# Table of Contents



# Homework 3

Aero 351 Liam Hood

```
function HW3()
```

# Clean up

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

# 4.15

```
disp( '4.15' )
mu_e = 398600 ;
r_e = 6378 ;

% Given
ecc = 1.5 ;
z_p = 300 ; % altitude at perigee km
inc = 35 ; % degrees
RAAN = 130 ; % degrees
omega = 115 ; % degrees
theta = 0 ;

r_p = z_p + r_e ;
h = sqrt( mu_e*r_p*(1+ecc) ) ; % angular momentum from orbit equation

[ r_peri , v_peri ] = coes2peri( h , ecc , theta , mu_e ) ; % Orbital
 elements to perifocal state
disp( 'In Perifocal' )
disp( 'r in km' )
disp( r_peri )
disp( 'v in km/s' )
disp( v_peri )
disp( 'These make sense because the the spacecraft is at periapse so
 position' )
```

```
disp( 'is at periapse and velocity is tangential.' )

[ r , v ] = peri2eci( r_peri , v_peri , RAAN , omega , inc ) ; %
 perifocal to eci
disp( 'In ECI' )
disp( 'r in km' )
disp( r )
disp( 'v in km/s' )
disp( v )
disp( 'These make sense as the magnitudes of the vectors are the same
 in ECI')
```

```
4.15
In Perifocal
r in km
   1.0e+03 *

    6.6780
         0
         0

v in km/s
         0
   12.2156
         0

These make sense because the the spacecraft is at periapse so position
is at periapse and velocity is tangential.
In ECI
r in km
   1.0e+03 *

   -1.9838
   -5.3488
    3.4715

v in km/s
   10.3559
   -5.7627
   -2.9611

These make sense as the magnitudes of the vectors are the same in ECI
```

# 4.26

```
clear

disp( '4.26' )
mu_e = 398600 ;
r_e = 6378 ;
J2 = 1.08263*10^(-3) ;

% Given
```

```
r0 = [ -2429.1 ; 4555.1 ; 4577 ] ; % km in ECI
v0 = [ -4.7689 ; -5.6113 ; 3.0535 ] ; % km/s in ECI
t_hr = 72 ; % time in hours

t = t_hr*60*60 ; % time in seconds

[ h , inc , ecc , RAAN , omega , theta , a ] = OrbitalElements( r0 ,
 v0 , mu_e ) ; % Find orbital elements
T = ( (2*pi)/sqrt(mu_e) ) * a^(3/2) ; % period in seconds
[ RAANdot , omegadot ] = oblatenessPerturbation( mu_e , J2 ,  r_e ,
 ecc , a , inc ) ; % Change in RAAN and argument of periapse
RAAN_new = RAAN + RAANdot*t ; % New RAAN
omega_new = omega + omegadot*t ; % New argument of periapse

% Time since sattellite passed final perigeee
[ time_from_perigee ] = theta2time( theta , T , ecc ) ; % finding time
 since satellite passed perigee at the beginning
time_to_next = T-time_from_perigee ; % Time to next perigee after the
 t0
number_full_orbits = floor( (t - time_to_next)/T ) ; % full orbits
 completed
time_final = t - time_to_next - number_full_orbits*T ; % time passed
 final perigee

% Find final true anomaly
[ theta_new ] = time2theta( time_final , T , ecc ) ;

% Find final state
[ r_peri , v_peri ] = coes2peri( h , ecc , theta_new , mu_e ) ;
[ r , v ] = peri2eci( r_peri , v_peri , RAAN_new , omega_new , inc ) ;

disp( 'r in km' )
disp( r )
disp( 'v in km/s' )
disp( v )
disp( 'These values seem good because the speed and radius are similar
 to the ' )
disp( 'starting values and the changges in RAAN and argument of
 periapse are ' )
disp( 'a few degrees a day' )
```

```
4.26
r in km
   1.0e+03 *

    4.5960
    5.7590
   -1.2665

v in km/s
   -3.6014
    3.1794
    5.6174
```

*These values seem good because the speed and radius are similar to the*
*starting values and the changges in RAAN and argument of periapse are a few degrees a day*

# 5.6

```
clear
mu_e = 398600 ;
r1 = [ 5644 -2830 -4170 ] ;
r2 = [ -2240 7320 -4980 ] ;
time = 20 ; % time in minutes
dt = time*60 ; % time in seconds
[ v1_long , v2_long , v1_short , v2_short ] = Lamberts( r1 , r2 , dt ,
 mu_e , 10^-8 , 0 ) ;
disp( 'v1 long' )
disp( norm(v1_long) )
disp( 'v2 long' )
disp( norm(v2_long) )
disp( 'v1 short' )
disp( norm(v1_short) )
disp( 'v2 short' )
disp( norm(v2_short) )
disp( 'definitely wrong because doesn''t match the book but I don''t
 know why' )
```

```
v1 long
    9.1663

v2 long
    8.4572

v1 short
    5.9681

v2 short
    4.8086

definitely wrong because doesn't match the book but I don't know why
```

# 6.8

```
clear

disp( '6.8' )
mu_e = 398600 ;
r_e = 6378 ;

% original and final
z0 = 300 ;
zf = 3000 ;
r0 = z0+r_e ;
rf = zf+r_e ;
```

```
v0 = sqrt( mu_e / r0 ) ;
vf = sqrt( mu_e / rf ) ;

% transfer
ecc = ( rf-r0 )/( rf+r0 ) ;
h = sqrt( r0*mu_e*(1+ecc) ) ;
vp = h/r0 ;
va = h/rf ;

% delta v
dvp = vp - v0 ;
dva = vf - va ;
dv = dva + dvp ;
disp([ 'The delta v for going from 300km circular to 3000km circular
 is ' , num2str( dv ) , ' km/s' ])
disp( 'This seems right because it is a big change in orbit and the
 apogee ')
disp( 'and perigee speeds made sense compared to the circular speeds '
 )

% time of transfer is one half ellipse period
a = .5*(r0+rf) ;
Tt = .5*(2*pi/sqrt(mu_e))*(a^1.5) ;
Tt_minutes = Tt/60 ;
disp([ 'The time of transfer is ' , num2str( Tt_minutes ) , ' minutes'
 ])
disp( 'I don''t have a sense of how long transfers should take' )
```

```
6.8
The delta v for going from 300km circular to 3000km circular is 1.1977
 km/s
This seems right because it is a big change in orbit and the apogee
and perigee speeds made sense compared to the circular speeds
The time of transfer is 59.6542 minutes
I don't have a sense of how long transfers should take
```

# 6.23

```
clear

disp( '6.23' )
mu_e = 398600 ;
r_e = 6378 ;

dapse = 45 ;
thetaB = 45 ;
thetaC = 150 ;
r_a1 = 18900 ;
r_p1 = 8100 ;

% More information about original orbit
a1 = .5*(r_a1+r_p1) ;
T1 = (2*pi/sqrt(mu_e))*a1^1.5 ;
```

```
ecc1 = ( r_a1-r_p1 )/( r_a1+r_p1 ) ;
v_p1 = sqrt( 2*( -.5*(mu_e/a1) + (mu_e/r_p1) ) ) ;
h1 = v_p1*r_p1 ;

% Speed information of B in original orbit
v1az = (mu_e/h1)*(1+ecc1*cosd(thetaB)) ;
v1r = (mu_e/h1)*(ecc1*sind(thetaB)) ;
v1 = sqrt( v1az^2 + v1r^2 ) ;
gamma1 = atand( v1r/v1az ) ;

% Time past periapse of each orbits
t_b1 = theta2time( 45 , T1 , ecc1 ) ;
t_c1 = theta2time( 150 , T1 , ecc1 ) ;

% Phasing
dt_bc = t_c1 - t_b1 ; % Time seperating b and c
T2 = T1 - dt_bc ; % Period of b phasing to catch c
a2 = ( T2 / (2*pi/sqrt(mu_e)) )^(2/3) ; % a of phasing
r_B = (h1^2/mu_e)*(1/(1+ecc1*cosd(thetaB))) ; % radius of B
r_p2 = r_B ;
r_a2 = 2*a2 - r_p2 ;
ecc2 = ( r_a2-r_p2 )/( r_a2+r_p2 ) ;
h2 = sqrt( r_p2*mu_e*(1+ecc2) ) ;

% Speed information of B in phasing orbit
v2az = (mu_e/h2)*(1+ecc2*cosd(0)) ;
v2r = (mu_e/h2)*(ecc2*sind(0)) ;
v2 = sqrt( v2az^2 + v2r^2 ) ;
gamma2 = atand( v2r/v2az ) ;

deltav_o2p = sqrt( v1^2 + v2^2 - 2*v1*v2*cosd( gamma2 - gamma1 ) ) ; %
 Original to phasing
deltav_p2o = sqrt( v1^2 + v2^2 - 2*v1*v2*cosd( -gamma2 +
 gamma1 ) ) ; % Phasing to original
deltav = deltav_o2p + deltav_p2o ;

disp([ 'The delta v for B to catch C in one orbit is ',
 num2str(deltav) , ' km/s' ])
disp( 'This answer seems correct as it is a big change in ecc and apse
 line ')
disp( 'and the delta v is high' )
```

*6.23*
*The delta v for B to catch C in one orbit is 3.4054 km/s*
*This answer seems correct as it is a big change in ecc and apse line*
*and the delta v is high*

# 6.25

```
clear
disp( '6.25' )
mu_e = 398600 ;
r_e = 6378 ;
```

```
% Given
z_p1 = 1270 ;
v_p1 = 9 ;
r_p1 = r_e+z_p1 ;
ecc2 = .4 ;
theta = 100 ;

% More starting values
h1 = r_p1*v_p1 ;
ecc1 = v_p1*(h1/mu_e) - 1 ;
r1 = (h1^2/mu_e)*(1/(1+ecc1*cosd(theta))) ;

h2 = sqrt( r1*mu_e*(1+ecc2*cosd(theta)) ) ;

% at theta = 100
v1r = (mu_e/h1)*(1+ecc1*cosd(theta)) ;
v1az = (mu_e/h1)*ecc1*sind(theta) ;
v1 = sqrt( v1r^2 + v1az^2 ) ;
gamma1 = atand( v1r/v1az ) ;

v2r = (mu_e/h2)*(1+ecc2*cosd(theta)) ;
v2az = (mu_e/h2)*ecc2*sind(theta) ;
v2 = sqrt( v2r^2 + v2az^2 ) ;
gamma2 = atand( v2r/v2az ) ;

deltav = sqrt( v1^2 + v2^2 - 2*v1*v2*cosd( gamma2 - gamma1 ) ) ;
delta_gamma = gamma1-gamma2 ;
disp([ 'The delta v is ' , num2str(deltav) , ' km/s and the change in
 flight ' ])
disp([ 'angle is ' , num2str(delta_gamma) , ' degrees' ])
disp( 'Seems like a reasonable delta v and delta flight path angle
 since just ' )
disp( 'eccentricity is being changed' )
```

```
6.25
The delta v is 0.91545 km/s and the change in flight
angle is -8.1813 degrees
Seems like a reasonable delta v and delta flight path angle since
 just
eccentricity is being changed
```

# 6.31

```
disp( 'on paper' )
```

```
on paper
```

# 6.44

```
clear

disp( '6.44' )
```

```
mu_e = 398600 ;
r_e = 6378 ;

% Given
z1 = 300 ;
z2 = 600 ;
r1 = r_e+z1 ;
r2 = r_e+z2 ;
dinc = 20 ;

% more beginning values
ecct = ( r2-r1 )/( r2+r1 ) ;
h1 = sqrt( r1*mu_e ) ;
h2 = sqrt( r2*mu_e ) ;
ht = sqrt( r1*mu_e*(1+ecct) ) ;
v1 = h1/r1 ;
v2 = h2/r2 ;
vpt = ht/r1 ;
vat = ht/r2 ;

% three burns
dv1_a = vpt - v1 ;
dv2_a = v2 - vat ;
dvinc_a = 2*v2*sind(dinc/2) ;
dv_a = dv1_a + dv2_a + dvinc_a ;
disp([ 'Delta v with 3 burns is ' , num2str( dv_a ) , ' km/s' ])

% two burns inc and final same time
dv1_b = dv1_a ;
dv2_b = sqrt( vat^2 + v2^2 - 2*vat*v2*cosd(dinc) ) ;
dv_b = dv1_b + dv2_b ;
disp([ 'Delta v with inc change at insertion into final orbit is ' ,
 num2str( dv_b ) , ' km/s' ])

% two burns inc and final same time
dv2_c = v2 - vat ;
dv1_c = sqrt( vpt^2 + v1^2 - 2*vpt*v1*cosd(dinc) ) ;
dv_c = dv1_c + dv2_c ;
disp([ 'Delta v with inc change at insertion into transfer orbit is '
 , num2str( dv_c ) , ' km/s' ])

disp( 'These answers seem correct because there is an inclination
 change so they' )
disp( 'should be large. There are all similar with the ones that
 combine burns ')
disp( 'being better and the best happening at apogee' )
```

```
6.44
Delta v with 3 burns is 2.7927 km/s
Delta v with inc change at insertion into final orbit is 2.696 km/s
Delta v with inc change at insertion into transfer orbit is 2.7826 km/
s
These answers seem correct because there is an inclination change so
 they
```

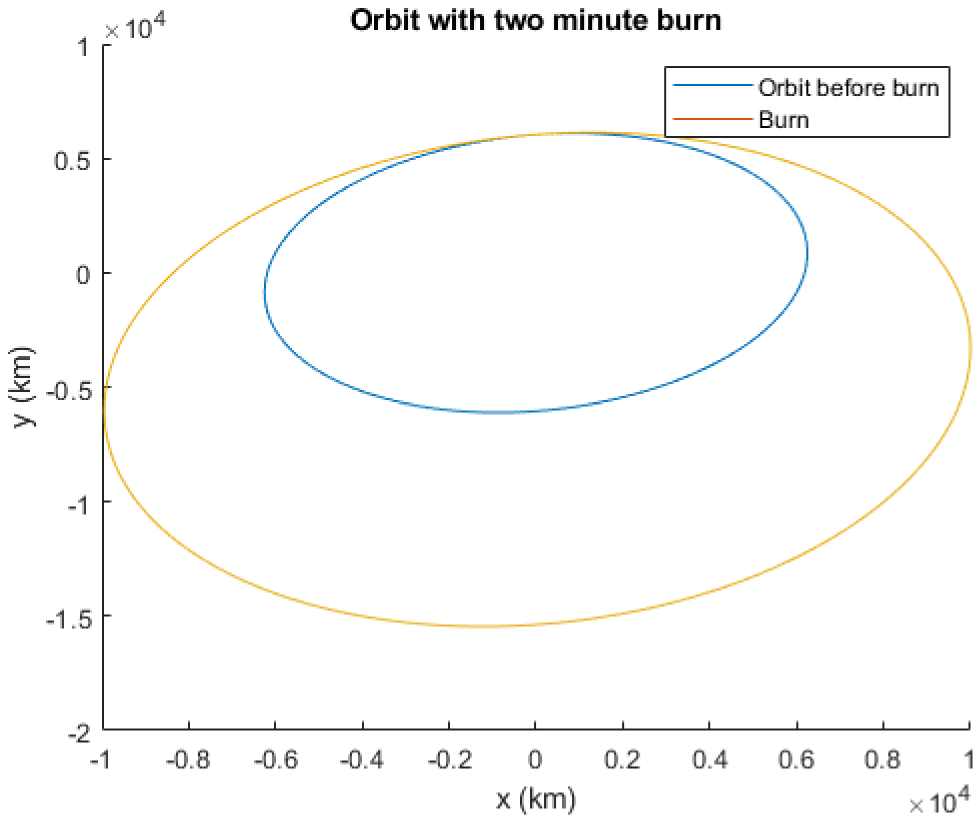
*should be large. There are all similar with the ones that combine burns being better and the best happening at apogee*

# 6.47

```
clear

disp( '6.47' )
mu_e = 398600 ;
r_e = 6378 ;

m = 1000 ; % mass satellite in kg
r0 = [ 436 6083 2529 ] ;
v0 = [ -7.34 -.5125 2.497 ] ;
tnoburn_minutes = 89 ;
tnoburn = tnoburn_minutes*60 ; % seconds
isp = 300 ; % seconds
thrust = 10 ; % newtons
tthrust = 120 ; % seconds
state0 = [ r0 , v0 ] ;
g0 = 9.81 ;
ooptions = odeset( 'AbsTol' , 1e-8 , 'RelTol' , 1e-8 ) ;

[ tpreburn , statepreburn ] = ode45( @TwoBodyMotion , [ 0 ,
 tnoburn ] , state0 , ooptions , mu_e ) ; % Motion without engine
len = length( tpreburn ) ;
state1 = [ statepreburn(len,:)' ; m ] ; % state as burn begins
[ tburn , statefinal ] = ode45( @NonImpulsive , [ 0 , tthrust ] ,
 state1 , ooptions , mu_e , thrust , isp , g0 ) ; % motion with burn
[ t_neworbit , stateneworbit ] = ode45( @TwoBodyMotion , [ 0 ,
 tnoburn*3 ] , statefinal(length(tburn),1:6) , ooptions , mu_e ) ; %
 Motion without engine
% Turn all positions into single column of radii
for ii = 1:length(tpreburn)
    rmag(ii) = norm(statepreburn(ii,1:3)) ;
end
for ii = 1:length(tburn)
    rmag(ii+length(tpreburn)) = norm(statefinal(ii,1:3)) ;
end
t = [ tpreburn ; tburn+tnoburn ] ; % single column of time
rmagmax = max(rmag) ; % find max radius
t_of_max_s = t(find(rmag == rmagmax)) ; % Time of max radius
t_of_max = t_of_max_s(1)/60 ; % time of max radius in seconds
zmax = rmagmax-r_e ; % max altitude
disp([ 'The maximum altitude is ' , num2str(zmax) , ' km at ' ,
 num2str(t_of_max) , ' minutes' ])
disp( 'This seems reasonable a reasonable answer because the burn puts
 it on ' )
disp( 'a new orbit that intersects with the old. It does not initially
 go far ' )
disp( 'from the original orbit because it has not traveled very far on
 the larger' )
```

```
disp( 'new orbit' )
% figure
hold on
plot3( statepreburn(:,1) , statepreburn(:,2) , statepreburn(:,3) )
plot3( statefinal(:,1) , statefinal(:,2) , statefinal(:,3) )
plot3( stateneworbit(:,1) , stateneworbit(:,2) , stateneworbit(:,3) )
xlabel( 'x (km)' )
ylabel( 'y (km)' )
zlabel( 'z (km)' )
title( 'Orbit with two minute burn' )
legend( 'Orbit before burn' , 'Burn' )
hold off
```

*6.47*
*The maximum altitude is 232.4446 km at 91 minutes*
*This seems reasonable a reasonable answer because the burn puts it on a new orbit that intersects with the old. It does not initially go far from the original orbit because it has not traveled very far on the larger new orbit*




# Functions

```
function [ r , v ] = coes2peri( h , ecc , theta , mu )
% find perifocal position and velocity from COES. Theta in degrees
```

```
    r = (h^2/mu) * ( 1/( 1 + ecc*cosd(theta) ) ) * [ cosd( theta ) ;
 sind( theta ) ; 0 ] ;
    v = (mu/h) * [ -sind( theta ) ; ecc+cosd(theta) ; 0 ] ;
end

function [ r , v ] = peri2eci( r_peri , v_peri , RAAN , omega , inc )
% finds state vectors from COES and perifocal coordinates. Angles in
% degrees
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 [ h , inc , ecc , RAAN , omega , theta , a ] =
 OrbitalElements( r , v , mu )
% all angles output in degrees
    r2d = 180/pi ; % radians to degrees

    Kh = [ 0 0 1 ] ; % K hat

    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 ;
    RAAN = r2d*RAAN ;
    omega = r2d*omega ;
    theta = r2d*theta ;
end

function [ RAANdot , omegadot ] = oblatenessPerturbation( mu , J2 ,
 R , ecc , a , inc )
% Inputs and outputs in degrees
RAANdotr = -((3/2)*((sqrt(mu)*J2*R^2)/((1-
ecc^2)^2*a^(7/2))))*cosd(inc) ;
omegadotr = RAANdotr*(((5/2)*sind(inc)^2-2)/cosd(inc)) ;
RAANdot = RAANdotr*(180/pi) ;
omegadot = omegadotr*(180/pi) ;
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 [ time ] = theta2time( theta , T , ecc )
theta = theta*(pi/180) ;
n = 2*pi/T ;
E = 2*atan( sqrt((1-ecc)/(1+ecc)) * tan( theta/2 ) ) ;
time = ( E - ecc*sin(E) ) / n ;
    if time < 0
        time = T+time ;
    end
end

function [ v1_long , v2_long , v1_short , v2_short ] = Lamberts( r1 ,
 r2 , dt , mu , tol , pro )
% pro is 1 or 0 for prograde or retrograde respectively
chi = @(y,C) sqrt(y/C) ;
    rcross = cross( r1 , r2 ) ;
    r1mag = norm(r1) ;
    r2mag = norm(r2) ;

% Find angle travelled
if pro == 1    % Prograde
    if rcross(3) >= 0
        dtheta = acosd(dot( r1 , r2 )/(r1mag*r2mag)) ;
    else
        dtheta = 360 - acosd(dot( r1 , r2 )/(r1mag*r2mag)) ;
    end
end
if pro == 0    % Retrograde
    if rcross(3) <= 0
        dtheta = acosd(dot( r1 , r2 )/(r1mag*r2mag)) ;
    else
        dtheta = 360 - acosd(dot( r1 , r2 )/(r1mag*r2mag)) ;
    end
end

% long way
tm = -1 ;
dtheta = asind( tm*sqrt(1-(cosd(dtheta))^2) ) ; % angle travelled long
 way
        A = ( sqrt( r1mag*r2mag )*sind( dtheta ))/sqrt(1-
cosd(dtheta)) ;
```

```
    % Bisection to find z
        z = 0 ;
        C = .5 ;
        S = 1/6 ;
        y = r1mag + r2mag + A*(( z * S - 1 )/sqrt(C)) ;
        zup = 4*pi^2 ;
        zlow = -4*pi^2 ;
        dtloop = (chi(y,C)^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 ) ;
            dtloop = (chi(y,C)^3*S)/sqrt(mu) + (A*sqrt(y))/sqrt(mu);
        end
        y = r1mag + r2mag + A*(( z * S - 1 )/sqrt(C)) ;

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

        % velocities
        v1_long = (1/g)*(r2-f*r1) ;
        v2_long = (1/g)*(gdot*r2-r1) ;

% short way
tm = 1 ;
dtheta = asind( tm*sqrt(1-(cosd(dtheta))^2) ) ;
        A = ( sqrt( r1mag*r2mag )*sind( dtheta ))/sqrt(1-
cosd(dtheta)) ;

    % Bisection to find z
        z = 0 ;
        C = .5 ;
        S = 1/6 ;
        y = r1mag + r2mag + A*(( z * S - 1 )/sqrt(C)) ;
        zup = 4*pi^2 ;
        zlow = -4*pi^2 ;
        dtloop = (chi(y,C)^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 ) ;
            dtloop = (chi(y,C)^3*S)/sqrt(mu) + (A*sqrt(y))/sqrt(mu);
        end
        y = r1mag + r2mag + A*(( z * S - 1 )/sqrt(C)) ;
```

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

        v1_short = (1/g)*(r2-f*r1) ;
        v2_short = (1/g)*(gdot*r2-r1) ;

end

function [ S , C ] = Stumpf( z )
% Stumpff Functions
sl = 10 ;
sc = zeros(1,sl) ;
cc = zeros(1,sl) ;
S = 0 ;
C = 0 ;

        for kk = 1:sl
            sc(kk) = (-1)^(kk-1) / factorial(2*(kk-1)+3) ;
        end
        for kk = 1:sl
            cc(kk) = (-1)^(kk-1) / factorial(2*(kk-1)+2) ;
        end

        for jj = 1:sl
            S = S + sc(jj)*z^(jj-1) ;
        end
        for jj = 1:sl
            C = C + cc(jj)*z^(jj-1) ;
        end
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))/(m*vel) ; %
 acceleration in x
    ddy = -(mu*state(2)/rad^3)+(thrust*state(5))/(m*vel) ; %
 acceleration in y
    ddz = -(mu*state(3)/rad^3)+(thrust*state(6))/(m*vel) ; %
 acceleration in z
    mdot = -thrust*1000/(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*