AERO 452: Spaceflight Dynamics II

HW 4

Justin Self

December 2, 2023

1 Handwork

2 MATLAB Results

3 MATLAB Script

4 MATLAB Functions

1: Work by hand

AERO 452 : HW # 4

DUE: 12/3/2023 : 22:59 PST.

JUSTIN SELF
FALL 2023

1. Calculate entry and exit eclipse times for a satellite for two week (August 10 to August 23, 1997) given the following information:

   Rvect = -26175.1034 Ihat + 12757.0706 Jhat +14626.6556 Khat (km)
   Vvect = 2.376641 Ihat + 0.139677 Jhat + 2.078097 Khat (km/s)

   What is the percentage increase for the solar-radiation pressure if we ignore the periods when the satellite is in the Earth's shadow?

ASSUME THESE ARE THE STATE VECTORS AT AUG. 10, 1997 AT 00:00:00 UTC.

SEE MATLAB.

BUT THE PRIMARY EQUATION USED FOR THE PERTURBATION ACCELERATION DUE SRP IS:

$$\vec{a}_{SRP} = \frac{-\nu P_{SR} C_R A_{S/C}}{m_{S/C}}$$

PART 1. ECLIPSE TIMES.

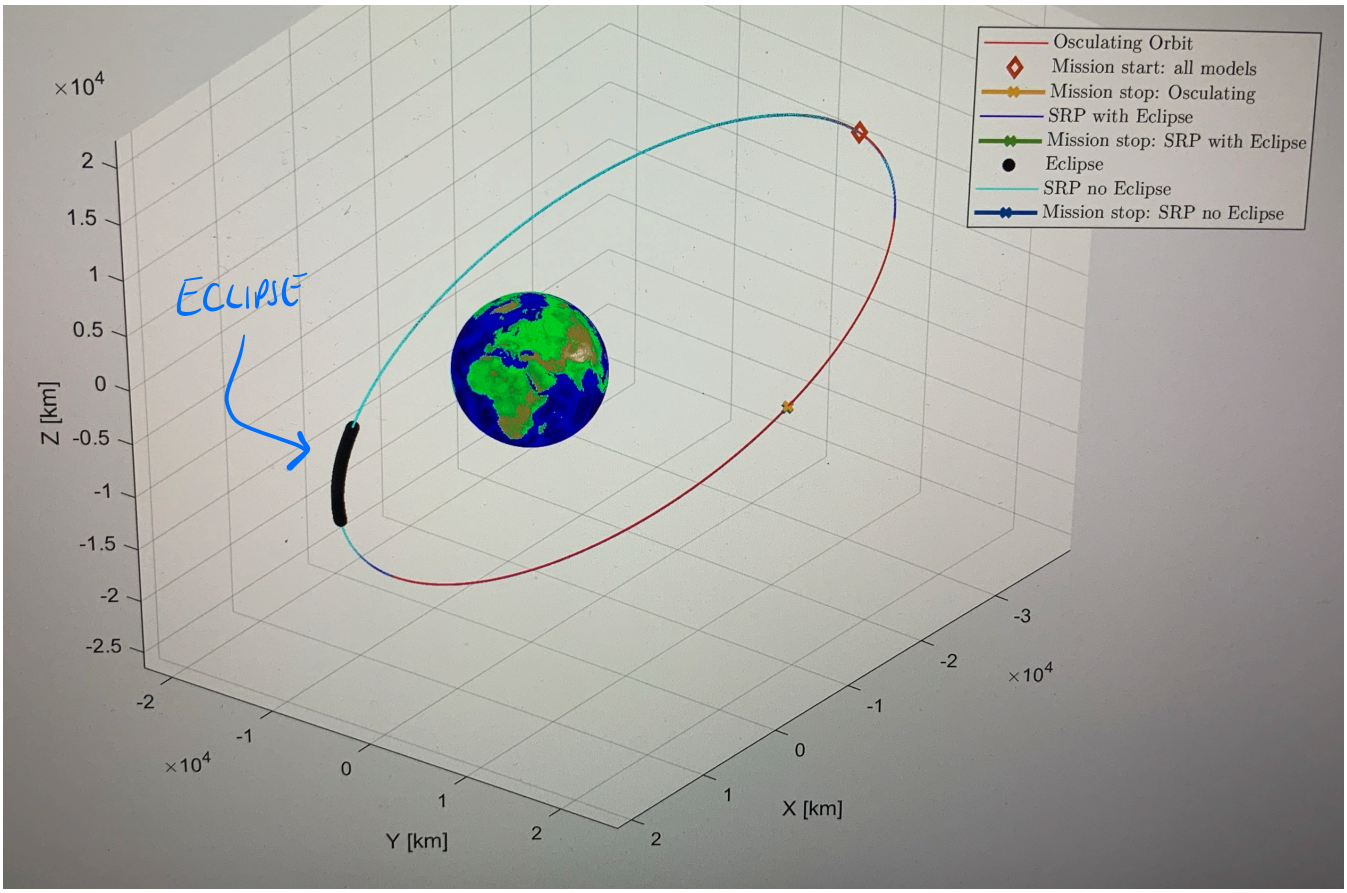


FIGURE 1. ECI FRAME OF OSCULATING AND SRP ORBITS.

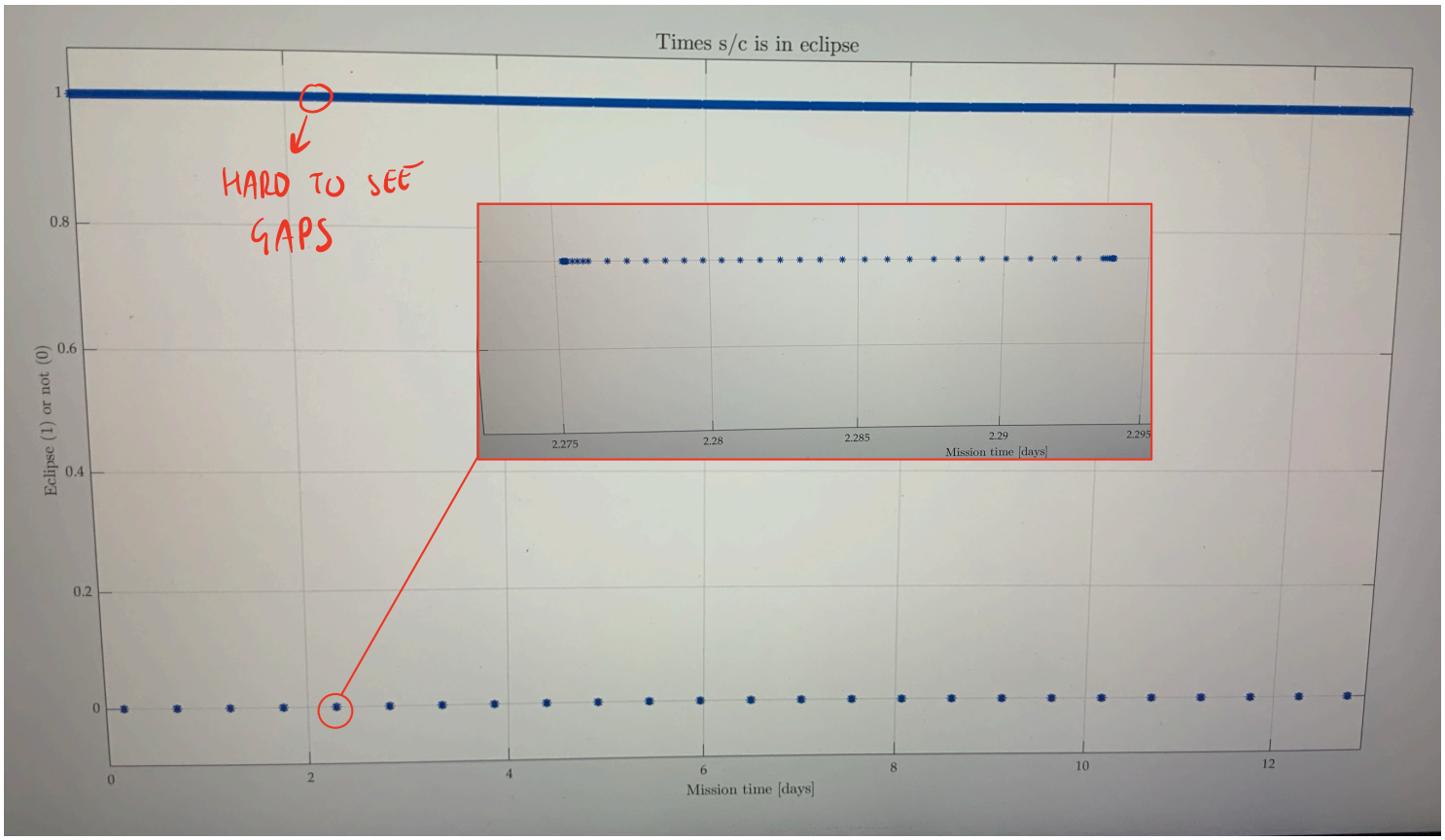


FIGURE 2. SIMPLE REPRESENTATION OF TIMES THE SPACECRAFT IS IN ECLIPSE. INSET SHOWS HIGHER DETAIL OF EACH EVENT.

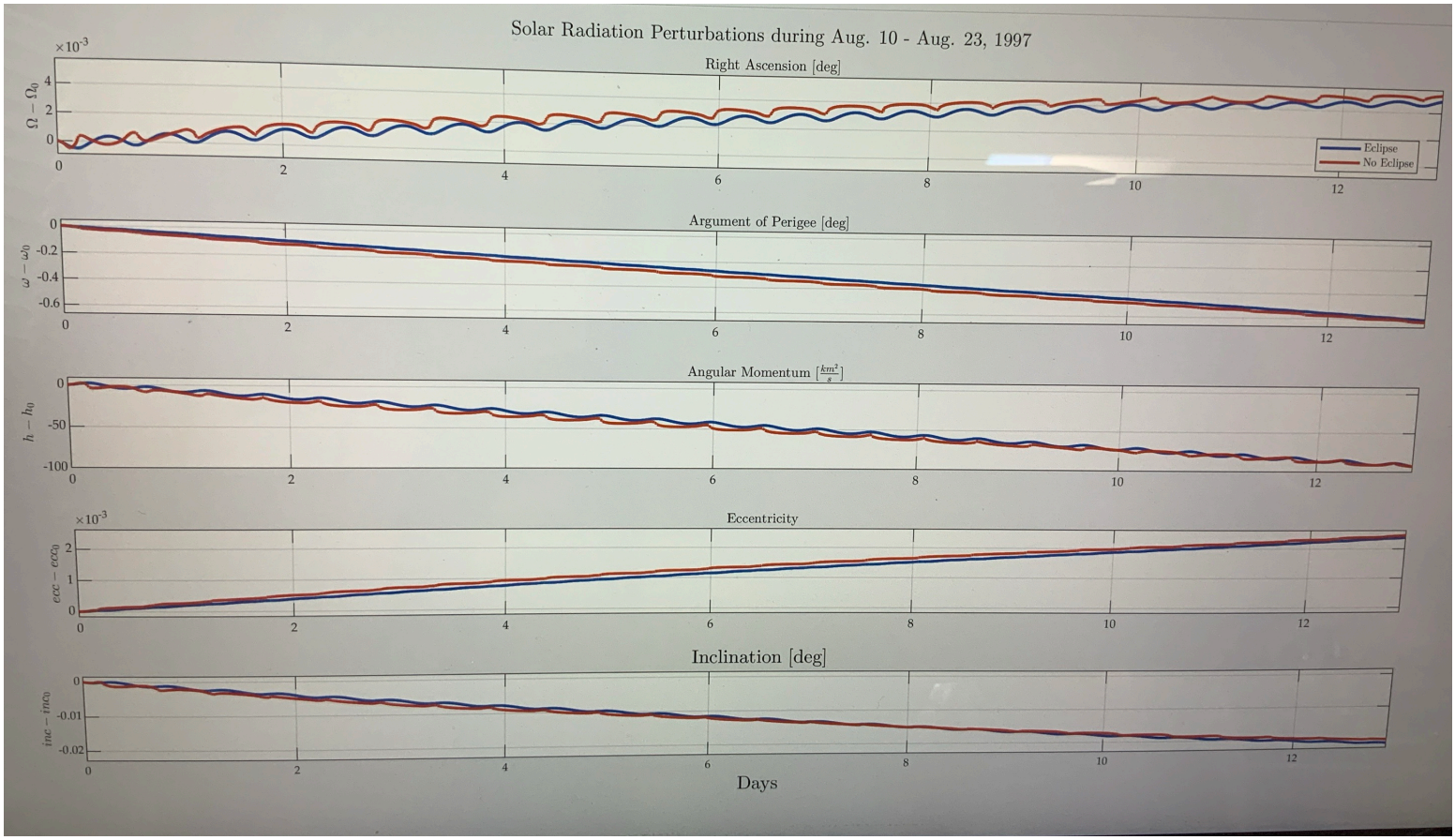
* FOR EXACT DATES/TIMES OF ECLIPSE, SEE ATTACHED PDF "ECLIPSE TIMES". (APPENDIX B: AFTER CODE).

PART 2. PERCENT INCREASE IN SRP WHEN WE IGNORE ECLIPSE.

```
Time s/c is in eclipse: 0.55242 days
Percent of time s/c is in eclipse (of 13 day mission): 4.2494 percent
fx >>
```

PLOT OF SRP EFFECTS ON COES.

(NEXT PAGE)



SRP ~1 AND ~10 ECLIPSE.

2. Calculate the effect of solar gravity (as in n-body) over 60 days on an object with the following parameters at t0 (see example ~~12~~ 10.12 in Curtis):

h = 69,084.1 km2/s;
ecc0 = 0.741;
raan0 = 0 degs
inc0 = 63.4 degs
omega0 = 270 degs
theta0 = 0 degs
a0 = 26,553.4 km
T0 = 11.9616 hours

SEE → MATLAB

PLOTS ↓

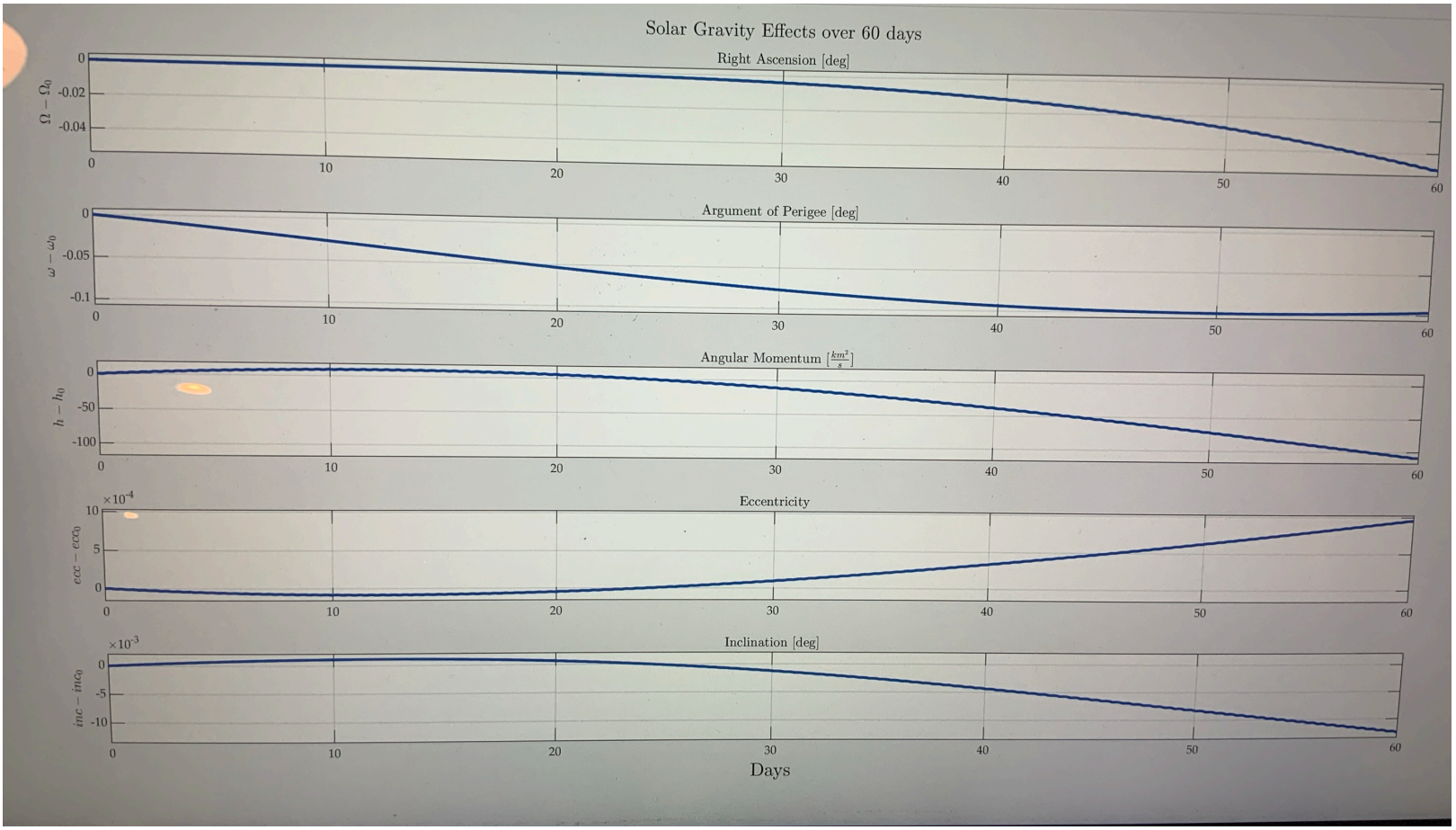


FIGURE 3. SOLAR GRAVITY (~800g) EFFECTS ON S/C ONLY.

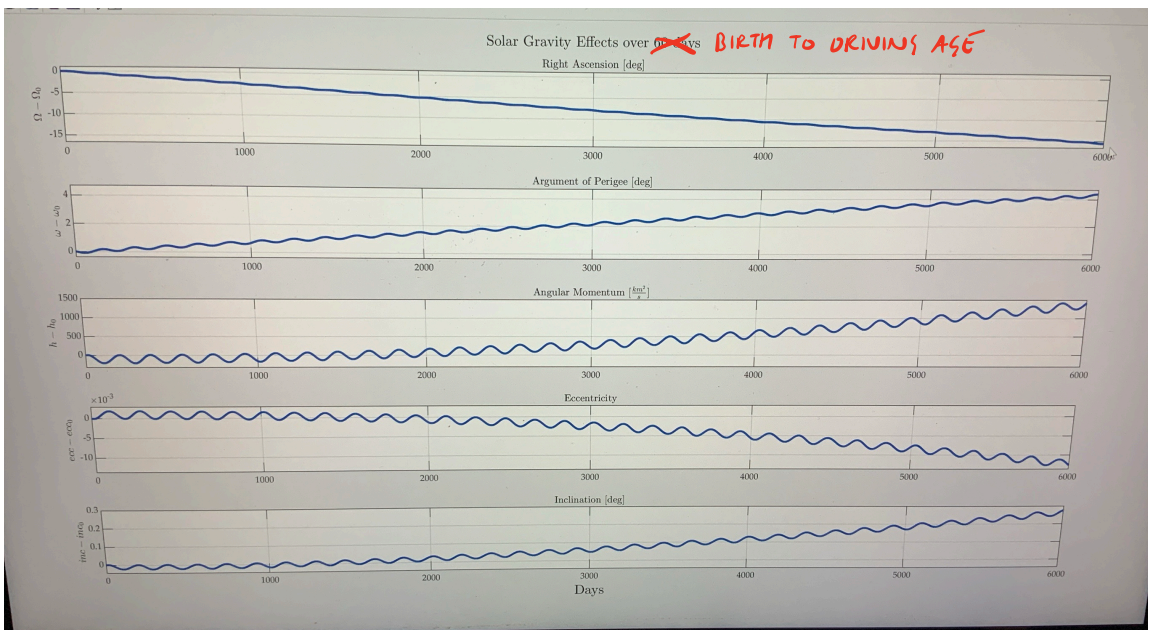
♡ CHECKS.

i. ecc, RAAN, inc : LONG PERIODIC.

= CAN'T TELL FOR 60 DAYS.
BUT AT 6000 DAYS (16 YEARS!)

* PERIODIC YET SECULAR

FINE →



3. Simulate the trajectory of a spacecraft passing through the following points and speed. Notice that this system is dimensionless. The time steps are set where 1 is one complete revolution of the primaries, which is about 30 days. Show the location of the earth, moon, and libration points where you believe are necessary. Plots for c-e can be on one plot and f and g on one plot. Discuss why cases c-e are called "free-return trajectories". Discuss the differences between the cases and when you might want your spacecraft to use the state vectors.

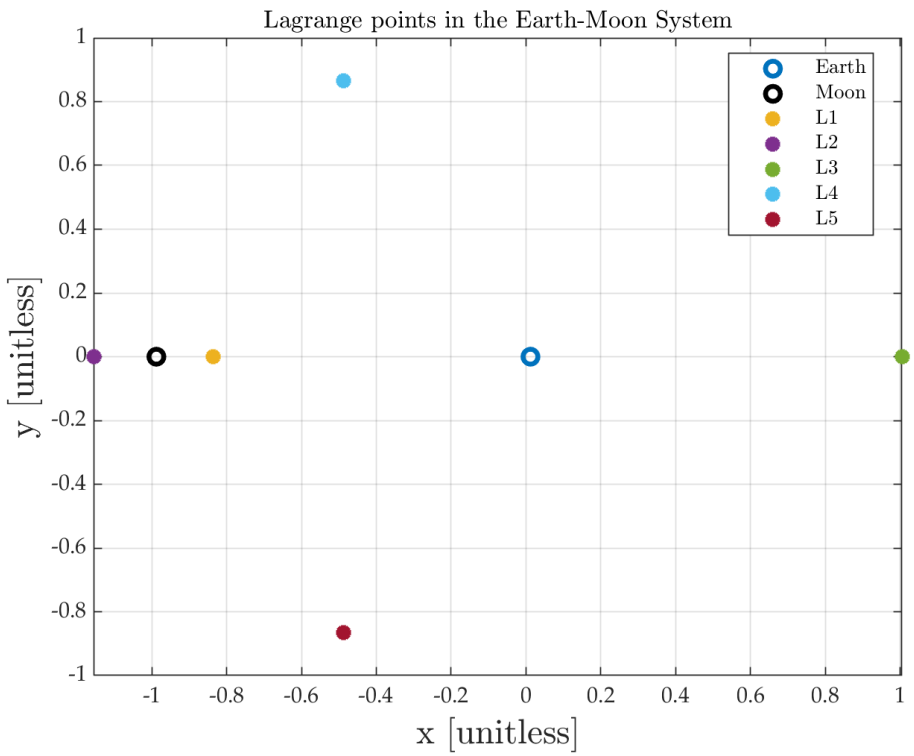
I PLOTTED THEM SEPARATE

a) x = -1.05, y = 0.0, vx = 0, vy = -0.066429; t=2*pi
b) x = -1.15, y = 0.0, vx = 0.0, vy = -0.0086882909; t=29.46
c) x = 0.10, y = 0, vx = -3.35, vy = 3; t=3.6
d) x = 0.10, y = 0, vx = -3.37; vy = 3; t=6
e) x = 0.10, y = 0, vx = -3.4; vy = 3; t=6
f) x = 1.25, y = 0, vx = 0, vy =0; t=2*pi
g) x = -0.5, y = 0, vx = 0, vy =0; t=2*pi
h) x = -1.10, y = 0, vx = 0, vy =0; t=2*pi

i) FIND AND PLOT LIBRATION POINTS ✓
ii) PLOT ↑ AND ⊕ AND ☾ ✓
iii) PLOT S/C TRAJECTORIES a) → h) ✓

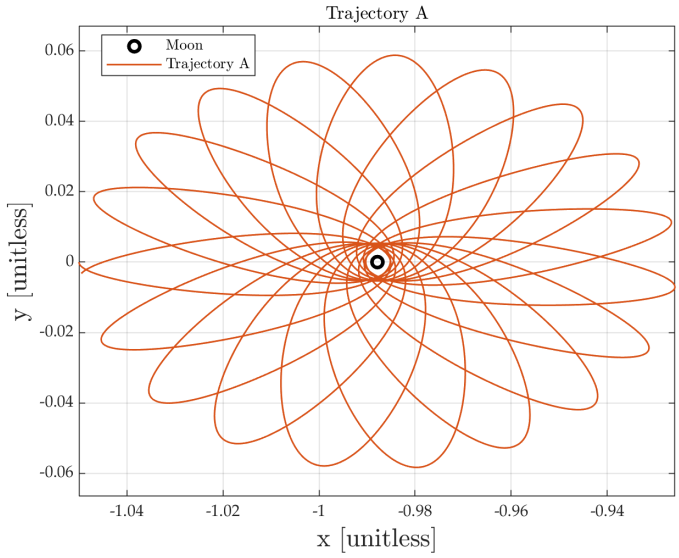
SEE MATLAB FOR WORK.

PLOTS AND DISCUSSION BELOW.



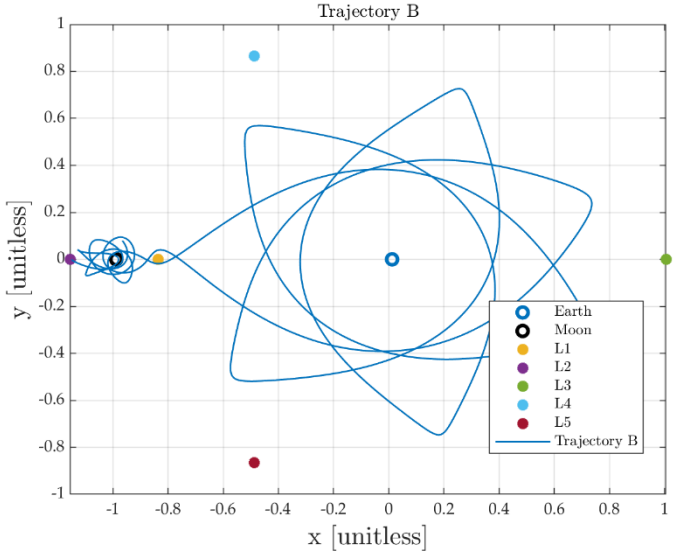
LEFT-HANDED ORIENTATION

DISCUSSION: *Differences between cases and potential usefulness of the trajectories / state vectors for spacecraft missions.*



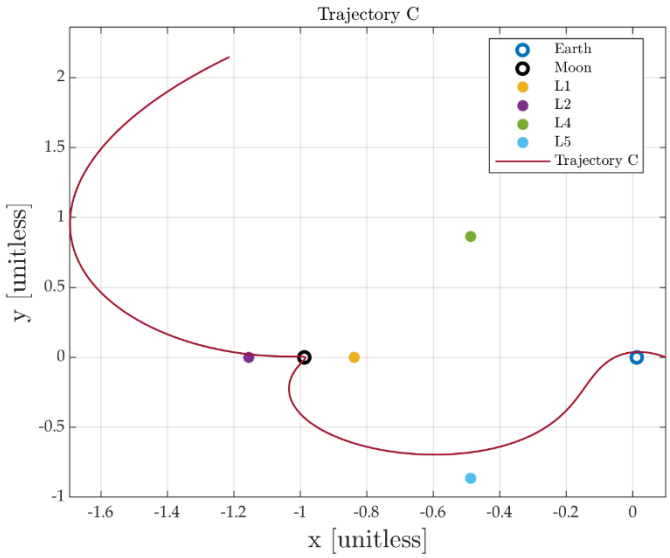
Case A

This is a lunar-centered trajectory with a maximum orbital distance of $\approx 0.06\ DU$ (where $1\ DU =$ distance from earth to moon). Over the time span shown here, the spacecraft enters and exits the system from the left (around y = 0). The geometry of this trajectory is smoother and more elliptical than the similar G trajectory. This trajectory has a velocity y-component, while G has no initial velocity components in any direction. Both A and G trajectories are plotted over the same time span. Trajectory A may be useful for lunar remote sensing missions or for missions requiring lunar proximity.



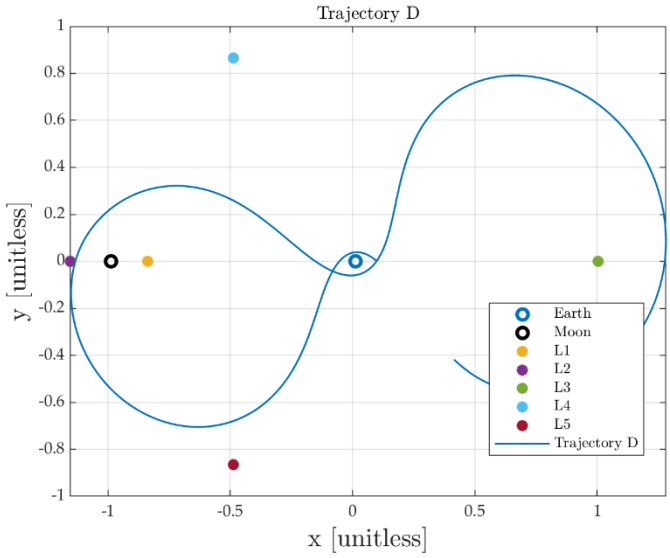
Case B

Case B is a lunar free-return trajectory in which the spacecraft starts *very near* to the L2 libration point with some velocity in the y-direction only. This interesting path then flies around the moon for a bit before skirting past the L1 point on its way to a visit to earth. After a few passes around the earth at $\approx 0.8\ DU$, the spacecraft will skirt by L1 on its way back to the moon system. This trajectory may be useful for cis-lunar missions.



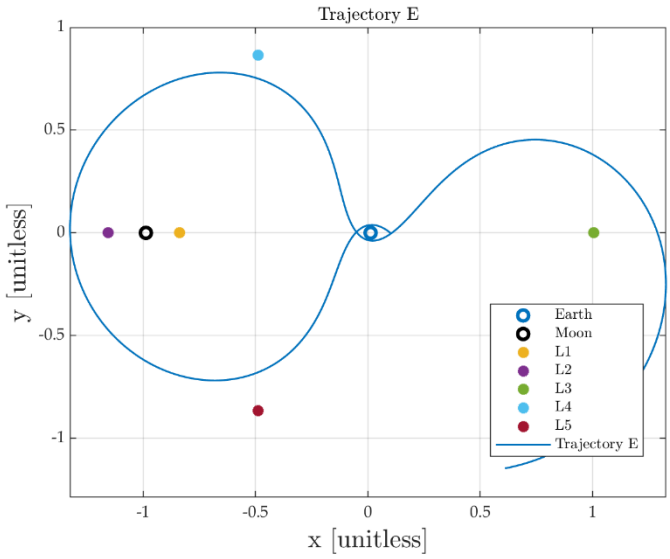
Case C

This is not a free-return trajectory, which can be seen when increasing the time interval to, say, 500 TU, which results in an increasing outward spiral trajectory. This path starts near the earth (on the right of this plot) and swings by the moon on its way out of the earth-moon system. The initial velocity for this trajectory is large relative to Cases A and B, being -3.35 in the x-direction and 3.0 in the y-direction. This may be a useful trajectory for interplanetary missions originating from earth.



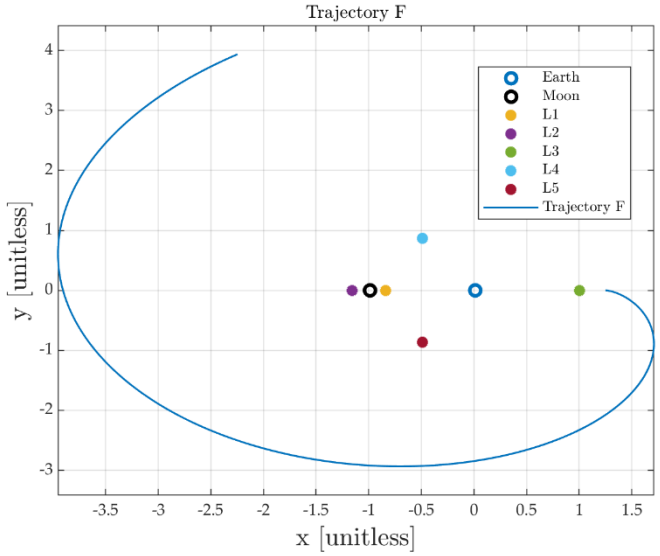
Case D

If the initial position and y-velocities were kept the same as Case C and only the x-direction velocity were increased by 0.02 (velocity in canonical units), the free-return trajectory in Case D would result. The extra initial *x*-velocity causes a far-side moon pass (as opposed to the Case C near-side flyby) which then allows the spacecraft to "fall" back toward earth for a close pass. This might be a useful trajectory for a technology (say, laser communication) demonstration mission.



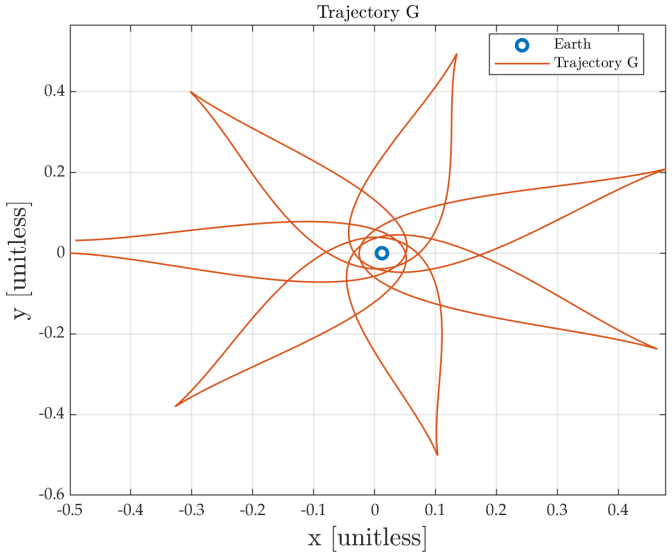
Case E

If we keep slightly increasing the initial velocity in the *x*-direction, the trajectory appears more symmetric about the moon-L1-L2 system than in Case D, while the free-return trajectory results in a nearer-to-earth flyby. This trajectory may be useful for the same reasons as in Case D and would result in a closer approach to earth on its way back.



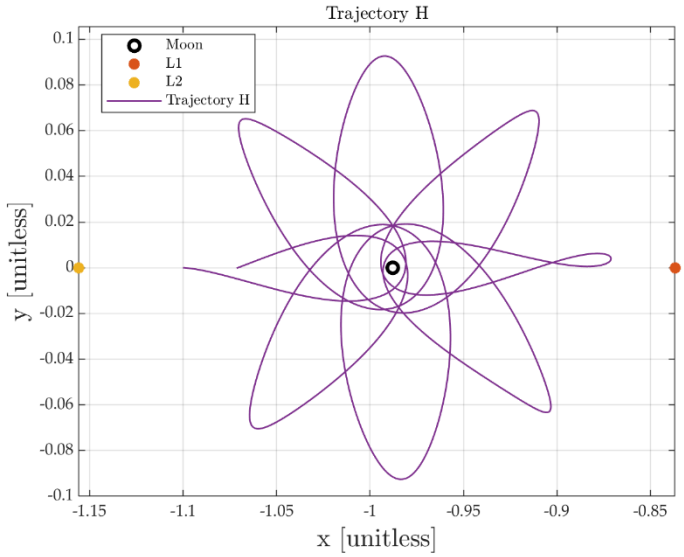
Case F

With a starting point just past L3 and no initial velocity, a spacecraft will loop far outside the earth-moon system and, unchecked, continue in an increasing outward spiral, leaving the system. This is not a free-return trajectory. This may be a candidate trajectory from an L3 Halo orbit to interplanetary systems.



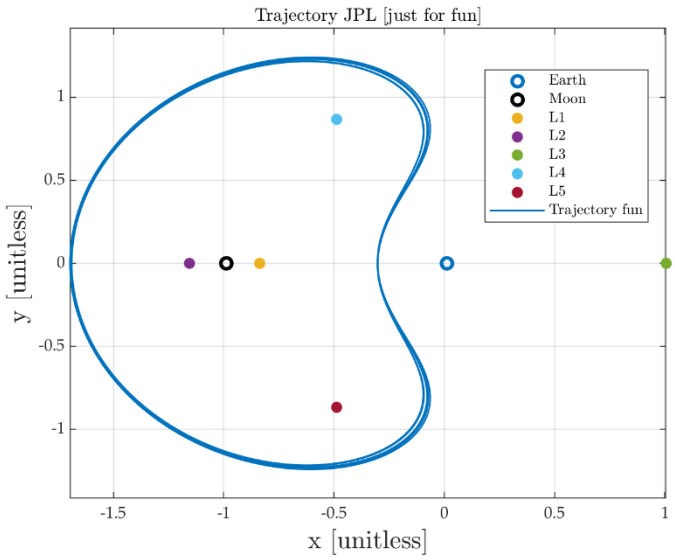
Case G

If only the initial position were changed from Case F to just between L4 and L5 and in line with the earth, keeping the initial velocities the same (zero), the resulting trajectory is a 'trapped' earth-centric trajectory (the epitome of 'free return', since it keeps coming back) that increases in amplitude with time but never escapes earth (in the Circular-Restricted-3-Body case considering only the earth-moon system). It is difficult to conceive of how this trajectory would be useful.



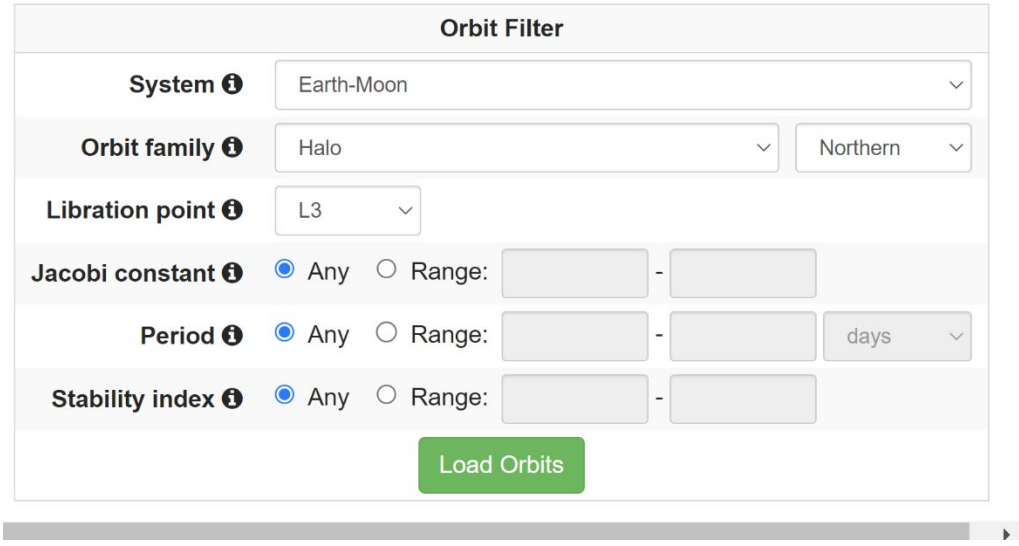
Case H

If only the initial position were adjusted from Case G's $-0.5\ DU$ to $-1.10\ DU$, Case H would result. The geometry of this trajectory is similar to Case G yet is clearly about the moon rather than the earth. This trajectory could be used for remote sensing, although like Case G, it is not clear how this would be a useful trajectory.



Case (BONUS)

Just for fun, I looked up what JPL had and found a [Solar System Dynamics page] that lets users choose system and orbit parameters and then they can use a tabulated list of outputs for plots, which I did, shown above—just for fun. I just grabbed one of the sets of initial conditions that were in the table. This does not seem like a very useful orbit, unless a space telescope or communications system had good reason to follow such a path.



JPL Solar System Dynamics: https://ssd.jpl.nasa.gov/tools/periodic_orbits.html

4. Plot the zero relative speed contours for the earth-moon system. Show the libration points as well as the position of the Earth and Moon.

FROM LECTURE, WE HAVE VELOCITY OBTAINED FROM THE JACOBI CONSTANT:

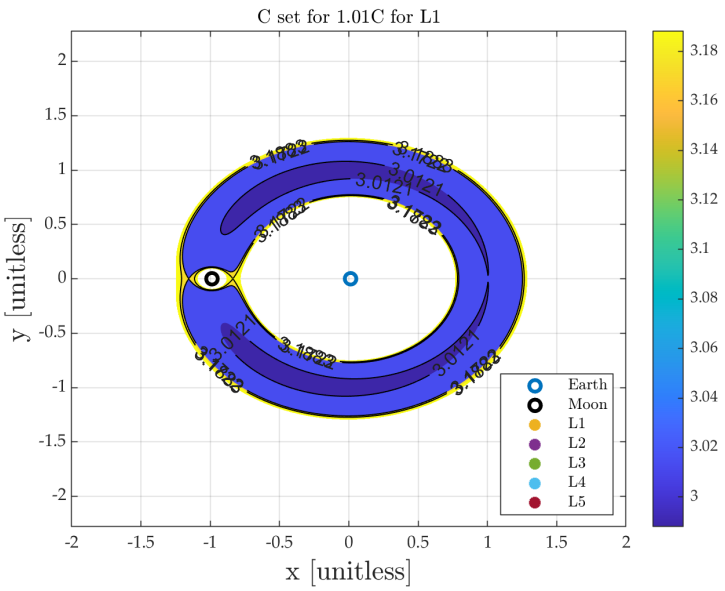
$$V^2 = (x^2 + y^2) + \frac{2(1-\mu^*)}{r_1} + \frac{2\mu^*}{r_2} - C$$

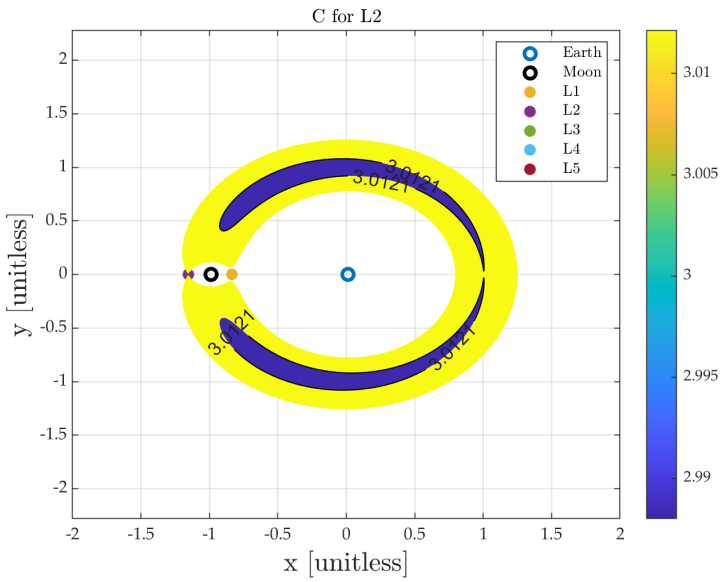
SET $V = 0$, FIND C KNOWING x, y. THAT IS,

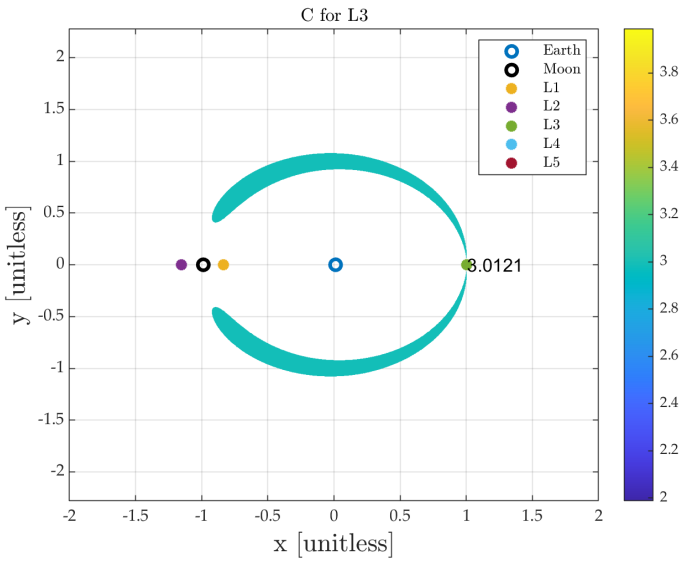
$\forall\, x, y \in [-2, 2)$ PLOT CONTOURS.

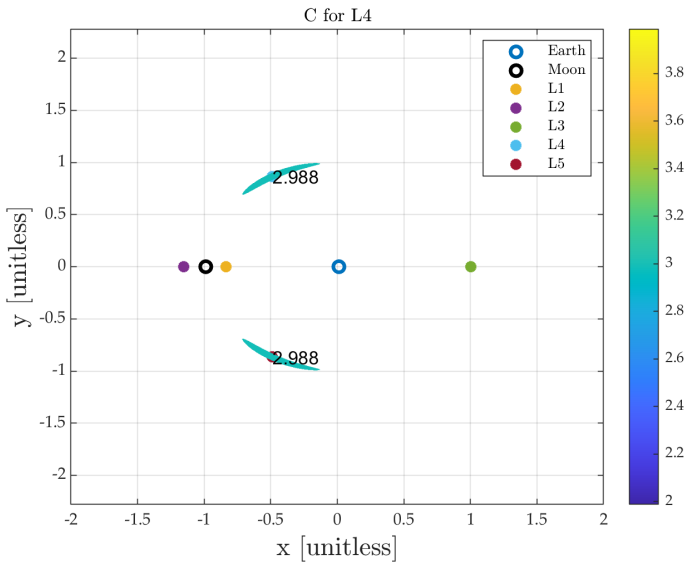
SEE MATLAB.

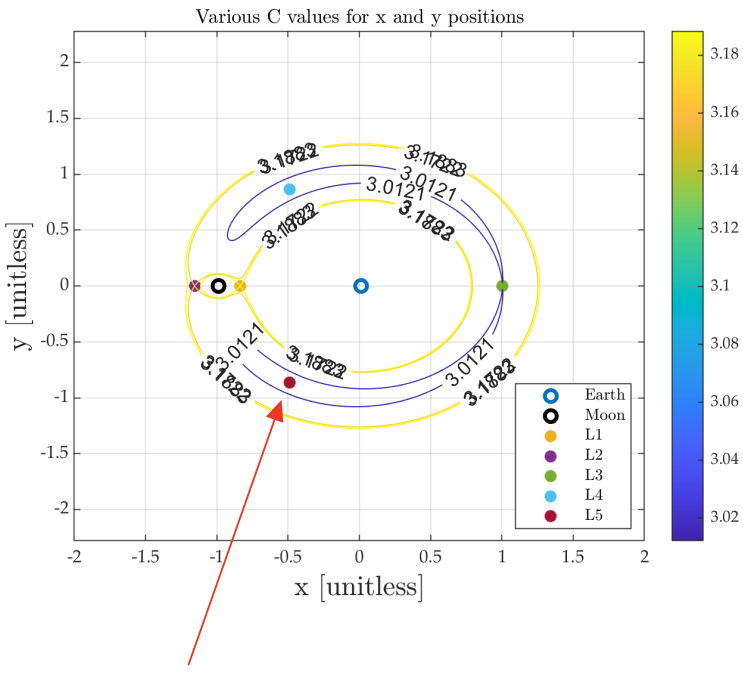
→ PLOTS HERE FOR CLARITY AND SIMPLICITY.





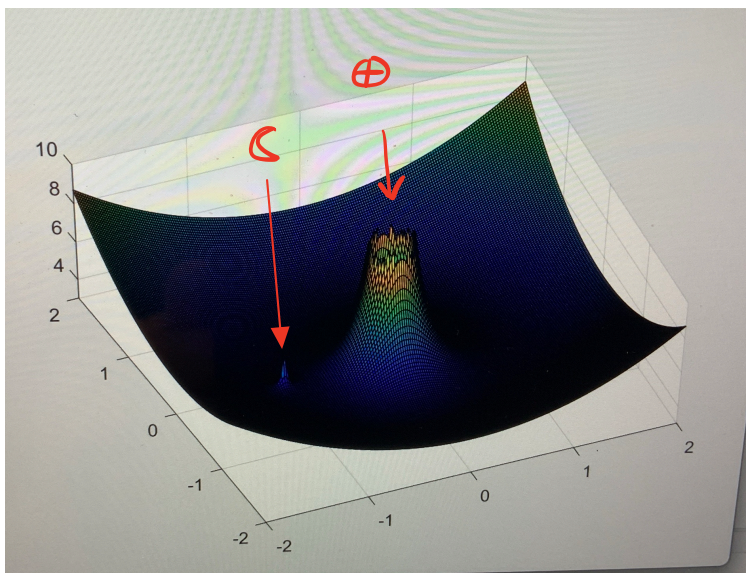






TWO LINES, HARD TO READ.

C ~ 3.17 - 3.18



JUST FOR FUN TO HELP VISUALIZE ENERGY LEVELS.

END HW # 4

2: MATLAB RESULTS

Table of Contents

..... 1
Problem 1: SRP with and without eclipse 1
P1: Part 1. 1
P1: Eclipse Times 1
P1: Plot trajectories in ECI 3
P1: Plot Apogee / Perigee (SRP with Eclipse only) 4
P1: Plot changes in COEs (SRP with AND without Eclipse) 6
Problem #2. 6
Problem 3.0: Libration points 8
Problem 3.a: Trajectory A 9
Problem 3.b 9
Problem 3.c 10
Problem 3.d 11
Problem 3.e 12
Problem 3.f 13
Problem 3.g 14
Problem 3.h 15
Problem 4. Energy Contour Plots 17
Contour for L1 18
Contour for L2 19
Contour for L3 20
Contour for L4 21

Problem 1: SRP with and without eclipse

```
~~~~~~~~~~~~~ PROBLEM 1: ECLIPSE TIMING ~~~~~~~~~~~~~
```

P1: Part 1.

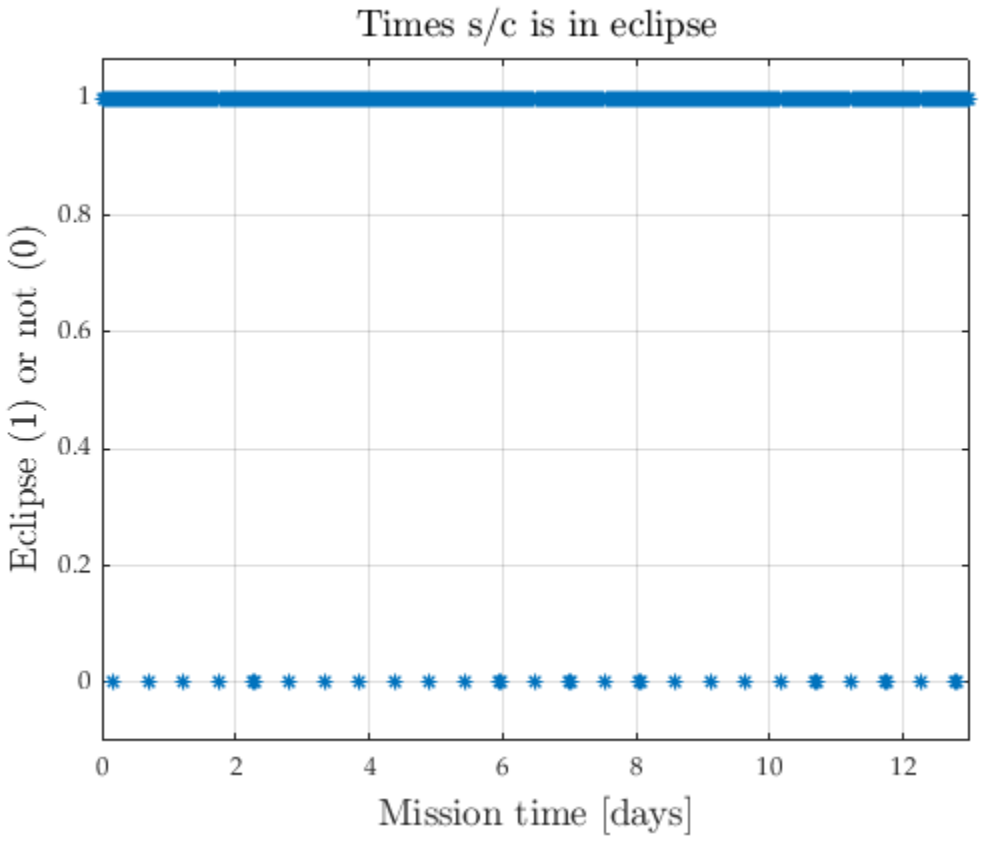
Part 1: Calculate entry and exit eclipse times for a TWO WEEK PERIOD: August 10 - August 23, 1997 PART 2: Find percentage increase for solar radiation pressure IF we ignore eclipses.

```
Time to run SRP model is: 0.18142 seconds
Time to run SRP (sans Eclipse) model is: 0.16769 seconds
Time to run OSCULATING model is: 0.12045 seconds
```

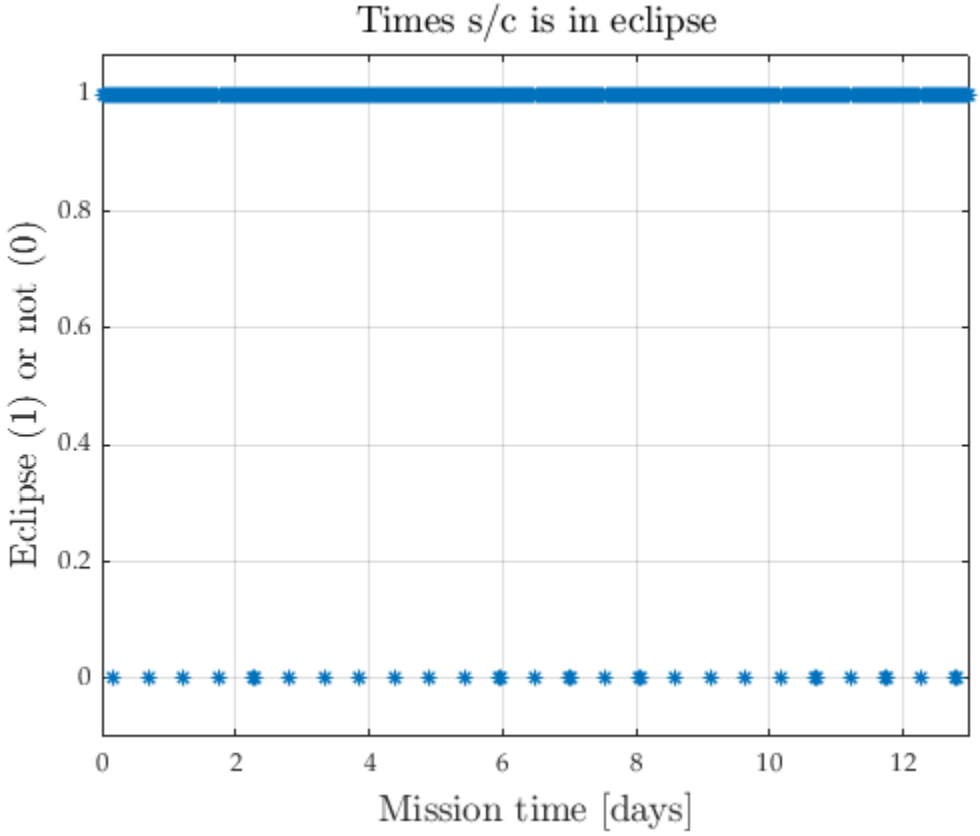
P1: Eclipse Times

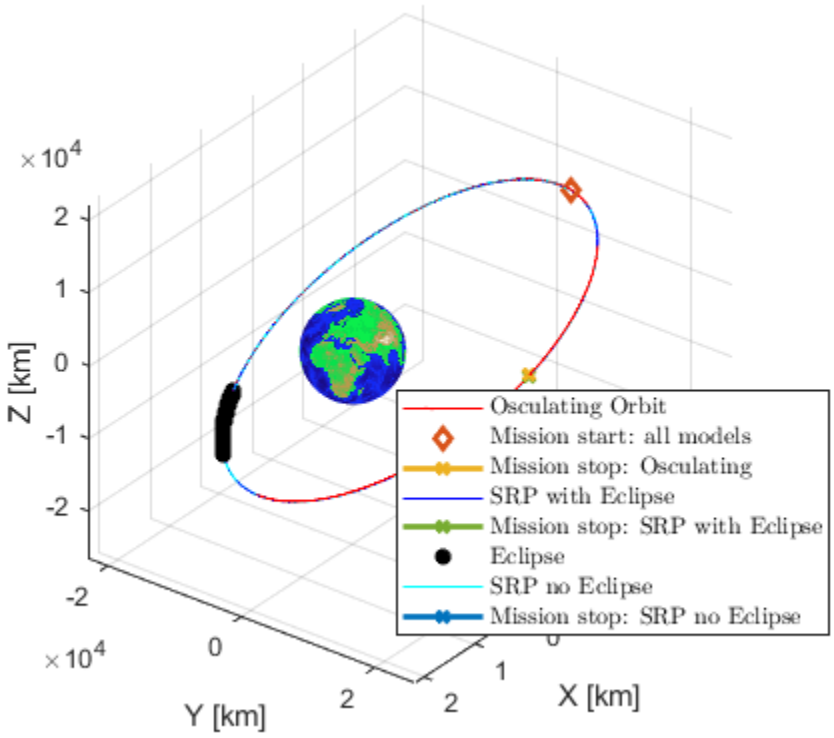
% DELIVERABLE CALLS FOR "entry and exit eclipse times" over this time period. DateVector gives exact times.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Time s/c is in eclipse: 0.55242 days <------------------------
Percent of time s/c is in eclipse (of 13 day mission): 4.2494 percent
<------------------------
```


Times s/c is in eclipse
Eclipse (1) or not (0)
1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
Mission time [days]

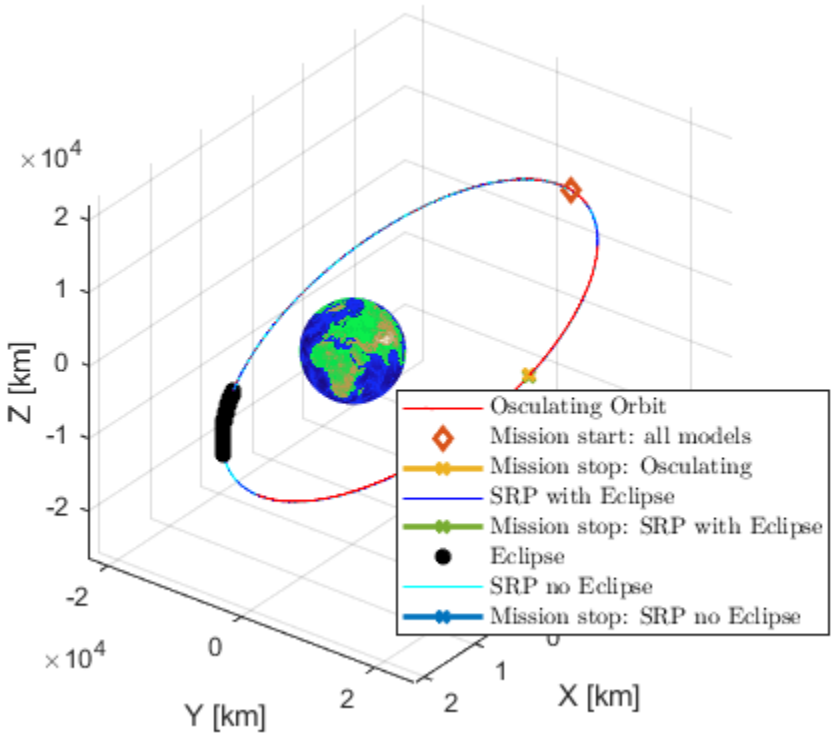
P1: Plot trajectories in ECI

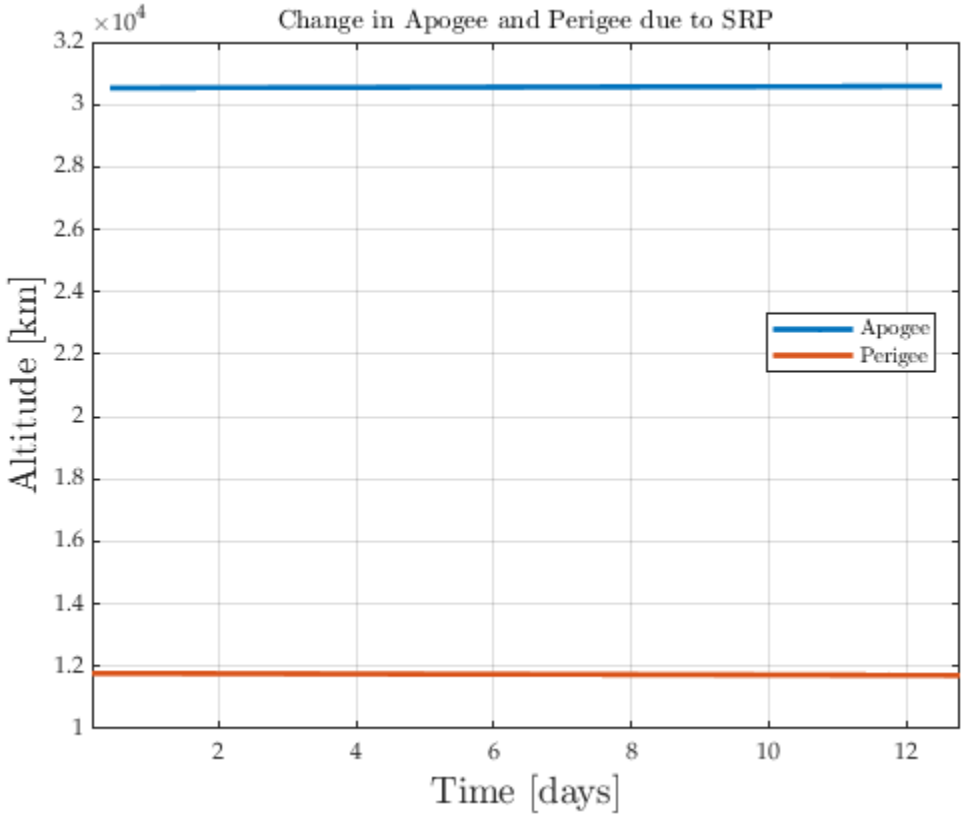




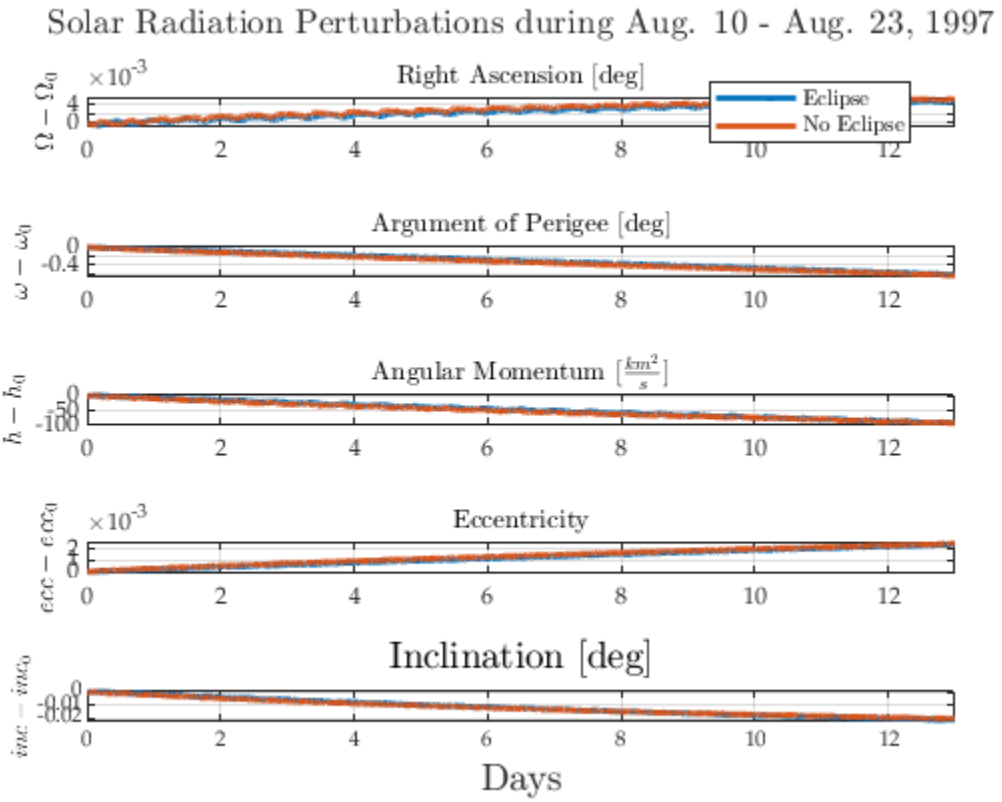
P1: Plot Apogee / Perigee (SRP with Eclipse only)

Extract outputs (RADIANS RADIANS RADIANS)


×10^4
2
1
0
-1
-2
Z [km]
-2
0
2
×10^4
Y [km]
2
1
0
X [km]
Osculating Orbit
Mission start: all models
Mission stop: Osculating
SRP with Eclipse
Mission stop: SRP with Eclipse
Eclipse
SRP no Eclipse
Mission stop: SRP no Eclipse

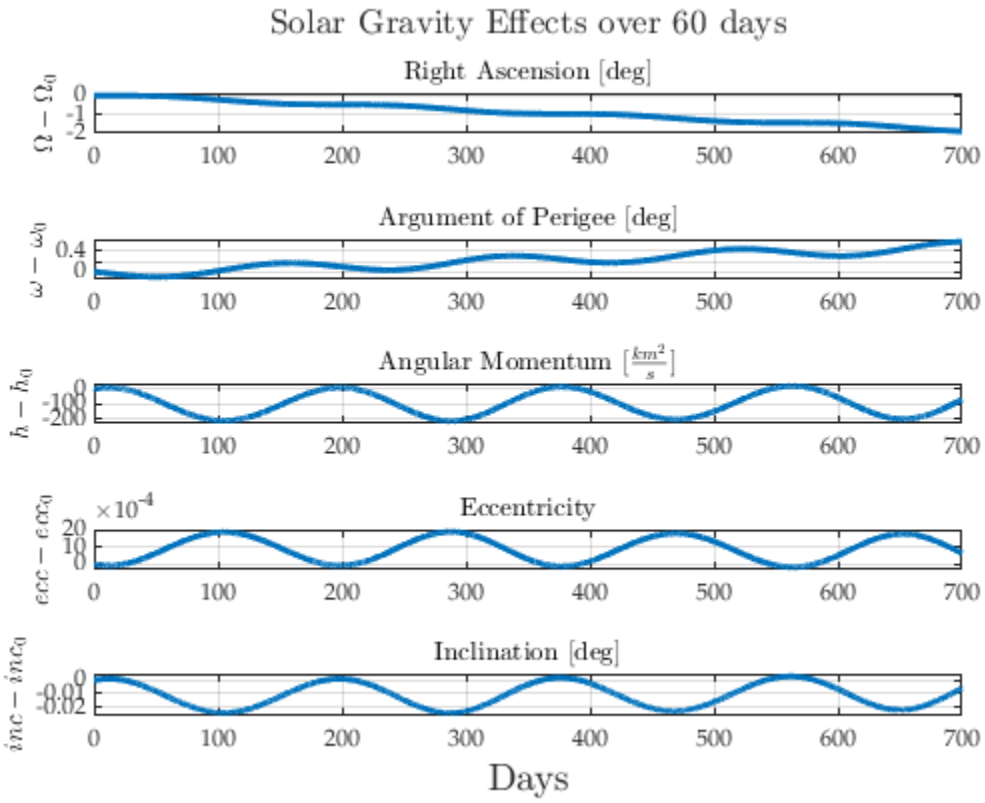

Change in Apogee and Perigee due to SRP
×10^4
3.2
3
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
Altitude [km]
2
4
6
8
10
12
Time [days]
Apogee
Perigee

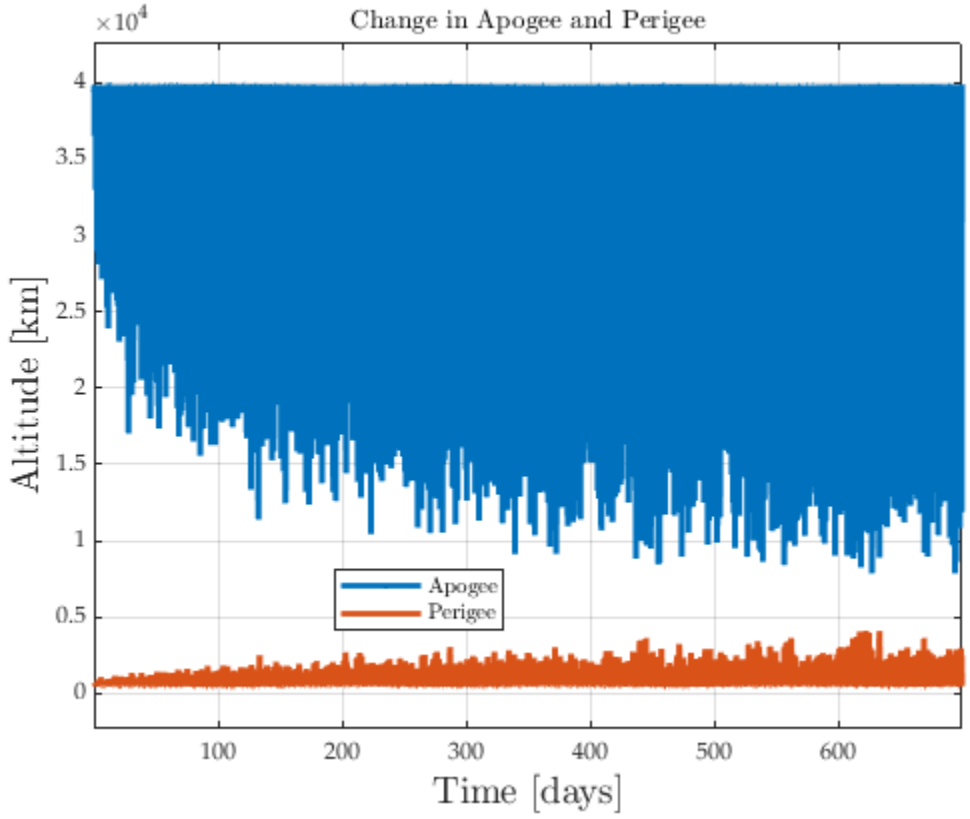
P1: Plot changes in COEs (SRP with AND with-out Eclipse)



Problem #2.

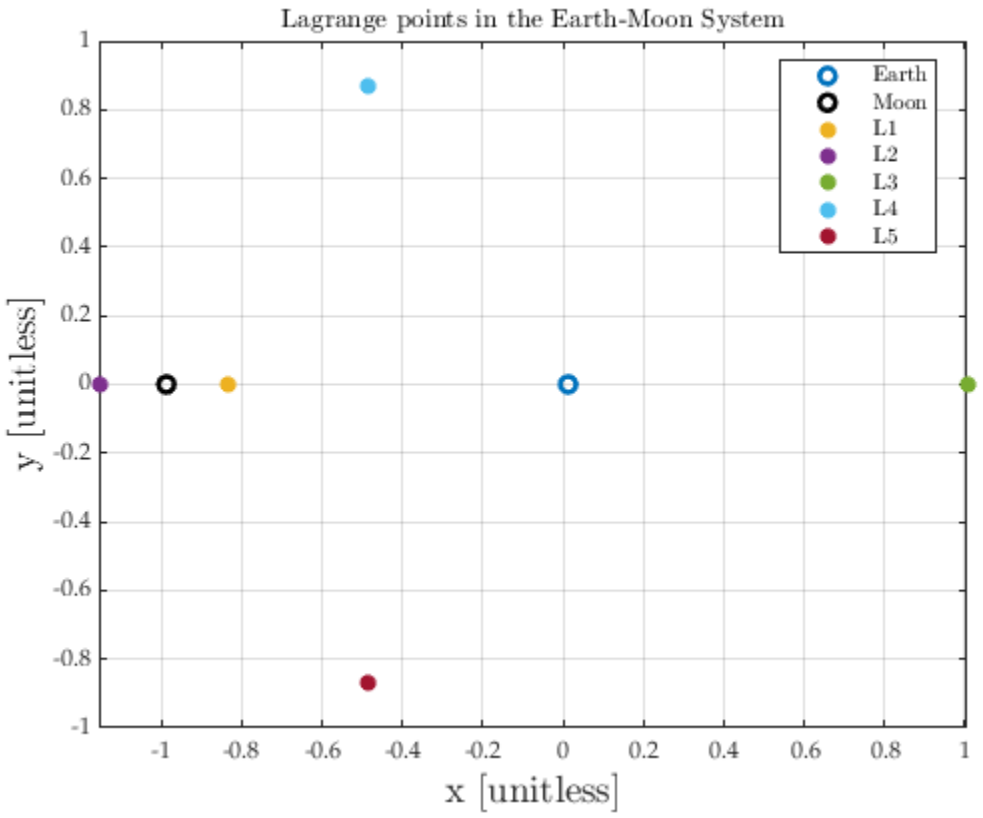
~~~~~~~~~~~~~~ PROBLEM 2: SOLAR GRAVITY (see plots) ~~~~~~~~~~~~~~
Time to run VoP for Solar Gravity model is: 5.8373 seconds


Solar Gravity Effects over 60 days
Right Ascension [deg]
$\Omega - \Omega_0$
Argument of Perigee [deg]
$\omega - \omega_0$
Angular Momentum $[\frac{km^2}{s}]$
$h - h_0$
Eccentricity
$\times 10^{-4}$
$ecc - ecc_0$
Inclination [deg]
$inc - inc_0$
Days

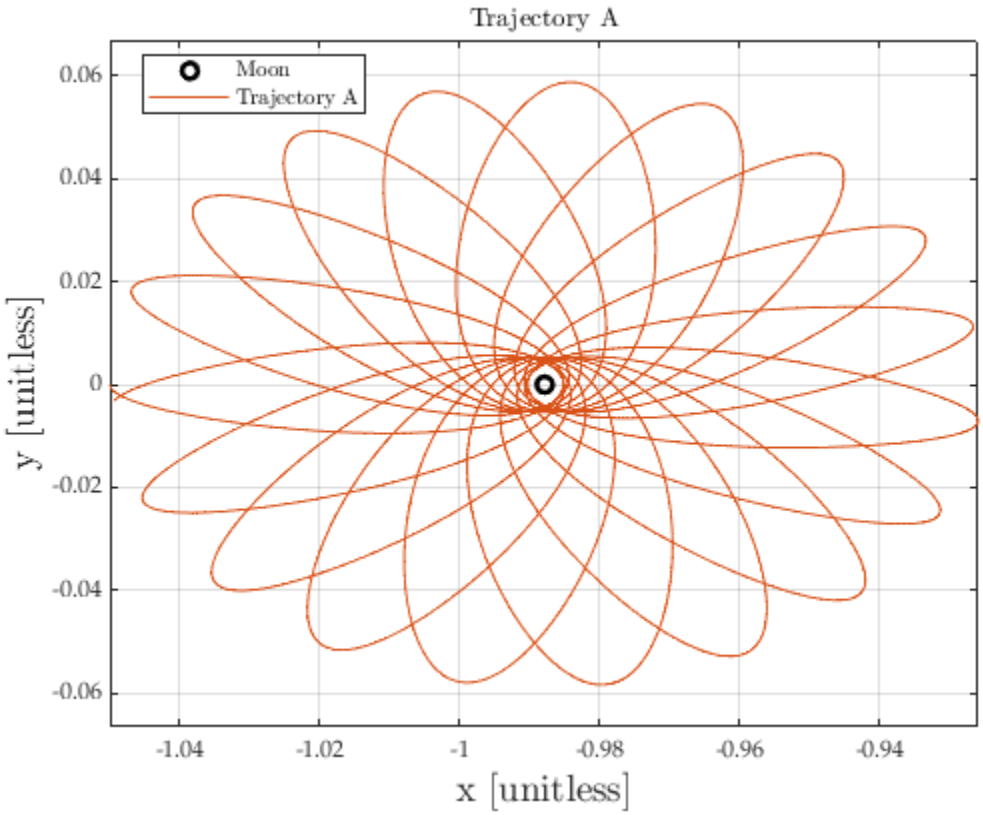

Change in Apogee and Perigee
$\times 10^4$
Altitude [km]
Apogee
Perigee
Time [days]

Problem 3.0: Libration points

~~~~~~~~~~~~~~ PROBLEM 3: Libration Points and S/C Trajectories ~~~~~~~~~~~~~~

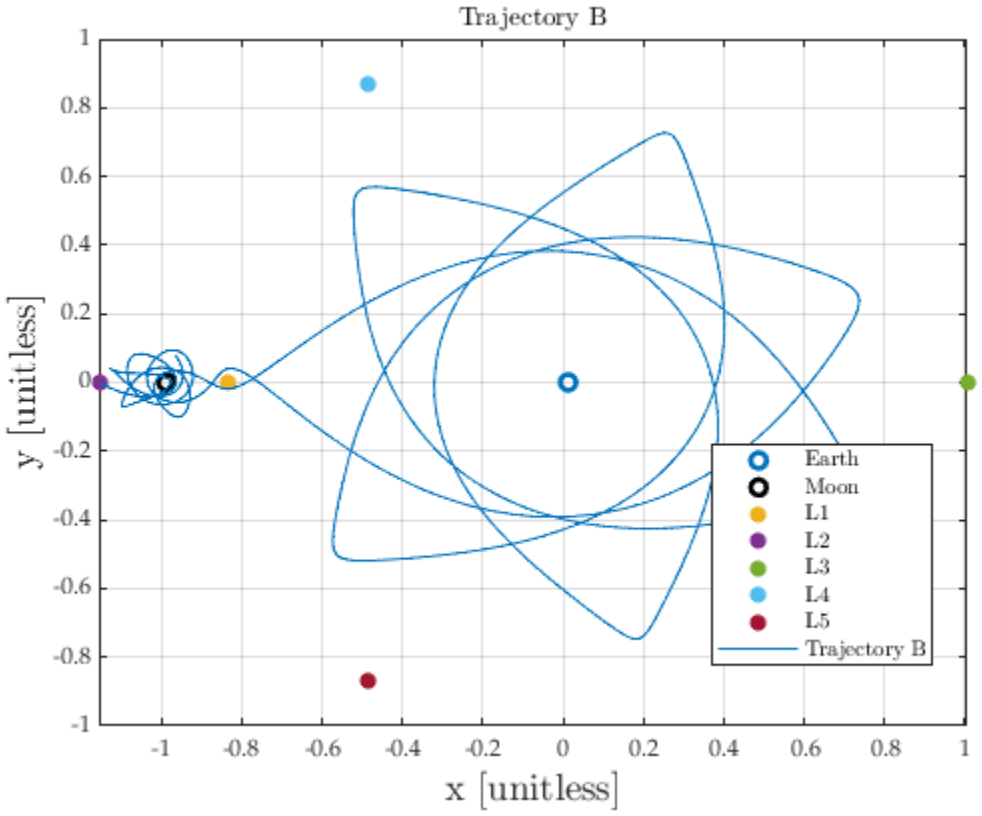


Problem 3.a: Trajectory A



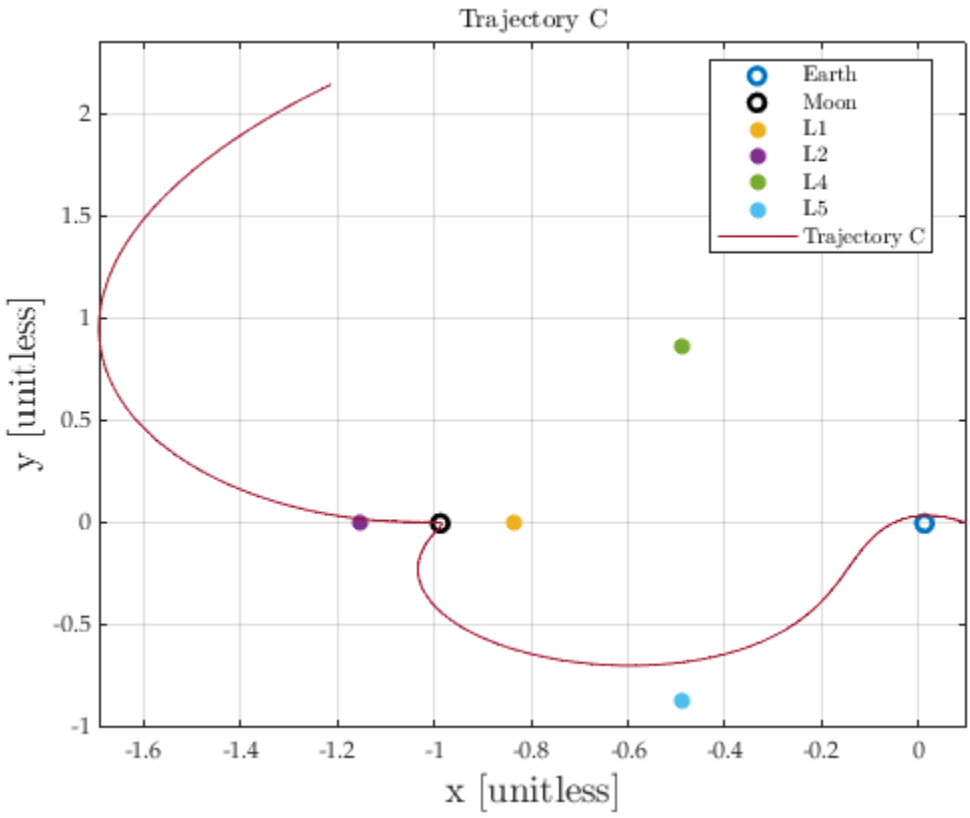
Problem 3.b

Trajectory B Define state vectors given in problem statement



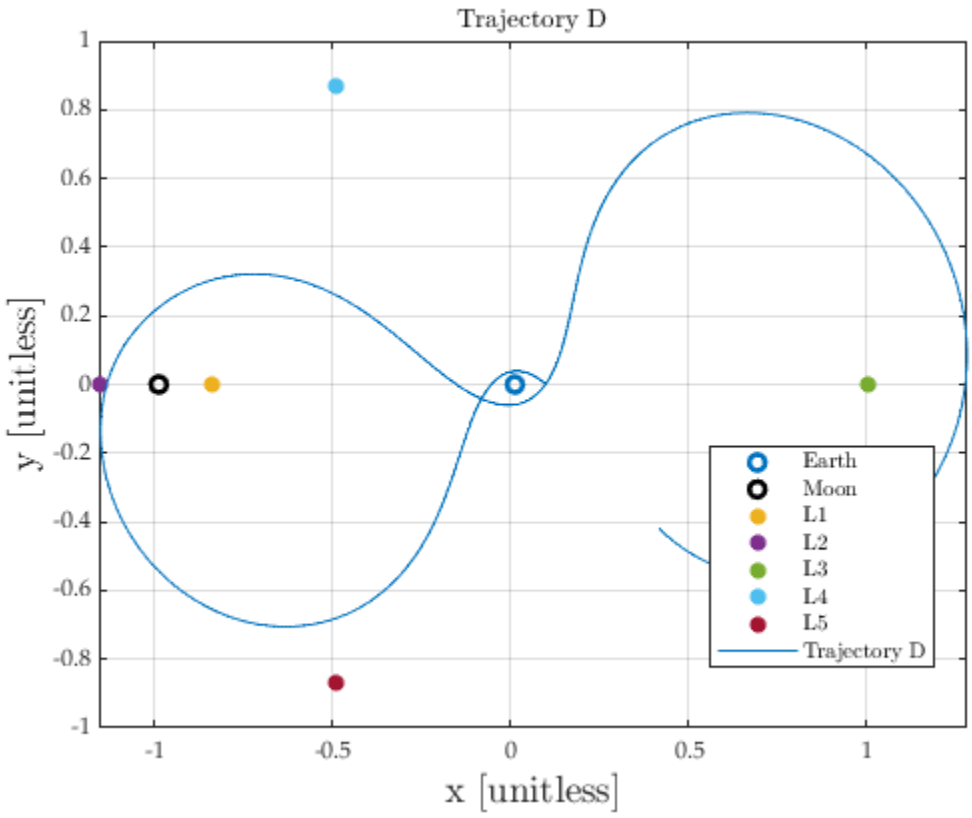
Problem 3.c

Trajectory C Define state vectors given in problem statement



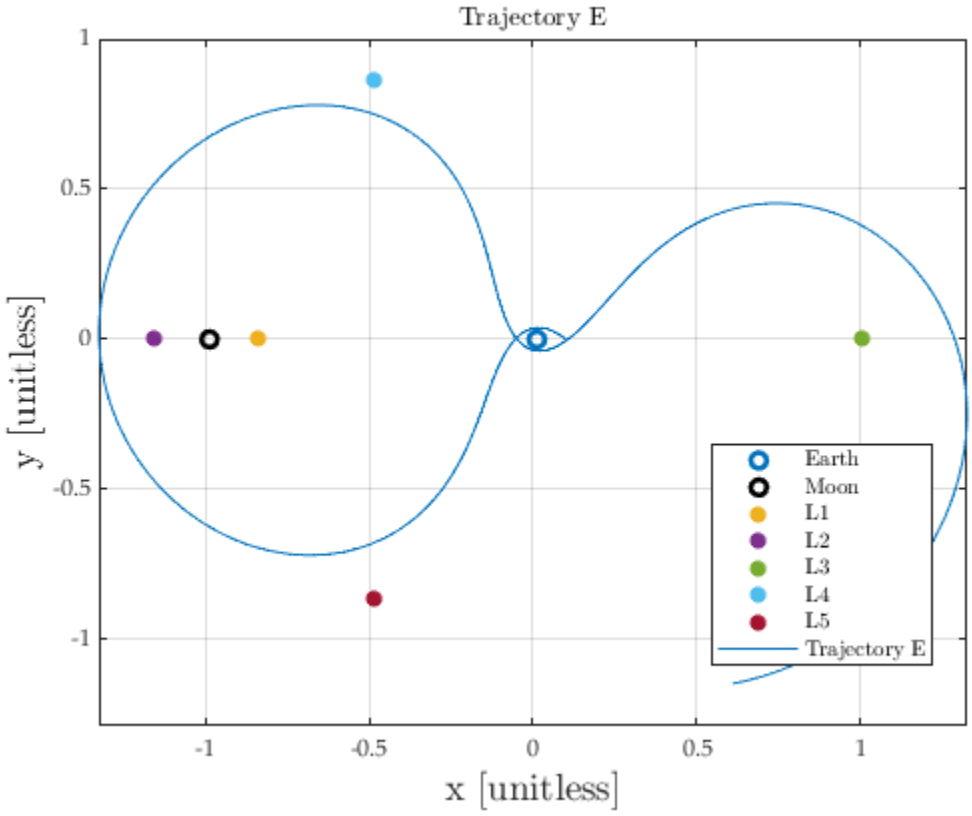
Problem 3.d

Trajectory D Define state vectors given in problem statement



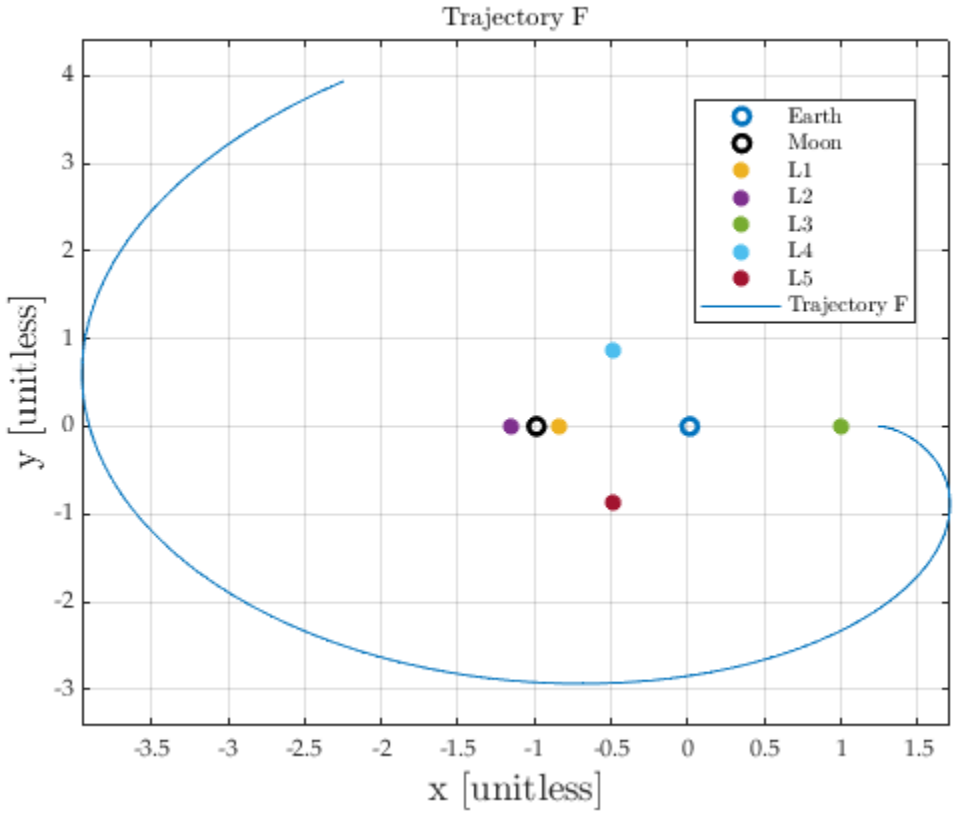
Problem 3.e

Trajectory E Define state vectors given in problem statement



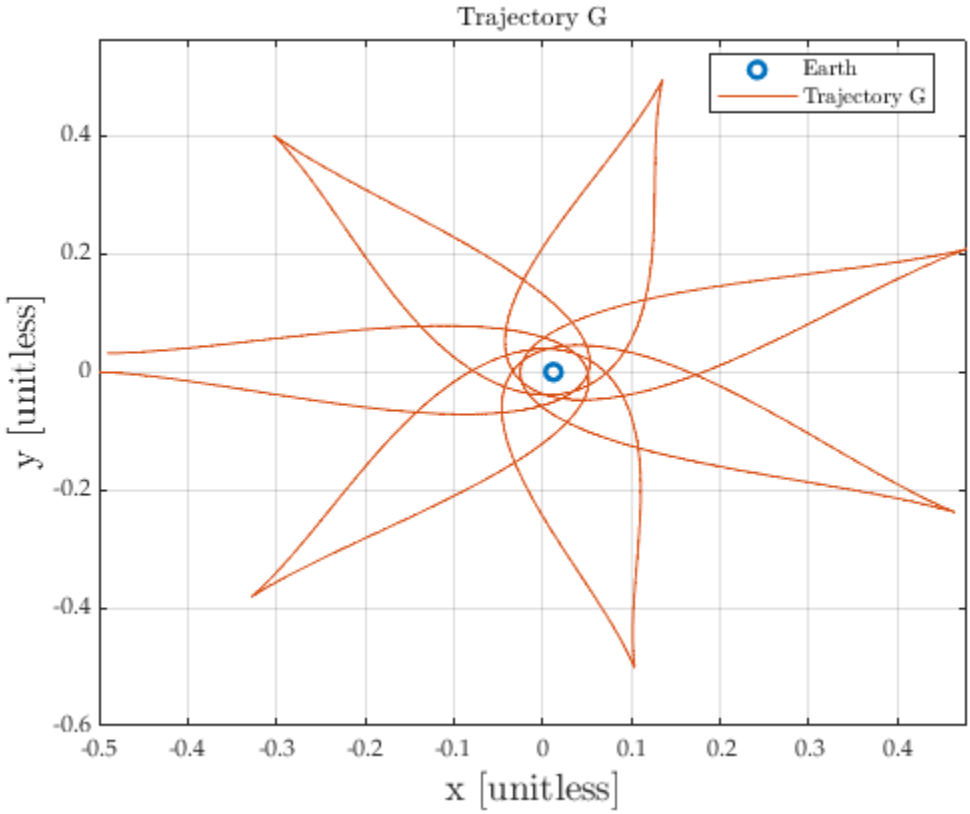
Problem 3.f

Trajectory F Define state vectors given in problem statement



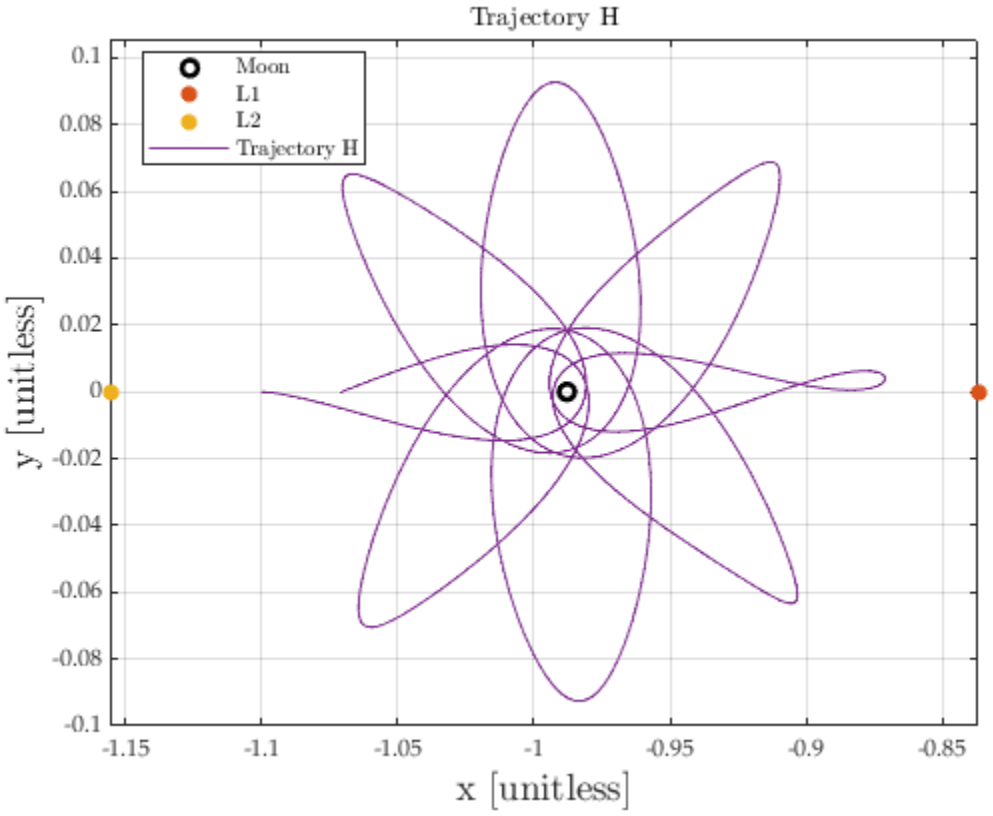
Problem 3.g

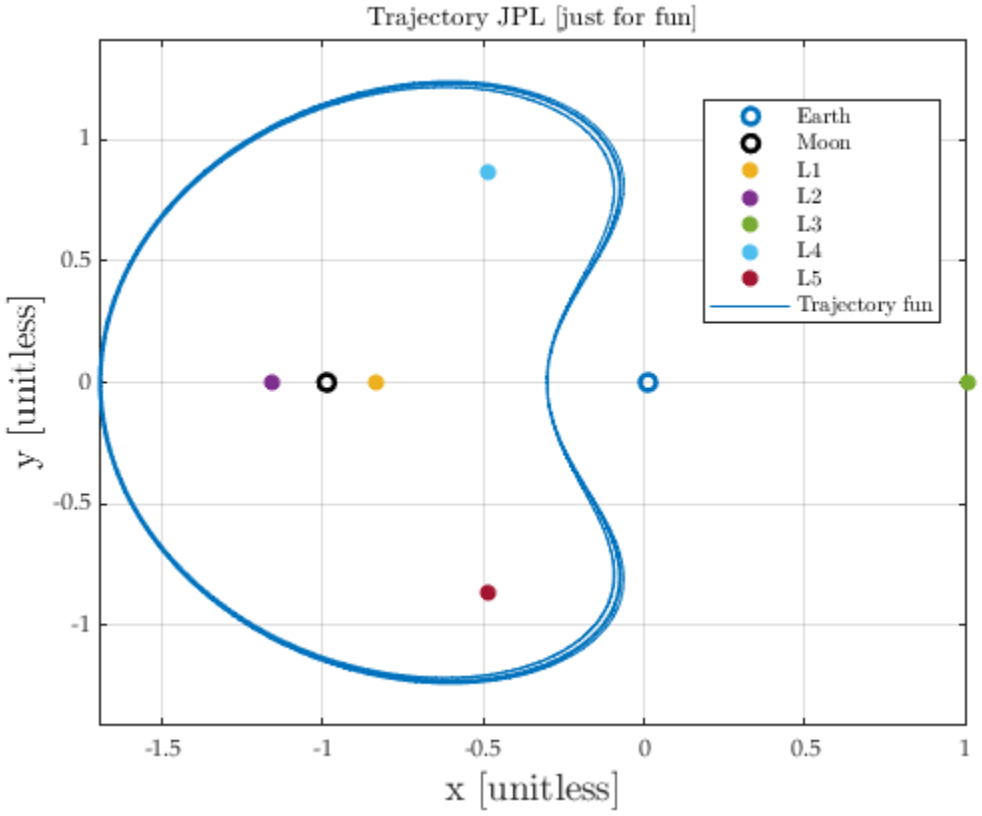
Trajectory G Define state vectors given in problem statement



Problem 3.h

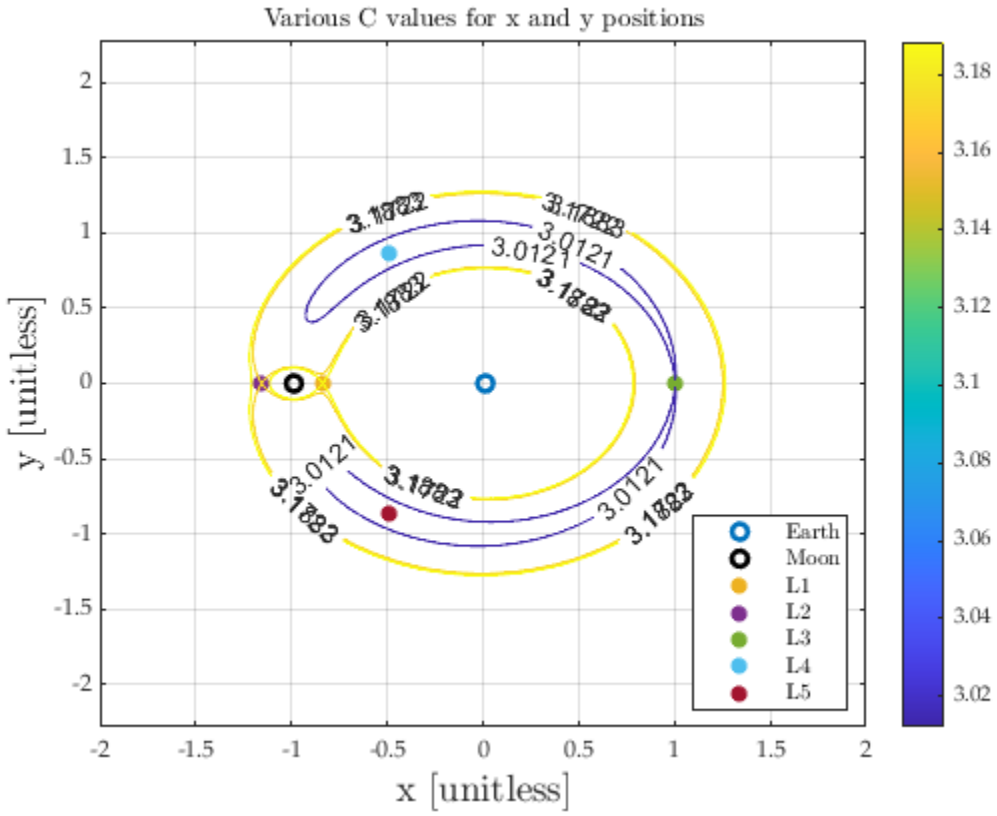
Trajectory H Define state vectors given in problem statement


Trajectory H
Moon
L1
L2
Trajectory H
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
-1.15
-1.1
-1.05
-1
-0.95
-0.9
-0.85
y [unitless]
x [unitless]

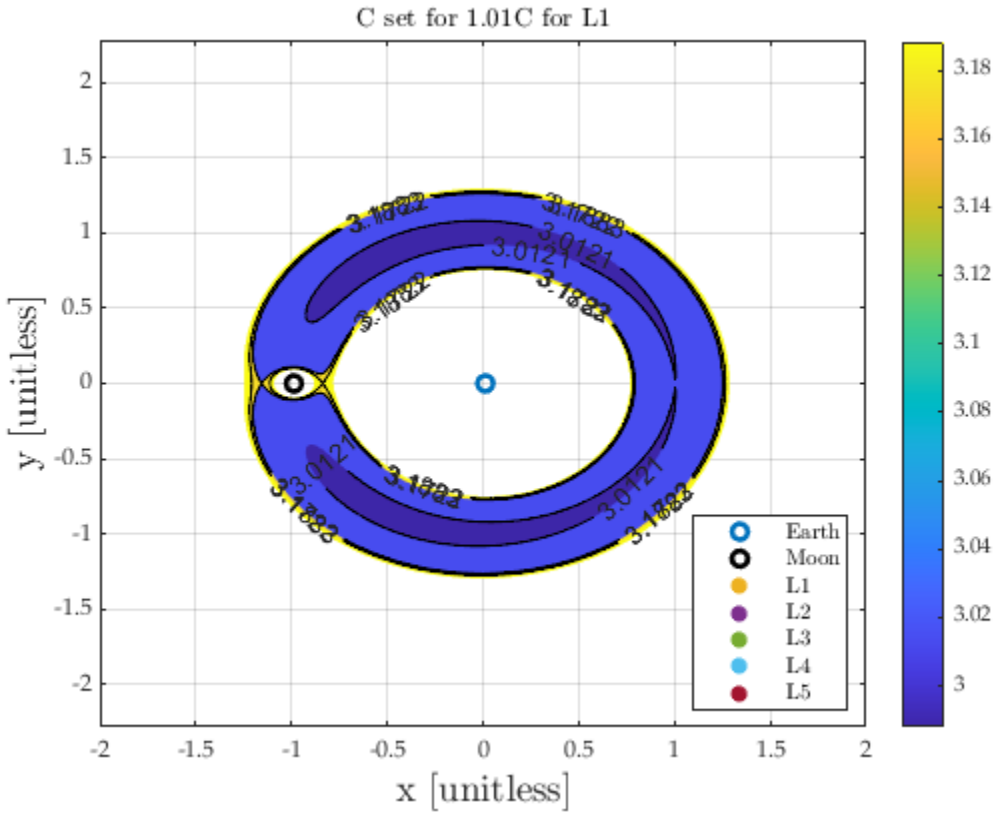

Trajectory JPL [just for fun]
Earth
Moon
L1
L2
L3
L4
L5
Trajectory fun
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
y [unitless]
x [unitless]

Problem 4. Energy Contour Plots

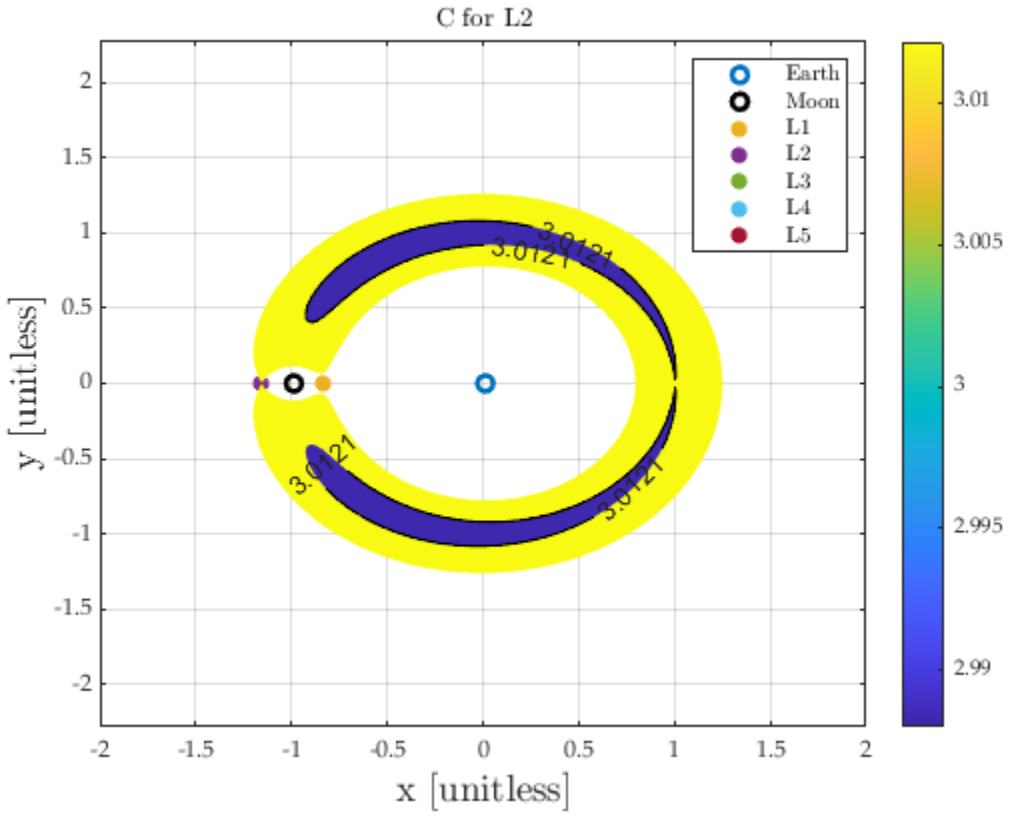
~~~~~~~~~~~~~~ PROBLEM 4: Zero Relative Speed Contours (Earth-Moon System) ~~~~~~~~~~~~~~



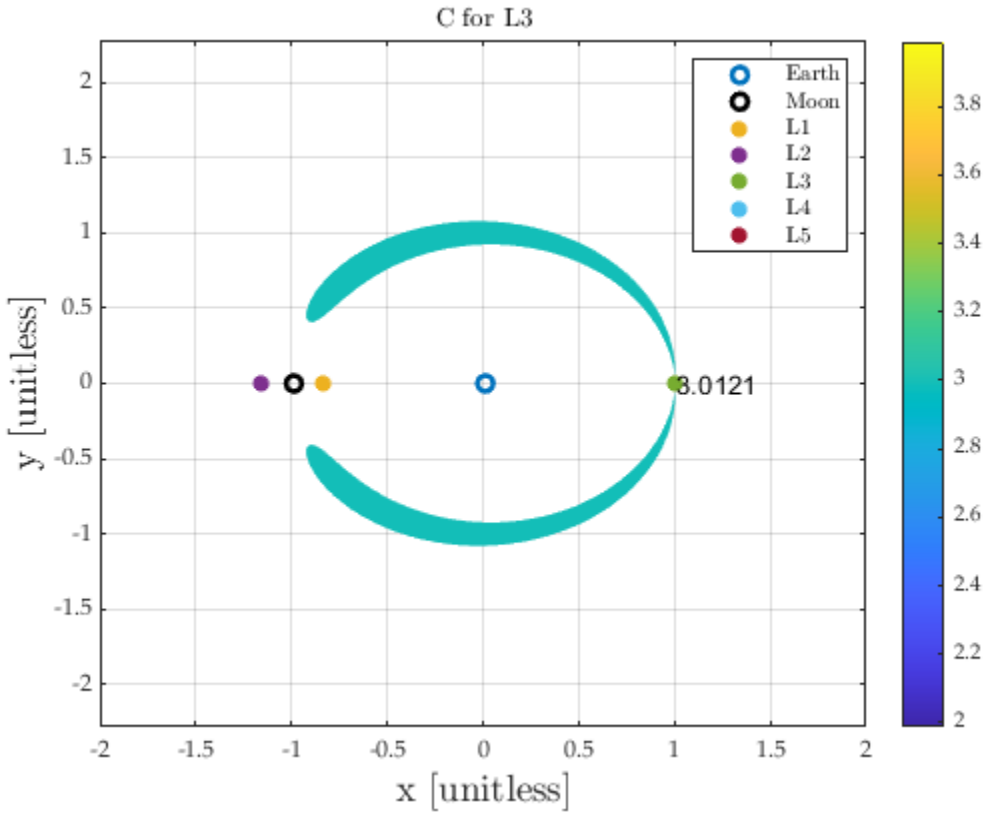
Contour for L1



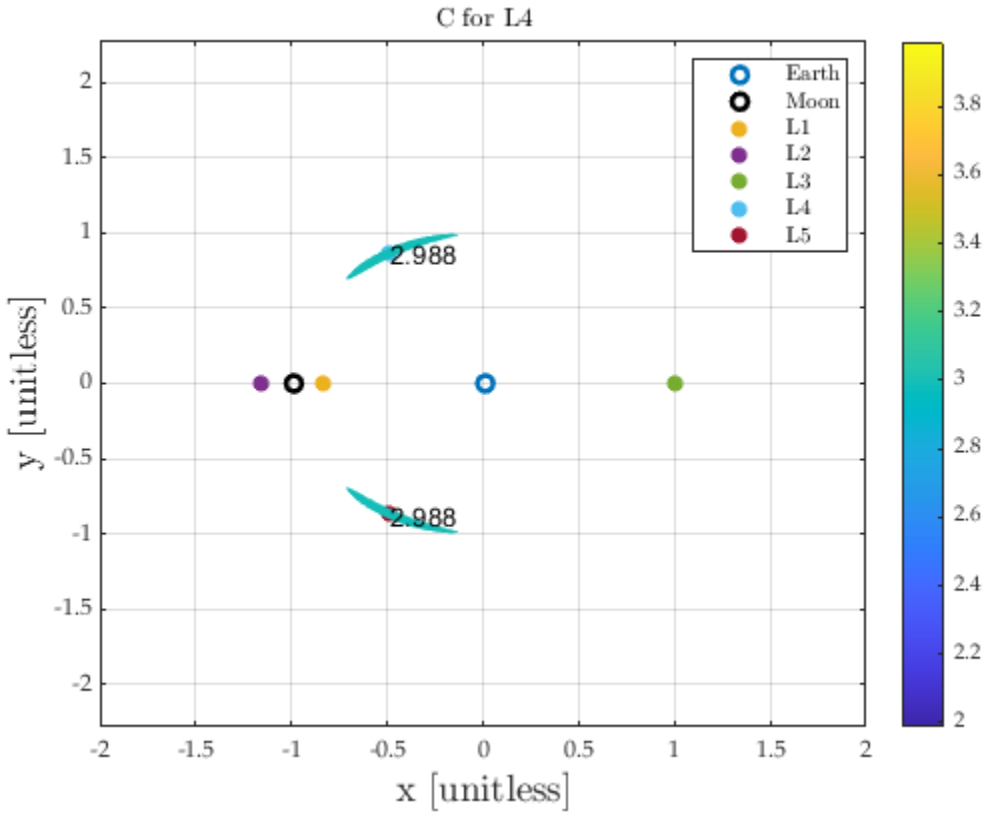
Contour for L2



Contour for L3



Contour for L4



Published with MATLAB® R2023b

3: MATLAB SCRIPT

Table of Contents

..... 1
Problem 1: SRP with and without eclipse 1
P1: Part 1. 2
P1: Eclipse Times 4
P1: Plot trajectories in ECI 5
P1: Plot Apogee / Perigee (SRP with Eclipse only) 6
P1: Plot changes in COEs (SRP with AND without Eclipse) 7
Problem #2. 9
Problem 3.0: Libration points 13
Problem 3.a: Trajectory A 14
Problem 3.b 15
Problem 3.c 16
Problem 3.d 17
Problem 3.e 18
Problem 3.f 18
Problem 3.g 19
Problem 3.h 20
Problem 4. Energy Contour Plots 22
Contour for L1 24
Contour for L2 25
Contour for L3 26
Contour for L4 27


```
%{
AERO 452 | Spaceflight Dynamics II
California Polytechnic State University, SLO
Dr. Kira Abercromby
Author: Justin Self

HW #4
%}

% Housekeeping
clear all; close all; clc;
addpath("C:/MATLAB_CODE/Orbits/")

% Constants
re = 6378;  % radius of earth, km
mu = 398600; % grav parameter, earth; km3/s2
sec2days = 1/86400;
```

Problem 1: SRP with and without eclipse

```
disp("~~~~~~~~~~~~~~ PROBLEM 1: ECLIPSE TIMING ~~~~~~~~~~~~~~")

% Knowns
R = [-26175.1034;12757.0706;14626.6556];     % km
V = [2.376641; 0.139677; 2.078097];          % km/s
```

```
[h0, inc, RAAN, ecc0, w, theta, epsilon, a, T] = rv2COEs(R,V,mu);

% Change to radians
inc0 = deg2rad(inc);
raan0 = deg2rad(RAAN);
w0 = deg2rad(w);
theta0 = deg2rad(theta);
COEs = [raan0;inc0;w0;h0;ecc0;theta0]; % in rad now
```

P1: Part 1.

Part 1: Calculate entry and exit eclipse times for a TWO WEEK PERIOD: August 10 - August 23, 1997 PART 2: Find percentage increase for solar radiation pressure IF we ignore eclipses.

```
%..... Step 1: Find geocentric position vector of the sun on the date given.

%..... Find JD
y = 1997;
m = 08;
d = 10;
UT = [0 0 0];

%..... Convert UTC date/time into Julian date
[jd_initial, ~, ~] = juliandateFunction(y,m,d,UT);

% Define inputs to new SRP function
A_sc = 200; % m2, for simplicity
mass_sc = 100; % kg for simplicity

%..... Define end date
y2 = 1997;
m2 = 08;
d2 = 23;
UT2 = [0 0 0];
[jd_final, ~, ~] = juliandateFunction(y2,m2,d2,UT2);

tf = (jd_final - jd_initial)*86400; % in seconds now

%..... ode45 options
options = odeset('RelTol', 1e-12, 'AbsTol',1e-12);
tspan = [0 tf];
state = COEs;

%.....Call ode for SRP WITH ECLIPSE version
tstart = tic;
[timenew, COEsnew] =
ode45(@SRP_VoP,tspan,state,options,jd_initial,A_sc,re,mu,mass_sc);
tsol = toc(tstart);
disp("Time to run SRP model is: " + tsol + " seconds")

%.....Call ode for SRP WITH NO ECLIPSE version
% Super weird. y and t were outputting in different lengths. No clue why.
```

```
% Thus, I forced the outputs.

nout = length(COEsnew(:,1));
tVector = linspace(0, tf, nout);
tstart2 = tic;
[time_NoEclipse, COEs_NoEclipse] =
ode45(@SRP_VoP_ignoreEclipse,tVector,state,options,jd_initial,A_sc,re,mu,mass
_sc);
tSRP_noEclipse = toc(tstart2);
disp("Time to run SRP (sans Eclipse) model is: " + tSRP_noEclipse + "
seconds")

%.....Call ode for OSCULATING version
tstart3 = tic;
[time_osc, COEs_osc] = ode45(@vop_NoPerts,tspan,COEs,options,mu);
tSRP_Osc = toc(tstart3);
disp("Time to run OSCULATING model is: " + tSRP_Osc + " seconds")

%{
%.....Find eclipse times (hack method, thanks Travis)
% travis = COEsnew(:,7);
% for i = 1:length(COEsnew)-1
%     if travis(i) == travis(i+1)
%         nu(i) = 0;
%     else
%         nu(i) = 1;
%     end
% end
% nu = nu';
%}

% ..... Extract data for ECI plot
% SRP WITH ECLIPSE
rEclipse = zeros(length(timenew),3);
nu = zeros(length(timenew),1);

for i = 1:length(timenew)
    [rEclipse(i,:),~] = rv_from_COESNew(COEsnew(i,:));
    jd = jd_initial + timenew(i)/86400;
    [~,~, R_sun] = solar_position(jd);
    nu(i) = checkeclipse(rEclipse(i,:),R_sun,re);
end

% SRP with NO ECLIPSES
r_NoEclipse = zeros(length(time_NoEclipse),3);

for i = 1:length(time_NoEclipse)
    [r_NoEclipse(i,:),~] = rv_from_COESNew(COEs_NoEclipse(i,:));
end

% NO SRP; osculating orbit
r_osc = zeros(length(time_osc),3);
for i = 1:length(time_osc)
```

```
    [r_osc(i,:),~] = rv_from_COESNew(COEs_osc(i,:));
end
```

P1: Eclipse Times

% DELIVERABLE CALLS FOR "entry and exit eclipse times" over this time period. DateVector gives exact times.

```
%..... Find where eclipses occur (where nu = 0)
eclipseYes = find(~nu);
eclipseNo = find(nu);

% Add up all the time s/c is in eclipse

%{
        |
       / \
      / _ \
     |.o '.|
     |'._.'|
     |     |
   ,'|  |  |`.
  /  |  |  |  \
  |,-'--|--'-.|

%}

% Batch all eclipse indices together
for i = 1:(length(nu)-1)
    if nu(i) - nu(i+1) > 0
        eclipseDuration.index(i) = (i+1);
    elseif nu(i) - nu(i+1) < 0
        eclipseDuration.index(i) = (i);
    end % if
end % for

% All eclipse enter and exit times
x = nonzeros(eclipseDuration.index);
A = zeros(length(x)-1);
for i = 1:2:(length(x)-1)
    A(i) = timenew(x(i+1)) - timenew(x(i));
end

% Time s/c is in eclipse
A = sum(nonzeros(A)) / 86400;
disp("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
disp("Time s/c is in eclipse: " + A + " days <------------------------") %
days
disp("Percent of time s/c is in eclipse (of 13 day mission): " + 100*A/13 +
" percent <------------------------")

% ..... Output DateTime spreadsheet with eclipse times in UTC
% Convert eclipse into jd
eclipseTimes.seconds = timenew(eclipseYes);
```

```
eclipseTimes.jd = jd_initial + eclipseTimes.seconds ./86400;

% Convert JD into UT time (VERY SPECIFIC TIMES)
DateVector = zeros(length(eclipseTimes.jd),6);
 for i = 1:length(eclipseTimes.jd)
 t(i) = datetime(eclipseTimes.jd(i),'ConvertFrom','juliandate'); % new time
in datetime() vector
 DateVector(i,:) = datevec(t(i));
 end
 t = t';

% Write eclipse data to spreadsheet
writematrix(t,'EclipseTimes.xls');

% Plot eclipse times visually
figure()
plot(timenew*sec2days,nu,'*', 'LineWidth',1)
% Graph pretty
ylim padded
xlim tight
xLab = xlabel('Mission time [days]','Interpreter','latex');
yLab = ylabel('Eclipse (1) or not (0)','Interpreter','latex');
plotTitle = title('Times s/c is in eclipse','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set([xLab, yLab],'FontSize', 14)
grid on
```

P1: Plot trajectories in ECI

```
figure()
% Earth
   h1 = gca;
   earth_sphere(h1)
   grid on
   hold on
% Osculating (no perts) orbit
plot3(r_osc(:,1),r_osc(:,2),r_osc(:,3),'r','LineWidth',1)
hold on
% Position at MISSION START
plot3(r_osc(1,1), r_osc(1,2), r_osc(1,3),'d','LineWidth',2)
% Position at OSCULATING ORBIT STOP
p = plot3(r_osc(end,1),r_osc(end,2),r_osc(end,3),'LineWidth',2);
%p.Color = 'c';
p.Marker = 'x';

% SRP WITH ECLIPSE
plot3(rEclipse(:,1),rEclipse(:,2),rEclipse(:,3),'b','LineWidth',1)
% Position at SRP with Ecl. stop
p = plot3(rEclipse(end,1),rEclipse(end,2),rEclipse(end,3),'LineWidth',2);
%p.Color = 'c';
p.Marker = 'x';
```

```
% ECLIPSE LOCATION
p3 =
plot3(rEclipse(eclipseYes,1),rEclipse(eclipseYes,2),rEclipse(eclipseYes,3),'*
','Linewidth',2);
p3.Color='k';

% SRP NO ECLIPSE
plot3(r_NoEclipse(:,1),r_NoEclipse(:,2),r_NoEclipse(:,3),'c','LineWidth',1)
% Position at SRP no Eclipse
p =
plot3(r_NoEclipse(end,1),r_NoEclipse(end,2),r_NoEclipse(end,3),'LineWidth',2)
;
%p.Color = 'c';
p.Marker = 'x';

legend('','Osculating Orbit','Mission start: all models','Mission stop:
Osculating','SRP with Eclipse',...
    'Mission stop: SRP with Eclipse','Eclipse','SRP no Eclipse','Mission
stop: SRP no Eclipse', ...
    'interpreter','latex','Location', 'best')
```

P1: Plot Apogee / Perigee (SRP with Eclipse only)

Extract outputs (RADIANS RADIANS RADIANS)

```
raanNew = COEsnew(:,1);
incNew = COEsnew(:,2);
wNew = COEsnew(:,3);
hNew = COEsnew(:,4);
eccNew = COEsnew(:,5);
thetaNew = COEsnew(:,6);

%..........Plot apogee/perigee change with time
% Find r, v vectors
r = zeros(3,length(timenew));
altitude = zeros(length(timenew),1);

for i = 1:length(timenew)
[r(:,i),~] =
r_and_v_from_COEs(raanNew(i),incNew(i),wNew(i),hNew(i),eccNew(i),thetaNew(i))
;
altitude(i) = norm(r(1:3,i)) - re;
end
r = r';

[max_altitude,imax] = findpeaks(altitude);
[~,imin] = findpeaks(-altitude);

min_altitude = zeros(length(imin),1);
for i = 1:length(imin)
```

```
    min_altitude(i) = norm(r(imin(i),1:3)) - re;
end

%..... Apogee and Perigee
apogee = [timenew(imax) max_altitude]; % Maximum altitudes and times
perigee = [timenew(imin) min_altitude]; % Min altitudes and times

% Plot apogee and perigee with time
figure()
plot(apogee(:,1)*sec2days,apogee(:,2),'LineWidth',2)
hold on
plot(perigee(:,1)*sec2days,perigee(:,2),'LineWidth',2)

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('Time [days]','Interpreter','latex');
yLab = ylabel('Altitude [km]','Interpreter','latex');
plotTitle = title("Change in Apogee and Perigee due to
SRP",'interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Apogee','Perigee', 'interpreter','latex','Location', 'best')
```

P1: Plot changes in COEs (SRP with AND with-out Eclipse)

```
%..... Extract outputs for SRP with NO ECLIPSE, plot overlay.
raanNew_noE = rad2deg(COEs_NoEclipse(:,1)); % degrees
incNew_noE = rad2deg(COEs_NoEclipse(:,2)); % degrees
wNew_noE = rad2deg(COEs_NoEclipse(:,3)); % degrees
hNew_noE = COEs_NoEclipse(:,4);
eccNew_noE = COEs_NoEclipse(:,5);
thetaNew_noE = rad2deg(COEs_NoEclipse(:,6)); % degrees

% Convert back to degrees for plots
raanNew = rad2deg(raanNew);
incNew = rad2deg(incNew);
wNew = rad2deg(wNew);

% For SRP WITH eclipse
dRAAN = raanNew - rad2deg(raan0);
dinc = incNew - rad2deg(inc0);
dw = wNew - rad2deg(w0);
dh = hNew - h0;
decc = eccNew - ecc0;

% For SRP WIHTOUT eclipse
```

```
dRAAN_noE = raanNew_noE - rad2deg(raan0);
dinc_noE = incNew_noE - rad2deg(inc0);
dw_noE = wNew_noE - rad2deg(w0);
dh_noE = hNew_noE - h0;
decc_noE = eccNew_noE - ecc0;

% Time vectors
time_eclipse = timenew .* sec2days; % in days
time_NoEclipse = timenew .* sec2days;

figure()
tiledlayout("vertical")

% Plot delta RAAN
nexttile
plot(time_eclipse,dRAAN,'LineWidth',2)
hold on
plot(time_NoEclipse,dRAAN_noE,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
yLab = ylabel('$\Omega - \Omega_0$','Interpreter','latex');
plotTitle = title('Right Ascension [deg]','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set(yLab,'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
grid on
legend('Eclipse','No Eclipse', 'interpreter','latex','Location', 'best')

% Plot delta omega
nexttile
plot(time_eclipse,dw,'LineWidth',2)
hold on
plot(time_NoEclipse,dw_noE,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
yLab = ylabel('$\omega - \omega_0$','Interpreter','latex');
plotTitle = title('Argument of Perigee [deg]','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set(yLab,'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
grid on

% Plot angular momentum
nexttile
plot(time_eclipse,dh,'LineWidth',2)
hold on
plot(time_NoEclipse,dh_noE,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
```

```
yLab = ylabel('$h - h_0$','Interpreter','latex');
plotTitle = title('Angular Momentum [$\frac{km^2}{s}
$]','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set(yLab,'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
grid on

% Plot eccentricity
nexttile
plot(time_eclipse,decc,'LineWidth',2)
hold on
plot(time_NoEclipse,decc_noE,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
yLab = ylabel('$ecc - ecc_0$','Interpreter','latex');
plotTitle = title('Eccentricity','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set(yLab,'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
grid on

% Plot delta inclination
nexttile
plot(time_eclipse,dinc,'LineWidth',2)
hold on
plot(time_NoEclipse,dinc_noE,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
xLab = xlabel('Days','Interpreter','latex');
yLab = ylabel('$inc - inc_0$','Interpreter','latex');
plotTitle = title('Inclination [deg]','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set(yLab,'FontName','Palatino Linotype')
set(xLab,'FontSize', 14)
grid on

% Overall title
sgtitle('Solar Radiation Perturbations during Aug. 10 - Aug. 23,
1997','interpreter','latex')
```

Problem #2.

```
addpath("C:/MATLAB_CODE/Orbits/")
disp(" ")
disp("~~~~~~~~~~~~~~ PROBLEM 2: SOLAR GRAVITY (see plots) ~~~~~~~~~~~~~~")

% Constants
```

```
re = 6378;  % radius of earth, km
muEarth = 398600; % grav parameter, earth; km3/s2
days2sec = 86400;
sec2days = 1/86400;
muSun = 132.712e9; % km3/s2

% Given parameters
h0 = 69084.1;           % km2/s
ecc0 = 0.741;           % whoa!
raan0 = 0;              % radians
inc0 = deg2rad(63.4);   % radians
w0 = deg2rad(270);      % radians
theta0 = 0;             % rad
a0 = 26553.4;           % km
T0 = 11.9616*3600;      % seconds
COEs = [raan0;inc0;w0;h0;ecc0;theta0];

% CALCULATE THE EFFECTS OF SOLAR BODY (n-body) OVER 60 DAYS

% Set up ODE
days = 700;

% Caleb's birthday for jd_initial
y = 2016;
m = 3;
d = 8;
UT = [02 50 00];

[jd_initial, ~, ~] = juliandateFunction(y,m,d,UT);
[~,~, R_sun] = solar_position(jd_initial);

tf = days*86400; % 60 days

%..... ode45 options
options = odeset('RelTol', 1e-12, 'AbsTol',1e-12);
tspan = [0 tf];
state = COEs;

%.....Call ode
tstart = tic;
[timenew, COEsnew] =
ode45(@VoP_nBody,tspan,state,options,jd_initial,muSun,muEarth);
tsol = toc(tstart);
disp("Time to run VoP for Solar Gravity model is: " + tsol + " seconds")

% HEART CHECKS:
% a: no secular, long-periodic, nor monthly variation
% perigee and n_hat == secular
% ecc, raan, inc: long periodic

%..... Extract outputs
raanNew = rad2deg(COEsnew(:,1)); % degrees
incNew = rad2deg(COEsnew(:,2)); % degrees
wNew = rad2deg(COEsnew(:,3)); % degrees
```

```
hNew = COEsnew(:,4);
eccNew = COEsnew(:,5);
thetaNew = rad2deg(COEsnew(:,6)); % degrees

% Take differences for plots
dRAAN = raanNew - rad2deg(raan0);
dinc = incNew - rad2deg(inc0);
dw = wNew - rad2deg(w0);
dh = hNew - h0;
decc = eccNew - ecc0;

% Time vectors
time = timenew .* sec2days; % in days

figure()
tiledlayout("vertical")

% Plot delta RAAN
nexttile
plot(time,dRAAN,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
yLab = ylabel('$\Omega - \Omega_0$','Interpreter','latex');
plotTitle = title('Right Ascension [deg]','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set(yLab,'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
grid on

% Plot delta omega
nexttile
plot(time,dw,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
yLab = ylabel('$\omega - \omega_0$','Interpreter','latex');
plotTitle = title('Argument of Perigee [deg]','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set(yLab,'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
grid on

% Plot angular momentum
nexttile
plot(time,dh,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
yLab = ylabel('$h - h_0$','Interpreter','latex');
plotTitle = title('Angular Momentum [$\frac{km^2}{s}
$]','interpreter','latex');
```

```
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set(yLab,'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
grid on

% Plot eccentricity
nexttile
plot(time,decc,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
yLab = ylabel('$ecc - ecc_0$','Interpreter','latex');
plotTitle = title('Eccentricity','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set(yLab,'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
grid on

% Plot delta inclination
nexttile
plot(time,dinc,'LineWidth',2)
% Graph pretty
ylim padded
xlim tight
xLab = xlabel('Days','Interpreter','latex');
yLab = ylabel('$inc - inc_0$','Interpreter','latex');
plotTitle = title('Inclination [deg]','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set(yLab,'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set(xLab,'FontSize', 14)
grid on

% Overall title
sgtitle('Solar Gravity Effects over 60 days','interpreter','latex')


%..........Plot apogee/perigee change with time
% Find r, v vectors
r = zeros(3,length(timenew));
altitude = zeros(length(timenew),1);

for i = 1:length(timenew)
[r(:,i),~] =
r_and_v_from_COEs(raanNew(i),incNew(i),wNew(i),hNew(i),eccNew(i),thetaNew(i))
;
altitude(i) = norm(r(1:3,i)) - re;
end
r = r';

[max_altitude,imax] = findpeaks(altitude);
```

```
[~,imin] = findpeaks(-altitude);

min_altitude = zeros(length(imin),1);
for i = 1:length(imin)
    min_altitude(i) = norm(r(imin(i),1:3)) - re;
end

%..... Apogee and Perigee
apogee = [timenew(imax) max_altitude]; % Maximum altitudes and times
perigee = [timenew(imin) min_altitude]; % Min altitudes and times

% Plot apogee and perigee with time
figure()
plot(apogee(:,1)*sec2days,apogee(:,2),'LineWidth',2)
hold on
plot(perigee(:,1)*sec2days,perigee(:,2),'LineWidth',2)

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('Time [days]','Interpreter','latex');
yLab = ylabel('Altitude [km]','Interpreter','latex');
plotTitle = title("Change in Apogee and Perigee",'interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Apogee','Perigee', 'interpreter','latex','Location', 'best')
```

Problem 3.0: Libration points

```
disp(" ")
disp("~~~~~~~~~~~~~~ PROBLEM 3: Libration Points and S/C Trajectories
~~~~~~~~~~~~~~")
clear all;

% Solve for Lagrange points using canonical units
mustar = 0.01215;

% L4 (let r1 = r2)
x.L4 = mustar - 0.5;
y.L4 = sqrt(3)/2;

% L5 (let r1 = r2)
x.L5 = x.L4;
y.L5 = -sqrt(3)/2;

% L3
syms t
xEqn.L3 = t - (1-mustar)/(t-mustar)^2 - mustar/(t - mustar + 1)^2;
x.L3 = vpasolve(xEqn.L3==0,t);
```

```
x.L3 = double(real(x.L3(1)));
y.L3 = 0;

% L1
xEqn.L1 = t + (1-mustar)/(t-mustar)^2 - mustar/(t - mustar + 1)^2;
x.L1 = vpasolve(xEqn.L1==0,t);
x.L1 = double(real(x.L1(1)));
y.L1 = 0;

% L2
xEqn.L2 = t + (1-mustar)/(t-mustar)^2 + mustar/(t - mustar + 1)^2;
x.L2 = vpasolve(xEqn.L2==0,t);
x.L2 = double(real(x.L2(1)));
y.L2 = 0;

% Plot libration Lagrange points
figure()
plot(mustar,0,'o','Linewidth',2) % earth
hold on
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
plot(x.L3,y.L3,'*','Linewidth',2) % L3
plot(x.L4,y.L4,'*','Linewidth',2) % L4
plot(x.L5,y.L5,'*','Linewidth',2) % L5

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('Lagrange points in the Earth-Moon
System','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Earth','Moon','L1','L2','L3','L4','L5',
'interpreter','latex','Location', 'best')
```

Problem 3.a: Trajectory A

```
% Define state vectors given in problem statement
state.a = [-1.05;0;0;0;-0.066429;0];

%..... ode45 options
options = odeset('RelTol', 1e-8, 'AbsTol',1e-8);
tf = 2*pi;
tspan = [0 tf];
```

```
%.....Call ode for SRP WITH ECLIPSE version
[~, statenew] = ode45(@CR3BP,tspan,state.a,options,mustar);
x.a = statenew(:,1);
y.a = statenew(:,2);
figure()
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
hold on
plot(x.a,y.a,'Linewidth',1) % Trajectory A

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('Trajectory A','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Moon','Trajectory A','interpreter','latex','Location', 'best')
```

Problem 3.b

Trajectory B Define state vectors given in problem statement

```
state.b = [-1.15;0;0;0;-0.0086882909;0];

%..... ode45 options
options = odeset('RelTol', 1e-8, 'AbsTol',1e-8);
tf = 29.46;
tspan = [0 tf];

%.....Call ode for SRP WITH ECLIPSE version
[~, statenew] = ode45(@CR3BP,tspan,state.b,options,mustar);
x.b = statenew(:,1);
y.b = statenew(:,2);

% Plot libration Lagrange points and Trajectory
figure()
plot(mustar,0,'o','Linewidth',2) % earth
hold on
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
plot(x.L3,y.L3,'*','Linewidth',2) % L3
plot(x.L4,y.L4,'*','Linewidth',2) % L4
plot(x.L5,y.L5,'*','Linewidth',2) % L5

% Plot trajectory state.b
```

```
plot(x.b,y.b,'Linewidth',1) % Trajectory B

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('Trajectory B','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Earth','Moon','L1','L2','L3','L4','L5','Trajectory B',
'interpreter','latex','Location', 'best')
```

Problem 3.c

Trajectory C Define state vectors given in problem statement

```
state.c = [0.10;0;0;-3.35;3;0];

%..... ode45 options
options = odeset('RelTol', 1e-8, 'AbsTol',1e-8);
tf = 3.6;
tspan = [0 tf];

%.....Call ode for SRP WITH ECLIPSE version
[~, statenew] = ode45(@CR3BP,tspan,state.c,options,mustar);
x.c = statenew(:,1);
y.c = statenew(:,2);

% Plot libration Lagrange points and Trajectory
figure()
plot(mustar,0,'o','Linewidth',2) % earth
hold on
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
% plot(x.L3,y.L3,'*','Linewidth',2) % L3
plot(x.L4,y.L4,'*','Linewidth',2) % L4
plot(x.L5,y.L5,'*','Linewidth',2) % L5

% Plot trajectory state.b
plot(x.c,y.c,'Linewidth',1) % Trajectory C

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
```

```
plotTitle = title('Trajectory C','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Earth','Moon','L1','L2','L4','L5','Trajectory C',
'interpreter','latex','Location', 'best')
```

Problem 3.d

Trajectory D Define state vectors given in problem statement

```
state.d = [0.10;0;0;-3.37;3;0];

%..... ode45 options
options = odeset('RelTol', 1e-8, 'AbsTol',1e-8);
tf = 6;
tspan = [0 tf];

%.....Call ode for SRP WITH ECLIPSE version
[~, statenew] = ode45(@CR3BP,tspan,state.d,options,mustar);
x.d = statenew(:,1);
y.d = statenew(:,2);

% Plot libration Lagrange points and Trajectory
figure()
plot(mustar,0,'o','Linewidth',2) % earth
hold on
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
plot(x.L3,y.L3,'*','Linewidth',2) % L3
plot(x.L4,y.L4,'*','Linewidth',2) % L4
plot(x.L5,y.L5,'*','Linewidth',2) % L5

% Plot trajectory state.d
plot(x.d,y.d,'Linewidth',1) % Trajectory D

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('Trajectory D','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
```

```
legend('Earth','Moon','L1','L2','L3','L4','L5','Trajectory D',
'interpreter','latex','Location', 'best')
```

Problem 3.e

Trajectory E Define state vectors given in problem statement

```
state.e = [0.10;0;0;-3.4;3;0];

%..... ode45 options
options = odeset('RelTol', 1e-8, 'AbsTol',1e-8);
tf = 6;
tspan = [0 tf];

%.....Call ode for SRP WITH ECLIPSE version
[~, statenew] = ode45(@CR3BP,tspan,state.e,options,mustar);
x.e = statenew(:,1);
y.e = statenew(:,2);

% Plot libration Lagrange points and Trajectory
figure()
plot(mustar,0,'o','Linewidth',2) % earth
hold on
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
plot(x.L3,y.L3,'*','Linewidth',2) % L3
plot(x.L4,y.L4,'*','Linewidth',2) % L4
plot(x.L5,y.L5,'*','Linewidth',2) % L5

% Plot trajectory state.e
plot(x.e,y.e,'Linewidth',1) % Trajectory E

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('Trajectory E','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Earth','Moon','L1','L2','L3','L4','L5','Trajectory E',
'interpreter','latex','Location', 'best')
```

Problem 3.f

Trajectory F Define state vectors given in problem statement

```
state.f = [1.25;0;0;0;0;0];

%..... ode45 options
options = odeset('RelTol', 1e-8, 'AbsTol',1e-8);
tf = 2*pi;
tspan = [0 tf];

%.....Call ode for SRP WITH ECLIPSE version
[~, statenew] = ode45(@CR3BP,tspan,state.f,options,mustar);
x.f = statenew(:,1);
y.f = statenew(:,2);

% Plot libration Lagrange points and Trajectory
figure()
plot(mustar,0,'o','Linewidth',2) % earth
hold on
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
plot(x.L3,y.L3,'*','Linewidth',2) % L3
plot(x.L4,y.L4,'*','Linewidth',2) % L4
plot(x.L5,y.L5,'*','Linewidth',2) % L5

% Plot trajectory state.e
plot(x.f,y.f,'Linewidth',1) % Trajectory f

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('Trajectory F','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Earth','Moon','L1','L2','L3','L4','L5','Trajectory F',
'interpreter','latex','Location', 'best')
```

Problem 3.g

Trajectory G Define state vectors given in problem statement

```
state.g = [-0.5;0;0;0;0;0];

%..... ode45 options
options = odeset('RelTol', 1e-8, 'AbsTol',1e-8);
tf = 2*pi;
tspan = [0 tf];
```

```
%.....Call ode for SRP WITH ECLIPSE version
[~, statenew] = ode45(@CR3BP,tspan,state.g,options,mustar);
x.g = statenew(:,1);
y.g = statenew(:,2);

% Plot libration Lagrange points and Trajectory
figure()
plot(mustar,0,'o','Linewidth',2) % earth
hold on
%p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
%plot(x.L1,y.L1,'*','Linewidth',2) % L1
%plot(x.L2,y.L2,'*','Linewidth',2) % L2
%plot(x.L3,y.L3,'*','Linewidth',2) % L3
%plot(x.L4,y.L4,'*','Linewidth',2) % L4
%plot(x.L5,y.L5,'*','Linewidth',2) % L5

% Plot trajectory state.g
plot(x.g,y.g,'Linewidth',1) % Trajectory G

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('Trajectory G','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Earth','Trajectory G', 'interpreter','latex','Location', 'best')
```

Problem 3.h

Trajectory H Define state vectors given in problem statement

```
state.h = [-1.1;0;0;0;0;0];

%..... ode45 options
options = odeset('RelTol', 1e-8, 'AbsTol',1e-8);
tf = 2*pi;
tspan = [0 tf];

%.....Call ode
[~, statenew] = ode45(@CR3BP,tspan,state.h,options,mustar);
x.h = statenew(:,1);
y.h = statenew(:,2);

% Plot libration Lagrange points and Trajectory
figure()
%plot(mustar,0,'o','Linewidth',2) % earth
```

```
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
hold on
plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
%plot(x.L3,y.L3,'*','Linewidth',2) % L3
%plot(x.L4,y.L4,'*','Linewidth',2) % L4
%plot(x.L5,y.L5,'*','Linewidth',2) % L5

% Plot trajectory state.g
plot(x.h,y.h,'Linewidth',1) % Trajectory H

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('Trajectory H','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Moon','L1','L2','Trajectory H', 'interpreter','latex','Location',
'best')

% Just for fun using a Halo orbit I found on JPL website
state.fun = [-1.6967233092486720;.1756007279014768e-23;
1.0136396075463166e-2;...
    -5.8932494929994690e-13;    1.2795419587406101; 0];

%..... ode45 options
options = odeset('RelTol', 1e-8, 'AbsTol',1e-8);
tf = 10*pi;
tspan = [0 tf];

%.....Call ode
[~, statenew] = ode45(@CR3BP,tspan,state.fun,options,mustar);
x.fun = statenew(:,1);
y.fun = statenew(:,2);

% Plot libration Lagrange points and Trajectory
figure()
plot(mustar,0,'o','Linewidth',2) % earth
hold on
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';

plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
plot(x.L3,y.L3,'*','Linewidth',2) % L3
plot(x.L4,y.L4,'*','Linewidth',2) % L4
```

```
plot(x.L5,y.L5,'*','Linewidth',2) % L5

% Plot trajectory state.g
plot(x.fun,y.fun,'Linewidth',1) % Trajectory H

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('Trajectory JPL [just for fun]','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Earth','Moon','L1','L2','L3','L4','L5','Trajectory fun',
'interpreter','latex','Location', 'best')
```

Problem 4. Energy Contour Plots

```
disp(" ")
disp("~~~~~~~~~~~~~~ PROBLEM 4: Zero Relative Speed Contours (Earth-Moon
System) ~~~~~~~~~~~~~~")

% Governing equation: velocity using (solving for) the Jacobi constant.
% v^2 = (x^2 + y^2) + 2*(1-mustar)/r1 + 2*mustar/r2 - c;

% .... Contours about the L points
mustar = 0.01215;

% .... Variation of x and y
nout = 1e3;
x.vect = linspace(2,-2,nout)';
y.vect = x.vect;
z = 0; % always
c.array = zeros(nout,nout);

for i = 1:length(x.vect)
    % Solve the velocity (v=0) equation for the Jacobi constant
    for j = 1:length(y.vect)
        r1 = sqrt(  (x.vect(i) - mustar)^2 + y.vect(j)^2 + z^2 );
        r2 = sqrt(  (x.vect(i) + 1 - mustar)^2 + y.vect(j)^2 + z^2 );
        c.array(j,i) = (x.vect(i)^2 + y.vect(j)^2) + 2*(1-mustar)/r1 +
2*mustar/r2; % == 0
           if c.array(j,i) > 4
                c.array(j,i) = NaN;
            end
    end
end

%..... Solve for Jacobian constant each libration point
```

```
x.libration = [x.L1 x.L2 x.L3 x.L4 x.L5];
y.libration = [y.L1 y.L2 y.L3 y.L4 y.L5];

for i = 1:5
    r1 = sqrt(  (x.libration(i) - mustar)^2 + y.libration(i)^2 + z^2 );
    r2 = sqrt(  (x.libration(i) + 1 - mustar)^2 + y.libration(i)^2 + z^2 );
    c.libration(i) = (x.libration(i)^2 + y.libration(i)^2) + 2*(1-mustar)/r1
+ 2*mustar/r2;
end


%[X,Y] = meshgrid(x.vect,y.vect,c.array);
figure
% Plot libration Lagrange points and Trajectory
plot(mustar,0,'o','Linewidth',2) % earth
hold on
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
plot(x.L3,y.L3,'*','Linewidth',2) % L3
plot(x.L4,y.L4,'*','Linewidth',2) % L4
plot(x.L5,y.L5,'*','Linewidth',2) % L5

levels = c.libration;

% .... Turn text on or off
% txt.L1 = num2str(c.libration(1));
% txt.L2 = num2str(c.libration(2));
% txt.L3 = num2str(c.libration(3));
% txt.L4 = num2str(c.libration(4));
% txt.L5 = num2str(c.libration(5));
% text(x.L1,y.L1,txt.L1)
% text(x.L2,y.L2,txt.L2)
% text(x.L3,y.L3,txt.L3)
% text(x.L4,y.L4,txt.L4)
% text(x.L5,y.L5,txt.L5)

% PLOT contours
contour(x.vect,y.vect,c.array,levels,'ShowText','on')
colorbar

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('Various C values for x and y
positions','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
```

```
grid on
legend('Earth','Moon','L1','L2','L3','L4','L5',...
    'interpreter','latex','Location', 'best')

% Just for fun
%figure()
%surf(x.vect,y.vect,c.array)
```

Contour for L1

```
% L1
for i = 1:length(x.vect)
    % Solve the velocity (v=0) equation for the Jacobi constant
    for j = 1:length(y.vect)
        r1 = sqrt(  (x.vect(i) - mustar)^2 + y.vect(j)^2 + z^2 );
        r2 = sqrt(  (x.vect(i) + 1 - mustar)^2 + y.vect(j)^2 + z^2 );
        c.array(j,i) = (x.vect(i)^2 + y.vect(j)^2) + 2*(1-mustar)/r1 +
2*mustar/r2; % == 0
            if c.array(j,i) > c.libration(1)*1.01
                c.array(j,i) = NaN;
            end
    end
end

figure
% Plot libration Lagrange points and Trajectory
plot(mustar,0,'o','Linewidth',2) % earth
hold on
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
plot(x.L3,y.L3,'*','Linewidth',2) % L3
plot(x.L4,y.L4,'*','Linewidth',2) % L4
plot(x.L5,y.L5,'*','Linewidth',2) % L5

% PLOT contours
levels = c.libration;
contourf(x.vect,y.vect,c.array,levels,'ShowText','on')
colorbar

% .... Turn text on or off
% txt.L1 = num2str(c.libration(1));
% txt.L2 = num2str(c.libration(2));
% txt.L3 = num2str(c.libration(3));
% txt.L4 = num2str(c.libration(4));
% txt.L5 = num2str(c.libration(5));
% text(x.L1,y.L1,txt.L1)
% text(x.L2,y.L2,txt.L2)
% text(x.L3,y.L3,txt.L3)
% text(x.L4,y.L4,txt.L4)
% text(x.L5,y.L5,txt.L5)
```

```
% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('C set for 1.01C for L1','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Earth','Moon','L1','L2','L3','L4','L5',...
    'interpreter','latex','Location', 'best')
```

Contour for L2

```
% L2
for i = 1:length(x.vect)
    % Solve the velocity (v=0) equation for the Jacobi constant
    for j = 1:length(y.vect)
        r1 = sqrt(  (x.vect(i) - mustar)^2 + y.vect(j)^2 + z^2 );
        r2 = sqrt(  (x.vect(i) + 1 - mustar)^2 + y.vect(j)^2 + z^2 );
        c.array(j,i) = (x.vect(i)^2 + y.vect(j)^2) + 2*(1-mustar)/r1 +
2*mustar/r2; % == 0
            if c.array(j,i) > c.libration(2)*1
                c.array(j,i) = NaN;
            end
    end
end

figure
% Plot libration Lagrange points and Trajectory
plot(mustar,0,'o','Linewidth',2) % earth
hold on
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
plot(x.L3,y.L3,'*','Linewidth',2) % L3
plot(x.L4,y.L4,'*','Linewidth',2) % L4
plot(x.L5,y.L5,'*','Linewidth',2) % L5

% PLOT contours
levels = c.libration;
contourf(x.vect,y.vect,c.array,levels,'ShowText','on')
colorbar

% .... Turn text on or off
% txt.L1 = num2str(c.libration(1));
% txt.L2 = num2str(c.libration(2));
% txt.L3 = num2str(c.libration(3));
% txt.L4 = num2str(c.libration(4));
```

```
% txt.L5 = num2str(c.libration(5));
% text(x.L1,y.L1,txt.L1)
% text(x.L2,y.L2,txt.L2)
% text(x.L3,y.L3,txt.L3)
% text(x.L4,y.L4,txt.L4)
% text(x.L5,y.L5,txt.L5)

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('C for L2','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Earth','Moon','L1','L2','L3','L4','L5',...
    'interpreter','latex','Location', 'best')
```

Contour for L3

```
% L3
for i = 1:length(x.vect)
    % Solve the velocity (v=0) equation for the Jacobi constant
    for j = 1:length(y.vect)
        r1 = sqrt(  (x.vect(i) - mustar)^2 + y.vect(j)^2 + z^2 );
        r2 = sqrt(  (x.vect(i) + 1 - mustar)^2 + y.vect(j)^2 + z^2 );
        c.array(j,i) = (x.vect(i)^2 + y.vect(j)^2) + 2*(1-mustar)/r1 +
2*mustar/r2; % == 0
            if c.array(j,i) > c.libration(3)*1.0
                c.array(j,i) = NaN;
            end
    end
end

figure
% Plot libration Lagrange points and Trajectory
plot(mustar,0,'o','Linewidth',2) % earth
hold on
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
plot(x.L3,y.L3,'*','Linewidth',2) % L3
plot(x.L4,y.L4,'*','Linewidth',2) % L4
plot(x.L5,y.L5,'*','Linewidth',2) % L5

% PLOT contours
levels = c.libration;
contourf(x.vect,y.vect,c.array,levels,'ShowText','on')
```

```
colorbar

% .... Turn text on or off
% txt.L1 = num2str(c.libration(1));
% txt.L2 = num2str(c.libration(2));
txt.L3 = num2str(c.libration(3));
% txt.L4 = num2str(c.libration(4));
% txt.L5 = num2str(c.libration(5));
% text(x.L1,y.L1,txt.L1)
% text(x.L2,y.L2,txt.L2)
text(x.L3,y.L3,txt.L3)
% text(x.L4,y.L4,txt.L4)
% text(x.L5,y.L5,txt.L5)

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('C for L3','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Earth','Moon','L1','L2','L3','L4','L5',...
    'interpreter','latex','Location', 'best')
```

Contour for L4

```
% L4
for i = 1:length(x.vect)
    % Solve the velocity (v=0) equation for the Jacobi constant
    for j = 1:length(y.vect)
        r1 = sqrt(  (x.vect(i) - mustar)^2 + y.vect(j)^2 + z^2 );
        r2 = sqrt(  (x.vect(i) + 1 - mustar)^2 + y.vect(j)^2 + z^2 );
        c.array(j,i) = (x.vect(i)^2 + y.vect(j)^2) + 2*(1-mustar)/r1 +
2*mustar/r2; % == 0
            if c.array(j,i) > c.libration(4)*1.001
                c.array(j,i) = NaN;
            end
    end
end

figure
% Plot libration Lagrange points and Trajectory
plot(mustar,0,'o','Linewidth',2) % earth
hold on
p = plot(mustar-1,0,'o','Linewidth',2); % moon
p.Color = 'k';
plot(x.L1,y.L1,'*','Linewidth',2) % L1
plot(x.L2,y.L2,'*','Linewidth',2) % L2
```

```
plot(x.L3,y.L3,'*','Linewidth',2) % L3
plot(x.L4,y.L4,'*','Linewidth',2) % L4
plot(x.L5,y.L5,'*','Linewidth',2) % L5

% PLOT contours
levels = c.libration;
contourf(x.vect,y.vect,c.array,levels,'ShowText','on')
colorbar

% .... Turn text on or off
% txt.L1 = num2str(c.libration(1));
% txt.L2 = num2str(c.libration(2));
% txt.L3 = num2str(c.libration(3));
txt.L4 = num2str(c.libration(4));
txt.L5 = num2str(c.libration(5));
% text(x.L1,y.L1,txt.L1)
% text(x.L2,y.L2,txt.L2)
% text(x.L3,y.L3,txt.L3)
text(x.L4,y.L4,txt.L4)
text(x.L5,y.L5,txt.L5)

% Graph pretty
ylim padded
xlim tight
xLab = xlabel('x [unitless]','Interpreter','latex');
yLab = ylabel('y [unitless]','Interpreter','latex');
plotTitle = title('C for L4','interpreter','latex');
set(plotTitle,'FontSize',14,'FontWeight','bold')
set(gca,'FontName','Palatino Linotype')
set([xLab, yLab],'FontName','Palatino Linotype')
set(gca,'FontSize', 9)
set([xLab, yLab],'FontSize', 14)
grid on
legend('Earth','Moon','L1','L2','L3','L4','L5',...
    'interpreter','latex','Location', 'best')
```

Published with MATLAB® R2023b

4: MATLAB FUNCTIONS

```
function [h, inc, RAAN, ecc, w, theta, epsilon, a, T] =
rv2COEs(r0Vect,v0Vect,mu)

% Outputs in degrees

% Finds the Six Classical Orbital Elements
r = norm(r0Vect);
v = norm(v0Vect);
vr = dot(r0Vect,v0Vect)/r;

% h, specific angular momentum
hVect = cross(r0Vect,v0Vect);
h = norm(hVect);

% inc, inclination
inc = acos(hVect(3)/h);

while inc > pi
    inc = inc - pi;
end
while inc < 0
    inc = inc + pi;
end

inc = inc*(180/pi);

% omega, right ascension of ascending node (RAAN)
NVect = cross([0;0;1],hVect);
N = norm(NVect);

if NVect(2) < 0
    RAAN = 2*pi - acos(NVect(1)/N);
else
    RAAN = acos(NVect(1)/N);
end

RAAN = RAAN*(180/pi);

% ecc, eccentricity
eccVect = (1/mu).*((v^2-(mu/r)).*r0Vect-r*vr.*v0Vect);
ecc = norm(eccVect);

% w, argument of perigee

if eccVect(3) < 0
    w = 2*pi - acos(dot(NVect,eccVect)/(N*ecc));
else
    w = acos(dot(NVect,eccVect)/(N*ecc));
end

w = w*(180/pi);
```

```
% theta, true anomaly
if vr >= 0
    theta = acos(dot(eccVect,r0Vect)/(ecc*r));
else
    theta = 2*pi - acos(dot(eccVect,r0Vect)/(ecc*r));
end

theta = theta*(180/pi);

% epsilon, specific energy
epsilon = (0.5*v^2) - (mu/r);
if epsilon > 0
    epsilon = nan;
end

% a, semi-major axis
a = -mu/(2*epsilon);

% T, period
T = 2*pi*sqrt((a^3)/mu);


end
```

Published with MATLAB® R2023b

```
function [juliandateReturn, UT_decimal, j0] = juliandateFunction(y,m,d,UT)
%   This function outputs the julian date (JD) with inputs year, month,
%   day, and GMT (UT)
% Author: Justin Self AERO 351 Fall 2022

% y = four digit year
% m = two digit month
% d = two digit day
% UT = array that contains the UT in the form [hour min sec]

j0 = 367*y - floor((7*(y+floor((m+9)/12)))/4) + floor((275*m)/9) + d +
1721013.5;

% j0 = Julian date at 0 hours UT (noon)
% Now ww need to convert HH:MM:SS to HH.HH (decimal hours)

UT_decimal = UT(1) + UT(2)/60 + UT(3)/3600; % 60 min / hour, 3600 sec / hour
juliandateReturn = j0 + (UT_decimal/24);

end
```

Published with MATLAB® R2023b

```
function [dCOEs] = SRP_VoP(time,state,jd_initial,A_sc,re,mu,mass_sc)

%{
Author: Justin Self
This function is for integration using ode45 to propogate the orbital
perturbation Solar Radiation Pressure (SRP) on a s/c of known parameters.

Assumptions: S/c is a cannonball (sphere, area = pi*r^2)

INPUTS:
    6x1 COEs initial state vector in RADIANS
    Order: [raan, inc,w, h, ecc, theta] (either row or column vect OK)
    jd_initial = initial Julian date.
    A_sc = cross-sectional (ram-direction) area of s/c affected by SRP
    * use tspan vector to handle span of dates, starting with jd_initial
    re = radius earth, km
    mu = mu earth, km3/s2

PROPOGATION METHOD:
Variation of Parameters method for propogating orbital perturbations.

OUTPUT: dCOEs for integration using ode45

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Justin Self, Cal Poly, Fall 2023: AERO 452 | Spaceflight Dynamics II
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%}
% Noemclature:
% curtis,ABERCROMBY ==  r,R; s,T; w,N
% ECI == X,Y,Z
```

Variation of Parameters Propogation Method Setup

```
raan = state(1);
inc = state(2);
w = state(3);
h = state(4);
ecc = state(5);
theta = state(6);

% Define argument of latitude
u = w + theta;

% Preallocate for speed
cr = cos(raan);
sr = sin(raan);
ci = cos(inc);
si = sin(inc);
cw = cos(w);
```

```
sw = sin(w);
ct = cos(theta);
st = sin(theta);
ti = tan(inc);
su = sin(u);
cu = cos(u);

% determine perifocal to GEO rotation matrix
QXx =   [  cr*cw - ci*sr*sw, cw*sr + ci*cr*sw, si*sw;
         - cr*sw - ci*cw*sr, ci*cr*cw - sr*sw, cw*si;
           si*sr,            -cr*si,           ci];
QXx = QXx';

% find r and v (Perifocal)
r_PF = (((h^2)/mu)*(1/(1+ecc*ct)))*[ct;st;0];
%v_PF = (mu/h)*[-st; (ecc+ct); 0];

% calc r and v relative to the geocentric reference frame
% GEO = ECI
rVect = QXx*r_PF;
%vVect = QXx*v_PF;

% Magnitude of r vector
r = norm(rVect);

%..........Update the Julian date
jd = jd_initial + time/86400; % seconds to days

%..........Find location of sun in ECI frame % THIS IS CURTIS CODE
[lambda,epsilon, R_S] = solar_position(jd);    % output in km

% Convert from degrees to radians
lambda = deg2rad(lambda);
epsilon = deg2rad(epsilon);

%..........Find earth-sun unit vectors from Curtis Eqns (10.105)
sl = sin(lambda);
se = sin(epsilon);
cl = cos(lambda);
ce = cos(epsilon);

%..........Define unit vectors
ur = sl*ce*cr*ci*su + sl*ce*sr*cu - cl*sr*ci*su + cl*cr*cu + sl*se*si*su;
us = sl*ce*cr*ci*cu - sl*ce*sr*su - cl*sr*ci*cu - cl*cr*su + sl*se*si*cu;
uw = -sl*ce*cr*si + cl*sr*si + sl*se*ci;

%..........Calculate value of shadow function nu (Find out if in eclispe or
not)
% if nu = 1, s/c NOT in eclipse
% if nu = 0; s/c IS IN ECLPISE

nu = checkeclipse(rVect,R_S,re);

%.......... Compute solar radiation perturbation
```

```
% Solar radiation pressure; Solar constant / speed of light
Psr = 4.57e-6; % Pa

% radiation pressure coefficient; 1<Cr<2
Cr = 1.2;

% ....SRP perturbation
psr = nu*Psr * Cr * (A_sc/mass_sc)/1000; % in km/s2 now
```

COEs

Calculate components of 'f' in Eqn (h) Recall: curtis,ABERCROMBY == r,R; s,T; w,N

```
% h
dh = -psr*r*us;

% ecc
decc = -psr * ((h/mu) * st*ur + (1/(mu*h)) * ((h^2 + mu*r)*ct +
mu*ecc*r)*us);

% theta
twobodymotion = h/r^2;
dtheta_pert = (-psr/(ecc*h)) * ( (h^2/mu)*ct*ur - ( (h^2/mu) + r)*st*us  );

% Final pert
dtheta = twobodymotion + dtheta_pert;

% inc
dinc =  -psr*(r/h) * cu * uw;

% raan
draan = -psr*((r*su) / (h*si)) * uw;

% w
dw = -psr * ( (-1/(ecc*h)) * ( (h^2/mu)*ct*ur - ( (h^2/mu) + r)*st*us ) -
((r*su) / (h*ti)) * uw);

% Final output
dCOEs = [draan;dinc;dw;dh;decc;dtheta];

end
```

Published with MATLAB® R2023b

```
function nu = checkeclipse(R_sc,R_sun,radiusBody)
%{
Author: Justin Self
November 19, 2023
This function (straight from Curtis p.526; Algorithm 10.3) takes:

INPUTS:
    radiusBody = radius of central body, km
    R_sc = orbital radius of spacecraft from central body, km (ECI, earth)
    R_sun = sun vector from central body frame (ECI for earth) km

OUTPUT:
    nu = 0 == s/c is in shadow
    nu = 1 == s/c is NOT in shadow
%}
re = radiusBody;
r = norm(R_sc);
rsun = norm(R_sun);

theta = acos( dot(R_sun,R_sc)/(rsun*r) );
theta1 = acos(re/r);
theta2 = acos(re/rsun);

if theta1 + theta2 <= theta
    nu = 0;
    %disp("Nu is: " + nu + "; s/c is in SHADOW")
else
    nu = 1;
    %disp("Nu is: " + nu + "; s/c NOT in shadow")
end
```

Published with MATLAB® R2023b

```
function [dCOEs] =
SRP_VoP_ignoreEclipse(time,state,jd_initial,A_sc,re,mu,mass_sc)

%{
Author: Justin Self
This function is for integration using ode45 to propogate the orbital
perturbation Solar Radiation Pressure (SRP) on a s/c of known parameters.

IGNORE ECLIPSE AS A WAY OF COMPARISON. THAT'S THE ONLY TIME WE SHOULD USE
THIS.

Assumptions: S/c is a cannonball (sphere, area = pi*r^2)

INPUTS:
    6x1 COEs initial state vector in RADIANS
    Order: [raan, inc,w, h, ecc, theta] (either row or column vect OK)
    jd_initial = initial Julian date.
    A_sc = cross-sectional (ram-direction) area of s/c affected by SRP
    * use tspan vector to handle span of dates, starting with jd_initial
    re = radius earth, km
    mu = mu earth, km3/s2

PROPOGATION METHOD:
Variation of Parameters method for propogating orbital perturbations.

OUTPUT: dCOEs for integration using ode45

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Justin Self, Cal Poly, Fall 2023: AERO 452 | Spaceflight Dynamics II
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%}
% Noemclature:
% curtis,ABERCROMBY ==  r,R; s,T; w,N
% ECI == X,Y,Z
```

Variation of Parameters Propogation Method Setup

```
raan = state(1);
inc = state(2);
w = state(3);
h = state(4);
ecc = state(5);
theta = state(6);

% Define argument of latitude
u = w + theta;

% Preallocate for speed
cr = cos(raan);
```

```
sr = sin(raan);
ci = cos(inc);
si = sin(inc);
cw = cos(w);
sw = sin(w);
ct = cos(theta);
st = sin(theta);
ti = tan(inc);
su = sin(u);
cu = cos(u);

% determine perifocal to GEO rotation matrix
QXx =   [  cr*cw - ci*sr*sw, cw*sr + ci*cr*sw, si*sw;
         - cr*sw - ci*cw*sr, ci*cr*cw - sr*sw, cw*si;
           si*sr,            -cr*si,           ci];
QXx = QXx';

% find r and v (Perifocal)
r_PF = (((h^2)/mu)*(1/(1+ecc*ct)))*[ct;st;0];
%v_PF = (mu/h)*[-st; (ecc+ct); 0];

% calc r and v relative to the geocentric reference frame
% GEO = ECI
rVect = QXx*r_PF;
%vVect = QXx*v_PF;

% Magnitude of r vector
r = norm(rVect);

%..........Update the Julian date
jd = jd_initial + time/86400; % seconds to days

%..........Find location of sun in ECI frame
[lambda,epsilon, R_S] = solar_position(jd);    % output in km

% Convert from degrees to radians
lambda = deg2rad(lambda);
epsilon = deg2rad(epsilon);

%..........Find earth-sun unit vectors from Curtis Eqns (10.105)
sl = sin(lambda);
se = sin(epsilon);
cl = cos(lambda);
ce = cos(epsilon);

%..........Define unit vectors
ur = sl*ce*cr*ci*su + sl*ce*sr*cu - cl*sr*ci*su + cl*cr*cu + sl*se*si*su;
us = sl*ce*cr*ci*cu - sl*ce*sr*su - cl*sr*ci*cu - cl*cr*su + sl*se*si*cu;
uw = -sl*ce*cr*si + cl*sr*si + sl*se*ci;

%..........Calculate value of shadow function nu (Find out if in eclispe or
not)

% NOPE NOT TODAY
```

```
nu = 1; % ignore eclipse

%.......... Compute solar radiation perturbation
% Solar radiation pressure; Solar constant / speed of light
Psr = 4.57e-6; % Pa

% radiation pressure coefficient; 1<Cr<2
Cr = 1.2;

% ....SRP perturbation
psr = nu*Psr * Cr * (A_sc/mass_sc)/1000; % in km/s2 now
```

COEs

Calculate components of 'f' in Eqn (h) Recall: curtis,ABERCROMBY == r,R; s,T; w,N

```
% h
dh = -psr*r*us;

% ecc
decc = -psr * ((h/mu) * st*ur + (1/(mu*h)) * ((h^2 + mu*r)*ct +
mu*ecc*r)*us);

% theta
twobodymotion = h/r^2;
dtheta_pert = (-psr/(ecc*h)) * ( (h^2/mu)*ct*ur - ( (h^2/mu) + r)*st*us  );

% Final pert
dtheta = twobodymotion + dtheta_pert;

% inc
dinc =  -psr*(r/h) * cu * uw;

% raan
draan = -psr*((r*su) / (h*si)) * uw;

% w
dw = -psr * ( (-1/(ecc*h)) * ( (h^2/mu)*ct*ur - ( (h^2/mu) + r)*st*us ) -
((r*su) / (h*ti)) * uw);

% Final output
dCOEs = [draan;dinc;dw;dh;decc;dtheta];

end
```

Published with MATLAB® R2023b

```
function dCOEs = vop_NoPerts(time,COEs,mu)

%{
Variation of Parameters method for propogating orbital perturbations.

INPUTS: 6x1 COEs initial state vector in RADIANS
        wE = angular velocity of the earth; ECI frame (3x1)

OUTPUT: dCOEs for integration using ode45

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Justin Self, Cal Poly, Fall 2023: AERO 452 | Spaceflight Dynamics II
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%}

% Noemclature:
% curtis,ABERCROMBY ==  r,R; s,T; w,N
% ECI == X,Y,Z

% Bring in COEs (RADIANS)
raan = COEs(1);
inc = COEs(2);
w = COEs(3);
h = COEs(4);
ecc = COEs(5);
theta = COEs(6);

% Calcuate r,v from COEs each time step.

% % Run rv from cose function here plainly for speed
%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%
muEarth = 398600;

% Preallocate for speed
cr = cos(raan);
sr = sin(raan);
ci = cos(inc);
si = sin(inc);
cw = cos(w);
sw = sin(w);
ct = cos(theta);
st = sin(theta);
ti = tan(inc);

% determine perifocal to GEO rotation matrix
QXx =   [  cr*cw - ci*sr*sw, cw*sr + ci*cr*sw, si*sw;
         - cr*sw - ci*cw*sr, ci*cr*cw - sr*sw, cw*si;
           si*sr,            -cr*si,           ci];
QXx = QXx';

% find r and v (Perifocal)
r_PF = (((h^2)/muEarth)*(1/(1+ecc*ct)))*[ct;st;0];
```

```
v_PF = (muEarth/h)*[-st; (ecc+ct); 0];

% calc r and v relative to the geocentric reference frame
% GEO = ECI
rVect = QXx*r_PF;
v = QXx*v_PF;

r = norm(rVect);
```

Direction Cosine Matrix from XYZ to RTN

```
% Q{ECI-->RTN} each time

% Convenient definition: "argument of latitude"
u = w + theta;
cu = cos(u);
su = sin(u);

% Drag acceleration IN ECI
p = [0;0;0];

% Change from ECI to RTN frame
R_hat = rVect/r;
N_hat = cross(rVect,v)/norm(cross(rVect,v));
T_hat = cross(N_hat,R_hat);
QXr = [R_hat T_hat N_hat];
QXr = QXr';

% /END ALTERNATE METHOD

p = QXr*p;

R = p(1);
T = p(2);
N = p(3);
```

COEs

```
% Recall: curtis,ABERCROMBY ==  r,R; s,T; w,N

% h
dh = r*T;

% ecc
decc = (h/mu) * st*R + (1/(mu*h)) * ((h^2 + mu*r)*cos(theta) + mu*ecc*r)*T;

% theta
twobodymotion = h/r^2;
dtheta_pert = (1/(ecc*h)) * ( (h^2/mu)*cos(theta)*R - ( (h^2/mu) +
r)*sin(theta)*T  );

% Final pert
dtheta = twobodymotion + dtheta_pert;
```

```
% inc
dinc =  (r/h) * cu * N;

% raan
draan = ((r*su) / (h*si)) * N;

% w
dw = -dtheta_pert - ((r*su) / (h*ti)) * N;

% Final output
dCOEs = [draan;dinc;dw;dh;decc;dtheta];

end
```

Published with MATLAB® R2023b

```
function [r,v] = rv_from_COESNew(COEs)


% Calcuate r,v from COEs each time step.

% % Run rv from cose function here plainly for speed
%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%
mu = 398600;

% Bring in COEs
raan = COEs(1);
inc = COEs(2);
w = COEs(3);
h = COEs(4);
ecc = COEs(5);
theta = COEs(6);

% Preallocate for speed
cr = cos(raan);
sr = sin(raan);
ci = cos(inc);
si = sin(inc);
cw = cos(w);
sw = sin(w);
ct = cos(theta);
st = sin(theta);

% determine perifocal to GEO rotation matrix
QXx =   [  cr*cw - ci*sr*sw, cw*sr + ci*cr*sw, si*sw;
         - cr*sw - ci*cw*sr, ci*cr*cw - sr*sw, cw*si;
           si*sr,            -cr*si,           ci];
QXx = QXx';

% find r and v (Perifocal)
r_PF = (((h^2)/mu)*(1/(1+ecc*ct)))*[ct;st;0];
v_PF = (mu/h)*[-st; (ecc+ct); 0];

% calc r and v relative to the geocentric reference frame
% GEO = ECI
r = QXx*r_PF;
v = QXx*v_PF;
```

Published with MATLAB® R2023b

```
function [xx,yy,zz] = earth_sphere(varargin)

%
%
%
% EARTH_SPHERE Generate an earth-sized sphere.
%   [X,Y,Z] = EARTH_SPHERE(N) generates three (N+1)-by-(N+1)
%   matrices so that SURFACE(X,Y,Z) produces a sphere equal to
%   the radius of the earth in kilometers. The continents will be
%   displayed.
%
%   [X,Y,Z] = EARTH_SPHERE uses N = 50.
%
%   EARTH_SPHERE(N) and just EARTH_SPHERE graph the earth as a
%   SURFACE and do not return anything.
%
%   EARTH_SPHERE(N,'mile') graphs the earth with miles as the unit rather
%   than kilometers. Other valid inputs are 'ft' 'm' 'nm' 'miles' and 'AU'
%   for feet, meters, nautical miles, miles, and astronomical units
%   respectively.
%
%   EARTH_SPHERE(AX,...) plots into AX instead of GCA.
%
%  Examples:
%    earth_sphere('nm') produces an earth-sized sphere in nautical miles
%
%    earth_sphere(10,'AU') produces 10 point mesh of the Earth in
%    astronomical units
%
%    h1 = gca;
%    earth_sphere(h1,'mile')
%    hold on
%    plot3(x,y,z)
%      produces the Earth in miles on axis h1 and plots a trajectory from
%      variables x, y, and z
%   Clay M. Thompson 4-24-1991, CBM 8-21-92.
%   Will Campbell, 3-30-2010
```

Input Handling

```
[cax,args,nargs] = axescheck(varargin{:}); % Parse possible Axes input
error(nargchk(0,2,nargs)); % Ensure there are a valid number of inputs
% Handle remaining inputs.
% Should have 0 or 1 string input, 0 or 1 numeric input
j = 0;
k = 0;
n = 50; % default value
units = 'km'; % default value
for i = 1:nargs
    if ischar(args{i})
        units = args{i};
        j = j+1;
```

```
    elseif isnumeric(args{i})
        n = args{i};
        k = k+1;
    end
end
if j > 1 || k > 1
    error('Invalid input types')
end
```

Calculations

Scale factors

```
Scale = {'km' 'm'  'mile'            'miles'           'nm'
'au'                  'ft';
         1    1000 0.621371192237334 0.621371192237334 0.539956803455724
6.6845871226706e-009 3280.839895};
% Identify which scale to use
try
    myscale = 6378.1363*Scale{2,strcmpi(Scale(1,:),units)};
catch %#ok<*CTCH>
    error('Invalid units requested. Please use m, km, ft, mile, miles, nm,
or AU')
end

% -pi <= theta <= pi is a row vector.
% -pi/2 <= phi <= pi/2 is a column vector.
theta = (-n:2:n)/n*pi;
phi = (-n:2:n)'/n*pi/2;
cosphi = cos(phi); cosphi(1) = 0; cosphi(n+1) = 0;
sintheta = sin(theta); sintheta(1) = 0; sintheta(n+1) = 0;
x = myscale*cosphi*cos(theta);
y = myscale*cosphi*sintheta;
z = myscale*sin(phi)*ones(1,n+1);
```

Plotting

```
if nargout == 0
    cax = newplot(cax);
    % Load and define topographic data
    load('topo.mat','topo','topomap1');
    % Rotate data to be consistent with the Earth-Centered-Earth-Fixed
    % coordinate conventions. X axis goes through the prime meridian.
    % http://en.wikipedia.org/wiki/
Geodetic_system#Earth_Centred_Earth_Fixed_.28ECEF_or_ECF.29_coordinates
    %
    % Note that if you plot orbit trajectories in the Earth-Centered-
    % Inertial, the orientation of the contintents will be misleading.
    topo2 = [topo(:,181:360) topo(:,1:180)]; %#ok<NODEF>

    % Define surface settings
    props.FaceColor= 'texture';
    props.EdgeColor = 'none';
```

```
    props.FaceLighting = 'phong';
    props.Cdata = topo2;
    % Create the sphere with Earth topography and adjust colormap
    surface(x,y,z,props,'parent',cax)
    colormap(topomap1)
% Replace the calls to surface and colormap with these lines if you do
% not want the Earth's topography displayed.
%     surf(x,y,z,'parent',cax)
%     shading flat
%     colormap gray

    % Refine figure
    axis equal
    xlabel(['X [' units ']'])
    ylabel(['Y [' units ']'])
    zlabel(['Z [' units ']'])
    view(127.5,30)
else
    xx = x; yy = y; zz = z;
end
```

```
function [dCOEs] = VoP_nBody(time,state,jd_initial,muSun,muEarth)

%{
Author: Justin Self
This function is for integration using ode45 to propogate the orbital
perturbation n-body (solar here) on a s/c of known parameters.

Assumptions: S/c is a cannonball (sphere, area = pi*r^2)

INPUTS:
    6x1 COEs initial state vector in RADIANS
    Order: [raan, inc,w, h, ecc, theta] (either row or column vect OK)
    jd_initial = initial Julian date.
    A_sc = cross-sectional (ram-direction) area of s/c affected by SRP
    * use tspan vector to handle span of dates, starting with jd_initial
    re = radius earth, km
    mu = mu earth, km3/s2

PROPOGATION METHOD:
Variation of Parameters method for propogating orbital perturbations.

OUTPUT: dCOEs for integration using ode45

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Justin Self, Cal Poly, Fall 2023: AERO 452 | Spaceflight Dynamics II
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%}
% Noemclature:
% curtis,ABERCROMBY ==  r,R; s,T; w,N
% ECI == X,Y,Z
```

Variation of Parameters Propogation Method Setup

```
mu = muEarth;
raan = state(1);
inc = state(2);
w = state(3);
h = state(4);
ecc = state(5);
theta = state(6);

% Define argument of latitude
u = w + theta;

% Preallocate for speed
cr = cos(raan);
sr = sin(raan);
ci = cos(inc);
si = sin(inc);
```

```
cw = cos(w);
sw = sin(w);
ct = cos(theta);
st = sin(theta);
ti = tan(inc);
su = sin(u);
cu = cos(u);

% determine perifocal to GEO rotation matrix
QXx =   [  cr*cw - ci*sr*sw, cw*sr + ci*cr*sw, si*sw;
         - cr*sw - ci*cw*sr, ci*cr*cw - sr*sw, cw*si;
           si*sr,            -cr*si,           ci];
QXx = QXx';

% find r and v (Perifocal)
r_PF = (((h^2)/mu)*(1/(1+ecc*ct)))*[ct;st;0];
v_PF = (mu/h)*[-st; (ecc+ct); 0];

% calc r and v relative to the geocentric reference frame
% GEO = ECI
rVect = QXx*r_PF;
vVect = QXx*v_PF;

% Magnitude of r vector
r = norm(rVect);

%..........Update the Julian date
jd = jd_initial + time/86400; % seconds to days

%..........Find location of sun in ECI frame
[~,~, R_S] = solar_position(jd);    % output in km
rS = norm(R_S);

% ..... Compute position of sun vector relative to the s/c
rSs = R_S - rVect;
rss = norm(rSs);

x = 2*R_S-rVect;
q = dot(rVect,x)/rS^2; % see Curtis Eqn. 10.131

% ..... Compute F(q) from Eq. (F.3)
Fq = q*(q^2 - 3*q + 3) / (1 + (1 - q)^(3/2));

% ..... Compute perturbing acceleration of sun
p = muSun * (Fq*R_S - rVect) / rss^3;

% .... Compute unit vectors of r,s,w (RTN)
R = rVect/r;
N = cross(rVect,vVect)/norm(cross(rVect,vVect));
T = cross(N,R)/norm(cross(N,R));
QXr = [R T N];
QXr = QXr';

p = QXr*p;
```

```
R = p(1);
T = p(2);
N = p(3);
```

COEs

Calculate components of 'f' in Eqn (h) Recall: curtis,ABERCROMBY == r,R; s,T; w,N

```
% h
dh = r*T;

% ecc
decc = (h/mu) * st*R + (1/(mu*h)) * ((h^2 + mu*r)*cos(theta) + mu*ecc*r)*T;

% theta
twobodymotion = h/r^2;
dtheta_pert = (1/(ecc*h)) * ( (h^2/mu)*cos(theta)*R - ( (h^2/mu) +
r)*sin(theta)*T  );

% Final pert
dtheta = twobodymotion + dtheta_pert;

% inc
dinc =  (r/h) * cu * N;

% raan
draan = ((r*su) / (h*si)) * N;

% w
dw =  (-1/(ecc*h)) * ( (h^2/mu)*ct*R - ( (h^2/mu) + r)*st*T ) - ((r*su) /
(h*ti)) * N;

% Final output
dCOEs = [draan;dinc;dw;dh;decc;dtheta];

end
```

Published with MATLAB® R2023b

```
% Circular Restricted 3 Body Problem ode function using canonical units
% Justin Self
% Cal Poly
% AERO452: Spaceflight Dynamics II
% Fall 2023

%{
INPUTS:     state = 6x1 vector: [r;v]
OUTPUTS:    new state vector propogated through orbit
%}

function dstate = CR3BP(time,state,mustar)

% define state inputs
x = state(1);
y = state(2);
z = state(3);
xdot = state(4);
ydot = state(5);
zdot = state(6);

% Define equation parameters
r1 = sqrt((x - mustar)^2 + y^2 + z^2);
r2 = sqrt((x + 1 - mustar)^2 + y^2 + z^2);

% EOMs
xddot = x + 2*ydot - (1-mustar)*(x-mustar) / (r1^3) - mustar*(x + 1 -
mustar)/(r2^3);
yddot = y - 2*xdot - (1-mustar)*y/(r1^3) - mustar*y/(r2^3);
zddot = -(1-mustar)*z/(r1^3) - mustar*z/(r2^3);

dstate = [xdot;ydot;zdot;xddot;yddot;zddot];

end % funct
```

Published with MATLAB® R2023b