Table of Contents

.... 1
Analytic Part 1
Numerical Part 1
Functions 6


```
clc
clear
close all
```

Analytic Part

```
MU = 3.986*10^5 ;

% values given in part B analytical part of project promt
a_1 = 13000;
a_2 = 7226.58;                % semi major axes of orbits 1 and 2

e_1 = 0.3;
e_2 = 0.444819;               % eccentricity of orbits 1 and 2

I_1 = 20;
I_2 = 20;                     % inclinations of orbits 1 and 2

RAAN_1 = 30;
RAAN_2 = 30;                  % right ascension of the ascending node for orbits
 1 and 2

AOP_1 = 50 ;
AOP_2 = 301.901 ;             % arguments of periapsis for orbits 1 and 2


delta_V_Analytic =
 deltaVCalc(a_1,e_1,I_1,RAAN_1,AOP_1,a_2,e_2,I_2,RAAN_2,AOP_2,MU)
```

Numerical Part

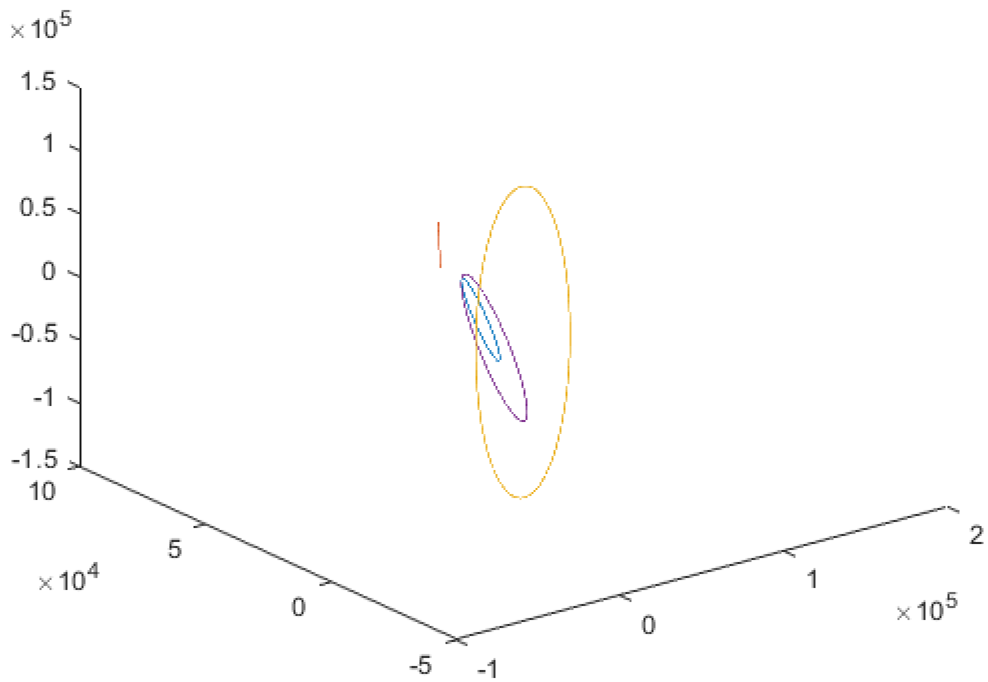
```
load('IODMeasurements.mat')
load('IODMeasurements2.mat')

l = length(AZIMUTH) ;
LOS = zeros(l,3) ;

MU = 3.986*10^5 ;

% calculating LOS
for i = 1:l
    LOS(i,1) = cosd(ELEVATION(i)) * cosd(AZIMUTH(i)) ;
```

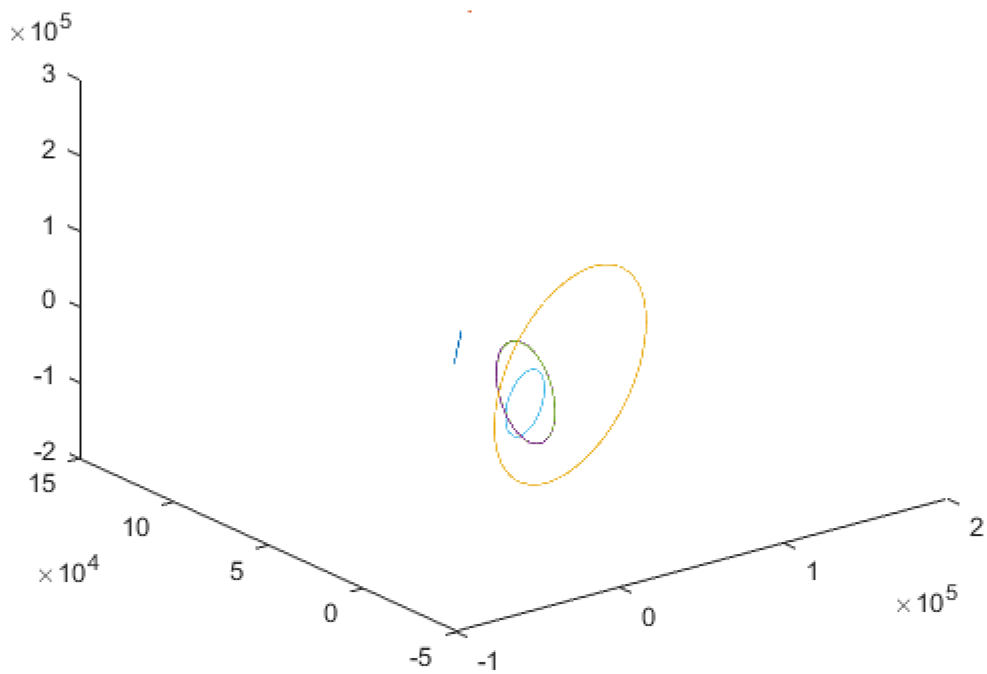
```
    LOS(i,2) = cosd(ELEVATION(i)) * sind(AZIMUTH(i)) ;
    LOS(i,3) = sind(ELEVATION(i)) ;
end

% obtaining orbital information from Gauss's and Gibbs' method
RSAT = zeros(3,3,3,l/3) ;
V2SAT = zeros(3,3,l/3) ;
ORBEL = zeros(3,6,l/3) ;
for i = 1:l/3
    [RSAT(:,:,:,i),V2SAT(:,:,i),ORBEL(:,:,i)] =
 gauss(TIMES((i*3-2:i*3)) ,RSITES((i*3-2:i*3),:) ,LOS((i*3-2:i*3),:),MU) ;
end

% ODE45 propogating all orbits
iters = 10000 ;
init = zeros(1,6) ;
RSAT_T = zeros(iters,6,l/3,3) ;
for i = 1:l/3
    for j = 1:3
        if V2SAT(j,:,i) ~= [0 0 0]
            totalT = 2*pi*sqrt(abs(ORBEL(j,1,i)^3)/MU) ;
            t = linspace(1,totalT,iters) ;
            options = odeset('reltol',1e-12,'abstol',1e-12) ;
            init((1:3)) = RSAT(2,:,j,i) ;
            init((4:6)) = V2SAT(j,:,i) ;
            [t,RSAT_T(:,:,i,j)] = ode45( @(t,RSAT_T) TwoBP(t,RSAT_T,MU) , t ,
 init, options) ;
        end
    end
end

% code for investing the orbits that have multiple r2 values
for i = 7:l/3
    f(i) = figure ;
    xlabel('X (KM)')
    ylabel('Y (KM)')
    zlabel('Z (KM)')
    for j = 1:3
        if V2SAT(j,:,i-1) ~= [0 0 0]
            subplot(1,1,1)
            plot3(RSAT_T(:,1,i-1,j), RSAT_T(:,2,i-1,j), RSAT_T(:,3,i-1,j))
            hold on
        end
        if V2SAT(j,:,i) ~= [0 0 0]
            subplot(1,1,1)
            plot3(RSAT_T(:,1,i,j), RSAT_T(:,2,i,j), RSAT_T(:,3,i,j))
            hold on
        end
    end
    hold off
end

exportgraphics(f(7),['r2Orbit7' '.jpg'])
exportgraphics(f(8),['r2Orbit8' '.jpg'])
```

```
% final plot after selecting valid orbits
ORBEL_R = zeros(l/3,6) ;
f = figure ;
subplot(1,1,1)
for i = 1:l/3

    xlabel('X (KM)')
    ylabel('Y (KM)')
    zlabel('Z (KM)')
    j = 1 ;
    if (i == 7)
        j = 3 ;
    elseif (i == 8)
        j = 2 ;
    end

    ORBEL_R(i,:) = ORBEL(j,:,i) ;

    plot3(RSAT_T(:,1,i,j), RSAT_T(:,2,i,j), RSAT_T(:,3,i,j))
    hold on
    quiver3(RSAT_T(1,1,i,j), RSAT_T(1,2,i,j), RSAT_T(1,3,i,j),RSAT_T(1,4,i,j),
 RSAT_T(1,5,i,j), RSAT_T(1,6,i,j),4000,'LineWidth',3)
    hold on


end
hold off
legend('Orbit 1','Orbit 1 vel','Orbit 2','Orbit 2 vel','Orbit 3','Orbit 3
 vel','Orbit 4','Orbit 4 vel','Orbit 5','Orbit 5 vel','Orbit 6','Orbit 6
 vel','Orbit 7','Orbit 7 vel','Orbit 8','Orbit 8 vel')
exportgraphics(f,['3D' '.jpg'])

% clean plot of only orbits that seem unique
f = figure ;
subplot(1,1,1)

xlabel('X (KM)')
ylabel('Y (KM)')
zlabel('Z (KM)')

lvals = [1,2,4,6,7] ;
ORBEL_F = zeros(7,6) ;
delta_V_Numeric = zeros(1,7)
j = 1 ;
delta_V_Numeric_tot = 0 ;
for i = lvals
    if (i == 7)
        j = 3;
    end
    plot3(RSAT_T(:,1,i,j), RSAT_T(:,2,i,j), RSAT_T(:,3,i,j))
    hold on
    quiver3(RSAT_T(1,1,i,j), RSAT_T(1,2,i,j), RSAT_T(1,3,i,j),RSAT_T(1,4,i,j),
 RSAT_T(1,5,i,j), RSAT_T(1,6,i,j),4000,'LineWidth',3)
```

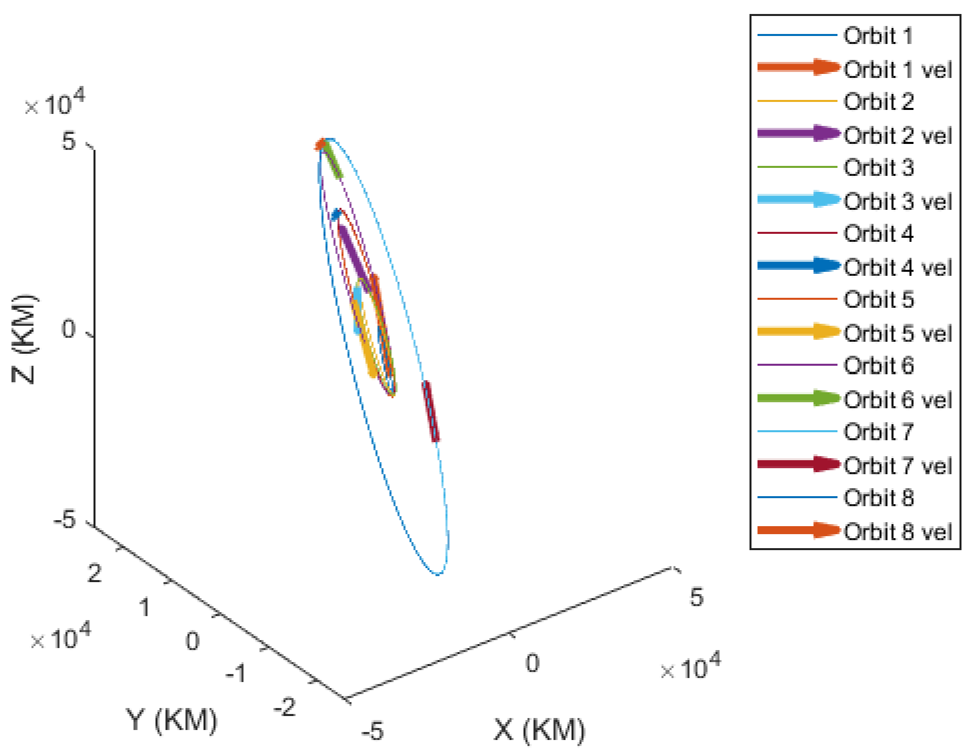
```
    hold on

    ORBEL_F(i,:) = ORBEL(j,:,i) ;
    if i ~= 1
        delta_V_Numeric(i) =
 deltaVCalc(ORBEL_F(ip,1),ORBEL_F(ip,2),ORBEL_F(ip,3),ORBEL_F(ip,4),ORBEL_F(ip,5),ORBEL_F(
        delta_V_Numeric_tot = delta_V_Numeric_tot + delta_V_Numeric(i) ;
    end
    ip = i ;
end
hold off
legend('Orbit 1','Orbit 1 vel','Orbit 2','Orbit 2 vel','Orbit 4','Orbit 4
 vel','Orbit 6','Orbit 6 vel','Orbit 7','Orbit 7 vel')
exportgraphics(f,['3D-clean' '.jpg'])

delta_V_Numeric = delta_V_Numeric' ;
```

```
delta_V_Numeric =

     0     0     0     0     0     0     0
```

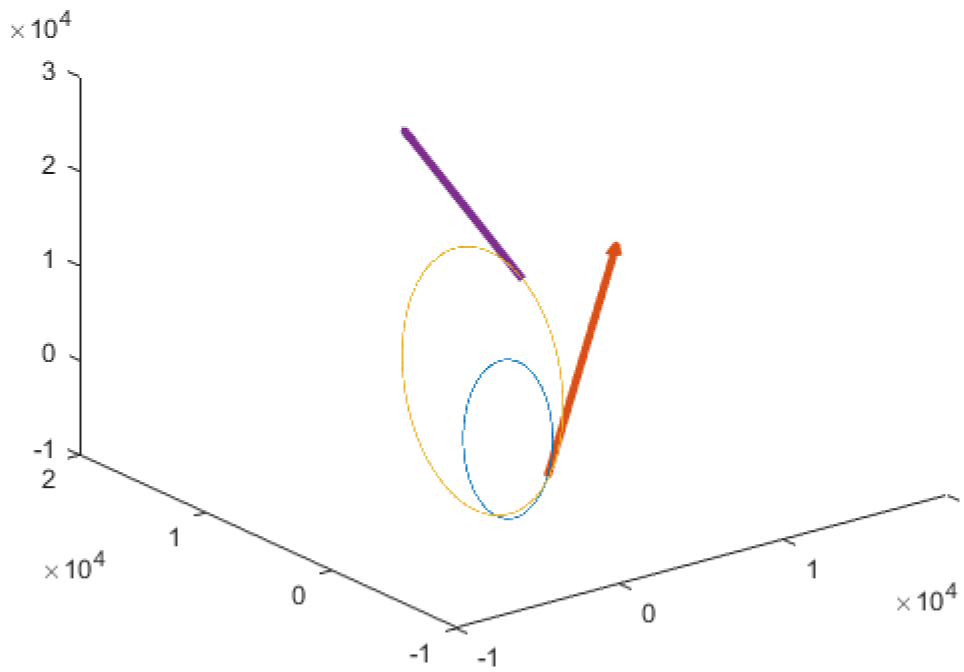




×10^5
3
2
1
0
-1
-2
15
10
×10^4
5
0
-5
-1
0
1
2
×10^5



Orbit 1
Orbit 1 vel
Orbit 2
Orbit 2 vel
Orbit 3
Orbit 3 vel
Orbit 4
Orbit 4 vel
Orbit 5
Orbit 5 vel
Orbit 6
Orbit 6 vel
Orbit 7
Orbit 7 vel
Orbit 8
Orbit 8 vel
×10^4
5
0
-5
Z (KM)
2
1
×10^4
0
-1
-2
Y (KM)
-5
0
5
×10^4
X (KM)



Functions

```
% function for the analytical section
function delta_V = deltaVCalc
 (a_1,e_1,I_1,RAAN_1,AOP_1,a_2,e_2,I_2,RAAN_2,AOP_2,MU)
    %calculation of semi latus rectum for both orbits
    p_1 = a_1 * (1 - e_1^2);
    p_2 = a_2 * (1 - e_2^2);

    % calculation of gamma, beta, and alpha (given in project guidelines for
    % simplified calculations)
    gamma = e_1 * e_2 * sind(AOP_1 - AOP_2);
    beta = e_1 * p_2 - e_2 * p_1 * cosd(AOP_1 - AOP_2);
    alpha = e_2 * cosd(AOP_1 - AOP_2) - e_1;

    % calculation of a, b, and c (determined from equating r^2sin^2f
 expressions)
    a = (e_1^2 - 1) / e_1^2 - alpha^2 / gamma^2;
    b = 2*p_1 / e_1^2 - 2*beta*alpha / gamma^2;
    c = - (p_1^2 / e_1^2 + beta^2 / gamma^2);

    %calculation of radius for orbits 1 and 2 using quadratic formula
    r_1 = (-b + sqrt(b^2 - 4*a*c)) / (2*a)
    r_2 = (-b - sqrt(b^2 - 4*a*c)) / (2*a)
```

```
    r_1 = real(r_1) ;
    r_2 = real(r_2) ;

    % calculation of true anaomaly at r
    f_1 = acosd((p_1-r_1)/(r_1*e_1));
    f_2 = acosd((p_2-r_2)/(r_2*e_2));

    % velocity and position calculations in the perifocal frame of each orbit
    rp_1 = r_1 * [cosd(f_1),sind(f_1),0] ;
    vp_1  = sqrt(MU/p_1) * [-sind(f_1),(e_1+cosd(f_1)),0] ;
    rp_2 = r_2 * [cosd(f_2),sind(f_2),0] ;
    vp_2  = sqrt(MU/p_2) * [-sind(f_2),(e_2+cosd(f_2)),0] ;

    % transformation into perifocal frame
    cEP_1 = dcm3axis(AOP_1)*dcm1axis(I_1)*dcm3axis(RAAN_1) ;
    cPE_1 = cEP_1' ;
    rECI_1 = cPE_1 * rp_1'
    vECI_1 = cPE_1 * vp_1'

    cEP_2 = dcm3axis(AOP_2)*dcm1axis(I_2)*dcm3axis(RAAN_2) ;
    cPE_2 = cEP_2' ;
    rECI_2 = cPE_2 * rp_2'
    vECI_2 = cPE_2 * vp_2'

    % delta V calculation
    delta_V = norm(vECI_2-vECI_1) ;
end



% Gauss's method, made from class handout as well as the wikipedia article on
 the subject
function [Rsat,V2sat,OE] = gauss(time,R,L,mu)
    z1 = time(1) - time(2) ;
    z3 = time(3) - time(2) ;
    z13  = time(3) - time(1) ;

    lcross(1,:) = cross(L(2,:),L(3,:)) ;
    lcross(2,:) = cross(L(1,:),L(3,:)) ;
    lcross(3,:) = cross(L(1,:),L(2,:)) ;

    D0 = dot(L(1,:),lcross(1,:)) ;

    D = zeros(3,3) ;
    for i = 1:3
        for j = 1:3
            D(i,j) = dot(R(i,:),lcross(j,:)) ;
        end
    end

    A = 1/D0 * (-D(1,2)*(z3/z13) + D(2,2) + D(3,2)*(z1/z13)) ;
    B = 1/(6*D0) * ( D(1,2)*(z3^2-z13^2)*(z3/z13) + D(3,2)*(z13^2-z1^2)*(z1/
z13)) ;
    E = dot(L(2,:),R(2,:)) ;
```

```
    R2 = norm(R(2,:)) ;
    a = - (A^2 + 2*A*E + R2) ;
    b = - 2*mu*B*(A+E) ;
    c = - mu^2*B^2 ;

    r2 = roots([1 0 a 0 0 b 0 0 c]) ;
    % rejects values that are not real and positive
    r2 = r2(r2==real(r2)) ;
    r2 = r2(r2>0) ;

    Rsat = zeros(3,3,3) ;
    V2sat = zeros(3,3) ;
    OE = zeros(3,6) ;
    for i = 1:length(r2)
        u2 = mu/(r2(i)^3) ;

        c1 = z3/z13 * (1 + (u2/6) * (z13^2 - z3^2)) ;
        c3 = - z1/z13 * (1 + (u2/6) * (z13^2 - z1^2)) ;

        rho(1) = 1/D0 * ( -D(1,1) + (1/c1)*D(2,1) - (c3/c1)*D(3,1) ) ;
        rho(2) = A + u2*B;
        rho(3) = 1/D0 * ( (-c1/c3)*D(1,3) + (1/c3)*D(2,3) - D(3,3) )  ;

        Rsat(1,:,i) = R(1,:) + rho(1) * L(1,:) ;
        Rsat(2,:,i) = R(2,:) + rho(2) * L(2,:) ;
        Rsat(3,:,i) = R(3,:) + rho(3) * L(3,:) ;

        V2sat(i,:) = gibbs(Rsat(:,:,i),mu) ;

        [a,e,I,O,W,f] = RV2OE(Rsat(2,:,i),V2sat(i,:),mu) ;
        OE(i,:) = [a,e,I,O,W,f] ;
    end
end


% Gibbs' Method, sourced from textbook
function v2 = gibbs(R, mu)
    r(1) = norm(R(1,:)) ;
    r(2) = norm(R(2,:)) ;
    r(3) = norm(R(3,:)) ;

    cR2R3 = cross(R(2,:),R(3,:)) ;
    cR3R1 = cross(R(3,:),R(1,:)) ;
    cR1R2 = cross(R(1,:),R(2,:)) ;

    D = cR2R3 + cR3R1 + cR1R2 ;
    N = r(1) * cR2R3 +  r(2) * cR3R1 + r(3) * cR1R2 ;
    S = (r(2)-r(3))*R(1,:) + (r(3)-r(1))*R(2,:) + (r(1)-r(2))*R(3,:) ;

    d = norm(D) ;
    n = norm(N) ;
    s = norm(S) ;
```

```
    v2 = 1/r(2) * sqrt( mu / (n*d)) * cross(D,R(2,:)) + sqrt(mu/(n*d)) * S ;
end


% Function to get orbital elements from r and v
function [a,e,I,RAAN,AOP,f] = RV2OE(r,v,mu)
    % declaring gravitational constant of earth and time provided and ECI
    ECI = [[1 0 0];[0 1 0];[0 0 1]];

    R1 = norm(r);
    V1 = norm(v);

    energy = (V1^2)/2-mu/R1;

    h = cross(r,v);
    H = norm(h);

    p = H^2/mu;

    % calculating semi major axis
    a = -mu/(2*energy);

    % calculating eccentricity
    eV = cross(v,h)/mu - (r/R1);
    e = norm(eV);

    % calculating orbital inclination
    I = acos(dot(ECI(3,:),h)/H);

    % calculating longitude of the ascending node
    n = cross(ECI(3,:),h)/norm(cross(ECI(3,:),h));

    if(dot(n,ECI(1,:)) >= 0)
        RAAN = atan(dot(n,ECI(2,:))/dot(n,ECI(1,:)));
    elseif (dot(n,ECI(1,:)) < 0)
        RAAN = atan(dot(n,ECI(2,:))/dot(n,ECI(1,:)))+pi;
    end

    % calculating argument of periapsis
    if(dot(eV,ECI(:,3)) >= 0)
        AOP = acos(dot(eV,n)/e);
    elseif (dot(eV,ECI(:,3)) < 0)
        AOP = -acos(dot(eV,n)/e);
    end

    if(dot(r,eV) >= 0)
        f = acosd(dot(eV,r)/(e*R1)) ;
    elseif(dot(r,eV) < 0)
        f = 360 - acosd(dot(eV,r)/(e*R1)) ;
    end
end


% Two body problem function
```

```
function dx = TwoBP(~, r, mu)
    x = r(1) ;
    y = r(2) ;
    z = r(3) ;
    xdot = r(4) ;
    ydot = r(5) ;
    zdot = r(6) ;

    R = sqrt(x^2 + y^2 + z^2) ;

    xdoubledot = -mu/R^3 * x ;
    ydoubledot = -mu/R^3 * y ;
    zdoubledot = -mu/R^3 * z ;

    dx = [ xdot ; ydot ; zdot ; xdoubledot ; ydoubledot ; zdoubledot ] ;
end

% creates a dcm for an angle about axis 1
function r = dcm1axis(ang)
r = [1 0 0 ; 0 cosd(ang) sind(ang) ; 0 -sind(ang) cosd(ang)];
end

% creates a dcm for an angle about axis 3
function r = dcm3axis(ang)
r = [cosd(ang) sind(ang) 0 ; -sind(ang) cosd(ang) 0 ; 0 0 1];
end
```

```
r_1 =

   1.0129e+04


r_2 =

   1.0132e+04


rECI_1 =

   1.0e+03 *

   -6.9890
    6.5318
    3.3308


vECI_1 =

   -5.7639
   -3.8460
   -0.1633
```

```
rECI_2 =

   1.0e+03 *

   -6.9984
    6.5255
    3.3305


vECI_2 =

   -4.0301
   -2.6965
   -0.1166


delta_V_Analytic =

    2.0808
```

Published with MATLAB® R2021b