```
% Perturbed Canonical CR3BP Dynamics with Linear State Feedback for Gateway - L4 SPO↙
Transfer Orbit
clc;
clear;
close all;

% Global Variables to be Saved to Workspace
global time_set t_final counter_1 counter_2 a_srp_set a_solar_set x_ref Poles↙
Days_Until_Deployment %#ok<*GVMIS>

%%

% Initial Conditions of Desired SPO
x_ref = [4.8784941344943100E-1; 1.3374629269150202E+0; 1.3237124335174697E-33;...
    7.0666616478037292E-1; -3.4384320489591991E-1; -1.2317770203396586E-33];

% Desired Poles (Change This Accordingly to Adjust Delta V and Transfer Time)
Poles = [-6 -6 -6 -4 -4 -4];

% Number of Days Until Deployment from Gateway
Days_Until_Deployment = 7;

% Number of Days For Orbit Propagation (Simulation). This is not the Orbit-to-Orbit↙
Transfer Time
Sim_Days = 15;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%↙
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

% Constants
Re = 6378.1; % Earth's Radius [km]
Rm = 1738.1; % Moon's Radius [km]
G = 6.67259e-20; % Gravitational Constant [km^3/(kg*s^2)]
M_e = 5.97219e24; % Earth's Mass [kg]
M_m = 7.34767e22; % Moon's Mass [kg]
M_s = 1.9891e30; % Sun's Mass [kg]
r_e_m = 384400; % Earth-Moon Distance [km]

% Global Variables
time_set = []; % Time [s]
counter_1 = 1; % Counter to Save Global Variables
counter_2 = 1; %#ok<NASGU> % Counter to Update Conttrol Gain
a_solar_set = []; % Acceleration due to Solar Gravity [km/s^2]
a_srp_set = []; % SRP Acceleration [km/s^2]

% Characteristic Mass [kg]
m_star = M_e+M_m;
% Characteristic Length [km]
l_star = r_e_m;
% Characteristic Time [s]
t_star = sqrt(l_star^3/(G*m_star));
% Characteristic Velocity [km/s]
v_star = l_star/t_star;

% Nondimensional Masses of Primaries
mu_e = M_e/(M_m+M_e); % Earth's Nondimensional Mass
mu_m = M_m/(M_m+M_e); % Moon's Nondimensional Mass
```

```
mu_s = M_s/(M_m+M_e); % Sun's Nondimensional Mass

% Simulation Time (Feel Free to Change)
t0 = 0/t_star; % Nondimensional Initial Simulation Time
t_step = 1000/t_star; % Nondimensional Time Step
t_final = 3600*24*Sim_Days/t_star; % Nondimensional Final Time

% Nondimensional Initial Conditions [Synodic Frame]

% Initial Condition of the NRHO Proposed for NASA's Gateway
x0 = [1.0273132294452039E+0;    3.8675798104630464E-27; -1.8555133506611556E-1;...
      2.2046034458619281E-14;   -1.1449886110106612E-1; 4.0753793820150663E-13];

% Simulation
options = odeset('MaxStep',t_step,'RelTol',1E-10,'AbsTol',1E-10);
[time, x] = ode15s(@Dynamics,t0:t_step:t_final,[x_ref; x0],options);

% Output

% Reference Orbit Nondimensional Position [Synodic Frame]
x_pos_ref = x(:,1);
y_pos_ref = x(:,2);
z_pos_ref = x(:,3);

% Reference Orbit Nondimensional Velocity [Synodic Frame]
x_vel_ref = x(:,4);
y_vel_ref = x(:,5);
z_vel_ref = x(:,6);

% Actual Orbit Nondimensional Position [Synodic Frame]
x_pos = x(:,7);
y_pos = x(:,8);
z_pos = x(:,9);

% Actual Orbit +Nondimensional Velocity [Synodic Frame]
x_vel = x(:,10);
y_vel = x(:,11);
z_vel = x(:,12);

% Control Acceleration [km/s^2]

u_set = []; % Control Acceleration Vector
u_set_norm = []; % Norm of Control Acceleration Vector

counter_2 = 1;

for ii = 1:1:length(time)

    if ii == 1

        % Linearized System in the Vicinity of L2
        A = [0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1; 7.3818 0 0 0 2 0; 0 -2.1909 0 -2 0 0;↙
0 0 0.0122 0 0 0]; % State Matrix
        B = [zeros(3,3); eye(3)]; % Input Matrix

        % Pole Placement
        K = place(A,B,Poles); % State Feedback Control Gain
```

```
        % States of the Equilibrium Point [nondimensional]
        x_e = [1.15568217; 0; 0; 0; 0; 0]; % For L2

    end

    if norm([x_pos(ii)-1+mu_m; y_pos(ii); z_pos(ii)])*l_star > 100000 && counter_2 == 1

        % Linearized System in the Vicinity of L4
        gamma = 3*sqrt(3)/2*(mu_m-1/2);
        A = [0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1; 3/4 -gamma 0 0 2 0; -gamma 9/4 0 -2 0↙
0; 0 0 -1 0 0 0]; % State Matrix
        B = [zeros(3,3); eye(3)]; % Input Matrix

        % Pole Placement
        K = place(A,B,Poles); % State Feedback Control Gain

        % States of the Equilibrium Point [nondimensional]
        x_e = [(r_e_m/2-mu_m*r_e_m)/l_star; (sqrt(3)/2*r_e_m)/l_star; 0; 0; 0; 0]; % For↙
L4

        counter_2 = 0;

    end

    % Control Law

    if time(ii)*t_star/(3600*24) < Days_Until_Deployment

        u = [0; 0; 0]; % This Shuts Down the Controller to Simulate Gateway's NRHO

    else

        % Tracking Error
        error = (x(ii,7:12)'-x_e) - (x(ii,1:6)'-x_e);

        % Control Law
        u = - K*error*(v_star/t_star);

    end

    % Saving Variables
    u_set = [u_set; u']; %#ok<*AGROW>
    u_set_norm = [u_set_norm norm(u)];

end

% Figures

figure (1) % Trajectories

plot3(x_pos,y_pos,z_pos,'LineWidth',1.5);
hold on
plot3(x_pos(1),y_pos(1),z_pos(1),'*','LineWidth',3);
hold on
plot3(x_pos_ref,y_pos_ref,z_pos_ref,'--','LineWidth',1.5);
hold on
plot3((r_e_m/2-mu_m*r_e_m)/l_star, (sqrt(3)/2*r_e_m)/l_star,0,'*','LineWidth',3);
hold on
```

```
title('Trajectories in Synodic Frame','Interpreter','latex');
xlabel('X [nondim]','Interpreter','latex');
ylabel('Y [nondim]','Interpreter','latex');
zlabel('Z [nondim]','Interpreter','latex');

% Adding Moon to 3D Plot

[X,Y,Z] = sphere;
Moon = surface((Rm*X+379729)/l_star, Rm*Y/l_star, -Rm*Z/l_star);
alpha = 1; % Transparency of Moon's Surface
cdata = imread('Moon_Texture.jpg');
set(Moon, 'FaceColor', 'texturemap', 'CData', cdata, 'FaceAlpha', alpha, 'EdgeColor',↙
'none');
hold on

% Adding Earth to 3D Plot

earth = surf((Re*X-4671)/l_star, Re*Y/l_star, -Re*Z/l_star);
alpha = 1; % Transparency of Earth's Surface
cdata = imread('Earth_Texture.jpg');
set(earth, 'FaceColor', 'texturemap', 'CData', cdata, 'FaceAlpha', alpha, 'EdgeColor',↙
'none');
hold off

legend('S/C Trajectory','S/C Initial Condition','Reference↙
SPO','L$_4$','Interpreter','latex');
axis equal
grid on
grid minor
xlim([-2.5 2.5]);
fontsize(gcf, 24, 'points')
fontname(gcf,"Times New Roman")
set(gcf,'color','w');

figure (2) % SRP Acceleration on Spacecraft [m/s^2]

plot(time_set/(3600*24),vecnorm(a_srp_set'),'LineWidth',1.5)
title('Synodic Frame','Interpreter','latex');
ylabel('SRP Acceleration [km/s$^2$]','Interpreter','latex');
xlabel('Time [days]','Interpreter','latex');
xlim([0 t_final*t_star/(3600*24)])
grid on
grid minor
fontsize(gcf, 24, 'points')
fontname(gcf,"Times New Roman")
set(gcf,'color','w');

figure (3) % Solar Gravity Acceleration on Spacecraft [m/s^2]

plot(time_set/(3600*24),vecnorm(a_solar_set'),'LineWidth',1.5)
title('Synodic Frame','Interpreter','latex');
ylabel('Solar Gravity [km/s$^2$]','Interpreter','latex');
xlabel('Time [days]','Interpreter','latex');
xlim([0 t_final*t_star/(3600*24)])
grid on
grid minor
fontsize(gcf, 24, 'points')
fontname(gcf,"Times New Roman")
```

```
set(gcf,'color','w');

figure (4) % Control Input

subplot(311)
plot(time/(3600*24)*t_star,u_set(:,1),'LineWidth',1.5)
title('Control Acceleration in Synodic Frame','Interpreter','latex');
ylabel('$u_x(t)$ [km/s$^2$]','Interpreter','latex');
xlim([0 t_final*t_star/(3600*24)])
grid on
grid minor
fontsize(gcf, 24, 'points')
fontname(gcf,"Times New Roman")
set(gcf,'color','w');

subplot(312)
plot(time/(3600*24)*t_star,u_set(:,2),'LineWidth',1.5)
ylabel('$u_y(t)$ [km/s$^2$]','Interpreter','latex');
xlim([0 t_final*t_star/(3600*24)])
grid on
grid minor
fontsize(gcf, 24, 'points')
fontname(gcf,"Times New Roman")
set(gcf,'color','w');

subplot(313)
plot(time/(3600*24)*t_star,u_set(:,3),'LineWidth',1.5)
ylabel('$u_z(t)$ [km/s$^2$]','Interpreter','latex');
xlabel('Time [days]','Interpreter','latex');
xlim([0 t_final*t_star/(3600*24)])
grid on
grid minor
fontsize(gcf, 24, 'points')
fontname(gcf,"Times New Roman")
set(gcf,'color','w');

% Calculating Delta V [km/s]

Delta_V_set = []; % Delta V

for ii = 2:1:length(time)

    time_int = time(1:1:ii)*t_star; % Time Interval [s]
    u_set_norm_int = u_set_norm(1:1:ii); % Control Input for Time Interval [km/s^2]
    Delta_V = trapz(time_int,u_set_norm_int); % [km/s]
    Delta_V_set = [Delta_V_set Delta_V]; % [km/s]

end

figure (5) % Delta V

plot(time(2:1:end)*t_star/(3600*24),Delta_V_set,'LineWidth',1.5)
title('Synodic Frame','Interpreter','latex');
ylabel('$\Delta$V [km/s]','Interpreter','latex');
xlabel('Time [days]','Interpreter','latex');
xlim([0 t_final*t_star/(3600*24)])
grid on
grid minor
```

```
fontsize(gcf, 24, 'points')
fontname(gcf,"Times New Roman")
set(gcf,'color','w');

ratio = l_star/t_star;
data_needed = [time(:,1)*t_star x_pos(:,1)*l_star y_pos(:,1)*l_star z_pos(:,1)*l_star↙
x_vel(:,1)*ratio y_vel(:,1)*ratio z_vel(:,1)*ratio ];

csvwrite('trajectory.csv',data_needed);

%%
% % Specify the filename of the CSV file
% csvFilename = 'ephemeris_data.csv';

% Specify the filename for the STK ephemeris file
ephemerisFilename = 'ephemeris_data.e';

% % Read data from the CSV file
% data = csvread(csvFilename, 1, 0); % Assumes the first row contains headers

% Open the ephemeris file for writing
fid = fopen(ephemerisFilename, 'w');

% Write header section
fprintf(fid, 'stk.v.12.0\n');
fprintf(fid, '# WrittenBy    MATLAB\n'); % Modify as needed
fprintf(fid, '\n');
fprintf(fid, 'BEGIN Ephemeris\n');
fprintf(fid, 'NumberOfEphemerisPoints\t\t %d\n', 1297);
fprintf(fid, 'ScenarioEpoch\t\t  31 Dec 2026 16:00:00.000000\n'); % Adjust as needed
fprintf(fid, 'InterpolationMethod\t\t Lagrange\n'); % Adjust as needed
fprintf(fid, 'CentralBody\t\t Earth\n'); % Adjust as needed
fprintf(fid, 'CoordinateSystem\t EarthLunaBarycentered\n'); % Custom coordinate system
fprintf(fid, 'EphemerisTimePosVel\t\t # Time x y z vx vy vz\n');
fprintf(fid, '\n');

% Write data section

for i = 1:size(data_needed, 1)
    fprintf(fid, '%.16e %.16e %.16e %.16e %.16e %.16e %.16e\n', time(i)*t_star,x_pos(i)↙
*l_star,y_pos(i)*l_star,z_pos(i)*l_star,x_vel(i)*ratio,y_vel(i)*ratio,z_vel(i)*ratio);
end


% Write footer section
fprintf(fid, 'END Ephemeris\n');

% Close the ephemeris file
fclose(fid);
close all
disp('Ephemeris file created successfully.');

% plot3(x_pos,y_pos,z_pos)
```

```
% Coordinate Frame Transformation ECI to Synodic Reference Frame
function r_Syn = ECI2Synodic(t,r_ECI)

% Input: Position Vector in ECI Basis
% Output: Position Vector in Synodic Frame Basis

% Constants
G = 6.67259e-20; % Gravitaional Constant [km^3/(kg*s^2)]
m1 = 5.97219e24; % Earth's Mass [kg]
m2 = 7.34767e22; % Moon's Mass [kg]
r12 = 384400; % Earth-Moon Distance [km]

% Characteristic Mass [kg]
m_star = m1+m2;
% Characteristic Length [km]
l_star = r12;
% Characteristic Time [s]
t_star = sqrt(l_star^3/(G*m_star));
% Dimentionless Time
theta = t/t_star;
% Inclination of Synodic Frame wrt Equator [rad]
inc = (5.145+23.44)*pi/180;

% Rotation Matrices
R_3 = [cos(theta) sin(theta) 0;...
    -sin(theta) cos(theta) 0;...
    0 0 1];
R_1 = [1 0 0;...
    0 cos(inc) sin(inc);...
    0 -sin(inc) cos(inc)];

% Rotation for Position Vector (Synodic Reference Frame Centered at Earth's Center) [km]
r_Syn = (R_3*R_1)*r_ECI;

% Translation for Position Vector (Synodic Reference Frame Centered at Barycenter) [km]
r_Syn = [r_Syn(1)-4671; r_Syn(2); r_Syn(3)];

end
```

```
% Computes Julian Date Given Calendar Date Vector
function d = Julian_Date(date)

% Calendar Date Vector
yr = date(1); % Year
mon = date(2); % Month
day= date(3); % Day

if length(date) == 6

    % Calendar Date Vector
    hr = date(4); % Hour
    min = date(5); % Minute
    sec = date(6); % Second

    % Julian Date
    d = 367.0*yr - floor((7*(yr + floor((mon+9)/12.0)))*0.25)...
        + floor(275*mon/9.0) + day + 1721013.5 + ((sec/60.0 + min)/60.0 + hr)/24.0;
else

    % Julian Date
    d = 367.0*yr - floor((7*(yr + floor((mon + 9)/12.0)))*0.25)...
        + floor(275*mon/9.0) + day + 1721013.5;
end

end
```

```
% CR3BP Dynamics to Integrate Using Numerical Solvers
function output = Dynamics(t,states)

% Global Variables to be Saved to Workspace
global time_set t_final counter_1 counter_2 a_srp_set a_solar_set Poles↙
Days_Until_Deployment %#ok<*GVMIS>

% Persistent Variables
persistent K x_e

% Progress
progress = t/t_final*100;
fprintf('%d%% \n',round(progress));

% Constants
G = 6.67259e-20; % Gravitational Constant [km^3/(kg*s^2)]
M_e = 5.97219e24; % Earth's Mass [kg]
M_m = 7.34767e22; % Moon's Mass [kg]
M_s = 1.9891e30; % Sun's Mass [kg]
r_e_m = 384400; % Earth-Moon Distance [km]
mu_m = M_m/(M_m+M_e); % Moon's Nondimensional Mass
mu_s = M_s/(M_m+M_e); % Sun's Nondimensional Mass

% Characteristic Quantities
M_star = M_e+M_m; % Characteristic Mass [kg]
l_star = r_e_m; % Characteristic Length [km]
t_star = sqrt(l_star^3/(G*M_star)); % Characteristic Time [s]
v_star = l_star/t_star; % Characteristic Velocity [km/s]

if t*t_star/(3600*24) < Days_Until_Deployment

    % Turns Off Perturbations
    on_off = 0;

else

    % Turns On Perturbations
    on_off = 1;

end

% Reference Unperturbed States

% Position of Reference Orbit [Nondimensional]
x_ref = states(1);
y_ref = states(2);
z_ref = states(3);

% Velocity of Reference Orbit [Nondimensional]
x_dot_ref = states(4);
y_dot_ref = states(5);
z_dot_ref = states(6);

% Earth's Nondimensional Relative Reference Position Vector wrt S/C
r_e_sc_ref = [x_ref+mu_m; y_ref; z_ref];

% Moon's Nondimensional Relative Reference Position Vector wrt S/C
r_m_sc_ref = [x_ref-1+mu_m; y_ref; z_ref];
```

```
% Reference Position Derivatives [Nondimensional]
output(1) = x_dot_ref;
output(2) = y_dot_ref;
output(3) = z_dot_ref;

% Reference Velocity Derivatives [Nondimensional]
output(4) = (2*y_dot_ref + x_ref - (1-mu_m)*(x_ref+mu_m)/(norm(r_e_sc_ref))^3 - mu_m*↙
(x_ref-1+mu_m)/(norm(r_m_sc_ref))^3);
output(5) = (-2*x_dot_ref + y_ref - (1-mu_m)*y_ref/(norm(r_e_sc_ref))^3 - mu_m*y_ref/↙
(norm(r_m_sc_ref))^3);
output(6) = (-(1-mu_m)*z_ref/(norm(r_e_sc_ref))^3 - mu_m*z_ref/(norm(r_m_sc_ref))^3);

% Actual States

% Actual Position [Nondimensional]
x = states(7);
y = states(8);
z = states(9);

% Actual Velocity [Nondimensional]
x_dot = states(10);
y_dot = states(11);
z_dot = states(12);

% Earth's Nondimensional Relative Position Vector wrt S/C
r_e_sc = [x+mu_m; y; z];

% Moon's Nondimensional Relative Position Vector wrt S/C
r_m_sc = [x-1+mu_m; y; z];

% SRP Acceleration (Cannonball Model)

AU = 149597870.691; % Astronomical Unit [km]
W = 1367; % Solar Flux [W/m^2]
c = 299792458; % Speed of Light in Vacuum [m/s]
C_R = 1.5; % Reflection Coefficient [Nondimensional]
r = 1; % S/C Radius [m]
Area = pi*r^2; % Exposed Area to Sunlight [m^2]
m = 14; % S/C Mass [kg]

% Sun's Position Vector in ECI [km]
r_sun_ECI = Sun_Position(t*t_star);

% Sun's Position Vector in Synodic Frame [km]
r_sun_syn = ECI2Synodic(t*t_star,r_sun_ECI);

% S/C Position Vector in Synodic Frame [km]
r_sat = l_star*[x; y; z];

% Sun's Unit Vector in the Synodic Frame Relative to S/C [km]
e_sun_hat = (r_sun_syn-r_sat)/norm(r_sun_syn-r_sat);

% SRP Acceleration [km/s^2]
a_srp = -on_off*W/c*(AU/norm(r_sun_syn-r_sat))^2*C_R*Area/m/1000*e_sun_hat;

% Nondimensional SRP Acceleration
a_srp_non = a_srp/(v_star/t_star);
```

```
% Solar Gravity

% Nondimensional Sun's Position Vector Components in the Synodic Frame
x_s = r_sun_syn(1)/l_star; y_s = r_sun_syn(2)/l_star; z_s = r_sun_syn(3)/l_star;

% Nondimensional Sun's Distance to the Earh-Moon Barycenter
a_s = norm(r_sun_syn)/l_star;

% Nondimensional S/C Distance to the Sun
r_s_sc = norm([x; y; z] - [x_s; y_s; z_s]);

% Nondimensional Gravitaional Pull of the Sun
a_solar = on_off*[-mu_s*(x-x_s)/(norm(r_s_sc))^3 - mu_s*x_s/a_s^3;...
    -mu_s*(y-y_s)/(norm(r_s_sc))^3 - mu_s*y_s/a_s^3;...
    -mu_s*(z-z_s)/(norm(r_s_sc))^3 - mu_s*z_s/a_s^3];

% Controller

if t == 0

    % Linearized System in the Vicinity of L2
    A = [0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1; 7.3818 0 0 0 2 0; 0 -2.1909 0 -2 0 0; 0 0↙
0.0122 0 0 0]; % State Matrix; % State Matrix
    B = [zeros(3,3); eye(3)]; % Input Matrix

    % Pole Placement
    K = place(A,B,Poles); % State Feedback Control Gain

    % States of the Equilibrium Point [nondimensional]
    x_e = [1.15568217; 0; 0; 0; 0; 0]; % For L2

end


if norm(r_m_sc)*l_star > 100000 && counter_2 == 1

        % Linearized System in the Vicinity of L4
        gamma = 3*sqrt(3)/2*(mu_m-1/2);
        A = [0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1; 3/4 -gamma 0 0 2 0; -gamma 9/4 0 -2 0↙
0; 0 0 -1 0 0 0]; % State Matrix
        B = [zeros(3,3); eye(3)]; % Input Matrix

        % Pole Placement
        K = place(A,B,Poles); % State Feedback Control Gain

        % States of the Equilibrium Point [nondimensional]
        x_e = [(r_e_m/2-mu_m*r_e_m)/l_star; (sqrt(3)/2*r_e_m)/l_star; 0; 0; 0; 0]; % For↙
L4

        counter_2 = 0;

end

% Control Law

if t*t_star/(3600*24) < Days_Until_Deployment
```

```
    u = [0; 0; 0]; % This Shuts Down the Controller to Simulate Gateway's NRHO

else

    % Tracking Error
    error = (states(7:12)-x_e) - (states(1:6)-x_e);

    % Control Law
    u = - K*error;

end

% Position Derivatives [Nondimensional]
output(7) = x_dot;
output(8) = y_dot;
output(9) = z_dot;

% Velocity Derivatives [Nondimensional]
output(10) = (2*y_dot + x - (1-mu_m)*(x+mu_m)/(norm(r_e_sc))^3 - mu_m*(x-1+mu_m)/(norm↙
(r_m_sc))^3) + a_srp_non(1) + a_solar(1) + u(1);
output(11) = (-2*x_dot + y - (1-mu_m)*y/(norm(r_e_sc))^3 - mu_m*y/(norm(r_m_sc))^3) +↙
a_srp_non(2) + a_solar(2) + u(2);
output(12) = (-(1-mu_m)*z/(norm(r_e_sc))^3 - mu_m*z/(norm(r_m_sc))^3) + a_srp_non(3) +↙
a_solar(3) + u(3);

% Function's Output to Integrate
output = output';

% Variables Saved to Workspace
time_set(counter_1) = t*t_star; % Time [s]
a_srp_set(counter_1,1:3) = a_srp; % SRP Acceleration [km/s^2]
a_solar_set(counter_1,1:3) = a_solar*v_star/t_star; % Acceleration due to Solar Gravity↙
[km/s^2]
counter_1 = counter_1 + 1; % Counter

end
```

```
% Coordinate Frame Transformation Synodic Reference Frame to ECI
function r_ECI = Synodic2ECI(t,r_Syn)

% Input: Position Vector in Synodic Frame Basis
% Output: Position Vector in ECI Basis

% Constants
G = 6.67259e-20; % Gravitaional Constant [km^3/(kg*s^2)]
m1 = 5.97219e24; % Earth's Mass [kg]
m2 = 7.34767e22; % Moon's Mass [kg]
r12 = 384400; % Earth-Moon Distance [km]

% Characteristic Mass [kg]
m_star = m1+m2;
% Characteristic Length [km]
l_star = r12;
% Characteristic Time [s]
t_star = sqrt(l_star^3/(G*m_star));
% Dimentionless Time
theta = t/t_star;
% Inclination of Synodic Frame wrt Equator [rad]
inc = (5.145+23.44)*pi/180;

% Rotation Matrix
R_3 = [cos(theta) sin(theta) 0;...
    -sin(theta) cos(theta) 0;...
    0 0 1];
R_1 = [1 0 0;...
    0 cos(inc) sin(inc);...
    0 -sin(inc) cos(inc)];

% Translating Synodic Reference Frame Barycenter to Earth's Center
r_Syn_prime = [r_Syn(1)+4671; r_Syn(2); r_Syn(3)];

% Final Rotation for Position Vector (ECI) [km]
r_ECI = (R_3*R_1)'*r_Syn_prime;

end
```

```
% Computes the Sun's Position Vector in km from an ECI Perspective
function r_sun = Sun_Position(t)

% Constants
AU = 149597870.691; % Earth-Sun Distance [km]

% Initial Time: 01 Jan 2022 at 0:00 AM UT (Modify The Initial Date as Needed)
initial_date = datetime(2022,1,1,0,0,0);
current_date = initial_date + seconds(t);

% Current Date Vector
[current_year, currect_month, current_day] = ymd(current_date);
[current_hour, current_minute, current_second] = hms(current_date);

Date = [current_year currect_month current_day current_hour current_minute↙
current_second];

% Julian Date
JD = Julian_Date(Date);

% Number of Days Since J2000
n = JD - 2451545;

% Mean Anomaly of the Sun [deg]
M = 357.529 + 0.98560023*n;
M = mod(M,360);

% Mean Longitude of the Sun [deg]
L = 280.459 + 0.98564736*n;
L = mod(L,360);

% Apparent Solar Ecliptic Longitude [deg]
lambda = L + 1.915*sind(M) + 0.0200*sind(2*M);
lambda = mod(lambda,360);

% Obliquity [deg]
epsilon = 23.439 - 3.56*(10^-7)*n;

% Unit Vector Direction from Earth to Sun
u_hat = [cosd(lambda); sind(lambda)*cosd(epsilon); sind(lambda)*sind(epsilon)];

% Distance b/t Sun and Earth [km]
r_sun_norm = (1.00014 - 0.0167*cosd(M) - 0.000140*cosd(2*M))*AU;

% Output: Sun's Geocentric Position Vector [km]
r_sun = r_sun_norm*u_hat;

end
```