Table of Contents

Roshan Jaiswal-Ferri

```
%Section - 01
%Aero 351 Final Exam Question 4: 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
Rmars = 3396; %km
Rearth = 6378; %km
tol = 1e-7;
Rpark = Rearth + 200;
```

Creating R & V Vectors

```
Vpark = sqrt(mu/Rpark);
Vesc = sqrt(2)*Vpark;
R = [Rpark,0,0];
V = [0,Vpark,0] + [0,1.075,0]; %account for first burn
```

Counting Burns

```
time = 0;
burns = 1; %add one extra for first burn

while V < Vesc
    [~,~,~,~,~,~,~,p] = rv2coes(R,V,mu,Rearth);
```

```
    time = time + p;
    V = V + [0,1.075,0];
    burns = burns + 1;
end
```

Display Results

```
disp(['Parking Velocity (km/s): ', num2str(Vpark)])
disp(['Escape Velocity (km/s): ', num2str(Vesc)])
disp(['Num of Perigee Burns: ', num2str(burns)])
disp(['Final Velocity at Perigee (km/s): ', num2str(norm(V))])
disp(['Total Time (hrs): ', num2str(time/3600)])
```

```
Parking Velocity (km/s): 7.7843
Escape Velocity (km/s): 11.0087
Num of Perigee Burns: 3
Final Velocity at Perigee (km/s): 11.0093
Total Time (hrs): 9.011
```

Functions:

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