Table of Contents

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
% ASEN 3200 Homework O-3
% Author: Caleb Bristol
% Date: 11/12/21
%
%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
clear
close all;
```

Problem 1

This problem involves conversions between solar time and sidereal time

Given

```
UT = 4.5 + 274 * 24;
theta_g0 = 100.35;
```

Part c: Calculate Greenwich Time

```
theta_g = theta_g0 + 360.98564724 * UT/24;
% Normalize within bounds
theta_g = theta_g - floor(theta_g/360)*360;
```

Part d: Georgia Tech

```
del = 360 - 84.3963; %East longitude of Georgia Tech

theta = theta_g + del;

% Normalize within bounds
```

```
theta = theta - floor(theta/360)*360;
```

Display Results

```
fprintf("Problem 1: \n")
fprintf("c) \n")
fprintf("The local siderial time of Greenwich [deg]: \n")
disp(theta_g)
fprintf("d) \n")
fprintf("The local siderial time of Georgia Tech [deg]: \n")
disp(theta)
```

```
Problem 1:
c)
The local siderial time of Greenwich [deg]:
   78.1022

d)
The local siderial time of Georgia Tech [deg]:
  353.7059
```

Problem 2

This problem involves a 3D plot in a-e-i space defined in 2D by a sun-synchronous orbit constraint

Given

```
R_e = 6378; %[km]
a = [R_e R_e+3000]; %[km]
e = [0 0.3];
i = [90 180]; %[deg]
interval = [a e i];
mu = 3.986e14 * 1e-9; %[km^3/s^2] 1e-9 to convert from m^3 to km^3
J_2 = 1.08262668e-03;
const = -3/2 * J_2 * mu^0.5 * R_e^2;
Omega_dot = deg2rad(0.9856) / (24*3600); %[rad/s]
load("catalog_TLEs.mat");
```

Define Constraint Equation

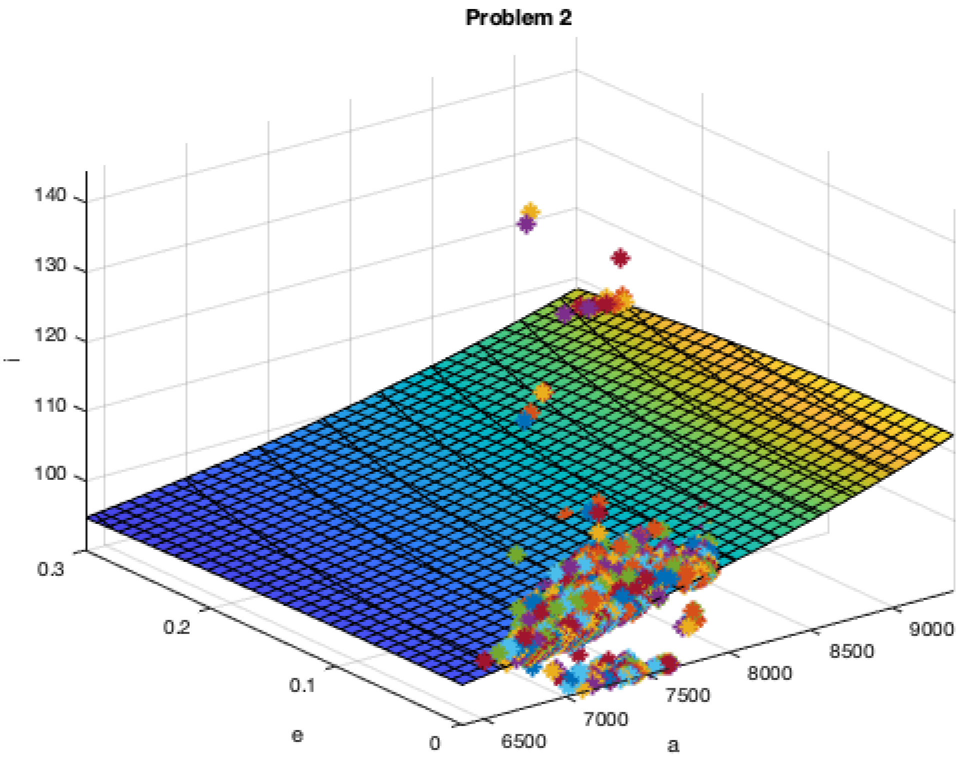
```
zero_func = @(a,e,i) const * cosd(i) / (a^(7/2) * (1-e^2)^2) -
Omega_dot;
```

Plot

```
figure()
fimplicit3(zero_func,interval); hold on
for j = 1:length(catalog_TLEs)
    obj = catalog_TLEs{j};
    int = [obj.semimajoraxis obj.eccentricity obj.inclination];
```

```
        if int(1)>a(1) && int(1)<a(2) && int(2)>e(1) && int(2)<e(2) &&
 int(3)>i(1) && int(3)<i(2)
            plot3(int(1),int(2),int(3),'*','LineWidth',2)
        end
    end
    title('Problem 2')
    xlabel('a')
    ylabel('e')
    zlabel('i')
    grid on
    hold off
```

```
Warning: Function behaves unexpectedly on array inputs. To improve
 performance,
properly vectorize your function to return an output with the same
 size and
shape as the input arguments.
```



Problem 3

This problem involves circular sun-synchronous orbits with certain constraints

Part a)

Spacecraft will pass the same points on Earth 2,3, and 4 times / day

```
day = 24; %[hr]
P_1 = day/2; %[hr]
P_1 = P_1 * 3600; %[s]
P_2 = day/3; %[hr]
P_2 = P_2 * 3600; %[s]
P_3 = day/4; %[hr]
P_3 = P_3 * 3600; %[s]

% Helper Function
a = @(P) (mu * (P/(2*pi))^2)^(1/3);

% Define Semi-major axes
a_1 = a(P_1);
a_2 = a(P_2);
a_3 = a(P_3);
```

Part b)

Published with MATLAB® R2020a