Table of Contents

Roshan Jaiswal-Ferri

```
%Section - 01
%Aero 351 Final Exam Question 3: 12/07/24
```

Workspace Prep

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

Constant/Global Vars

```
options = odeset('RelTol',1e-8,'AbsTol',1e-8);
muSun = 1.327e11; %mu values from curtis
muMars = 42828;
mu = 398600; %earth
muJup = 126686534;
Rmars = 3396; %km
Rearth = 6378; %km
Rjup = 71490;
tol = 1e-7;
T = -0.006; %thrust in kN, ISP & Thrust negative because we are slowing down
Isp = -5000; % (ISP = F/(mdot*g0) ) so if F (thrust) is neg Isp is too
g = 9.807e-3; % gravity in km/s
mi = 600; %initial mass 600 kilos
```

Defining State Vectors & Propogation of 3.5 Day Burn

```
r1 = [26578,0,0];
v1 = [0,3.8726,0];
```

```
state = [r1,v1,mi];
tspan1 = [0,3.5*86400]; %tspan of burn one for 3.5 days in seconds
[~,burn1] = ode45(@contThrust,tspan1,state,options, T, Isp, mu, g);
```

Coast Phase - 5 Days

```
r2 = [burn1(end,1),burn1(end,2),burn1(end,3)];
v2 = [burn1(end,4),burn1(end,5),burn1(end,6)];
m2 = burn1(end,7); %new mass after burning fuel
state2 = [r2,v2];
tspan2 = [0,5*86400]; %five days in seconds
[~,coast] = ode45(@twobodymotion,tspan2,state2,options,mu);
```

Burn 2 - 0.6 Days

```
r3 = [coast(end,1),coast(end,2),coast(end,3)];
v3 = [coast(end,4),coast(end,5),coast(end,6)];
state3 = [r3,v3,m2];
tspan3 = [0,86400*0.6]; %0.6 days in seconds
[~,burn2] = ode45(@contThrust,tspan3,state3,options, T, Isp, mu, g);
```

Finding New Coes

```
rf = [burn2(end,1),burn2(end,2),burn2(end,3)];
vf = [burn2(end,4),burn2(end,5),burn2(end,6)];
mf = burn2(end,7);
[~,~,~,~,~,~,~,~,~,~,~,Altp] = rv2coes(rf,vf,mu,Rearth);
```

Display Results

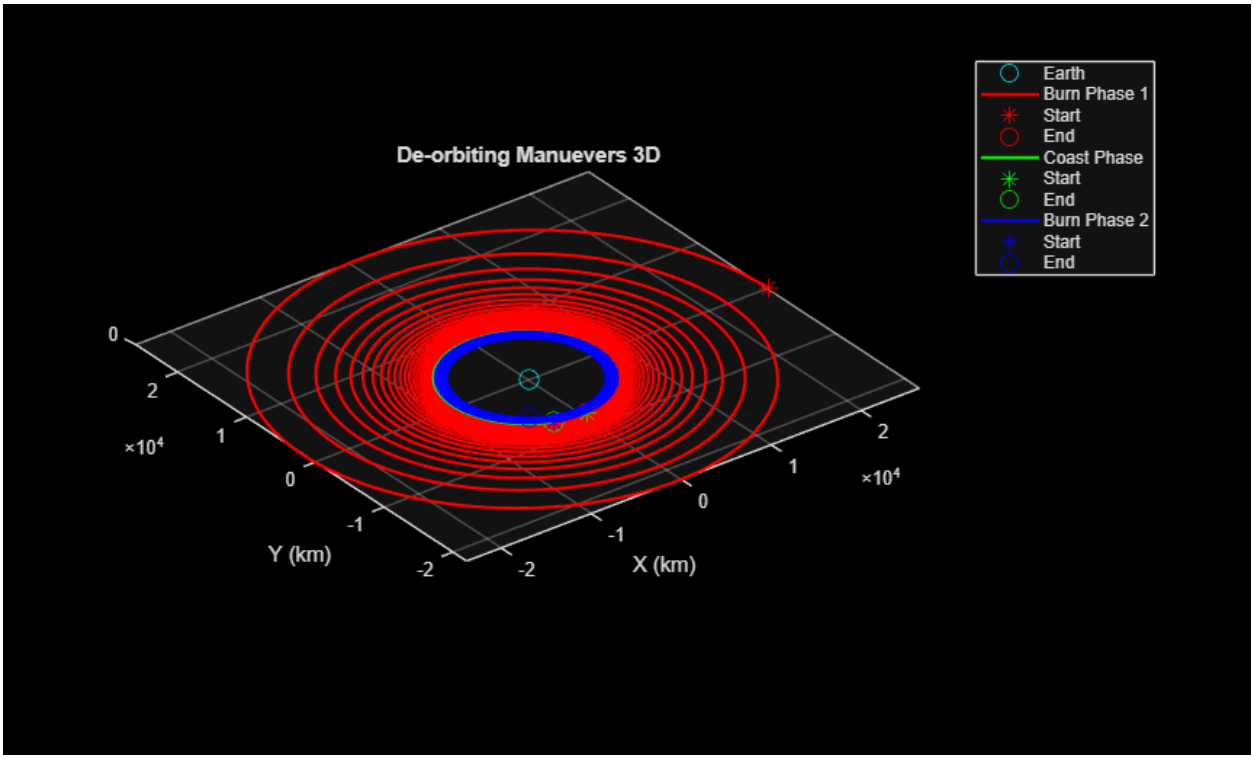
```
disp(['Final Mass (kg): ', num2str(mf)])
disp(['Altitude of Perigee (km): ', num2str(Altp)])
```

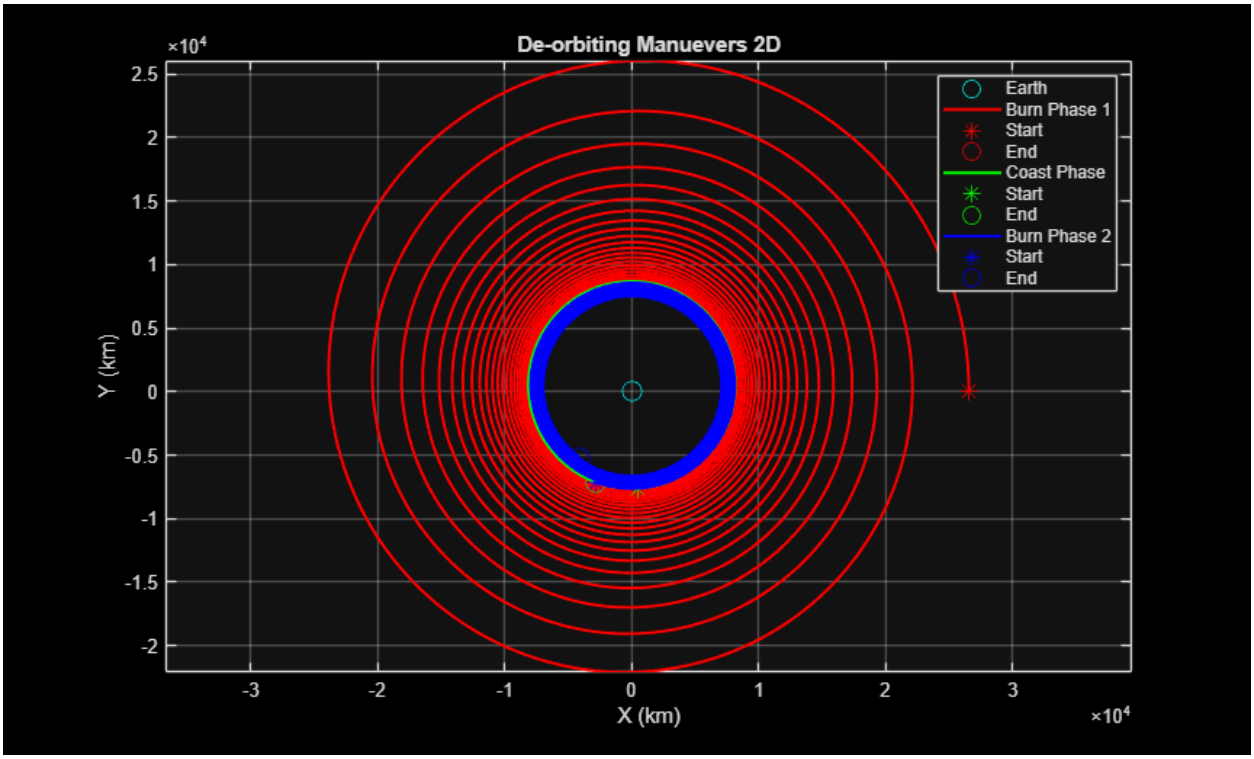
```
Final Mass (kg): 556.6546
Altitude of Perigee (km): 143.9152
```

Plotting

```
figure('Name', 'De-orbiting Manuevers 3D');
plot3(0,0,0,'co', 'MarkerSize',10); %Earth
hold on;

plot3(burn1(:, 1), burn1(:, 2), burn1(:, 3), 'r', 'LineWidth', 1.5); %Thr 1
plot3(burn1(1,1),burn1(1,2),burn1(1,3),'r*', 'MarkerSize',10); %start thr 1
plot3(burn1(end,1),burn1(end,2),burn1(end,3),'ro', 'MarkerSize',10); %end

plot3(coast(:, 1), coast(:, 2), coast(:, 3), 'g', 'LineWidth', 1.5); %coast
plot3(coast(1,1),coast(1,2),coast(1,3),'g*', 'MarkerSize',10); %start
plot3(coast(end,1),coast(end,2),coast(end,3),'go', 'MarkerSize',10); %end
```

```
plot3(burn2(:, 1), burn2(:, 2), burn2(:, 3), 'b', 'LineWidth', 1.5); %Thr 2
plot3(burn2(1,1),burn2(1,2),burn2(1,3),'b*', 'MarkerSize',10); %start
plot3(burn2(end,1),burn2(end,2),burn2(end,3),'bo', 'MarkerSize',10); %end

xlabel('X (km)');
ylabel('Y (km)');
zlabel('Z (km)');
grid on;
legend('Earth',...
    'Burn Phase 1','Start','End',...
    'Coast Phase','Start','End',...
    'Burn Phase 2','Start','End')
title('De-orbiting Manuevers 3D');
axis equal

figure('Name', 'De-orbiting Manuevers 2D');
plot(0,0,'co', 'MarkerSize',10); %Earth
hold on;

plot(burn1(:, 1), burn1(:, 2), 'r', 'LineWidth', 1.5); %Thr 1
plot(burn1(1,1),burn1(1,2),'r*', 'MarkerSize',10); %start thr 1
plot(burn1(end,1),burn1(end,2),'ro', 'MarkerSize',10); %end

plot(coast(:, 1), coast(:, 2), 'g', 'LineWidth', 1.5); %coast
plot(coast(1,1),coast(1,2),'g*', 'MarkerSize',10); %start
plot(coast(end,1),coast(end,2),'go', 'MarkerSize',10); %end

plot(burn2(:, 1), burn2(:, 2), 'b', 'LineWidth', 1.5); %Thr 2
plot(burn2(1,1),burn2(1,2),'b*', 'MarkerSize',10); %start
plot(burn2(end,1),burn2(end,2),'bo', 'MarkerSize',10); %end

xlabel('X (km)');
ylabel('Y (km)');
zlabel('Z (km)');
grid on;
legend('Earth',...
    'Burn Phase 1','Start','End',...
    'Coast Phase','Start','End',...
    'Burn Phase 2','Start','End')
title('De-orbiting Manuevers 2D');
axis equal
```


De-orbiting Manuevers 3D
Earth
Burn Phase 1
Start
End
Coast Phase
Start
End
Burn Phase 2
Start
End
×10^4
Y (km)
X (km)
×10^4


De-orbiting Manuevers 2D
×10^4
Earth
Burn Phase 1
Start
End
Coast Phase
Start
End
Burn Phase 2
Start
End
Y (km)
X (km)
×10^4

Functions:

Continuous Thrust

```
function [tstate] = contThrust(time, state, T, Isp, mu, g0)
    %CONTTHRUST: Propogates a new orbit trajectory during thrusting phase
    %   [tstate] = contThrust(time, state, T, Isp, mu, g0)
    %
    %   Make sure g0 (usually 9.807 m/s^2) is in km/s^2

    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
```

twobodymotion

```
function dstate = twobodymotion(time,state,muEarth) %dstate is derivitve of
state
%FUNCTION put in descrip

    %define vars
    x = state(1);
    y = state(2);
    z = state(3);
    dx = state(4); %vel
    dy = state(5); %vel
    dz = state(6); %vel

    %mag of pos vector
    r = norm([x y z]);

    %accel: !!eqs of motion!!
    ddx = -muEarth*x/r^3;
    ddy = -muEarth*y/r^3;
    ddz = -muEarth*z/r^3;
```

```
    dstate = [dx; dy; dz; ddx; ddy; ddz];

end
```

rv2coes

```
function [hM,a,e,nu,i,RAAN,w,p,t,en,Alta,Altp] = rv2coes(R,V,mu,r)
%Function for finding orbital state vectors RV
%   Input is in SI & %ALL ANGLES IN RADIANS!!
%   [hM,a,e,nu,i,RAAN,w,p,t,en,Ra,Rp] = rv2coes(R,V,mu,r)
%   hM = specific angular momentum
%   a = semi-major axis
%   e = eccentricity
%   nu = true anamoly
%   i = inc
%   RAAN = Right angle asending node
%   w = argument of periapsis
%   p = period (s)
%   t = time since perigee passage
%   en = orbit energy
%   Ra = Radius of Apogee
%   Rp = Radius of Perigee
%   r = radius of orbiting planet


RM = norm(R); %Magnitude of R
VM = norm(V); %Magnitude of V

ui = [1,0,0];
uj = [0,1,0];
uk = [0,0,1];
h = cross(R,V);
h2 = dot(R,V);

uiM = norm(ui); %the magnitudes of the values above
ujM = norm(uj);
ukM = norm(uk);
hM = norm(h); %Calculating specific energy


% PART 1: Initial Calculations for later

ep = ((VM^2)/2)-((mu)/RM); %Calculating Epsilon (specific mechanical energy)
in J/kg


% PART 2: Calculating semi-major axis

a = -((mu)/(2*ep)); %in km

% PART 3: Genreal equation calculation for period

p = (2*pi)*sqrt((a^3)/(mu)); %period of orbit in seconds (ellipse & circ)
```

```
% PART 4: Calculating eccentricity
eV = (1/mu)*((((VM^2)-((mu)/(RM)))*R)-(dot(R,V)*V)); %eccentricity vector is
from origin to point of periapsis

e = norm(eV);

% PART 5: inclination in rad

i = acos((dot(uk,h))/((hM)*(ukM))); %in rad not deg


% PART 6: RAAN in rad

n = cross(uk,h); %projection of momentum vector in orbital plane and node
line?
nM = norm(n);

if n(2) >= 0
    RAAN = acos((dot(ui,n))/((uiM)*(nM))); %original equation
else
    RAAN = (2*pi)-(acos((dot(ui,n))/((uiM)*(nM))));
end

% PART 7: Argument of Periapsis in rad

if eV(3) >= 0 %k component of eccentricity vector (height)
    w = acos(dot(n,eV)/(nM*e));
else
    w = (2*pi)-(acos(dot(n,eV)/(nM*e)));
end

% PART 8: nu (or theta) true anomaly in rad

if h2 >= 0 %dot product of R and V idk what it represents
    nu = acos(dot(eV,R)/(e*RM));
else
    nu = (2*pi)-(acos(dot(eV,R)/(e*RM)));
end

% PART 9: Time since perigee passage

E = 2*atan(sqrt((1-e)/(1+e))*tan(nu/2));
Me = E - e*sin(E);
n = (2*pi)/p;
t = Me/n; %in seconds

if t < 0 %If it is negative it is other way around circle think 360-angle
    t = p + t; %this shows adding but it is adding a negative
end

% PART 10: Calculating Energy

energy = (VM^2)/2 - mu/RM; %km^2/s^2
```

```
en = energy;

% PART 11: Calculating Apogee and Perigee Altitude

Alta = a*(1+e)-r;
Altp = a*(1-e)-r;


end
```

Published with MATLAB® R2024b