Table of Contents


```
clear; clc; close all;
```

Constants

```
G = 6.6742e-11; % gravitational constant [m^3/kg/s^2]
M_Earth = 5.974e24; % mass of Earth [kg]
M_Luna = 7.348e22; % mass of Moon [kg]
r_12 = 384400e3; % distance between Earth and Moon [m]

% Gravitational parameters
Mu_Earth = G * M_Earth; % [m^3/s^2]
Mu_Luna  = G * M_Luna; % [m^3/s^2]

% Distances of Earth and Moon from barycenter
Pi_Earth = M_Earth / (M_Earth + M_Luna);
Pi_Luna = M_Luna / (M_Earth + M_Luna);
d_Earth = -Pi_Luna * r_12; % Earth position offset [m]
d_Luna = Pi_Earth * r_12; % Moon position offset [m]
```

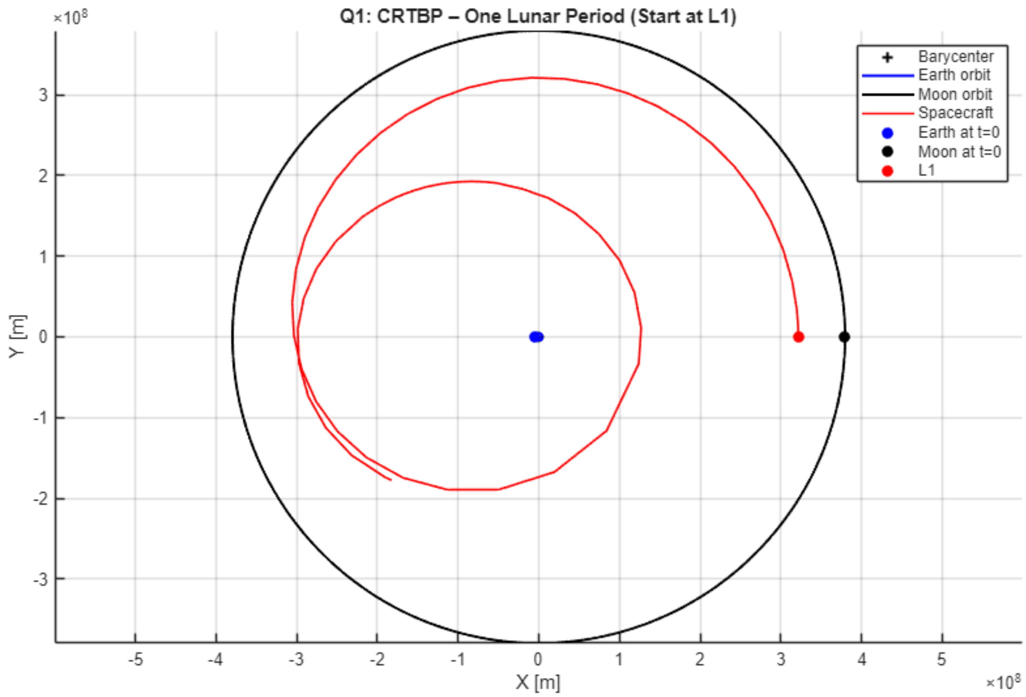
Question 1

```
% Motion and period
Omega = sqrt(G * (M_Earth + M_Luna) / r_12^3); % angular velocity [rad/s]
T_sys = 2 * pi / Omega; % lunar period [s]

% Lagrange Point L1
mu = M_Luna/(M_Earth + M_Luna);
d_from_moon = r_12*(mu/3)^(1/3);
x_guess = d_Luna - d_from_moon;

L1_fun = @(x) (Mu_Earth/(x - d_Earth)^2 - Mu_Luna/(d_Luna - x)^2 - Omega^2 *
x);
xL1 = fzero(L1_fun, x_guess);

% Initial position at L1
r0x = xL1;
r0y = 0;
r0z = 0;
```

```
% Initial velocity at L1
v0x = 0;
v0y = Omega * xL1;
v0z = 0;

% State vector
r0 = [r0x; r0y; r0z];
v0 = [v0x; v0y; v0z];
y0 = [r0; v0];

% Time span: one lunar cycle
tspan = [0 T_sys];

% Solve system
[t, y] = ode113(@(t,y) crtbp_inertial(t, y, Mu_Earth, Mu_Luna, d_Earth,
d_Luna, Omega), tspan, y0);

% Smooth plot lines
tt = linspace(0, T_sys, 2000).';
R_E = [d_Earth * cos(Omega * tt), d_Earth * sin(Omega * tt)];
R_M = [d_Luna * cos(Omega * tt), d_Luna * sin(Omega * tt)];

% Run Simulink model
simOut = sim('AER821_LAB_1_S', 'StopTime', num2str(T_sys));

% Get x position
x_sim = squeeze(simOut.x_sc.Data);

% Get y position
y_sim = squeeze(simOut.y_sc.Data);

% Get time
t_sim = simOut.x_sc.Time;

% Inertial-frame plot with spacecraft motion from Simulink
figure;
hold on;
plot(0,0,'k+','LineWidth',1.5); % barycenter
plot(R_E(:,1), R_E(:,2), 'b','LineWidth',1.5); % Earth path
plot(R_M(:,1), R_M(:,2), 'k','LineWidth',1.5); % Moon path
plot(x_sim, y_sim, 'r','LineWidth',1.2); % spacecraft path
plot(d_Earth,0,'bo','MarkerFaceColor','b'); % Earth at t0
plot(d_Luna, 0,'ko','MarkerFaceColor','k'); % Moon at t0
plot(xL1,0,'ro','MarkerFaceColor','r'); % L1
legend('Barycenter','Earth orbit','Moon orbit','Spacecraft','Earth at
t=0','Moon at t=0','L1');
xlabel('X [m]');
ylabel('Y [m]');
title('Q1: CRTBP - One Lunar Period (Start at L1)');
axis equal;
grid on;
hold off;

% Discussion Question 1
```

```
% Simulate the system motion in Simulink, calculated in the inertial frame.
You
% will need to specify initial conditions for the spacecraft. Using the
textbook as a guide select initial
% conditions for the appropriate orbital system (see the introduction
above). You should be able to
% find masses and distances for these bodies online or in print(please cite
your sources). Clearly
% state these initial conditions and generate a plot showing the position of
the two primaries and the
% spacecraft over one period of the primaries% motion. Comment on these
results

% Answer
% The initial conditions for this simulation were selected by placing the
spacecraft at the Lagrange Point
% L1 of the Earth-Luna system. This location was determined numerically
using the CRTBP equations and
% provides a reasonable test case, since it lies along the Earth-Moon line
where gravitational and
% centrifugal forces balance. The spacecraft was initialized with zero out-
of-plane position and velocity,
% and its tangential velocity was set to the rotating-frame angular velocity
at L1. These choices were
% critical because the CRTBP is highly sensitive to initial conditions; even
small deviations can lead to
% significantly different trajectories.

% The system of equations of motion was implemented in Simulink, as shown in
the block diagram. The model
% uses two sets of nested integrators: the inner set integrates
accelerations to obtain velocities, while
% the outer set integrates velocities to obtain positions. A custom MATLAB
Function block computes
% accelerations using the gravitational attractions from Earth and the Moon
along with the systems angular
% velocity. This makes a straightforward process to modify constants (e.g.,
masses, distances, angular
% velocity) for different planetary systems while maintaining the same
overall model.

% The plot shows the barycenter at the origin, with Earth and Moon following
their expected circular orbits.
% The spacecraft was initialized at the L1 Lagrange Point with a tangential
velocity in the +y direction,
% corresponding to the system’s angular rotation. While this setup reflects
the equilibrium condition in
% the rotating frame, in the inertial frame it introduces a nonzero velocity
that causes the spacecraft to
% drift. As the simulation progresses, the spacecraft does not remain fixed
at L1 but is gradually pulled
% inward by the gravitational fields of Earth and Moon. Its trajectory
curves toward their orbital paths,
% demonstrating the strong forces produced by the primaries and how
```

```
sensitive spacecraft motion is to
% initial conditions near collinear Lagrange points.
```

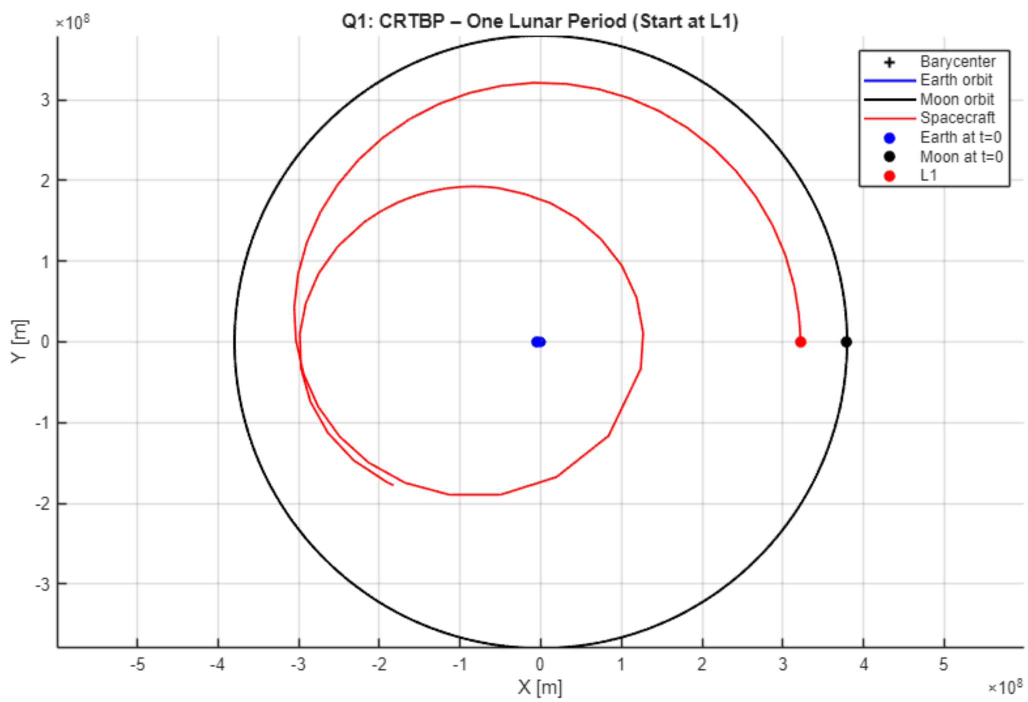


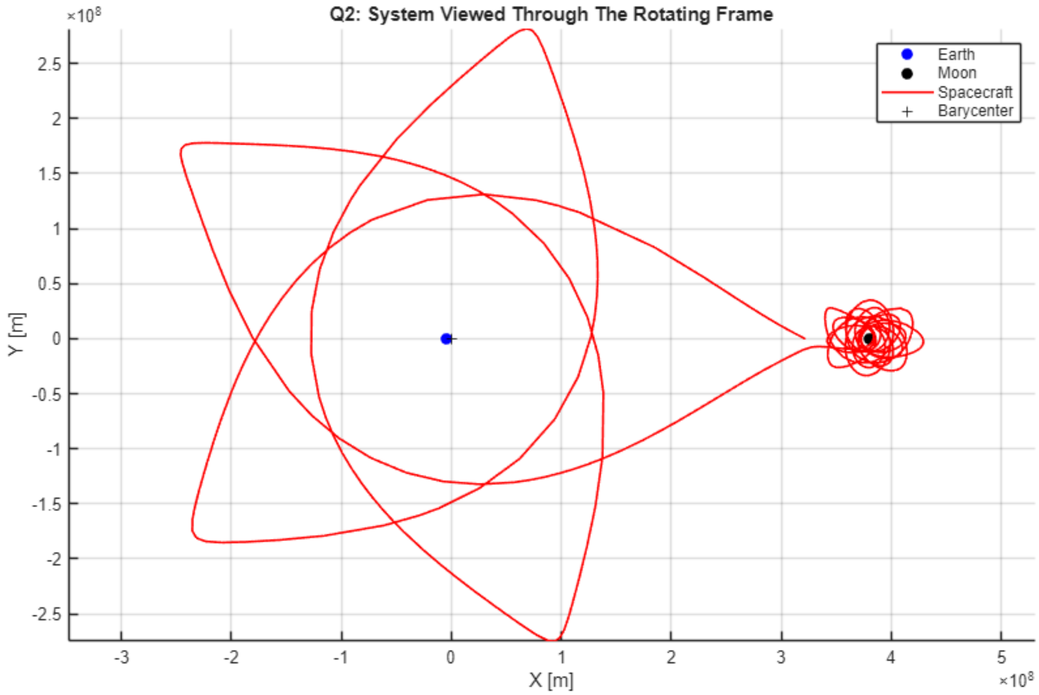
Question 2

```
% Redefine the time of the system to better display the orbit of the
% spacecraft in the rotating frame

tspan = tspan * 5;                          % Changes the time span to four
lunar cycles
[t, y] = ode113(@(t,y) crtbp_inertial(t, y, Mu_Earth, Mu_Luna, d_Earth,
d_Luna, Omega), tspan, y0);    % Reruns the derivation of the x and y
coordinates using the 5 cycles span

% Define spacecraft position arrays and preallocate

X_SC_Rot = zeros(length(t),1);              % Preallocate for x components of
the spacecraft in the rotating frame
Y_SC_Rot= zeros(length(t),1);               % Preallocate for y components of
the spacecraft in the rotating frame

% Creates a Loop to Transform The Inertial Frame Spacecraft Positions to the
rotated frame

for i = 1:length(t)                         % Creates the index

    % Creates the transformation matrix

    R = [cos(Omega * t(i)), sin(Omega * t(i));
```

```
        -sin(Omega * t(i)), cos(Omega * t(i))];

    SC_Pos_Inertial = y(i,1:2)';        % Selects the x and y coordinates
for the spacecraft in the inertial frame at time equal to index (i), and
stores it as array, then transposes for later algebra
    SC_Pos_Rot = R * SC_Pos_Inertial;   % Transfroms these coordinates
through multiplication with the matrix and stores as new array

    X_SC_Rot(i) = SC_Pos_Rot(1);        % Creates list of x coordinates post
rotation
    Y_SC_Rot(i) = SC_Pos_Rot(2);        % Creates list of y coordinates post
rotation

end

% Since in the rotating from, the earth and the moon do not move, they are
fixed at their initial position, this is performed below.

E_Stationary = [d_Earth; 0];     % Earth's stationary position
Luna_Stationary  = [d_Luna;  0]; % Moon's stationary position

% Plotting the system

figure;
hold on;
plot(E_Stationary(1), E_Stationary(2), 'bo', 'MarkerFaceColor', 'b');
% Plots the Earth
plot(Luna_Stationary(1), Luna_Stationary(2), 'ko', 'MarkerFaceColor', 'k');
% Plots the Moon
plot(X_SC_Rot, Y_SC_Rot, 'r','LineWidth',1.2);
% Plots the spacecraft's path
plot(0, 0, 'k+');
% Plots the Barycenter of the system
xlabel('X [m]');
% Creates the x axis label
ylabel('Y [m]');
% Creates the y axis label
legend('Earth', 'Moon', 'Spacecraft', 'Barycenter');
% Creates the legend
title('Q2: System Viewed Through The Rotating Frame');
% Creates the title
grid on;
% Turns on the background grid on the plot
axis equal;
% Sets the units of both axis equal to eachother
hold off;

% Discussion Question 2
% Do these match your expectations? Discuss your results. You may need to
% adjust the Simulink model to get your results to match expectations.
Discuss your findings during
% this process

% Answer
```

```
% These results align with expectations. Initially, the spacecraft's
trajectory
% shows it orbiting the Earth, similar to what is observed in the inertial
frame.
% In this plot, however, the orbit appears to rotate due to the rotating
reference
% frame, even though the rotation itself is not directly visible.
Eventually, as
% the spacecraft enters the Moon's gravitational field, it becomes captured
and
% begins to orbit the Moon.
%
% One confusing aspect is that extending the simulation duration in Simulink
% does not consistently show the spacecraft being captured by the Moon,
whereas
% plotting the results using ODE113 does. I was unable to determine the exact
% cause of this discrepancy, although the capture behavior intuitively seems
% like the correct physical outcome.
%
% I did not need to modify the Simulink model to produce these results, as I
used
% the differential equations from Part One to model the system. The only
significant
% challenge was constructing the array to store the X and Y coordinates in a
form
% that was algebraically compatible, as implemented in line 112.
```

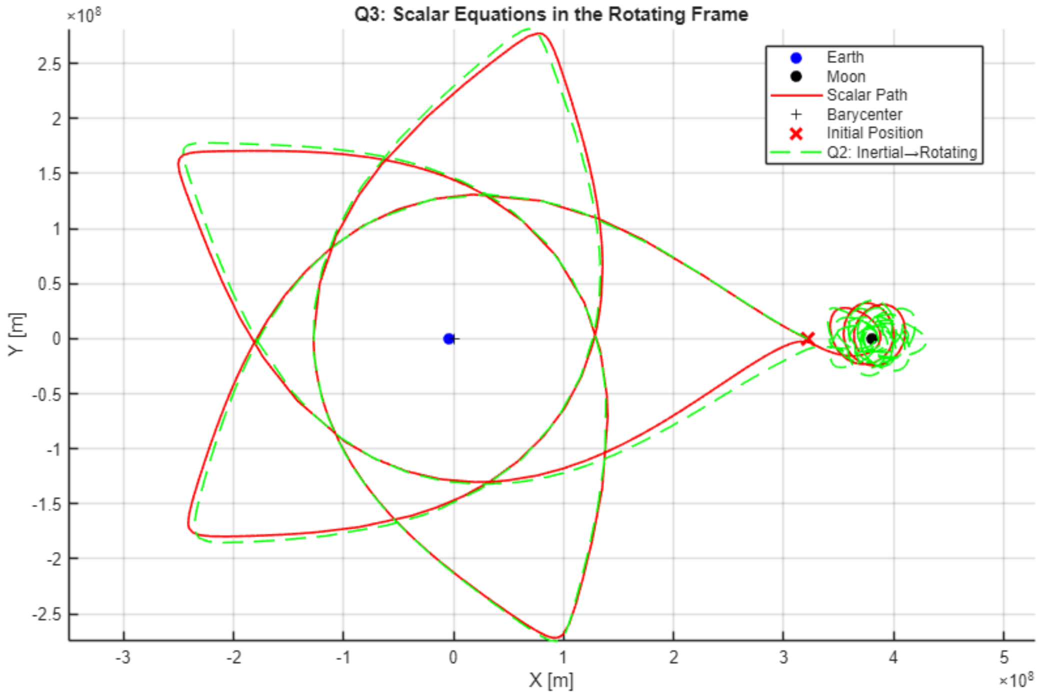




Question 3 - Scalar equations

```
% tspan = tspan;
y0(5) = 0; % since the scalar equations are defined in the rotating frame,
            % initial velocity conditions need to be removed


[t, y] = ode113(@(t,y) scalarAccel(t, y, d_Luna, Mu_Luna, d_Earth, Mu_Earth,
Omega), tspan, y0);

% Plotting Graph
figure;
hold on;
plot(E_Stationary(1), E_Stationary(2), 'bo', 'MarkerFaceColor', 'b'); %
Plots the Earth
plot(Luna_Stationary(1), Luna_Stationary(2), 'ko', 'MarkerFaceColor', 'k');
% Plots the Moon

plot(y(:,1), y(:,2), 'r', 'LineWidth', 1.2); % Plot scalar eqns

plot(0,0, 'k+'); % Barycenter
xlabel('X [m]'); % Creates the x axis label
ylabel('Y [m]'); % Creates the y axis label

plot(y(1,1), y(1,2), 'rx', 'MarkerSize', 10, 'LineWidth', 2); % Initial
Position

if exist('X_SC_Rot','var') && exist('Y_SC_Rot','var')
```

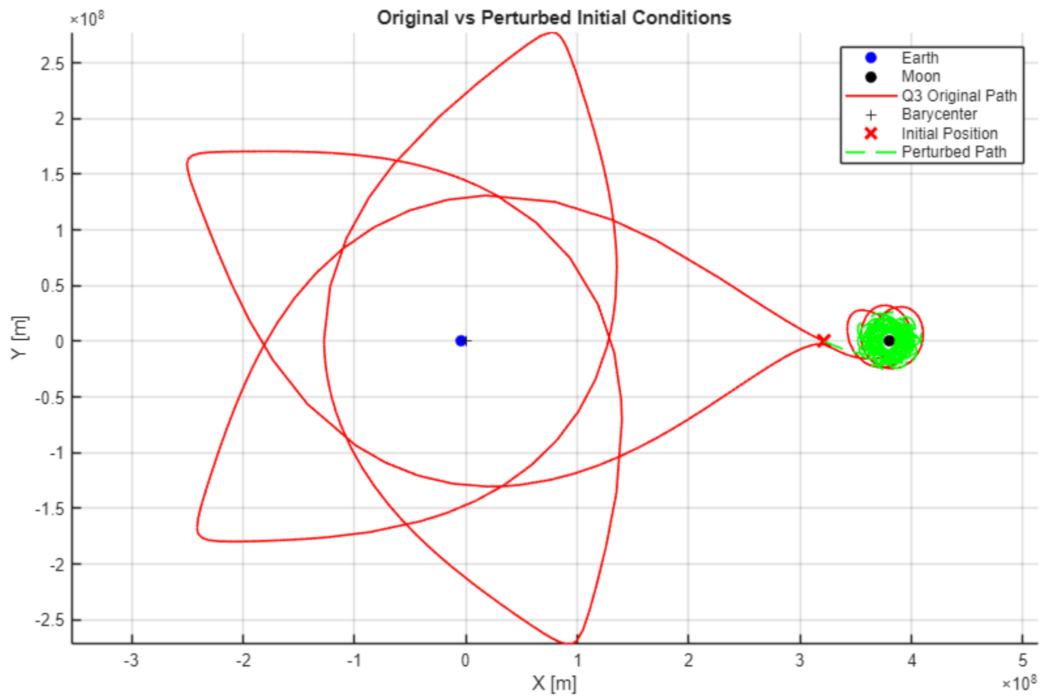
```
    plot(X_SC_Rot, Y_SC_Rot, 'g--','LineWidth',0.2);
    legend('Earth','Moon','Scalar Path','Barycenter', 'Initial
Position','Q2: Inertial→Rotating','Location','best');
else
    legend('Earth','Moon','Scalar Path','Barycenter', 'Initial
Position','Location','best');
end

% legend('Earth', 'Moon', 'Spacecraft', 'Barycenter'); % Legend
title('Q3: Scalar Equations in the Rotating Frame'); % Title

grid on;axis equal;hold off;


% Discussion - Question 3
%
% The paths do not exactly align due to the different approaches used to
achieve the answer.
%
% Question 1, and by extension Question 2, calculate the results based on
vector algebra and simulink.
% Coordinates and values are calculated based on the polar coordinate system
rather than a cartesian one.
% Scalar equations are also obtained iteratively without angles but instead
vector components.
% The approach to these calculations can results in minute initial
differences, which are magnified over the course
% of calculation. Simulink calculations may also differ since we use two
integrate blocks rather than an ode MATLAB function.
% Furthermore, Question 2 is obtained by transformations performed on
Question 1 data, which may produce more
% noise in the data and magnify discrepancies, whereas the scalar
calculations are based entirely on the set of initial conditions.
```



Question 4

```
% set initial conditon perturbations

px = 1;
py = 0;
pz = 0;

vpx = 0;
vpy = 0;
vpz = 0;

% Plot original scalar path
figure;
hold on;
plot(E_Stationary(1), E_Stationary(2), 'bo', 'MarkerFaceColor', 'b'); %
Plots the Earth
plot(Luna_Stationary(1), Luna_Stationary(2), 'ko', 'MarkerFaceColor', 'k');
% Plots the Moon

plot(y(:,1), y(:,2), 'r', 'LineWidth', 1.2); % Plot scalar eqns

plot(0,0, 'k+'); % Barycenter
xlabel('X [m]'); % Creates the x axis label
ylabel('Y [m]'); % Creates the y axis label

plot(y(1,1), y(1,2), 'rx', 'MarkerSize', 10, 'LineWidth', 2); % Initial
Position
```

```
% Perturb initial conditions vector
y0 = [y0(1) + px; y0(2) + py; y0(3) + pz;
    y0(4) + vpx; y0(5) + vpy; y0(6) + vpz];

[t, y] = ode113(@(t,y) scalarAccel(t, y, d_Luna, Mu_Luna, d_Earth, Mu_Earth,
Omega), tspan, y0);

% Plot perturbed path

plot(y(:,1), y(:,2), '--g', 'LineWidth', 1.2);                              %
Plot scalar eqns

plot(0,0, 'k+');
% Barycenter
xlabel('X [m]');
% Creates the x axis label
ylabel('Y [m]');
% Creates the y axis label
legend('Earth', 'Moon', 'Q3 Original Path', 'Barycenter', 'Initial
Position', 'Perturbed Path');      % Creates the legend
title('Original vs Perturbed Initial Conditions');
% Creates the title
grid on;
% Turns on the background grid on the plot
axis equal;
% Sets the units of both axis equal to eachother
hold off;

% Discussion - Question 4

% Minute changes to the initial conditions result in drastic differences in
the orbit.
% For example, when perturbing the spacecraft 1m toward the moon, the
spacecraft orbit
% spirals towards the body.
```



Functions

```
% Function: Equations of motion
function dydt = crtbp_inertial(t, y, Mu_Earth, Mu_Luna, d_Earth, d_Luna,
Omega)

    % Position and velocity
    r = y(1:3); % spacecraft position [m]
    v = y(4:6); % spacecraft velocity [m/s]

    % Earth and Moon positions
    R_Earth = [d_Earth * cos(Omega * t); d_Earth * sin(Omega * t); 0];
    R_Luna  = [d_Luna  * cos(Omega * t); d_Luna  * sin(Omega * t); 0];

    % Relative positions
    r1 = r - R_Earth;
    r2 = r - R_Luna;

    % Accelerations
    a = -Mu_Earth * r1 / norm(r1)^3 - Mu_Luna * r2 / norm(r2)^3;

    % Derivative
    dydt = [v; a];

end

% Scalar Eqns of Motion
function dadt = scalarAccel(~, Y, d_Luna, Mu_Luna, d_Earth, Mu_Earth, Omega)
```

```
    % position
    x = Y(1);
    y = Y(2);
    z = Y(3);

    % velocity
    vx = Y(4);
    vy = Y(5);
    vz = Y(6);

    % define distance to bodies

    r_earth = sqrt( (x-d_Earth)^2 + y^2 + z^2);
    r_moon = sqrt( (x-d_Luna)^2 + y^2 + z^2);



    % Function

    ax = 2*Omega*vy + (Omega^2)*x - (Mu_Earth / r_earth^3)*(x - d_Earth) -
(Mu_Luna / r_moon^3)*(x-d_Luna);

    ay = -(Mu_Earth / r_earth^3)*y - (Mu_Luna / r_moon^3)*y + (Omega^2)*y -
2*Omega*vx;

    az = -(Mu_Earth / r_earth^3)*z - (Mu_Luna / r_moon^3)*z;

    dadt = [vx; vy; vz; ax; ay; az];

end
```

Published with MATLAB® R2025a