Table of Contents

```
%%%%%%%%%%%%%%%%%%%%%%%%%
% Homework 0-1          %
% Author: Caleb Bristol %
% Date: 10/29/21        %
%%%%%%%%%%%%%%%%%%%%%%%%%

clc
clear
close all;
```

Establish Initial Conditions / Constants

```
r_i = [7642;170;2186]; %[km]
r_dot_i = [0.32;6.91;4.29]; %[km/s]
mu_earth = 3.986e14 * 1e-9; %[km^3/s^2] 1e-9 to convert from m^3 to
 km^3

%Set up state as scalars
r_x = r_i(1);
r_y = r_i(2);
r_z = r_i(3);
r_dot_x = r_dot_i(1);
r_dot_y = r_dot_i(2);
r_dot_z = r_dot_i(3);

%State Vector
r_0 = [r_x r_y r_z r_dot_x r_dot_y r_dot_z];

%t vector
t = [0 13000]; %[s]
```

Part 1: Position

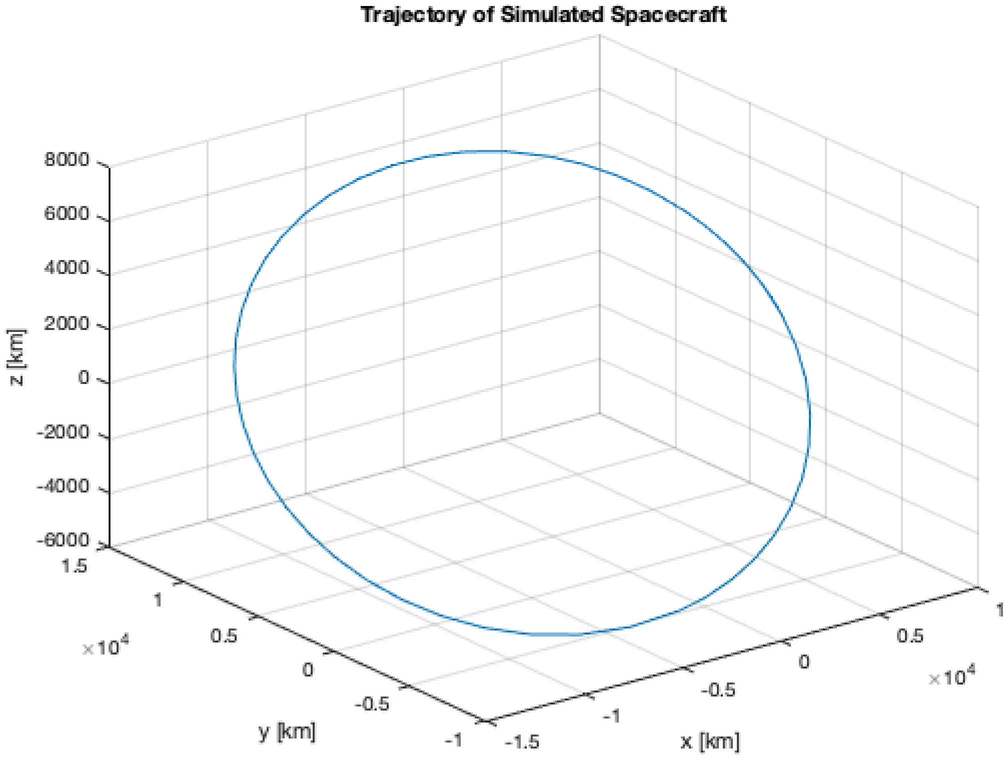
Call ODE45

```
    [t,X_i] = ode45(@(t,X_i) positionfunc(t,X_i,mu_earth),t,r_0);
```

```
r = X_i(:,1:3);
r_dot = X_i(:,4:6);
```

Plotting

```
figure()
plot3(r(:,1),r(:,2),r(:,3)); hold on
xlabel("x [km]")
ylabel("y [km]")
zlabel("z [km]")
title("Trajectory of Simulated Spacecraft")
grid on
hold off
```



Part 2: Integration Tolerances

Call ODE45

```
   % First Set of Tolerances
   opts_1 = odeset('RelTol',1e-3,'AbsTol',1e-3);
   [t,X_1] = ode45(@(t,X_1)
positionfunc(t,X_1,mu_earth),t,r_0,opts_1);
```

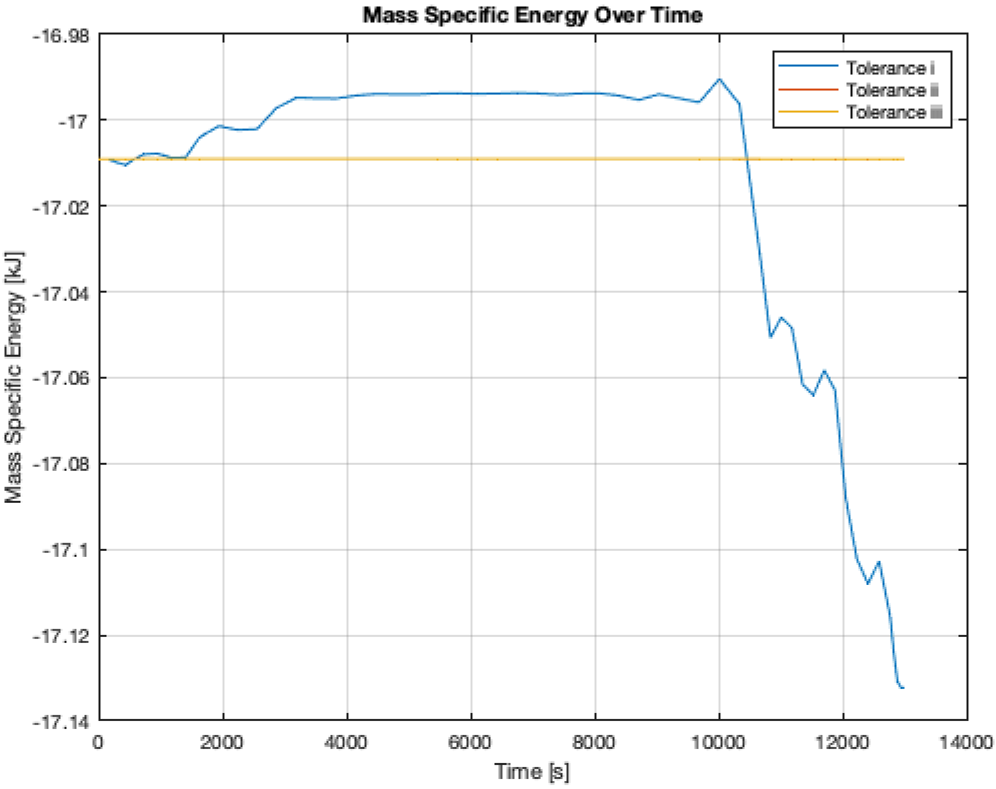
```
   r_1 = X_1(:,1:3);
   r_dot_1 = X_1(:,4:6);

   % Second Set of Tolerances
   opts_2 = odeset('RelTol',1e-6,'AbsTol',1e-6);
   [t,X_2] = ode45(@(t,X_2)
positionfunc(t,X_2,mu_earth),t,r_0,opts_2);

   r_2 = X_2(:,1:3);
   r_dot_2 = X_2(:,4:6);

   % Third Set of Tolerances
   opts_3 = odeset('RelTol',1e-12,'AbsTol',1e-12);
   [t,X_3] = ode45(@(t,X_3)
positionfunc(t,X_3,mu_earth),t,r_0,opts_3);

   r_3 = X_3(:,1:3);
   r_dot_3 = X_3(:,4:6);
```

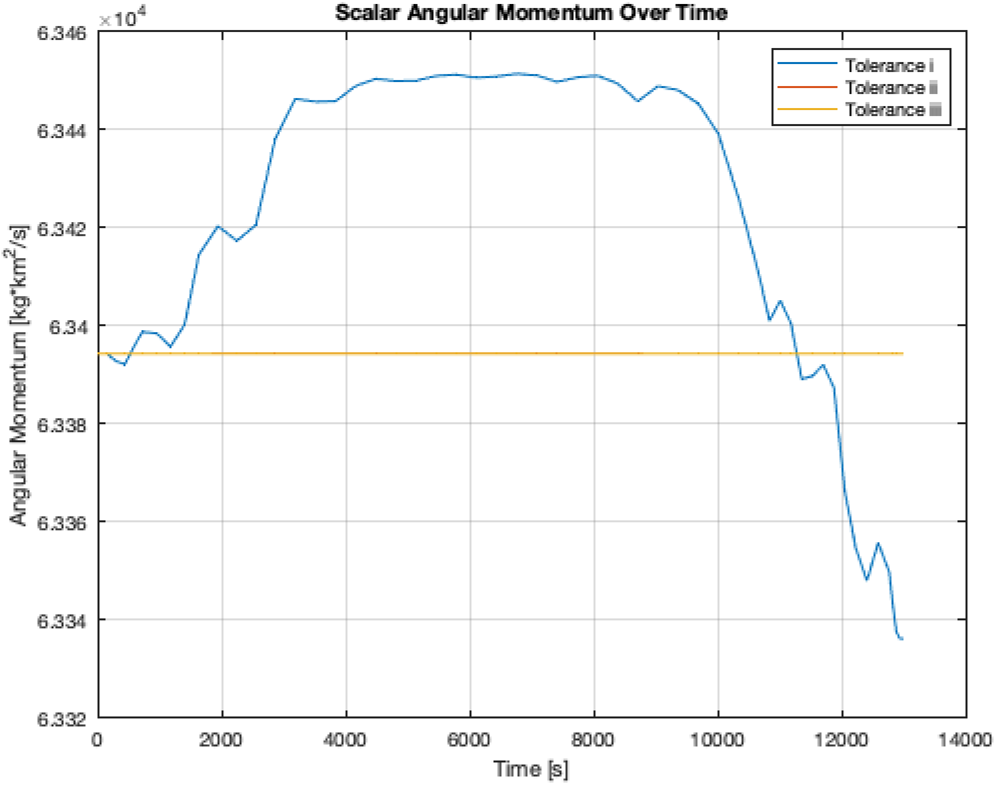
Calculate energy, momentum, eccentricity

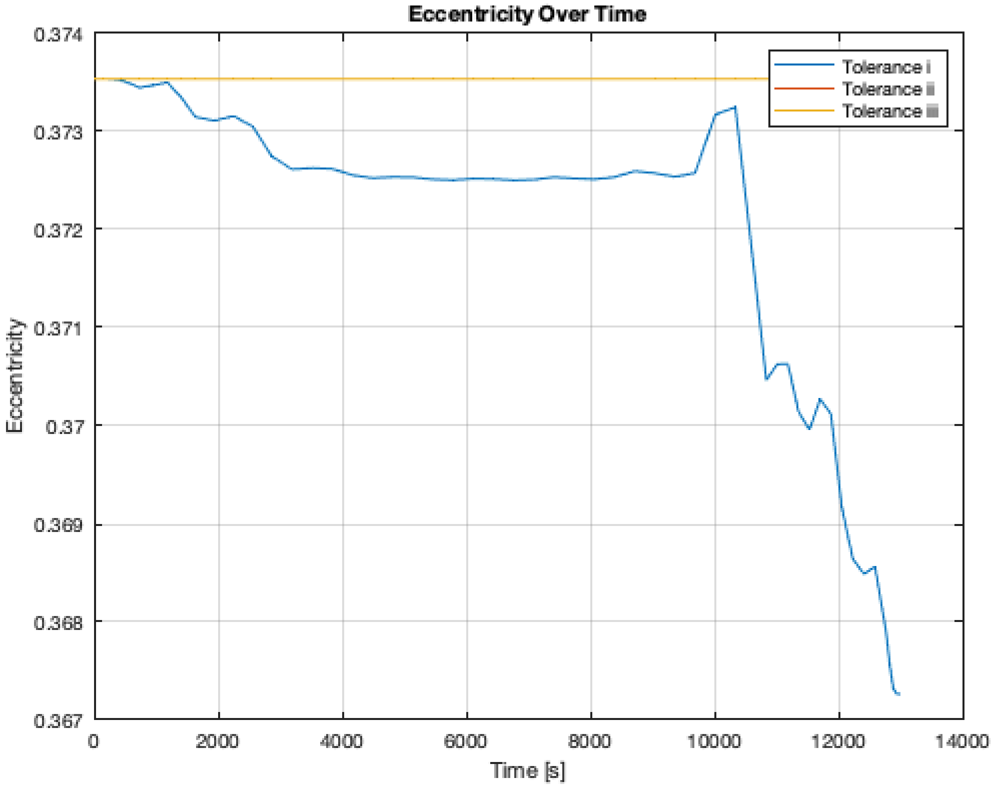
```
epsilon_1 = MSE(r_1,r_dot_1,mu_earth);
hs_1 = AMs(r_1,r_dot_1);
hv_1 = AMv(r_1,r_dot_1);
e_1 = Eccs(r_1,r_dot_1,hv_1,mu_earth);

epsilon_2 = MSE(r_2,r_dot_2,mu_earth);
hs_2 = AMs(r_2,r_dot_2);
hv_2 = AMv(r_2,r_dot_2);
e_2 = Eccs(r_2,r_dot_2,hv_2,mu_earth);

epsilon_3 = MSE(r_3,r_dot_3,mu_earth);
hs_3 = AMs(r_3,r_dot_3);
hv_3 = AMv(r_3,r_dot_3);
e_3 = Eccs(r_3,r_dot_3,hv_3,mu_earth);
```

Plotting

```
figure()
plot(t,epsilon_1); hold on
plot(t,epsilon_2)
plot(t,epsilon_3)
title("Mass Specific Energy Over Time")
xlabel("Time [s]")
ylabel("Mass Specific Energy [kJ]")
legend("Tolerance i","Tolerance ii","Tolerance iii")
grid on
hold off

figure()
plot(t,hs_1); hold on
plot(t,hs_2)
plot(t,hs_3)
title("Scalar Angular Momentum Over Time")
xlabel("Time [s]")
ylabel("Angular Momentum [kg*km^2/s]")
legend("Tolerance i","Tolerance ii","Tolerance iii")
grid on
hold off

figure()
plot(t,e_1); hold on
plot(t,e_2)
plot(t,e_3)
title("Eccentricity Over Time")
xlabel("Time [s]")
ylabel("Eccentricity")
legend("Tolerance i","Tolerance ii","Tolerance iii")
grid on
hold off
```




Scalar Angular Momentum Over Time
×10^4
6.346
6.344
6.342
6.34
6.338
6.336
6.334
6.332
Angular Momentum [kg*km^2/s]
0
2000
4000
6000
8000
10000
12000
14000
Time [s]
Tolerance i
Tolerance ii
Tolerance iii


Eccentricity Over Time
0.374
0.373
0.372
0.371
0.37
0.369
0.368
0.367
Eccentricity
0
2000
4000
6000
8000
10000
12000
14000
Time [s]
Tolerance i
Tolerance ii
Tolerance iii

Define Function(s)

```
% ODE45 function (dif eqns)
function drdt = positionfunc(t,r_0,mu)
    r_x = r_0(1);
    r_y = r_0(2);
    r_z = r_0(3);
    r_mag = norm(r_0);

    v_x = r_0(4);
    v_y = r_0(5);
    v_z = r_0(6);
    a_x = -(mu / (r_mag^3)) * r_x;
    a_y = -(mu / (r_mag^3)) * r_y;
    a_z = -(mu / (r_mag^3)) * r_z;

    drdt = [v_x;v_y;v_z;a_x;a_y;a_z];
end

% Mass Specific Energy
function epsilon = MSE(r,r_dot,mu)
    r_mag = (r(:,1).^2 + r(:,2).^2 + r(:,3).^2).^0.5;

    energy = (0.5 * (r_dot(:,1).^2 + r_dot(:,2).^2 + r_dot(:,3).^2)) -
 (mu./r_mag);

    epsilon = energy;
end

% Angular Momentum Vector
function h = AMv(r,r_dot)
    momentum = cross(r,r_dot);

    h = momentum;

end

% Scalar Angular Momentum
function h = AMs(r,r_dot)
    h_vec = cross(r,r_dot);

    momentum_scalar = (h_vec(:,1).^2 + h_vec(:,2).^2 +
 h_vec(:,3).^2).^0.5;

    h = momentum_scalar;
end

% Eccentricity Vector
function e = Eccv(r,r_dot,h,mu)
    r_mag = (r(:,1).^2 + r(:,2).^2 + r(:,3).^2).^0.5;

    eccentricity = (cross(r_dot,h) - mu * (r./r_mag)) / mu;
    e = eccentricity;
end

% Eccentricity Scalar
function e = Eccs(r,r_dot,h,mu)
    r_mag = (r(:,1).^2 + r(:,2).^2 + r(:,3).^2).^0.5;

    eccentricity = (cross(r_dot,h) - mu * (r./r_mag)) / mu;

    e_s = (eccentricity(:,1).^2 + eccentricity(:,2).^2 +
 eccentricity(:,3).^2).^0.5;

    e = e_s;
end
```

Published with MATLAB® R2020a