Table of Contents

Homework 4

Aero 351 Liam Hood

```
clear ;
close all ;
clc ;
```

8.2

```
problem_8_2() ;
```

8.4

```
problem_8_4() ;
```

```
8.4
The synodic period of Jupiter and Mars is 2.2356 years
This makes sense because Mars has a longer period than Earth so the
 synodic
is longer than Earth and Jupiter
```

8.6

```
problem_8_6()
```

```
8.6
The radius of the sphere of influence of Saturn is 54787876.075 km
The radius of the sphere of influence of Uranus is 51784339.6741 km
The radius of the sphere of influence of Neptune is 86575650.9187 km
These answers make sense because the sphere of influence of Neptune is
 large
because it is far from the sun where as Jupiter is closer to the sun
 so even
though it is larger the sun has a greater effect on objects near it.
 Saturn
```

```
has the smallest because it is the farther from the sun than Jupiter
 and smaller
than Neptune
```

8.7

```
problem_8_7()
```

```
8.7
The hyperbolic excess velocity is 1.5789 km/s
The delta v is 3.337 km/s
This seems accurate because it is a relatively low vinf and there is
relatively little change in orbit. There is a large difference between
a LEO circular orbit and a hyperbolic orbit.
```

8.12

```
problem_8_12() ;
```

```
8.12
Orbital Elements of heliocentric elliptical orbit
h 2736378480.0904 km^2/s
eccentricity 0.93226
inclination 180 km^2/s
theta 183.0762 degrees
semi-major axis 431033868.9337 m
RAAN and argument of periapse are undefined because I don't know where
 the coordinate frame points
Delta v is 10.8516 km/s
Answers seem like they are in the realm of possibility but they are
 wrong
```

8.16

```
problem_8_16() ;
```

```
8.16
Total delta v 5.9368 km/s
The delta v seems reasonable for a transfer to Mars.
```

Problems

```
function problem_8_2()
disp( '8.2' )
    % set up
    rs2m = 227.9e6 ; % radius from sun to mars in km
    rs2j = 778.6e6 ; % radius from sun to jupiter in km
    mus = 132.712e9 ; % mu of sun in km^3/s^2

    [ ~,~,~,~, dvi , dvo ] = HohmannC2C( mus , rs2j , rs2m ) ;
```

```
    dv = dvi + dvo ;
    disp([ 'The total delta v is ' , num2str( dv ) , ' km/s ' ])
    disp( 'This answer seems reasonable for a planetary transfer since
 it ' )
    disp( 'is less than the transfer from Earth to Jupiter and Mars is
 closer' )
    disp( 'to Jupiter than Earth ' )
end

function problem_8_4()
disp( ' ' )
disp( '8.4' )
    Tjupiter = 11.86 ; % Sidereal Period in years
    Tmars = 1.881 ; % Sidereal period in years
    Tsyn = ( Tjupiter*Tmars ) / abs( Tjupiter - Tmars ) ;
    disp([ 'The synodic period of Jupiter and Mars is ' ,
 num2str( Tsyn ) , ' years' ])
    disp( 'This makes sense because Mars has a longer period than
 Earth so the synodic ' )
    disp( 'is longer than Earth and Jupiter' )
end

function problem_8_6()
disp( ' ' )
disp( '8.6' )
    massSun = 1.989e30 ;
    % Saturn
        massSaturn = 568.5e24 ;
        rs2s = 1.433e9 ;
        RsoiS = rs2s*( massSaturn / massSun )^.4 ;
        disp([ 'The radius of the sphere of influence of Saturn is ' ,
 num2str( RsoiS ) , ' km' ])
    % Uranus
        massUranus = 86.83e24 ;
        rs2u = 2.872e9 ;
        RsoiU = rs2u*( massUranus / massSun )^.4 ;
        disp([ 'The radius of the sphere of influence of Uranus is ' ,
 num2str( RsoiU ) , ' km' ])
    % Neptune
        massNeptune = 102.4e24 ;
        rs2n = 4.495e9 ;
        RsoiN = rs2n*( massNeptune / massSun )^.4 ;
        disp([ 'The radius of the sphere of influence of Neptune is '
 , num2str( RsoiN ) , ' km' ])
disp( 'These answers make sense because the sphere of influence of
 Neptune is large ' )
disp( 'because it is far from the sun where as Jupiter is closer to
 the sun so even ' )
disp( 'though it is larger the sun has a greater effect on objects
 near it. Saturn ' )
disp( 'has the smallest because it is the farther from the sun than
 Jupiter and smaller' )
disp( 'than Neptune' )
end
```

```
function problem_8_7()
disp( ' ' )
disp( '8.7' )
    rs2e = 147.4e6 ; % radius of sun to earth at initial time
    zpark = 200 ; % altitude of parking orbit
    re = 6378 ; % radius of earth
    rpark = re + zpark ; % radius of parking orbit
    rper = 120e6 ; % radius of perihelion
    mus = 132.712e9 ; % mu of sun in km^3/s^2
    mue = 398600 ; % mu of earth in km^3/s^2

    vpark = sqrt( mue / rpark ) ;

    vinf = sqrt( mus / rs2e )*( sqrt( (2*rper)/(rs2e+rper) ) - 1 ) ; %
 v infinite of hyperbola
    vpHyp = sqrt( vinf^2 + (2*mue)/rpark ) ;
    dv = vpHyp - vpark ;
    disp([ 'The hyperbolic excess velocity is ' ,
 num2str( abs(vinf) ) , ' km/s' ])
    disp([ 'The delta v is ' , num2str( dv ) , ' km/s' ])
    disp( 'This seems accurate because it is a relatively low vinf and
 there is' )
    disp( 'relatively little change in orbit. There is a large
 difference between' )
    disp( 'a LEO circular orbit and a hyperbolic orbit.' )
end

function problem_8_12()
disp( ' ' )
disp( '8.12' )
    mus = 1.32712e11 ;
    muj = 1.26686e8 ;
    aJ = 778.6e6 ;
    aE = 149.6e6 ;
    eccJ = .0489 ;
    eccE = .0167 ;
    rs2j_a = aJ*(1-eccJ^2)/(1-eccJ); % sun to jupiter radius at
 aphelion
    rs2e_p = aE*(1-eccE^2)/(1+eccE) ; % sun to earth radius at
 perihelion
    rfly = 200000 ;

    % information about the fly by
    [  ~ , vscj , ~ , vj , ~ , ~ ] = HohmannC2C( mus , rs2j_a ,
 rs2e_p ) ;
    vinf1 = vscj - vj ; % v infinite = v of s/c at jupiter - v of
 jupiter
    aHyp = muj / vinf1^2 ;
    eccHyp = 1 + rfly/aHyp ;
    delta = 2*asin(1/eccHyp) ;
    vinf2 = [ vinf1*cos(delta) , vinf1*sin(delta) ] ; % [ planet
 direction , sun direction ]
    dv = norm( [ vinf1 , 0 ] - vinf2 ) ;
```

```
    vleave = vinf2 - [ vscj , 0 ] ;

    % information about orbit after flyby
    state1 = [ rs2j_a ; 0 ; 0 ; vleave' ; 0 ] ;
    [ hleave , incleave , eccleave , RAANleave , omegaleave ,
 thetaleave , aleave ] = state2COE( state1 , mus ) ;
    disp( 'Orbital Elements of heliocentric elliptical orbit' )
    disp([ 'h ' , num2str( hleave ) , ' km^2/s' ])
    disp([ 'eccentricity ' , num2str( eccleave ) ])
    disp([ 'inclination ' , num2str( incleave ) , ' km^2/s' ])
    disp([ 'theta ' , num2str( thetaleave ) , ' degrees' ])
    disp([ 'semi-major axis ' , num2str( aleave ) , ' m' ])
    disp( 'RAAN and argument of periapse are undefined because I
 don''t know where the coordinate frame points' )
    disp([ 'Delta v is ' , num2str( dv ) , ' km/s' ])
    disp( 'Answers seem like they are in the realm of possibility but
 they are wrong' )
end

function problem_8_16()
disp( ' ' )
disp( '8.16' )
mus = 1.32712e11 ;
mue = 398600 ;
mum = 42828 ;
re = 6378 ; % radius of earth
rm = 3396 ; % radius of mars
% [a;ecc;inc;raan;w_hat;L]
    %Earth
    dateBegin = [ 15 , 8 , 2005 ] ; % date in days
    [ ~ , ~ , ~ , JdBegin ] = Julian( [12,0,0] , dateBegin ) ;
    [EarthCOES] = planetary_elements(3,JdBegin/365.25) ;
    hE = sqrt( EarthCOES(1)*mus*(1-EarthCOES(2)^2) ) ;
    omegaE = EarthCOES(5) - EarthCOES(4) ;
    ME = EarthCOES(6)-EarthCOES(5) ;
    TE = (2*pi)/sqrt(mus)*EarthCOES(1) ;
    tE = ME*TE/(2*pi) ;
    [ thetaE ] = time2theta( tE , TE , EarthCOES(2) ) ;
    [ rE , vE ] = coes2state(  hE , EarthCOES(2) , thetaE ,
 EarthCOES(4) , omegaE , EarthCOES(3) , mus ) ;

    % Initial Parking Orbit
    zparkE = 190 ;
    rparkE = re + zparkE ;
    incPE = 52 ;
    vPE = sqrt(mue/rparkE) ;

    % Mars
    dateEnd = [ 15 , 3 , 2006 ] ;
    [ ~ , ~ , ~ , JdEnd ] = Julian( [12,0,0] , dateEnd ) ;
    [MarsCOES] = planetary_elements(4,JdEnd/365.25) ;
    hM = sqrt( MarsCOES(1)*mus*(1-MarsCOES(2)^2) ) ;
    omegaM = MarsCOES(5) - MarsCOES(4) ;
    MM = MarsCOES(6)-MarsCOES(5) ;
```

```
    TM = (2*pi)/sqrt(mus)*MarsCOES(1) ;
    tM = MM*TM/(2*pi) ;
    [ thetaM ] = time2theta( tM , TM , MarsCOES(2) ) ;
    [ rM , vM ] = coes2state(  hM , MarsCOES(2) , thetaM ,
 MarsCOES(4) , omegaM , MarsCOES(3) , mus ) ;

    % Final parking orbit
    rpPM = 300 + rm ; % radius of periapse of parking orbit at mars
    TparkHoursM = 35 ; % period of parking orbit in hours
    TparkM = 35*60*60 ; % period of parking orbit in seconds
    aPM = ( TparkM*sqrt(mum)/(2*pi) )^(2/3) ;
    raPM = 2*aPM - rpPM ;
    eccPM = ( raPM - rpPM )/( raPM + rpPM ) ;
    vpPM = sqrt( mum*raPM/(aPM*rpPM)) ;

    % Transfer
    DaysTransfer = JdEnd - JdBegin ;
    tTrans = DaysTransfer*24*60*60 ; % seconds of transfer
    tol = 1e-8 ;
    [ v1 , v2 ] = Lamberts2( rE , rM , tTrans , mus , tol , 1 ) ;

    % Find delta v to get from parking orbit to transfer to park
    vinfE = v1 - vE ;
    vinfM = v2 - vM ;
    vpHE = sqrt( dot( vinfE , vinfE ) + ( 2*mue )/(rparkE) ) ;
    vpHM = sqrt( dot( vinfM , vinfM ) + ( 2*mum )/(rpPM) ) ;
    dvE = vpHE - vPE ;
    dvM = vpHM - vpPM ;
    dv = dvM + dvE ;
    disp([ 'Total delta v ' , num2str( dv ) , ' km/s' ])
    disp( 'The delta v seems reasonable for a transfer to Mars.' )
end
```

8.2

Functions

```
function [ vp , va , vinner , vouter , dvi , dvo ] = HohmannC2C( mu ,
 router , rinner )

    ecc = ( router - rinner ) / (  router + rinner ) ;
    h = sqrt( rinner*mu*(1+ecc) ) ;

    va = (mu/h)*(1-ecc) ;
    vp = (mu/h)*(1+ecc) ;

    vinner = sqrt( mu / rinner ) ;
    vouter = sqrt( mu / router ) ;

    dvi = vp - vinner ;
    dvo = vouter - va ;

end
```

```
function [ h , inc , ecc , RAAN , omega , theta , a ] =
 state2COE( state, mu )
% all angles output in degrees
    r2d = 180/pi ; % radians to degrees

    Kh = [ 0 0 1 ] ; % K hat
    r = state(1:3) ;
    v = state(4:6) ;
    distance = norm( r ) ;
    speed = norm( v ) ;
    vr = dot( r , v )/distance ; % radial velocity
    h = cross( r , v ) ; % specific angular momentum
    hmag = norm( h ) ; % specific angular momentum
    inc = acos(h(3)/norm(h)) ; %inclination
    eccv = (1/mu)*( cross(v,h)-mu*(r/distance) ) ; %eccentricity
 vector
    ecc = norm( eccv ) ; % eccentricity
    Nv = cross( Kh , h ) ; % Node line
    N = norm( Nv ) ;

    if Nv(2) > 0
        RAAN = acos(Nv(1)/N) ; %Right ascension of ascending node
    elseif Nv(2) < 0
        RAAN = 2*pi - acos(Nv(1)/N) ; %Right ascension of ascending
 node
    else
        RAAN = 'Undefined' ;
    end

    if eccv(3) > 0
        omega = acos(dot(Nv,eccv)/(N*ecc)) ; % Argument of perigee
    elseif eccv(3) < 0
        omega = 2*pi - acos(dot(Nv,eccv)/(N*ecc)) ; % Argument of
 perigee
    else
        omega = 'Undefined' ;
    end

    % True anomaly
    if vr >= 0
        theta = acos( dot(eccv,r)/(ecc*distance) ) ;
    else
        theta = 2*pi - acos( dot(eccv,r)/(ecc*distance) ) ;
    end

    epsilon = speed^2/2 - mu/distance ; % specific energy
    a = - mu/(2*epsilon) ; % semi-major axis

    hvec = h ;
    h = hmag ;
    inc = r2d*inc ;
    if Nv(3) ~= 0
        RAAN = r2d*RAAN ;
```

```
    end
    if eccv(3) ~= 0
        omega = r2d*omega ;
    end
    theta = r2d*theta ;
end

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)

% 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 [ 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

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
```

The total delta v is 10.1542 km/s
This answer seems reasonable for a planetary transfer since it is less than the transfer from Earth to Jupiter and Mars is closer to Jupiter than Earth

Published with MATLAB® R2018b