Table of Contents

Orbits Project 1
Clean Up 1
Import Data 1
Orbital Path 1
Visibility Animation 3
Plot the animation for 1 day 3
Run the following section ONLY FOR THE FULL WEEK, UNCOMENT CODE 5
Get the angles and observability for a specific facet number 7
Plotting 8
Functions 10
Triangulation method with vectors 14

Orbits Project

```
%Author: Thomas Dunnington
%Date: 4/20/2023
```

Clean Up

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

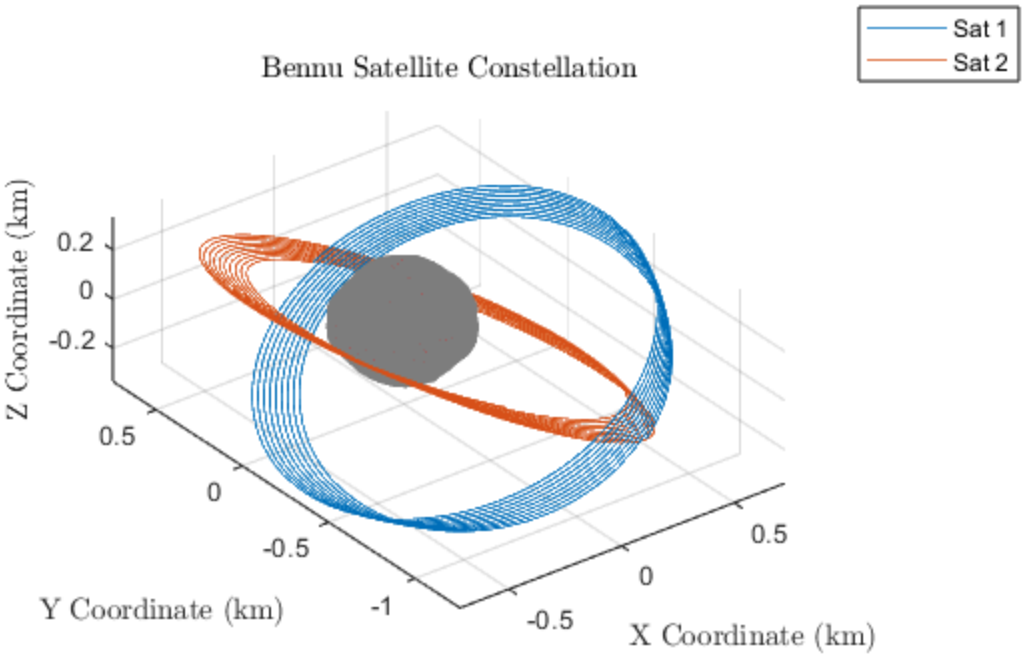
Import Data

```
fileName = "BennuFull.obj";

%Reading in data
[facets, vertices] = read_obj(fileName);
```

Orbital Path

```
%Declaring details of constellation design with two satellites
%Satellite Parameters
t0 = 0;
tf = 60*60*24*7; %1 week timespan
A = 3.25^2 / 1000^2;
m = 2000;

%Satellite 1 initial conditions
v0 = 0.1 / 1000; %km/s
bennu_r = 0.2625;
X0_1 = [0; bennu_r*2; 0; v0; 0; 0.5*v0];

%Satellite 2 initial conditions
v0 = 0.1 / 1000; %km/s
bennu_r = 0.2625;
```

```
X0_2 = [0; bennu_r*2; 0; -v0; 0; 0.5*v0];


%Propogate the spacecraft position
[Xout_1, OEout_1, Tout_1] = propagate_spacecraft(X0_1, t0, tf, A, m);
[Xout_2, OEout_2, Tout_2] = propagate_spacecraft(X0_2, t0, tf, A, m);


%Plotting orbit around Bennu for a full week
figure();
set(0, 'defaulttextinterpreter', 'latex');
patch('Faces', facets, 'Vertices',
 vertices, 'FaceColor', 'red', 'EdgeColor', [0.5 0.5
 0.5], 'linestyle', ':');
view(3);
hold on
grid on
s1 = plot3(Xout_1(:,1), Xout_1(:,2), Xout_1(:,3)); %Sat 1
s2 = plot3(Xout_2(:,1), Xout_2(:,2), Xout_2(:,3)); %Sat 2

axis equal
xlabel('X Coordinate (km)');
ylabel('Y Coordinate (km)');
zlabel('Z Coordinate (km)');
title('Bennu Satellite Constellation');
legend([s1 s2], 'Sat 1', 'Sat 2');
```



Visibility Animation

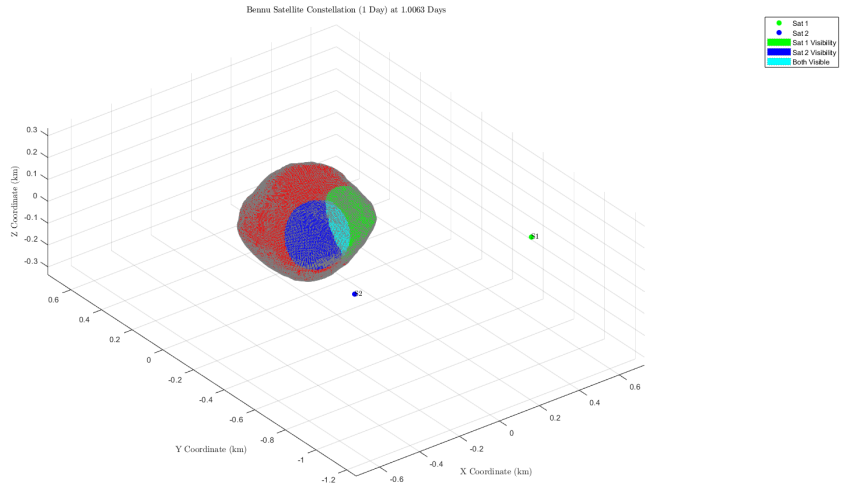
```
%Determining Visibility
rOut_1 = Xout_1(:,1:3);
rOut_2 = Xout_2(:,1:3);

%Create vector of rotation of the vertices due to bennu's rotation
period = 4.297461 * 3600; %Time period of Bennu
angRate = 2*pi / period;
thetaVec = t0*angRate:60*angRate:tf*angRate;
vRotated = zeros(size(vertices));
```

Plot the animation for 1 day

```
%Run a for loop to calculate the visible faces for the spacecraft
 during a
%week
warning('off');
fig2 = figure();
fig2.WindowState = 'maximized';

%Plot for 1 day
day1 = 24*60*60;

%Creating animation of visibility
for i = 10:10:length(rOut_1(:,1))
    %Clear the figure
    clf;

    %Rotate Bennu vertices
    rotMat = [cos(thetaVec(i)), -sin(thetaVec(i)), 0;
 sin(thetaVec(i)), cos(thetaVec(i)), 0; 0, 0, 1];
    for j = 1:length(vertices(:,1))
        vRotated(j,:) = (rotMat * vertices(j,:)')';
    end

    %Call check view for spacecraft one and two
    [observable_1, elevationAngle_1, cameraAngle_1] =
 check_view(rOut_1(i,:), 0xDEADBEEF, facets, vRotated, 2);
    [observable_2, elevationAngle_2, cameraAngle_2] =
 check_view(rOut_2(i,:), 0xDEADBEEF, facets, vRotated, 2);

    %Determine the visible facets at the given position
    obsLog_1 = observable_1 == 1;
    obsLog_2 = observable_2 == 1;

    %Get the faces that are visible
    visFaces_1 = facets(obsLog_1, :);
    visFaces_2 = facets(obsLog_2, :);
    visFaces_both = facets(obsLog_1 & obsLog_2, :);
    invisFaces = facets(~(obsLog_1 | obsLog_2), :);
```

```
   %Plot the figure with patch
   vis1 = patch('Faces', visFaces_1, 'Vertices',
vRotated, 'FaceColor', 'green', 'EdgeColor', [0.5 0.5
0.5], 'linestyle', ':');
   axis equal
   hold on
   vis2 = patch('Faces', visFaces_2, 'Vertices',
vRotated, 'FaceColor', 'blue', 'EdgeColor', [0.5 0.5
0.5], 'linestyle', ':');
   visBoth = patch('Faces', visFaces_both, 'Vertices',
vRotated, 'FaceColor', 'cyan', 'EdgeColor', [0.5 0.5
0.5], 'linestyle', ':');
   patch('Faces', invisFaces, 'Vertices',
vRotated, 'FaceColor', 'red', 'EdgeColor', [0.5 0.5
0.5], 'linestyle', ':');
   view(3);

   %Plot the satellite location
   sc_1 = scatter3(rOut_1(i,1), rOut_1(i,2),
rOut_1(i,3), 'filled', 'markerEdgecolor', 'g', 'markerFaceColor', 'g');
   text(rOut_1(i,1), rOut_1(i,2), rOut_1(i,3), 'S1');
   sc_2 = scatter3(rOut_2(i,1), rOut_2(i,2),
rOut_2(i,3), 'filled', 'markerEdgecolor', 'b', 'markerFaceColor', 'b');
   text(rOut_2(i,1), rOut_2(i,2), rOut_2(i,3), 'S2');
   grid on


   %Set axis limits
   xlim([min([rOut_1(:,1); rOut_2(:,1)]) max([rOut_1(:,1);
rOut_2(:,1)])]);
   ylim([min([rOut_1(:,2); rOut_2(:,2)]) max([rOut_1(:,2);
rOut_2(:,2)])]);
   zlim([min([rOut_1(:,3); rOut_2(:,3)]) max([rOut_1(:,3);
rOut_2(:,3)])]);

   %Calculate the time for the title in days for the title
   dayTime = Tout_1(i) / 24 / 3600;
   dayString = num2str(dayTime);

   %Add labels
   xlabel('X Coordinate (km)');
   ylabel('Y Coordinate (km)');
   zlabel('Z Coordinate (km)');
   title(strcat("Bennu Satellite Constellation (1 Day) at ",
dayString, " Days"));
   legend([sc_1 sc_2 vis1 vis2 visBoth], 'Sat 1', 'Sat 2', 'Sat 1
Visibility', 'Sat 2 Visibility', 'Both Visible');

   drawnow limitrate;

   if(Tout_1(i) >= 24*60*60) %Only run for 1 day, break loop
      break;
   end
```

```
end
```



Run the following section ONLY FOR THE FULL WEEK, UNCOMENT CODE

```
%WARNING: THIS SECTION TAKES A LONG TIME TO RUN
%
%Run a for loop to calculate the visible faces for the spacecraft
 during a
%week
% fig3 = figure();
% fig3.WindowState = 'maximized';
%
% %Creating animation of visibility
% for i = 10:10:length(rOut_1(:,1))
%     %Clear the figure
%     clf;
%
%     %Rotate Bennu vertices
%     rotMat = [cos(thetaVec(i)), -sin(thetaVec(i)), 0;
 sin(thetaVec(i)), cos(thetaVec(i)), 0; 0, 0, 1];
%     for j = 1:length(vertices(:,1))
%         vRotated(j,:) = (rotMat * vertices(j,:)')';
%     end
%
%     %Call check view for spacecraft one and two
%     [observable_1, elevationAngle_1, cameraAngle_1] =
 check_view(rOut_1(i,:), 0xDEADBEEF, facets, vRotated, 2);
%     [observable_2, elevationAngle_2, cameraAngle_2] =
 check_view(rOut_2(i,:), 0xDEADBEEF, facets, vRotated, 2);
%
%     %Determine the visible facets at the given position
%     obsLog_1 = observable_1 == 1;
```

```
%     obsLog_2 = observable_2 == 1;
%
%     %Get the faces that are visible
%     visFaces_1 = facets(obsLog_1, :);
%     visFaces_2 = facets(obsLog_2, :);
%     visFaces_both = facets(obsLog_1 & obsLog_2, :);
%     invisFaces = facets(~(obsLog_1 | obsLog_2), :);
%
%
%     %Plot the figure with patch
%     vis1 = patch('Faces', visFaces_1, 'Vertices', vRotated,
 'FaceColor', 'green', 'EdgeColor', [0.5 0.5 0.5], 'linestyle', ':');
%     axis equal
%     hold on
%     vis2 = patch('Faces', visFaces_2, 'Vertices', vRotated,
 'FaceColor', 'blue', 'EdgeColor', [0.5 0.5 0.5], 'linestyle', ':');
%     visBoth = patch('Faces', visFaces_both, 'Vertices', vRotated,
 'FaceColor', 'cyan', 'EdgeColor', [0.5 0.5 0.5], 'linestyle', ':');
%     patch('Faces', invisFaces, 'Vertices', vRotated, 'FaceColor',
 'red', 'EdgeColor', [0.5 0.5 0.5], 'linestyle', ':');
%     view(3);
%
%     %Plot the satellite location
%     sc_1 = scatter3(rOut_1(i,1), rOut_1(i,2), rOut_1(i,3), 'filled',
 'markerEdgecolor', 'g', 'markerFaceColor', 'g');
%     text(rOut_1(i,1), rOut_1(i,2), rOut_1(i,3), 'S1');
%     sc_2 = scatter3(rOut_2(i,1), rOut_2(i,2), rOut_2(i,3), 'filled',
 'markerEdgecolor', 'b', 'markerFaceColor', 'b');
%     text(rOut_2(i,1), rOut_2(i,2), rOut_2(i,3), 'S2');
%     grid on
%
%
%     %Set axis limits
%     xlim([min([rOut_1(:,1); rOut_2(:,1)]) max([rOut_1(:,1);
 rOut_2(:,1)])]);
%     ylim([min([rOut_1(:,2); rOut_2(:,2)]) max([rOut_1(:,2);
 rOut_2(:,2)])]);
%     zlim([min([rOut_1(:,3); rOut_2(:,3)]) max([rOut_1(:,3);
 rOut_2(:,3)])]);
%
%     %Calculate the time for the title in days for the title
%     dayTime = Tout_1(i) / 24 / 3600;
%     dayString = num2str(dayTime);
%
%     %Add labels
%     xlabel('X Coordinate (km)');
%     ylabel('Y Coordinate (km)');
%     zlabel('Z Coordinate (km)');
%     title(strcat("Bennu Satellite Constellation at ", dayString, "
 Days"));
%     legend([sc_1 sc_2 vis1 vis2 visBoth], 'Sat 1', 'Sat 2', 'Sat 1
 Visibility', 'Sat 2 Visibility', 'Both Visible');
%
%     drawnow limitrate;
```

```
%
%     movieVec(i/10) = getframe(fig3);
% end
%
% %Save animation to a file
% vidWrite = VideoWriter('VisibilityOrbit_1Week');
% vidWrite.FrameRate = 20;
%
% open(vidWrite);
% writeVideo(vidWrite, movieVec);
% close(vidWrite);
%
```

Get the angles and observability for a specific facet number

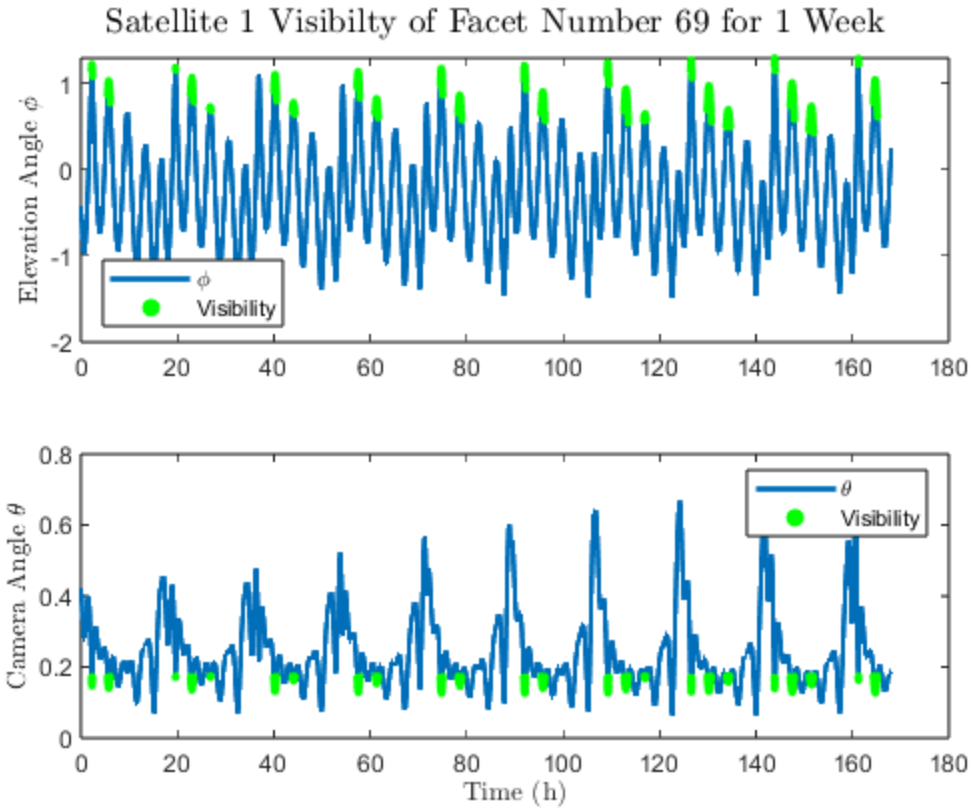
```
%Determining angles and observation for facet
facetNum = 69;

%Preallocating memory for visibility for the facet
facetElevationAngles_1 = zeros(length(Tout_1),1);
facetCameraAngles_1 = zeros(length(Tout_1),1);
facetObservability_1 = zeros(length(Tout_1),1);

facetElevationAngles_2 = zeros(length(Tout_1),1);
facetCameraAngles_2 = zeros(length(Tout_1),1);
facetObservability_2 = zeros(length(Tout_1),1);

%Looping through every position of the constellation
for i = 1:length(Tout_1)

    %Rotate Bennu vertices
    rotMat = [cos(thetaVec(i)), -sin(thetaVec(i)), 0;
 sin(thetaVec(i)), cos(thetaVec(i)), 0; 0, 0, 1];
    for j = 1:length(vertices(:,1))
        vRotated(j,:) = (rotMat * vertices(j,:)')';
    end

    %Call check view for spacecraft one and two
    [observable_1, elevationAngle_1, cameraAngle_1] =
 check_view(rOut_1(i,:), 0xDEADBEEF, facets, vRotated, 2);
    [observable_2, elevationAngle_2, cameraAngle_2] =
 check_view(rOut_2(i,:), 0xDEADBEEF, facets, vRotated, 2);

    %Satellite 1 observability
    facetElevationAngles_1(i) = elevationAngle_1(facetNum);
    facetCameraAngles_1(i) = cameraAngle_1(facetNum);
    facetObservability_1(i) = observable_1(facetNum);

    %Satellite 2 observability
    facetElevationAngles_2(i) = elevationAngle_2(facetNum);
    facetCameraAngles_2(i) = cameraAngle_2(facetNum);
```

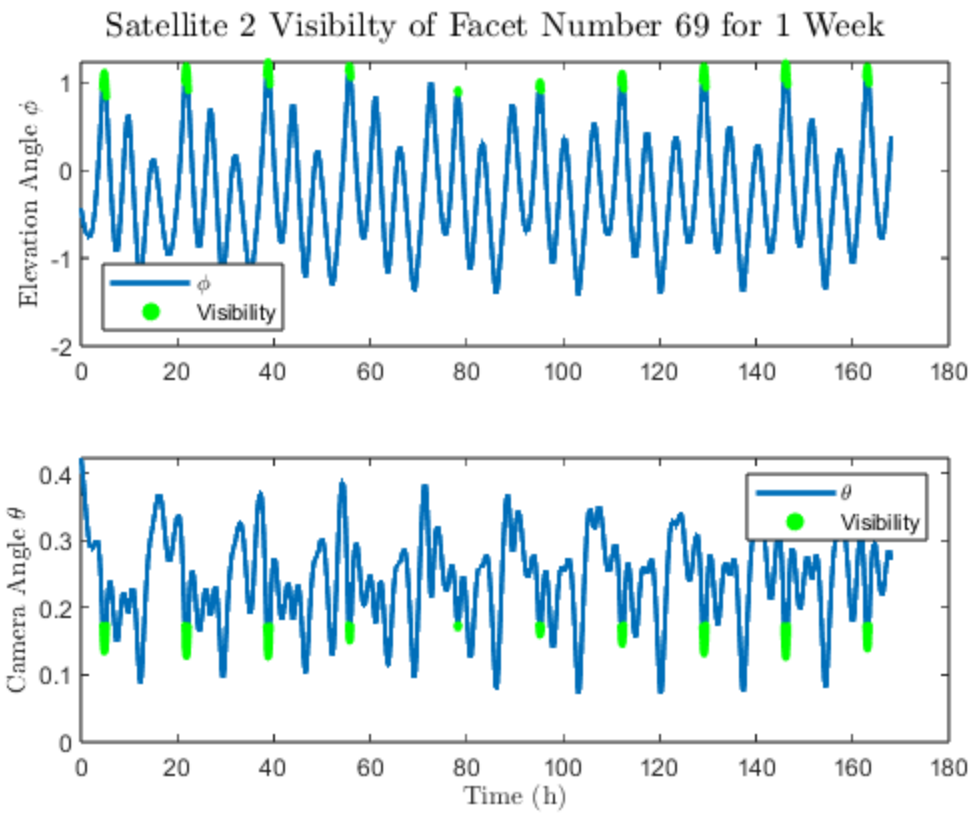
```
    facetObservability_2(i) = observable_2(facetNum);
end
```

Plotting

```
%Getting visible angles
facetObservability_1 = facetObservability_1 == 1;
facetObservability_2 = facetObservability_2 == 1;

Tout_1Vis = Tout_1(facetObservability_1);
Tout_2Vis = Tout_2(facetObservability_2);

facetElevationAngles_1Vis =
 facetElevationAngles_1(facetObservability_1);
facetElevationAngles_2Vis =
 facetElevationAngles_2(facetObservability_2);

facetCameraAngles_1Vis = facetCameraAngles_1(facetObservability_1);
facetCameraAngles_2Vis = facetCameraAngles_2(facetObservability_2);

%Plotting the visibility
%SAT 1
figure();
subplot(2,1,1);
plot(Tout_1 ./ 3600, facetElevationAngles_1, 'linewidth', 2);
hold on
test = scatter(Tout_1Vis ./ 3600,
 facetElevationAngles_1Vis, 'filled', 'MarkerFaceColor', 'g', 'MarkerEdgeColor', '
 5);

sgtitle('Satellite 1 Visibilty of Facet Number 69 for 1 Week');
ylabel('Elevation Angle $$\phi$$');
legend('\phi', 'Visibility', 'location', 'sw');

subplot(2,1,2);
plot(Tout_1 ./ 3600, facetCameraAngles_1, 'linewidth', 2);
hold on
scatter(Tout_1Vis ./ 3600,
 facetCameraAngles_1Vis, 'filled', 'MarkerFaceColor', 'g', 'MarkerEdgeColor', 'g',
 5);

xlabel('Time (h)');
ylabel('Camera Angle $$\theta$$');
legend('\theta', 'Visibility');

%SAT 2
figure();
subplot(2,1,1);
plot(Tout_2 ./ 3600, facetElevationAngles_2, 'linewidth', 2);
hold on
test = scatter(Tout_2Vis ./ 3600,
 facetElevationAngles_2Vis, 'filled', 'MarkerFaceColor', 'g', 'MarkerEdgeColor', '
 5);
```

```
sgtitle('Satellite 2 Visibilty of Facet Number 69 for 1 Week');
ylabel('Elevation Angle $$\phi$$');
legend('\phi', 'Visibility', 'location', 'sw');

subplot(2,1,2);
plot(Tout_2 ./ 3600, facetCameraAngles_2, 'linewidth', 2);
hold on
scatter(Tout_2Vis ./ 3600,
 facetCameraAngles_2Vis, 'filled', 'MarkerFaceColor', 'g', 'MarkerEdgeColor', 'g',
 5);

xlabel('Time (h)');
ylabel('Camera Angle $$\theta$$');
legend('\theta', 'Visibility');
```





Functions

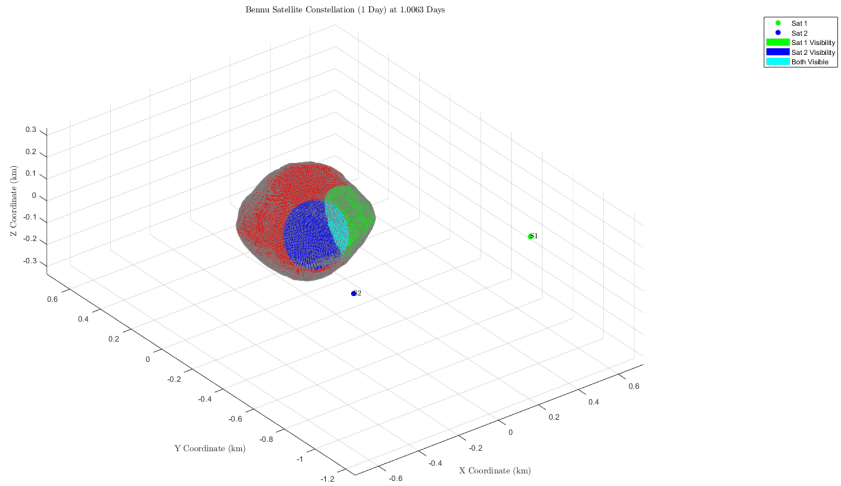
```
function [facets, vertices] = read_obj(fileName)
%Summary: This function reads in an object file and outputs the
 vertices
%and facets.
%Author: Thomas Dunnington
%Date: 4/6/2023
%INPUTS: fileName = name of the object file
%OUTPUTS:
%   facets = n x 3 matrix of vertices that form each face
%   vertices = m x 3 matrix of vertex locations in implied body frame

% fileID = fopen(fileName);
% C = textscan(fileID, '%s %f %f %f');
%
% strings = string([C{:,1}]);
% data = [C{:,2}, C{:,3}, C{:,4}];
%
% vertices = data(strings == 'v', :);
% facets = data(strings == 'f', :);

fileID = fopen(fileName);
raw = textscan(fileID, '%s', 'delimiter', '\n');
strings = string([raw{1,1}]);
```

```
%Get rid of comments
checker = 1;
ind = 1;
while(checker)
    checker = 0;
    string1 = strings{ind};
    if(string1(1) == '#') %Condition for comment
        strings(1) = [];
        checker = 1;
    end
end

%Parsing through the strings with spaces
strings_split = split(strings);

%First column is v or f, the other three are the faces and vertices
%coordinates
vOrf = strings_split(:,1);
data = strings_split(:,2:end);
data = str2double(data);

vertices = data(vOrf == 'v', :);
facets = data(vOrf == 'f', :);

end

function outVec = EOM(t, state, mu, area, mass)
    %Two body problem EOM
    %State is [x;y;z;vx;vy;vz]
    %outVec is [vx;vy;vz;ax;ay;az] where a is acceleration
    %t is time
    %mu is gravitational constant

    %Getting states
    r = state(1:3);
    rDot = state(4:end);

    %Two Body Problem
    dr = rDot;
    rDotDot = -1*mu*r ./ (norm(r))^3;

    %Calculate rDotDot due to SRP, SRP is in the negative x direction,
 is
    %constant
    rSat = [-1; 0; 0];
    pSR = 4.57e-6;
    CR = 1.2;

    %Solar radiation pressure
    aSR = -((pSR * CR * area) ./ mass) .* rSat / 1000; %km/s^2

    %Final rate of change
    outVec = [dr; rDotDot + aSR];
```

```
end


function [Xout, OEout, Tout] = propagate_spacecraft(X0, t0, tf, A, m)
%PROPAGATE_SPACECRAFT
%Author: Thomas Dunnington
%Date: 4/6/2023
%Summary: Description: This function accepts an initial state, initial
 time, final time, SRP area, and spacecraft mass in km,
% kg, s units and outputs the state and orbital elements at the final
 time.
% Inputs:
%   X0 - [6 by 1 ] spacecraft Cartesian state vector in the ACI frame
 [x, y, z,  #x,  #y,  #z]T at the initial time
%   t0 - scalar, initial time in seconds
%   tf - scalar, final time in seconds
%   A - scalar, SRP area in km2
%   m - scalar, spacecraft mass in kg
% Outputs:
%   Xout - [6 by 1 ] spacecraft Cartesian state vector in the ACI
 frame [x; y; z; x;#y; z] at the final time
%   OEout - [6 by 1 ] spacecraft orbital elements relative to the ACI
 frame [a; e; i; OMEGA; omega; Theta] at the final
%   time tf in km and degrees on [#180, 180]

%Assume bennu is located at the 0, 0, 0


%Create function handle
mu = 4.892 / 1000^3;
A = A * 1000^2; %Convert to m^2
funHandle = @(t, state)EOM(t, state, mu, A, m);

%Define tspan
tspan = t0:60:tf;

%Call ode45 to propagate state variables
options = odeset('RelTol',1e-12,'AbsTol',1e-12);
[Tout, stateOut] = ode45(funHandle, tspan, X0, options);

%Output final state
Xout = stateOut;

%Velocity
Vout = Xout(:,4:end);
speed = vecnorm(Vout, 2, 2);

%Distance
Rout = Xout(:,1:3);
distance = vecnorm(Rout, 2, 2);

%Computing the specific agular momentum
hVec = cross(Rout, Vout, 2);
h = vecnorm(hVec, 2, 2);
```

```
%Calculating line of nodes vector
Zhat = [zeros(length(stateOut(:,1)), 1), zeros(length(stateOut(:,1)),
 1), ones(length(stateOut(:,1)), 1)];
nVec = cross(Zhat, hVec, 2);
n = vecnorm(nVec, 2, 2);

%Calculate a
energy = 0.5*speed.^2 - mu./distance;
a = -mu ./ (2*energy);

%Computing the eccentricity vector
eVec = (1/mu)*cross(Vout, hVec, 2) - (1./vecnorm(Rout, 2, 2)).*Rout;
e = vecnorm(eVec, 2, 2);

%Calculating i
hz = hVec(:,3);
i = acosd(hz ./ h);

%Calculating OMEGA
nx = nVec(:,1);
OMEGA = acosd(nx ./ n);

%Quadrant Check
nyLog = nVec(:,2) < 0;
OMEGA(nyLog) = 360 - OMEGA(nyLog);

%Calculating omega
nDote = dot(nVec, eVec, 2) ./ (vecnorm(eVec, 2, 2) .* vecnorm(nVec, 2,
 2));
omega = acosd(nDote);

%Quadrant Check
ezLog = eVec(:,3) < 0;
omega(ezLog) = 360 - omega(ezLog);

%Calculating theta
eDotr = dot(eVec, Rout, 2) ./ (vecnorm(eVec, 2, 2) .* vecnorm(Rout, 2,
 2));
theta = acosd(eDotr);

%Quadrant Check
rDotv = dot(Rout, Vout, 2);
tempLog = rDotv < 0;
theta(tempLog) = 360 - theta(tempLog);


%Output elements
OEout = [a, e, i*pi/180, OMEGA*pi/180, omega*pi/180, theta*pi/180];


end
```

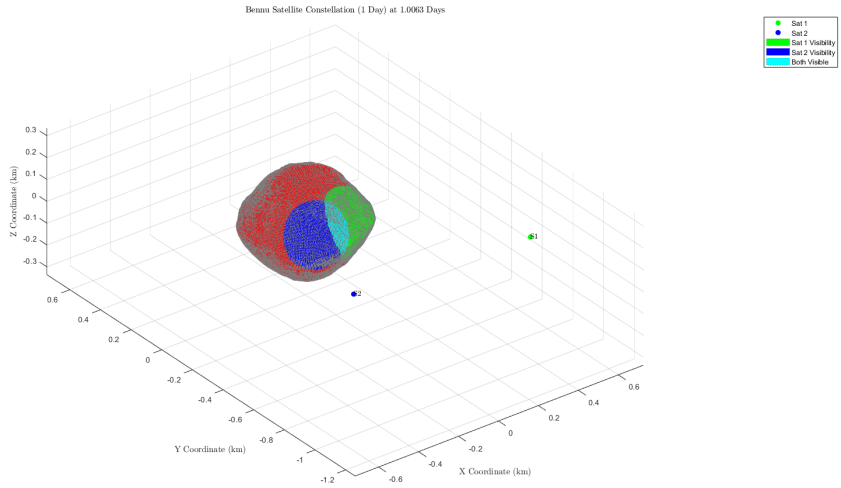
```
function [observable, elevationAngle, cameraAngle] = check_view(r,
 facetNumber, F, V, N)

%CHECK_VIEW
%Author: Thomas Dunnington
% Description:
% This function accepts an initial state, facet number, list of
 facets, and a list of verticies in km and
% outputs the if the facet is observable, the elevation angle of the
 facet, and the camera angle of the
% facet.
% Inputs:
%   r - [3 by 1 ] spacecraft Cartesian position vector in the body
 frame [x, y, z]T
%   facetNumber - scalar, facet index
%   F - [n x 3] matrix of vertices that form each face
%   V - [m x 3] matrix of vertex locations in implied body frame
%   N - number of satellites in the constellation
% Outputs:
%   observable - int, 0 for unobservable, 1 for observable
%   elevationAngle - scalar, elevation of spacecraft relative to the
 facet plane in radians
%   cameraAngle - scalar, angle of facet center relative to camera
 boresight

%Define FOV
FOV = pi / 9 / N;
```

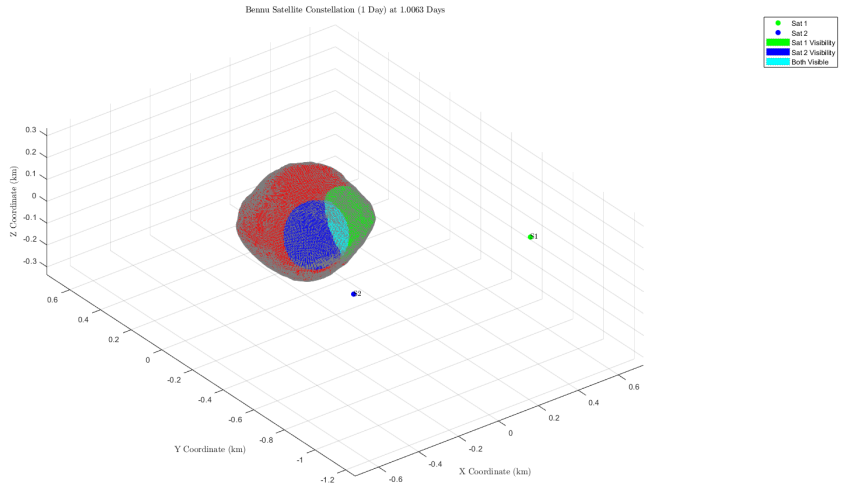


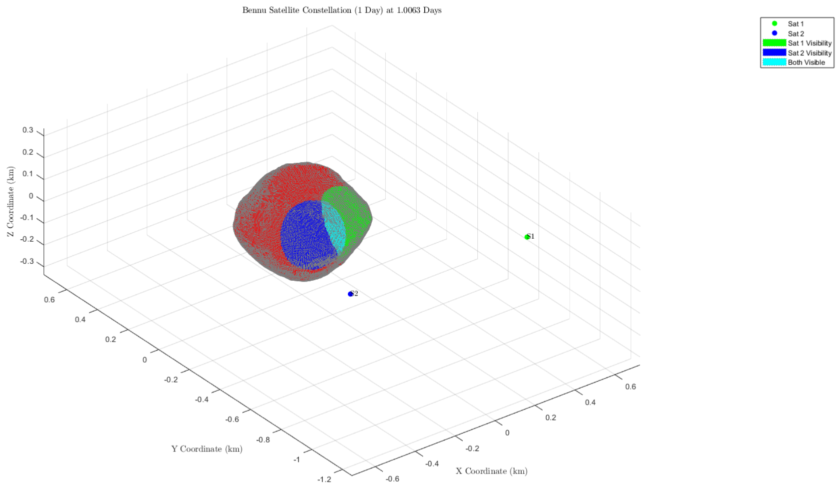
Triangulation method with vectors

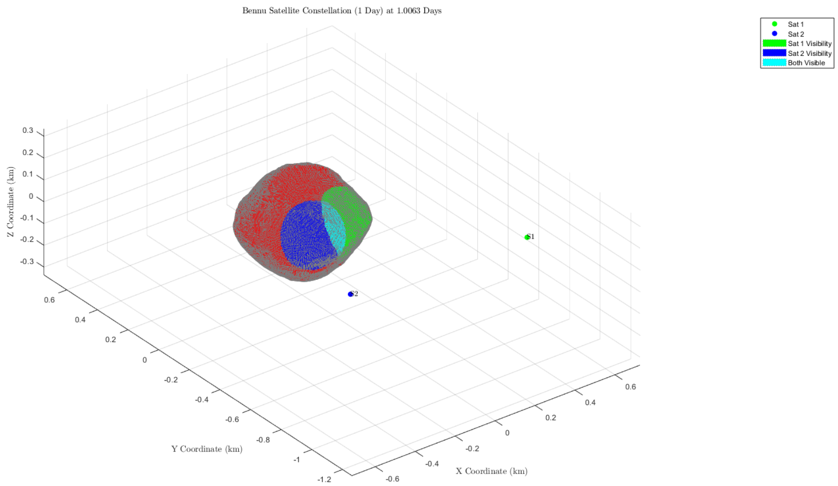
```
TR = triangulation(F, V);
rFVec = incenter(TR);
rVec = r.*ones(length(F(:,1)), 3);
norms = faceNormal(TR);
```

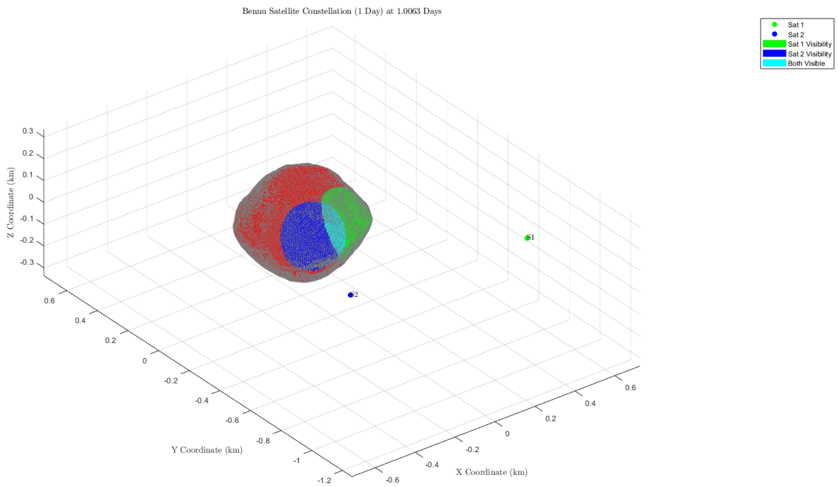
```
%Getting the vector to the face and the face normal
rF = rFVec;
n = norms;

%Defining the vector from the face to the spacecraft
rFS = rVec - rF;

%Finding angle between rFS and the normal of the face
elevationAngle = pi/2 - acos(dot(rFS, n, 2) ./ (vecnorm(rFS, 2, 2) .*
 vecnorm(n, 2, 2)));

%Finding field of view angle
cameraAngle = acos(dot(-1*rFS, -1*rVec, 2) ./ (vecnorm(rFS,2,2) .*
 vecnorm(rVec,2,2)));

%Determining whether or not the facet is visible
observable = elevationAngle >= 15*pi/180 & cameraAngle <= FOV;
```






Bennu Satellite Constellation (1 Day) at 1.0063 Days
Sat 1
Sat 2
Sat 1 Visibility
Sat 2 Visibility
Both Visible
Z Coordinate (km)
Y Coordinate (km)
X Coordinate (km)


Bennu Satellite Constellation (1 Day) at 1.0063 Days
Sat 1
Sat 2
Sat 1 Visibility
Sat 2 Visibility
Both Visible
Z Coordinate (km)
Y Coordinate (km)
X Coordinate (km)


Bennu Satellite Constellation (1 Day) at 1.0063 Days
Sat 1
Sat 2
Sat 1 Visibility
Sat 2 Visibility
Both Visible
Z Coordinate (km)
Y Coordinate (km)
X Coordinate (km)

```
end
```

Published with MATLAB® R2020b