ASEN 3200 Orbital Project Part 1

Table of Contents

Problem Statement:

Author: Caleb Bristol Collaborators: N/A Date: 12/01/21

Workspace Cleaning

```
clc
clear
close all;
```

Preliminary Work

Stuff not specified in any part but required for calculations.

Constants

```
mu = 3.986e14 * 1e-9; %[km^3/s^2] 1e-9 to convert from m^3 to km^3
Re = 6378.137; %[km]
J2 = 1082.63*10^(-6);
```

Read in Cities & Coastline

Used for plotting later need conversion to cartesian for 3D

```
cities_data = readtable('worldcities.csv');
cities_ll = [cities_data.lng cities_data.lat];
coastlines_ll = load('world_coastline_low.txt');
```

```
    [cities(:,1),cities(:,2),cities(:,3)] =
sph2cart(deg2rad(cities_ll(:,1)),deg2rad(cities_ll(:,2)),Re);

    [coastlines(:,1),coastlines(:,2),coastlines(:,3)] =
sph2cart(deg2rad(coastlines_ll(:,1)),deg2rad(coastlines_ll(:,2)),Re);
```

Part i)

Read in a JSON constellation design file

Call loadConstellation Function

```
    [num_launches, num_spacecraft, satellite_list] =
loadConstellation('example_constellation.json');
```

Part ii)

Propagate the constellation through time for a full mean solar day (in 30 second time intervals)

Define Mean Solar Day (30s time steps)

```
    MSD = 0:30:24*3600;
```

Propagate Constellation with propagateState function

```
    for i = 1:length(satellite_list)
        x_ = zeros(6,length(MSD));
        for j = 1:length(MSD)
            x_(:,j) =
propagateState(satellite_list(i).oe0,MSD(j),MSD(1),mu,J2,Re);
        end
        satellite_list(i).x = x_;
    end
```

Part iii)

Compute the number of spacecraft in line of site for each city i at each time step

Rotate Cities/Coastline in ECI

The spacecraft should remain the same, I'm going to do everything in ECI because it's easier testLoS function called at each time, city, and spacecraft

```
    % Earth Rotation: Sidereal Time
    omega_earth = 2*pi/(23*3600 + 56*60 + 4.1); %[rad/s]
    theta_earth = omega_earth * MSD; %[rad]
```

```
% Create num_LoS matrix, rows are times, columns are cities
num_LoS = zeros(length(MSD),length(cities));

% Iterate every time, city, and spacecraft for LoS
for i = 1:length(MSD)
    % Account for rotation of Earth
    cities_ = (angle2dcm(theta_earth(i),0,0,'ZXZ') * cities')';
    for j = 1:length(cities_)
        for k = 1:length(satellite_list)
            num_LoS(i,j) = num_LoS(i,j) +
testLoS(cities_(j,:),satellite_list(k).x(1:3,i),pi/8);
        end
    end
end
```

Part iv)

Plot a 3D render of constellation orbits and the earth (with coastlines and cities) at the final time

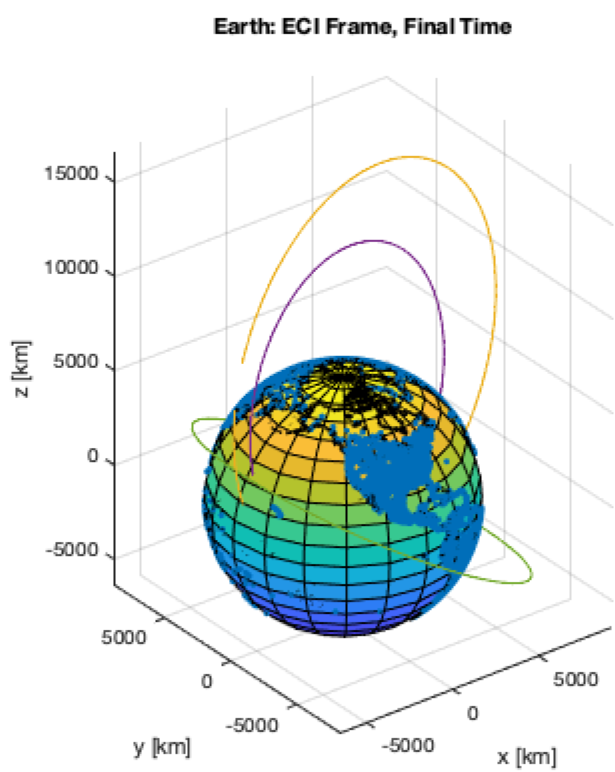
Propogate Final Orbits with ODE45

```
for i = 1:length(satellite_list)

    % Some orbital elements
    a = satellite_list(i).oe0(1);
    P = 2*pi*sqrt(a^3/mu);
    r_0 = satellite_list(i).x(:,end);

    % Propagate exactly one period
    t = [0 P];

    [t,X_i] = ode45(@(t,X_i)
positionfunc(t,X_i,mu),t,r_0,odeset('RelTol',1e-9,'AbsTol',1e-9));

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

    satellite_list(i).finalorbit = r;
end
```

Plotting

```
% Account for rotation of Earth
cities_ = (angle2dcm(theta_earth(end),0,0,'ZXZ') * cities')';
coastlines_ = (angle2dcm(theta_earth(end),0,0,'ZXZ') *
coastlines')';

% Unit Sphere
[ex,ey,ez] = sphere;

% Plot:
figure()
```

```
    % Earth
    surf(Re*ex,Re*ey,Re*ez); hold on
    % Cities
    scatter3(cities_(:,1),cities_(:,2),cities_(:,3),'.')
    % Coastlines

plot3(coastlines_(:,1),coastlines_(:,2),coastlines_(:,3),'k','Linewidth',2)
    % Spacecraft Orbits
    for i = 1:length(satellite_list)

plot3(satellite_list(i).finalorbit(1,:),satellite_list(i).finalorbit(2,:),satelli
    end
    grid on
    xlabel('x [km]')
    ylabel('y [km]')
    zlabel('z [km]')
    title('Earth: ECI Frame, Final Time')
    axis equal
    hold off
```



Functions

```
% For ODE45
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
```

Published with MATLAB® R2020a