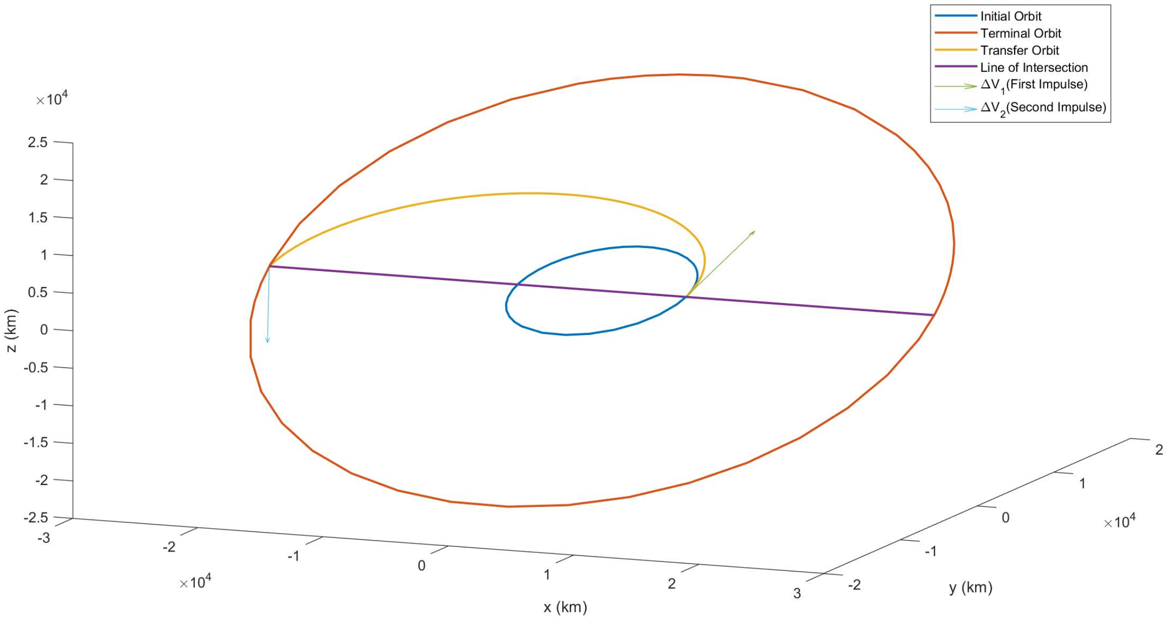
(e) The eccentricity of the transfer orbit.

Eccentricity of Transfer Orbit: 0.5957

(f) The angle θ that defines the rotation of the orbital plane due to the application of the second impulse.

Crank angle (θ): 27 degrees

(g) Using MATLAB, plot on the same three-dimensional plot the initial orbit, the terminal orbit, the transfer orbit, and the line of intersection between the initial and terminal orbits. Include the impulses required to accomplish the orbit transfer on your plot using the MATLAB command quiver3.



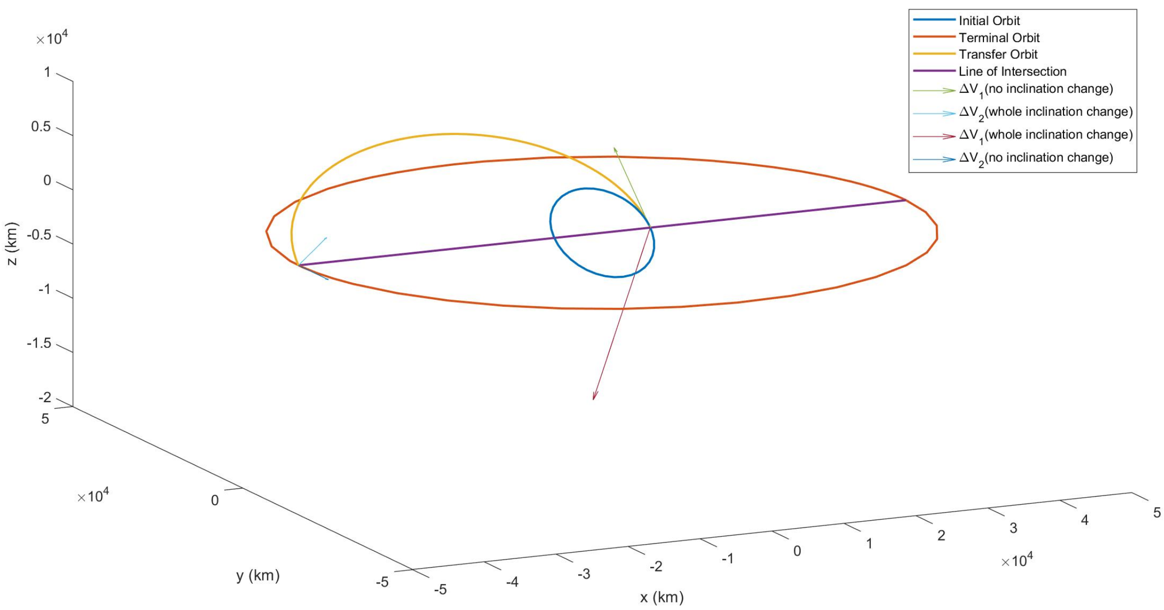
Code for question 5:

```
%Astro HW#5 Problem 5
clc; clear;

mu              = 398600;
Re              = 6378.145;

%% Orbit 1
alt1            = 350; %[km]
a1              = alt1 + Re;
tau1            = 2*pi*sqrt(a1^3/mu);
tspan1          = [0 tau1];
e               = 0;
inc_deg         = 28; %[deg]
inc1            = inc_deg*pi/180; %[rad]
omega           = 0; %UNDEFINED FOR CIRCULAR ORBIT
nu              = 0; %UNDEFINED FOR CIRCULAR ORBIT
Omega           = 0;
oe1             = [a1,e,Omega,inc1,omega,nu];
[rin0, vin0]    = oe2rv_Gusman_Lucas(oe1,mu);
hv1             = cross(rin0,vin0);
uhv1            = hv1/norm(hv1,2);
[tout1,pout1]   = orbit_path_main (rin0,vin0,mu,tspan1);

%% Orbit 2
a2              = 26558; %[km]
tau2            = 2*pi*sqrt(a2^3/mu);
tspan2          = [0 tau2];
inc_deg         = 55; %[deg]
inc2            = inc_deg*pi/180; %[rad]
oe2             = [a2,e,Omega,inc2,omega,nu];
[rf0, vf0]      = oe2rv_Gusman_Lucas(oe2,mu);
hv2             = cross(rf0,vf0);
uhv2            = hv2/norm(hv2,2);
[tout2,pout2]   = orbit_path_main (rf0,vf0,mu,tspan2);


%% Transfer Orbit
lineint         = cross(hv1,hv2)/norm(cross(hv1,hv2),2);
at              = (a1+a2)/2;
taut            = 2*pi*sqrt(at^3/mu);
tspant          = [0 taut/2];
et              = (a2-a1)/(a1+a2);
oet             = [at,et,Omega,inc1,omega,nu];
[rt0, vt0]      = oe2rv_Gusman_Lucas(oet,mu);
[toutt,poutt]   = orbit_path_main (rt0,vt0,mu,tspant);

%% Impulse Calculations
speed1_minus    = sqrt(mu/a1);
```

```
speed1_plus    = sqrt(2*mu/a1-mu/at);
u1v            = cross(hv1,rin0)/norm(cross(hv1,rin0),2);
delv1          = (speed1_plus - speed1_minus)*u1v;

speed2_minus   = sqrt(2*mu/a2-mu/at);
speed2_plus    = sqrt(mu/a2);
u2v            = cross(hv2,rf0)/norm(cross(hv2,rf0),2);
delv2          = speed2_plus*u2v - speed2_minus*u1v;

delv_total     = norm(delv1,2)+norm(delv2,2);

%% Crank Angle
theta          = acos((hv1.'*hv2)/(norm(hv1,2)*norm(hv2,2))); %rad
theta_deg      = theta*180/pi;


%% Plotting

plot3(pout1(:,1),pout1(:,2),pout1(:,3),pout2(:,1),pout2(:,2),pout2(:,3),...
    poutt(:,1), poutt(:,2),poutt(:,3),'LineWidth',1.5)
hold on
% Line of Intersection
line           = [rf0';-rf0'];
plot3(line(:,1),line(:,2),line(:,3),'LineWidth',1.5)
% Impulse Vectors
quiver3(rin0(1),rin0(2),rin0(3),delv1(1),delv1(2),delv1(3),5000)
quiver3(-rf0(1),-rf0(2),-rf0(3),-delv2(1),-delv2(2),-delv2(3),5000)
xlabel('x (km)')
ylabel('y (km)')
zlabel('z (km)')
legend('Initial Orbit','Terminal Orbit', 'Transfer Orbit',...
    'Line of Intersection','\DeltaV_1(First Impulse)','\DeltaV_2(Second Impulse)')
hold off

%% Print Statements
fprintf('---------------------------------------------------------------\n');
fprintf('-----------Hohmann Transfer with a Plane Change-----------\n');
fprintf('---------------------------------------------------------------\n');
fprintf('---------------------------------------------------------------\n');
fprintf('-----------------Data Supplied for This Run---------------\n');
fprintf('---------------------------------------------------------------\n');
fprintf('Initial Semi-Major Axis \t\t= %8.4f (km)\n',a1);
fprintf('Terminal Semi-Major Axis \t\t= %8.4f (km)\n',a2);
fprintf('Initial Inclination \t\t\t= %8.4f (deg)\n',inc1*180/pi);
fprintf('Terminal Inclination \t\t\t= %8.4f (deg)\n',inc2*180/pi);
fprintf('Initial Longitude of Ascending Node \t= %8.4f (deg)\n',Omega*180/pi);
fprintf('Terminal Longitude of Ascending Node \t= %8.4f (deg)\n',Omega*180/
```

```
    pi);
fprintf('──────────────────────────────────────────────────────────────\n');
fprintf('──────────────────────────────────────────────────────────────\n');
fprintf('────────────────────────Values of interest────────────────────\n');
fprintf('──────────────────────────────────────────────────────────────\n');
fprintf('Time of flight for transfer:  %8.4f (hrs)\n',taut/(3600*2));
fprintf('Eccentricity of transfer orbit:%8.4f \n',et);
fprintf('Crank angle: %8.4f (deg) \n',theta_deg);
fprintf('──────────────────────────────────────────────────────────────\n');
```

5-6 A spacecraft is launched from the Kennedy Space Center in Florida into an initial circular low-Earth orbit (LEO) with altitude 300 km and an inclination $i = 28.5$ deg. The goal is to transfer the spacecraft to a geostationary orbit (GEO), where it is noted that a geostationary orbit is an circular equatorial orbit with a period of 24 hours). Suppose that it is desired to transfer the spacecraft from the given LEO to GEO using a two-impulse transfer that consists of two energy change impulses along with up to two inclination change impulses. Suppose further that f and 1 - f denote, respectively, the fraction of the inclination change that is accomplished at the initial LEO and the apoapsis of the transfer orbit (that is, the inclination change is divided into an inclination change that is accomplished at the radius of the initial LEO while the remainder of the inclination change is accomplished at the apoapsis of the transfer orbit). Using the information provided, determine the following:



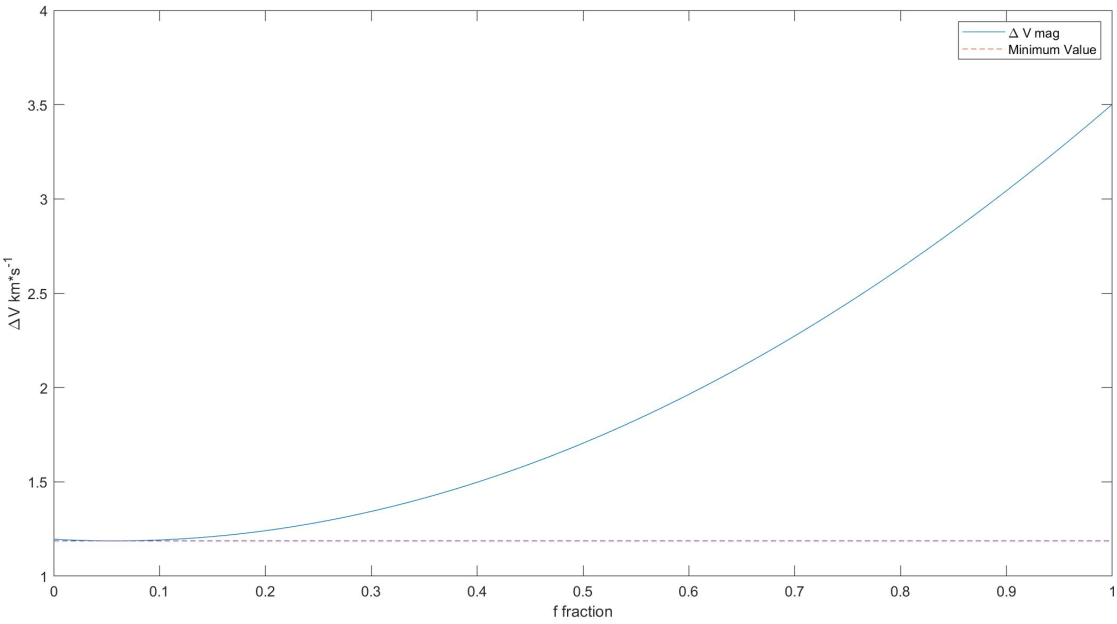
(a) The magnitude of the total impulse assuming that all of the inclination change is accomplished at the apoapsis of the elliptic transfer orbit.

In the case of accomplishing all of the inclination change at apoapsis: $\|\ \Delta V\ \|= \boxed{4.2563 \text{ km } s^{-1}}$

(b) The magnitude of the total impulse assuming that all of the inclination change is accomplished at the initial LEO.

In the case of accomplishing all of the inclination change at the initial LEO: $\|\ \Delta V\ \|= \boxed{6.4568 \text{ km } s^{-1}}$

(c) A two-dimensional plot in MATLAB that shows the total impulse normalized by the initial circular speed (that is $\Delta v/v_{c1}$) as a function of the fraction f of the total inclination change that is accomplished at the initial LEO.



(d) From the plot generated in part (c) determine the value of f that results in the smallest total impulse for the maneuver.

According to the plotted data above, the value of f that results in the smallest total impulse for the maneuver is $f = \boxed{0.06}$

Code for question 6:

```
%Astro HW#5 Problem 6
clc; clear;

mu              = 398600;
Re              = 6378.145;

%% Orbit 1 (LEO)
alt1            = 300; %[km]
a1              = alt1 + Re;
tau1            = 2*pi*sqrt(a1^3/mu);
tspan1          = [0 tau1];
e               = 0;
inc_deg         = 28.5; %[deg]
inc1            = inc_deg*pi/180; %[rad]
omega           = 0; %UNDEFINED FOR CIRCULAR ORBIT
nu              = 0; %UNDEFINED FOR CIRCULAR ORBIT
Omega           = 0;
oe1             = [a1,e,Omega,inc1,omega,nu];
[rin0, vin0]    = oe2rv_Gusman_Lucas(oe1,mu);
hv1             = cross(rin0,vin0);
uhv1            = hv1/norm(hv1,2);
[tout1,pout1]   = orbit_path_main (rin0,vin0,mu,tspan1);


%% Orbit 2 (Geostationary)
tau2            = 24; %hrs sidereal
tau2            = tau2*3600;
tspan2          = [0 tau2];
a2              = ((tau2/(2*pi))^2 * mu)^(1/3);
Omega2          = 0; %UNDEFINED FOR EQUATORIAL ORBIT
inc2            = 0; %0 FOR EQUATORIAL ORBIT
oe2             = [a2,e,Omega2,inc2,omega,nu];
[rf0,vf0]       = oe2rv_Gusman_Lucas(oe2,mu);
hv2             = cross(rf0,vf0);
uhv2            = hv2/norm(hv2,2);
[tout2,pout2]   = orbit_path_main (rf0,vf0,mu,tspan2);


%% Transfer Orbit
lineint         = cross(hv1,hv2)/norm(cross(hv1,hv2),2);
at              = (a1+a2)/2;
taut            = 2*pi*sqrt(at^3/mu);
tspant          = [0 taut/2];
et              = (a2-a1)/(a1+a2);
oet             = [at,et,Omega,inc1,omega,nu];
[rt0, vt0]      = oe2rv_Gusman_Lucas(oet,mu);
[toutt,poutt]   = orbit_path_main (rt0,vt0,mu,tspant);

```

```
%% Crank Angle
theta          = acos((hv1.'*hv2)/(norm(hv1,2)*norm(hv2,2))); %rad
theta_deg      = theta*180/pi;

%% Impulse Calculations

% All inclination change at APOAPSIS
speed1_minus   = sqrt(mu/a1);
speed1_plus    = sqrt(2*mu/a1-mu/at);
u1v            = cross(hv1,rin0)/norm(cross(hv1,rin0),2);
delv1          = (speed1_plus - speed1_minus)*u1v;

speed2_minus   = sqrt(2*mu/a2-mu/at);
speed2_plus    = sqrt(mu/a2);
u2v            = cross(hv2,rf0)/norm(cross(hv2,rf0),2);
delv2          = speed2_plus*u2v - speed2_minus*u1v;

delva_total     = norm(delv1,2)+norm(delv2,2)

% All inclination change at LEO
delv1_LEO       = speed1_plus*u2v - speed1_minus*u1v;
delv2_LEO       = (speed2_plus - speed2_minus)*u2v;
delvLEO_total   = norm(delv1_LEO,2)+norm(delv2_LEO,2)

% Normalized Impulses
delv1_f          = @(f) ((speed1_minus)^2 + (speed1_plus)^2 - 2*speed1_minus
    *...
     speed1_plus*cos(theta.*f))/(sqrt(mu/a1));
delv2_f          = @(f) ((speed2_minus)^2 + (speed2_plus)^2 - 2*speed2_minus
    *...
     speed2_plus*cos(theta.*(1-f)))/(sqrt(mu/a1));
fv               = 0:.01:1;
impulsev         = zeros(length(fv));
delvtotal_f      = delv1_f(fv) + delv2_f(fv);
delv_min         = min(delvtotal_f);
f_minspot        = find(delvtotal_f == delv_min);
f_min            = fv(f_minspot)
delv_minv        = zeros(length(fv));
delv_minv(:)     = delv_min;

%% Plotting
figure(1)
plot3(pout1(:,1),pout1(:,2),pout1(:,3),pout2(:,1),pout2(:,2),pout2(:,3),...
     poutt(:,1), poutt(:,2),poutt(:,3),'LineWidth',1.5)
hold on
% Line of Intersection
line           = [rf0';-rf0'];
plot3(line(:,1),line(:,2),line(:,3),'LineWidth',1.5)
```

```
% Impulse Vectors
quiver3(rin0(1),rin0(2),rin0(3),delv1(1),delv1(2),delv1(3),5000)
quiver3(-rf0(1),-rf0(2),-rf0(3),-delv2(1),-delv2(2),-delv2(3),5000)
quiver3(rin0(1),rin0(2),rin0(3),delv1_LEO(1),delv1_LEO(2),delv1_LEO(3)
    ,5000)
quiver3(-rf0(1),-rf0(2),-rf0(3),-delv2_LEO(1),-delv2_LEO(2),-delv2_LEO(3)
    ,6000)
xlabel('x (km)')
ylabel('y (km)')
zlabel('z (km)')
legend('Initial Orbit','Terminal Orbit', 'Transfer Orbit',...
    'Line of Intersection','\DeltaV_1(no inclination change)',...
    '\DeltaV_2(whole inclination change)','\DeltaV_1(whole inclination
        change)',...
    '\DeltaV_2(no inclination change)')
hold off

figure(2)
plot(fv,delvtotal_f,fv,delv_minv,'--')
xlabel('f fraction')
ylabel('\DeltaV km*s^{-1}')
legend('\Delta V mag','Minimum Value')
```

```
function [rPCI,vPCI] = oe2rv_Gusman_Lucas(oe,mu)
%-%-----------------------------------------------------------------%-
%-% Input:  orbital elements              (6 by 1 column vector)   %-
%-%   oe(1): Semi-major axis.                                        %-
%-%   oe(2): Eccentricity.                                           %-
%-%   oe(3): Longitude of the ascending node (rad)                   %-
%-%   oe(4): Inclination (rad)                                       %-
%-%   oe(5): Argument of the periapsis (rad)                         %-
%-%   oe(6): True anomaly (rad)                                      %-
%-%   mu:    Planet gravitational parameter      (scalar)            %-
%-% Outputs:                                                         %-
%-%   rPCI:  Planet-Centered Inertial (PCI) Cartesian position       %-
%-%          (3 by 1 column vector)                                  %-
%-%   vPCI:  Planet-Centered Inertial (PCI) Cartesian inertial velocity %-
%-%          (3 by 1 column vector)                                  %-
%-%-----------------------------------------------------------------%-


%%
%Orbital Elements
a     = oe(1);
e     = oe(2);
Omega = oe(3);
inc   = oe(4);
omega = oe(5);
nu    = oe(6);
```

```matlab

%%
%Position and Velocity in Perifocal Basis
p      = a*(1-e^2);
r      = p/(1+e*cos(nu));
rpv    = [r*cos(nu);r*sin(nu);0];
vpv    = [-sin(nu); e+cos(nu);0];
vpv    = sqrt(mu/p)*vpv;

%%
%313 Euler Angle Sequence

%Rotation 1: by Omega (about Iz)
TnI    = [cos(Omega),-sin(Omega),0;sin(Omega),cos(Omega),0;0,0,1];

%Rotaion 2: by inc (about nx)
Tqn    = [1,0,0;0, cos(inc),-sin(inc);0,sin(inc),cos(inc)];

%Rotation 3: by omega (about qz)
Tpq    = [cos(omega),-sin(omega),0;sin(omega),cos(omega),0;0,0,1];

%%
%Final Transformation
TpI    = TnI*Tqn*Tpq;
rPCI   = TpI*rpv;
vPCI   = TpI*vpv;
end
```

```matlab
function [tout,pout] = orbit_path_main (r0,v0,mu,tspan)
%NEED TO CHANGE MU ON ORBIT_PATH_CALC.M FILE
P0           = [r0;v0]; %Column vector of initial conditions
options      = odeset('RelTol',1e-6);
[tout,pout] = ode113(@orbit_path_calc,tspan,P0,options); %using ODE113 and
    calling my function to calculate pout
end
```

```matlab
function [rPCIf,vPCIf,E0,nu0,E,nu] = propagateKepler_Gusman_Lucas(t0,t,
    rPCI0,vPCI0,mu)
% % ------------------------------------------------------------------------ %
% % ---------------------------- propagateKepler.m ------------------------- %
% % -------- Propagate Spacecraft Orbit Using Kepler's Equation -------- %
% % ------------------------------------------------------------------------ %
% % Given a position and inertial velocity at a time t0 expressed in    %
% % planet-centered inertial (PCI) coordinates, determine the position %
% % and inertial velocity at a later time t on an elliptic orbit by    %
% % solving Kepler's equation.                                          %
% % ------------------------------------------------------------------------ %
% % --------------- Inputs (Supplied Data) for Test Cases --------------- %
% % ------------------------------------------------------------------------ %
```

```matlab
%%      t0 = initial time                                                %
%%       t = terminal time                                               %
%% rPCI0 = Initial PCI Position                                          %
%% vPCI0 = Initial PCI Inertial Velocity                                 %
%%      mu = planet gravitational parameter                              %
%% --------------------------------------------------------------------- %
%% ---------- Output (Computed Quantities) from Test Cases ------------- %
%% --------------------------------------------------------------------- %
%% rPCIf = Terminal PCI Position                                         %
%% vPCIf = Terminal PCI Inertial Velocity                                %
%%      E0 = Eccentric Anomaly at Time t0                                %
%%     nu0 = True Anomaly at Time t0                                     %
%%       E = Eccentric Anomaly at Time t                                 %
%%      nu = True Anomaly at Time t                                      %
%% --------------------------------------------------------------------- %
%% --------- Note: all quantities must be in consistent units ---------- %
%% --------------------------------------------------------------------- %

%% Calculating orbital elements from initial position and velocity    %%
oe              = rv2oe_Gusman_Lucas(rPCI0,vPCI0,mu);

%% Defining each orbital element and initial Nu
a               = oe(1);
e               = oe(2);
Omega           = oe(3);
inc             = oe(4);
omega           = oe(5);
nu0             = oe(6);

%% Calculating intial Eccentric Anomaly
E0              = atan2(sqrt(1-e)*sin(nu0/2),sqrt(1+e)*cos(nu0/2));

%% Using Kepler Solver
[E,nu]          = KeplerSolver_Gusman_Lucas(a,e,t0,t,nu0,mu);

%% Final set of oe
oe(6)           = nu;

%% Getting final position and velocity from new oe set
[rPCIf,vPCIf] = oe2rv_Gusman_Lucas(oe,mu);
end
```