# Table of Contents



```
%Roshan Jaiswal-Ferri
%Section - 01
%Aero 351 Homework 3: 11/09/24
```

# Workspace Prep

```
format long      %Allows for more accurate decimals
close all;       %Clears all
clear all;       %Clears Workspace
clc;             %Clears Command Window
```

# PART 1: 4.15

```
mu = 398600;
Rearth = 6378;

ecc = 1.5;
Rp = Rearth + 300;
inc = 35;
RAAN = 130;
w = 115; %arg of peri
theta = 0; %at perigee

a = Rp/(1-ecc); %km
h = (mu*(a*(1-ecc^2)))^(1/2); %<3:h is much higher to expect for leo,
% makes sense b/c it is hyperbolic

R1 = [(((h^2)/mu)*(1/(1+ecc*cosd(theta))))*cosd(theta);...
    (((h^2)/mu)*(1/(1+ecc*cosd(theta))))*sind(theta);0];
V = [(mu/h)*-sind(theta);(mu/h)*(ecc+cosd(theta));0];

[~,PtoE] = ECI2PERI(w,inc,RAAN);
```

```
R_eci = PtoE*R1; %<3: Magnitude of position is the same in both frames
V_eci = PtoE*V; %<3: Magnitude of velocity is the same in each frame
%<3: velocity at perigee is very high, matches the hyperbolic ecc (1.5)

disp('Problem 4.15:')
disp('Perifocal Frame Position:')
disp(['Mag (km): ', num2str(norm(R1))])
disp(['Vector (km): ', num2str(R1')])
disp('Perifocal Frame Velocity:')
disp(['Mag (km/s): ', num2str(norm(V))])
disp(['Vector (km/s): ', num2str(V')])
disp('Geocentric Frame Position:')
disp(['Mag (km): ', num2str(norm(R_eci))])
disp(['Vector (km): ', num2str(R_eci')])
disp('Geocentric Frame Velocity:')
disp(['Mag (km/s): ', num2str(norm(V_eci))])
disp(['Vector (km/s): ', num2str(V_eci')])
disp(' ')
```

# PART 2: 5.6

```
R1 = [5644, 2830, 4170];
R2 = [-2240, 7320, 4980];

%parameters for lambert w/ uv function:
tol = 1e-8;
dt = 20*60;
tm = 1; %<3: Short way around

[V1,V2] = lambUVBi(R1,R2,dt,tm,mu,tol);
%<3: f*gd - fd*g = 1

%<3:V1 is larger than V2, b/c h is conserved and the r2 is greater than R1

disp('Problem 5.6:')
disp('Starting Velocity:')
disp(['Mag (km/s): ', num2str(norm(V1))])
disp(['Vector (km/s): ', num2str(V1)])
disp('Ending Velocity:')
disp(['Mag (km/s): ', num2str(norm(V2))])
disp(['Vector (km/s): ', num2str(V2)])
disp(' ')
```

```
Problem 5.6:
Starting Velocity:
Mag (km/s): 8.333
Vector (km/s): -4.8864      6.0226      3.0479
Ending Velocity:
Mag (km/s): 7.1675
Vector (km/s): -6.9168      1.2549     -1.3988
```

# PART 3: 6.8

```
%Calculating dV
mu = 398600; %for hrt check: h between leo and meo
rpt = 6378+300; %km
rat = 6378+3000; %km

V1 = sqrt(mu/rpt); %<3: V is between 7 and 8 km/s!
V3 = sqrt(mu/rat);

ecc = (rat-rpt)/(rat+rpt); %<3 should be between 0 and 1!
ht = sqrt(rpt*mu*(1+ecc*cos(0))); %<3 check: h is between leo and meo

Vpt = ht/rpt; %<3:V perigee is faster than apogee
Vat = ht/rat;

dVp = Vpt - V1;
dVa = V3 - Vat;
dVt = dVa + dVp; %<3: delta V positive for inner to outer orbit

%Finding Orbit time
a = (rat+rpt)/2;
P = pi*sqrt((a^3)/mu);
tP = P/60; %<3 is shorter than one hour

disp('Problem 6.8:')
disp(['Required delta V (km/s): ', num2str(dVt)]);
disp(['Transfer Orbit Period (min): ', num2str(tP)]);
disp(' ')
```

```
Problem 6.8:
Required delta V (km/s): 1.1977
Transfer Orbit Period (min): 59.6542
```

# Reset Workspace Vars

```
clear all;      %Clears Workspace
mu = 398600;
Rearth = 6378;
```

# PART 4: 6.23

```
%Orbit 1:
Rp1 = 8100; %km
Ra1 = 18900; %km
theta1B = 45; %deg
theta1C = 150; %deg
ecc1 = (Ra1-Rp1)/(Ra1+Rp1);
a1 = (Rp1+Ra1)/2;
h2 = sqrt(mu*a1*(1-ecc1^2));
T1 = 2*pi*sqrt((a1^3)/mu);
```

```
Rb1 = ((h2^2)/mu)/(1+ecc1*cosd(theta1B)); %current position of s/c B
Vb1t = (mu/h2)*(1+ecc1*cosd(theta1B)); %V tangential
Vb1r = (mu/h2)*ecc1*sind(theta1B); %V radial
Vt1 = sqrt((Vb1t^2)+(Vb1r^2));
FPAb1 = atand(Vb1r/Vb1t); %flight path angle in deg

%For s/c B
E = 2*atan(sqrt((1-ecc1)/(1+ecc1))*tand(theta1B/2)); %eccentric anomaly
Me = E-ecc1*sin(E); %Mean anomaly
ts = (Me*T1)/(2*pi); %time since periapsis

%For s/c C
E2 = 2*atan(sqrt((1-ecc1)/(1+ecc1))*tand(theta1C/2)); %eccentric anomaly
Me2 = E2-ecc1*sin(E2); %Mean anomaly
ts2 = (Me2*T1)/(2*pi); %time since periapsis

%Orbit 2:
dt = T1-(ts2-ts); %<3: Transfer orbit period is less than og orbit
a2 = (dt^(2/3)*mu^(1/3))/(2^(2/3)*pi^(2/3)); %<3: semi-maj axis is smaller
Rp2 = Rb1; %treating position of s/c as periapsis
Ra2 = 2*a2-Rp2;
ecc2 = (Ra2-Rp2)/(Ra2+Rp2); %<3: is elliptical
h2 = sqrt(mu*a2*(1-ecc2^2));
Vt2 = h2/Rp2; %<3: Velocity on smaller transf orbit is less than og larger
%(at the same point)

%DeltaV EQ:
%2nd Flight path angle is 0:
dVt = 2*sqrt((Vt1^2)+(Vt2^2)-2*Vt1*Vt2*cosd(0-FPAb1));
% It looks like the answer is twice the dV equation, probably once for
% speeding up and once for slowing down

disp('Problem 6.23:')
disp(['Total delta V (km/s): ', num2str(dVt)])
disp(' ')
```

```
Problem 6.23:
Total delta V (km/s): 3.4054
```

# Reset Workspace Vars

```
clear all;      %Clears Workspace
mu = 398600;
Rearth = 6378;
```

# PART 5: 6.25

```
%Calculating Orbital Elements
Rp2 = 1270 + Rearth; %km
Vp = 9; %km/s
theta2 = 100;
```

```
h2 = Vp*Rp2;
ecc2 = (h2^2)/(mu*Rp2)-1;
ecct = .4; %target ecc

%Calculating R & V
R1m = (h2^2/mu)*(1/(1+ecc2*cosd(100)));
V1t = (h2/R1m);
V1r = (mu/h2)*ecc2*sind(100);
FPA1 = atand(V1r/V1t);
V1m = sqrt(V1r^2+V1t^2);
hf = sqrt(R1m*mu*(1+ecct*cosd(100)));
V2t = hf/R1m;
V2r = (mu/hf)*ecct*sind(100);
V2m = sqrt(V2r^2+V2t^2); %<3:V2 smaller than V1 (orbit is becoming smaller)
FPA2 =atand(V2r/V2t); %<3: Smaller than the first FPA

%Delta V EQ:
dVt = sqrt((V1m^2)+(V2m^2)-2*V1m*V2m*cosd(FPA2-FPA1));
%<3: Delta gamma (FPA) is negative, pointing more towards the earth (ecc is
%dropping)

disp('Problem 6.25:')
disp(['Total Delta V (km/s): ', num2str(dVt)])
disp(['Flight Path Angle (deg): ', num2str((FPA2-FPA1))])
disp(' ')
```

*Problem 6.25:*
*Total Delta V (km/s): 0.91545*
*Flight Path Angle (deg): -8.1813*

# Reset Workspace Vars

```
clear all;      %Clears Workspace
mu = 398600;
Rearth = 6378;
```

# PART 6: 6.44

```
Rpa = 6678; %Radius of parking orbit
Rt = 6978; %Target radius
dinc = 20; %change in inc (deg)

Vpa = sqrt(mu/Rpa); %vel parking orbit
ecch = (Rt-Rpa)/(Rt+Rpa); %ecc of hohman T orbit
ht = sqrt(Rpa*mu*(1+ecch)); %h of ht orbit
Vt1 = ht/Rpa; %total velocity onto/in transfer orbit
Vt2 = ht/Rt; %total velocity off of transfer orbit

Vb1 = abs(Vt1-Vpa); %delta V from burn one
Vt = sqrt(mu/Rt); %target velocity
Vb2 = abs(Vt-Vt2); %delta V from burn two
```

```
dVi = 2*Vt*sind(dinc/2); %delta V to change inclination
dVt = Vb1+Vb2+dVi; %<3: Most expensive b/c of two independant burns

%Simultaneous Plane change & insertion
dVir = sqrt((Vt-Vt2)^2 + 4*Vt2*Vt*sind(dinc/2)^2);
dVirt = dVir+Vb1; %<3: least expensive b/c of combined burns
%(inc change plus transf)

%Plane Change Departing
dVl = sqrt((Vt1-Vpa)^2 + 4*Vt1*Vpa*sind(dinc/2)^2);
dVlt = dVl+Vb2;

%<3: All Delta Vs are close together

disp('Problem 6.44')
disp(['Part A Delta V (km/s): ', num2str(dVt)])
disp(['Part B Delta V (km/s): ', num2str(dVirt)])
disp(['Part C Delta V (km/s): ', num2str(dVlt)])
disp(' ')
```

```
Problem 6.44
Part A Delta V (km/s): 2.7927
Part B Delta V (km/s): 2.696
Part C Delta V (km/s): 2.7826
```

# Reset Workspace Vars

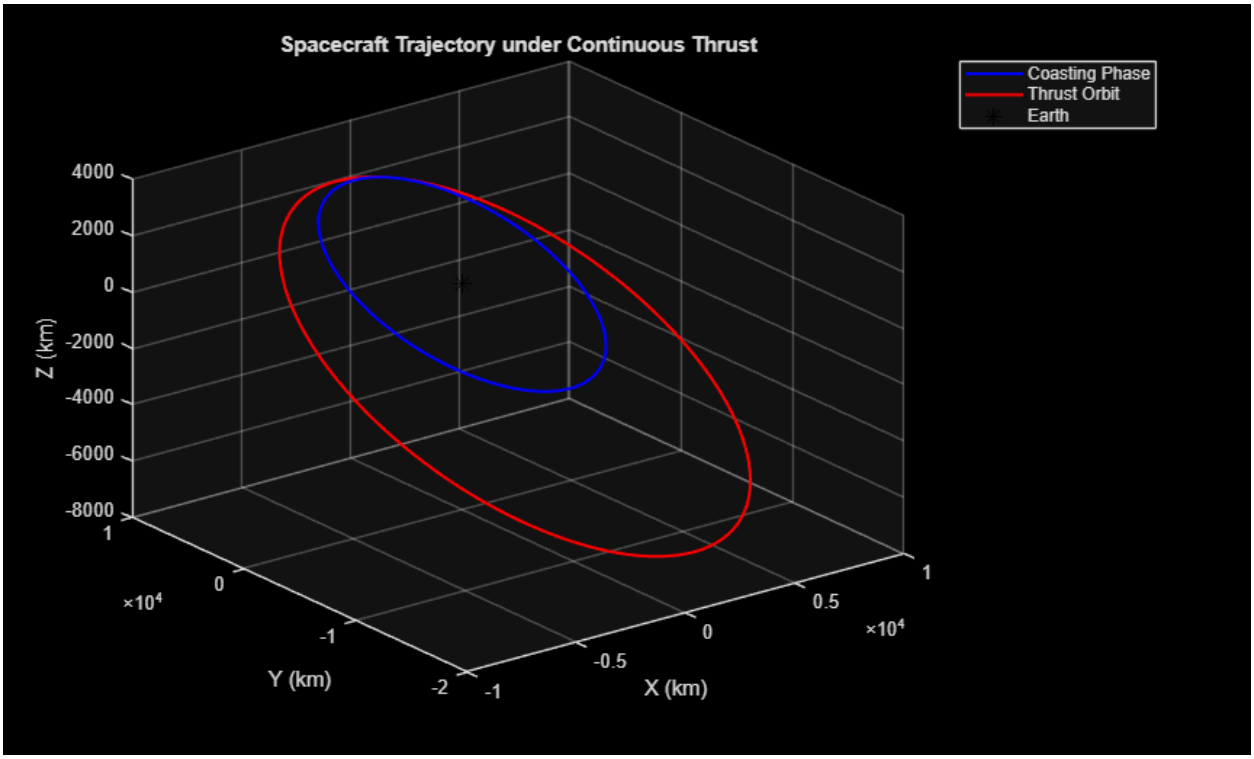
```
clear all;      %Clears Workspace
mu = 398600;
Rearth = 6378;
options = odeset('RelTol',1e-8,'AbsTol',1e-8);
```

# PART 7: 6.47

```
m = 1000; %kg
g = 9.807e-3; % gravity in km/s
R = [436, 6083, 2529]; %km
V = [-7.340, -0.5125, 2.497]; %km/s
[~,~,~,~,~,~,~,p1,~,~,Ra1] = rv2coes(R, V, mu, Rearth);

Isp = 300; %s
T = 10; % kN Thrust
tspan = [0, 89*60] ;%s
Tt = [89*60, 89*60+120];%Thrust time ins

state = [R, V];
[~,stateNew] = ode45(@twobodymotion2,tspan,state,options,mu);

R1 = [stateNew(end,1), stateNew(end,2), stateNew(end,3)];
V1 = [stateNew(end,4), stateNew(end,5), stateNew(end,6)];

state2 = [R1, V1, m]; %Beginning of burn
```

```
[~,state3] = ode45(@contThrust,Tt,state2,options, T, Isp, mu, g);

R2 = [state3(end,1), state3(end,2), state3(end,3)];
V2 = [state3(end,4), state3(end,5), state3(end,6)];
%mf = state3(end, 7); %final mass not needed but for later reference
[~,~,~,~,~,~,~,p,t,~,Ra] = rv2coes(R2, V2, mu, Rearth);
%<3: Apogee of 2nd orbit is greater than starting

ToA = ((p/2)-t)+120+(89*60); %time of apogee
ToA = ToA/3600; %seconds to hours
%<3: period of final orbit is longer than initial

R1n = norm(R); %<3: R vector is larger in final orbit
R2n = norm(R2);

disp('Problem 6.47')
disp(['Highest Altitude (km): ', num2str(Ra)])
disp(['Time of Altitude (hrs): ', num2str(ToA)])
disp(' ')

% Plot the trajectory
tspan = [0, 6*60*60];
state2 = [R2, V2];
[~,stateNew4] = ode45(@twobodymotion2,tspan,state2,options,mu);

figure('Name', 'Continuous Thrust Trajectory');
plot3(stateNew(:, 1), stateNew(:, 2), stateNew(:, 3), 'b', 'LineWidth',
1.5); % Coasting phase
hold on;
plot3(stateNew4(:, 1), stateNew4(:, 2), stateNew4(:, 3), 'r', 'LineWidth',
1.5); % Thrust phase
plot3(0, 0, 0, 'k*', 'MarkerSize', 10); % Earth at the origin
xlabel('X (km)');
ylabel('Y (km)');
zlabel('Z (km)');
grid on;
legend('Coasting Phase', 'Thrust Orbit', 'Earth');
title('Spacecraft Trajectory under Continuous Thrust');
hold off;
```

```
Problem 6.47
Highest Altitude (km): 10433.1114
Time of Altitude (hrs): 3.252
```




# Functions:

```
function [EtoP, PtoE] = ECI2PERI(omega,inc,RAAN)
    EtoP = inv(rotz(omega))*inv(rotx(inc))*inv(rotz(RAAN));
    PtoE = inv(EtoP);
end

function [V1,V2] = lambUVBi(R1,R2,dtime,Tm,mu,tol)
    R1n = norm(R1);
    R2n = norm(R2);
    Z = 0;
    C = 1/2;
    S = 1/6;
    Zu = 4*pi^2; %upper bound
    Zl = -4*pi^2; %lower bound
    dtl = 1; %change in time of loop (random guess)
    deltaTheta = acos((dot(R1,R2)/(R1n*R2n)));
    A = sin(deltaTheta)*sqrt((R1n*R2n)/(1-cos(deltaTheta)));

    while abs(dtl-dtime) > tol
        Y = R1n+R2n+(A*((Z*S-1)/sqrt(C)));
        UV = sqrt(Y/C);
        dtl = (((UV^3)*S)/sqrt(mu))+((A*sqrt(Y))/sqrt(mu));

        if dtl < dtime
            Zl = Z; %reset zlower
        elseif dtl > dtime
            Zu = Z; %reset zupper
```

```
        end

        Z = 0.5*(Zu+Zl); %update z to midpoint
        [C,S] = stumpff(Z); %update stumpff c(z) s(z)

        f = 1-(((UV^2)/R1n)*C);
        g = dtl - (((UV^3)/sqrt(mu)) * S);
        fd = (sqrt(mu)/(R1n*R2n))*UV*((Z*S)-1);
        gd = 1-(((UV^2)/R2n)*C);
        %<3: f*gd - fd*g = 1

        %g = (1/mu)*(((Y/C)^(3/2))*S+(A*sqrt(Y)))-(1/mu)*((Y/C)^(3/2))*S;
        %g = A*sqrt(Y/mu);

        for i = 1:3
            V1(i) = (1/g)*(R2(i)-f*R1(i));
            V2(i) = (fd*R1(i))+(gd*V1(i));
        end

    end
end

function [C, S] = stumpff(z)
    if z > 0
        S = (sqrt(z)-sin(sqrt(z)))/((sqrt(z))^3);
        C = (1-cos(sqrt(z)))/z;
    elseif z < 0
        S = (sinh(sqrt(-z))-sqrt(-z))/((sqrt(-z))^-3);
        C = (cosh(sqrt(-z))-1)/-z;
    elseif z == 0
        S = 1/6;
        C = 1/2;
    else
        error('stumpff broke? (not a number?)')
    end
end

function tstate = contThrust(time, state, T, Isp, mu, g0)

    x = state(1);
    y = state(2);
    z = state(3);
    dx = state(4);
    dy = state(5);
    dz = state(6);
    m = state(7);
    rMag = norm([x y z]);
    vMag = norm([dx dy dz]);
    ddx = -mu*x/rMag^3 +(T/m)*dx/vMag;
    ddy = -mu*y/rMag^3 +(T/m)*dy/vMag;
    ddz = -mu*z/rMag^3 +(T/m)*dz/vMag;
    mdot = -T/(g0*Isp);
```

```
    tstate = [dx;dy;dz;ddx;ddy;ddz;mdot];

end

function dstate = twobodymotion2(time,state,muEarth) %dstate is derivitve of
state

    %define vars

    x = state(1);
    y = state(2);
    z = state(3);

    dx = state(4); %vel
    dy = state(5); %vel
    dz = state(6); %vel


    r = norm([x y z]);

    ddx = -muEarth*x/r^3;
    ddy = -muEarth*y/r^3;
    ddz = -muEarth*z/r^3;
    dstate = [dx; dy; dz; ddx; ddy; ddz];
end
```

```
Problem 4.15:
Perifocal Frame Position:
Mag (km): 6678
Vector (km): 6678              0              0
Perifocal Frame Velocity:
Mag (km/s): 12.2156
Vector (km/s): 0      12.2156            0
Geocentric Frame Position:
Mag (km): 6678
Vector (km): -1983.7706       -5348.76      3471.4701
Geocentric Frame Velocity:
Mag (km/s): 12.2156
Vector (km/s): 10.3559     -5.76267     -2.96111
```

*Published with MATLAB® R2024b*