Mini Project #3

Samantha Harris

April 6, 2020

Examination Rules

The rules for this take-home quiz are as follows:

(1) You are not permitted to communicate with anyone about the contents of this quiz during the entire duration of this quiz.

(2) You cannot get help from anyone in any form whatsoever.

(3) You are permitted to use any resources from books, lecture notes, or any online resources you find useful provided you abide by (1) above.

(4) All numerical solutions must be obtained using MATLAB in accordance with the instructions in the questions.

(5) You must type your solutions neatly in either Microsoft Word or LaTeX.

(6) You must be crystal clear with every step of your solution. In other words, every step in a derivation or statement you write must be unambiguous (that is, each step must have one and only one meaning). If any step is ambiguous, it will be assumed to be incorrect.

Anyone suspected of violating the quiz rules will be subject to the rules and regulations of the University of Florida Academic Honesty policy found by clicking **here**.

Point Distribution

The exam consists of a series of questions with the value of each question clearly indicated. Unless otherwise stated, full credit will be given for a proper application of a relevant concept. Contrariwise, no credit will be given for a concept applied incorrectly, ***even if the final answer is correct***.

University of Florida Honor Code

On your exam you must state and sign the University of Florida honor pledge as follows:

We, the members of the University of Florida community, pledge to hold ourselves and our peers to the highest standards of honesty and integrity. On all work submitted for credit by students at the university, the following pledge is either required or implied: On my honor, I have neither given nor received unauthorized aid in doing this quiz.

Name: Samantha Harris **UF-ID:** 11428884

Signature:  **Date:** April 7, 2020

Argument of the periapsis = 90 degrees

The spacecraft spends the majority of its time in the Northern Hemisphere (about 8.333 hours). When the longitude of the ascending node = 0 degrees the spacecraft is over Russia. When the longitude of the ascending node = 180 degrees the spacecraft is over North America. See figures 1 and 3.

Argument of the periapsis = 270 degrees

The spacecraft spends the majority of its time in the Southern Hemisphere (about 8.333 hours). When the longitude of the ascending node = 180 degrees the spacecraft is over Australia. When the longitude of the ascending node = 45 degrees the spacecraft is over South America. See figures 2 and 4.

Script

MiniProject3main.m

```
clc
clear all
close all

mu     =  398600;            %gravitational parameter
tau    = 24*60*60;           %orbital period
OmegaE = 2*pi/(24*60*60);   %earth rotation rate
e      = .25;                %eccentricity
a = nthroot(mu*(tau/(2*pi))^2,3);  %semi-major axis
inc    = deg2rad(63.4);     %orbital inclination
omega  = [deg2rad(90) deg2rad(270)]; %argument of periapsis
Omega  = [deg2rad(0) deg2rad(45) deg2rad(90) deg2rad(135) deg2rad(180)];
nu0    = 0; %start at periapsis
t0     = 0;
tValues  = [t0:5*60:tau]; %time span

omegaCell = {};
positioncell = {};
for j = 1:length(omega)
    for i = 1:length(Omega)
        oe(i,:)     = [a e Omega(i) inc omega(j) nu0]; %orbital elements
        [rPCI0(i,:),vPCI0(i,:)]  = oe2rv_Harris_Samantha(oe(i,:),mu);
        [rPCIf,vPCIf,E0,nu0,E,nu] = propagateKepler(t0,tau,tValues, rPCI0(i,:),vPCI0(i,:),mu);
        positioncell{i} = rPCIf;
        rvValuesECEF = eci2ecef(tValues(1,1:length(rPCIf)),rPCIf,OmegaE);
        [lonE(i,:),lat(i,:)] = ecef2LonLat(rvValuesECEF);
    end
   omegaCell{j} = positioncell;
end
```

```
%FIGURE 1
omega90 = omegaCell{1};
ra90 = omega90{1};
rb90 = omega90{2};
rc90 = omega90{3};
rd90 = omega90{4};
re90 = omega90{5};

figure
earthSphere
hold on
plot3(ra90(:,1),ra90(:,2), ra90(:,3),'LineWidth',2)
plot3(rb90(:,1),rb90(:,2), rb90(:,3),'LineWidth',2)
plot3(rc90(:,1),rc90(:,2), rc90(:,3),'LineWidth',2)
plot3(rd90(:,1),rd90(:,2), rd90(:,3),'LineWidth',2)
plot3(re90(:,1),re90(:,2), re90(:,3),'LineWidth',2)
title('Argument of Periapsis 90 Degrees');

    %mercator plotting
    rECEFa90 = eci2ecef(tValues,ra90,OmegaE);
    rECEFb90 = eci2ecef(tValues,rb90,OmegaE);
    rECEFc90 = eci2ecef(tValues,rc90,OmegaE);
    rECEFd90 = eci2ecef(tValues,rd90,OmegaE);
    rECEFe90 = eci2ecef(tValues,re90,OmegaE);
    [lonE(1,:),lat(1,:)] = ecef2LonLat(rECEFa90);
    [lonE(2,:),lat(2,:)] = ecef2LonLat(rECEFb90);
    [lonE(3,:),lat(3,:)] = ecef2LonLat(rECEFc90);
    [lonE(4,:),lat(4,:)] = ecef2LonLat(rECEFd90);
    [lonE(5,:),lat(5,:)] = ecef2LonLat(rECEFe90);
    mercatorDisplay5(lonE,lat)
    legend('Omega = 0','Omega = 45','Omega = 90', 'Omega = 135', 'Omega = 180');

% FIGURE 2
omega270 = omegaCell{2};
ra270 = omega270{1};
rb270 = omega270{2};
rc270 = omega270{3};
rd270 = omega270{4};
re270 = omega270{5};

figure
earthSphere
hold on
plot3(ra270(:,1),ra270(:,2), ra270(:,3),'LineWidth',2)
plot3(rb270(:,1),rb270(:,2), rb270(:,3),'LineWidth',2)
plot3(rc270(:,1),rc270(:,2), rc270(:,3),'LineWidth',2)
plot3(rd270(:,1),rd270(:,2), rd270(:,3),'LineWidth',2)
plot3(re270(:,1),re270(:,2), re270(:,3),'LineWidth',2)
title('Argument of Periapsis 270 Degrees');
```

```
%mercator plotting
rECEFa270 = eci2ecef(tValues,ra270,OmegaE);
rECEFb270 = eci2ecef(tValues,rb270,OmegaE);
rECEFc270 = eci2ecef(tValues,rc270,OmegaE);
rECEFd270 = eci2ecef(tValues,rd270,OmegaE);
rECEFe270 = eci2ecef(tValues,re270,OmegaE);
[lonEb(1,:),latb(1,:)] = ecef2LonLat(rECEFa270);
[lonEb(2,:),latb(2,:)] = ecef2LonLat(rECEFb270);
[lonEb(3,:),latb(3,:)] = ecef2LonLat(rECEFc270);
[lonEb(4,:),latb(4,:)] = ecef2LonLat(rECEFd270);
[lonEb(5,:),latb(5,:)] = ecef2LonLat(rECEFe270);
mercatorDisplay5(lonEb,latb)
legend('Omega = 0','Omega = 45','Omega = 90', 'Omega = 135', 'Omega = 180');


%time between crossing point
mean = mean(lonE(1,:));
find = find(lonE(1,:) > (mean - .007) & lonE(1,:) < (mean + .007));
seconds = tValues(find(4)) - tValues(find(2));
hours = seconds/60/60;
```

propagateKepler.m

```
function [rPCIf,vPCIf,E0,nu0,E,nu] = propagateKepler(t0,t, trange, rPCI0,vPCI0,mu)
% % ---------------------------------------------------------------- %
% % ----------------------- propagateKepler.m ----------------------- %
% % ------- Propagate Spacecraft Orbit Using Kepler's Equation ------- %
% % ---------------------------------------------------------------- %
% % Given a position and inertial velocity at a time t0 expressed in   %
% % planet-centered inertial (PCI) coordinates, determine the position %
% % and inertial velocity at a later time t on an elliptic orbit by    %
% % solving Kepler's equation.                                         %
% % ---------------------------------------------------------------- %
% % ------------- Inputs (Supplied Data) for Test Cases -------------- %
% % ---------------------------------------------------------------- %
% %    t0 = initial time                                               %
% %     t = terminal time                                              %
% % rPCI0 = Initial PCI Position                                       %
% % vPCI0 = Initial PCI Inertial Velocity                              %
% %    mu = planet gravitational parameter                             %
% % ---------------------------------------------------------------- %
% % --------- Output (Computed Quantities) from Test Cases ---------- %
% % ---------------------------------------------------------------- %
% % rPCIf = Terminal PCI Position                                      %
% % vPCIf = Terminal PCI Inertial Velocity                             %
% %    E0 = Eccentric Anomaly at Time t0                               %
```

```
% %   nu0 = True Anomaly at Time t0                                        %
% %     E = Eccentric Anomaly at Time t                                    %
% %    nu = True Anomaly at Time t                                         %
% % ------------------------------------------------------------------- %
% % -------- Note: all quantities must be in consistent units -------- %
% % ------------------------------------------------------------------- %
% % ---------------------------------------------------------------------- %
% % FUNCTIONS THAT WERE USED:
% %                     t0, t, r0, v0, mu -> rPCI0, vPCI0      r0v02rv.m
% %              t0, t, rsteps, vsteps, mu -> [rPCIf, vPCIf]     rvsteps.m
% % ---------------------------------------------------------------------- %
%find orbital elements with inital position and velocity
oe0 = rv2oe_Harris_Samantha(rPCI0,vPCI0,mu);
a     = oe0(1);
e     = oe0(2);
Omega = oe0(3);
inc   = oe0(4);
omega = oe0(5);
nu0   = oe0(6);

E0 = 2*atan2(sqrt(1-e)*sin(nu0/2),sqrt(1+e)*cos(nu0/2));

%trange = [t0,5*60 ,t];
for j = 1:length(trange)
    %use kepler solver to find E? and nu
    [E,nu] = KeplerSolver(a,e,t0,trange(j),nu0,mu);
    %use this new value of nu along with the previous orbital elements
    oe = [a, e, Omega, inc, omega, nu];
    %use these orbital elemtents to find position and velocity
    [rPCIf(j,1:3),vPCIf(j,1:3)]  = oe2rv_Harris_Samantha(oe,mu);
end
```

KeplerSolver.m

```
function [E,nu] = KeplerSolver(a,e,t0,t,nu0,mu)

% % ---------------------------------------------------------------------- %
% % --------- Kepler Solver for Propagation Along Elliptic Orbits -------- %
% % Solve Kepler's equation for the eccentric anomaly and the true anomaly %
% % given an initial time, t0, the initial true anomaly, nu0, the terminal %
% % time, t, the semi-major axis, a, eccentricity, e, and the planet       %
% % gravitational parameter, mu.  This function takes into account the     %
% % number of periapsis crossings en route from t0 to t and assumes that   %
% % the orbit is elliptic (that is, is is assumed that the eccentricity is %
```

```
% % strictly less than unity.                                                %
% % ------------------------------------------------------------------------ %
% % Inputs: a, e, mu, t, t0, and nu0                                         %
% %   a     = semi-major axis                                                %
% %   e     = eccentricity                                                   %
% %   t0    = initial time                                                   %
% %   t     = terminal time                                                  %
% %   nu0   = true anomaly at t0                                             %
% %   mu    = gravitational parameter of planet                              %
% % ------------------------------------------------------------------------ %
% % Outputs: E and nu                                                        %
% % ------------------------------------------------------------------------ %
% %   E     = the eccentric anomaly at time t                                %
% %   nu    = the true anomaly at time t                                     %
% % ------------------------------------------------------------------------ %
% % IMPORTANT: the units of a, mu, t0, and t must be consistent              %
% % ------------------------------------------------------------------------ %
% % FUNCTIONS THAT WERE USED:
% %                         e, nu0  -> E0      nu2E.m
% %          e, E0, t0, t, mu, a  -> E       E02E.m
% %                           e, E  -> nu      E2nu.m
% % ------------------------------------------------------------------------ %
% e, n0 -> E0
E0 = 2*atan2(sqrt(1-e)*sin(nu0/2),sqrt(1+e)*cos(nu0/2));
% e, E0, t0, t, mu, a  -> E
tp = t0 - sqrt(a^3/mu)*(E0-e*sin(E0)); %time at periapsis
tau = 2*pi*sqrt((a^3)/mu);               %orbital period
k = floor((t-tp)/tau);                 %number of times it passes periapsis
C = sqrt(mu/a^3)*(t-t0)+(E0-e*sin(E0))-2*pi*k; %contstant to help find E

%fixed point interation to find E
N = 10*floor(1/(1-e));
E = E0;
for j = 1:N
    E = e*sin(E) + C;
end

%use E to find nu
nu = 2*atan2(sqrt(1+e)*sin(E/2), sqrt(1-e)*cos(E/2));
```

eci2ecef.m

```
function rvValuesECEF = eci2ecef(tValues,rvValuesECI,OmegaE)

% ------------------------------------------------------------%
```

```
% ------------------------ ECI2ECEF.M ------------------------%
% ------------------------------------------------------------%
% Transform the values of the position along a spacecraft orbit %
% from Earth-centered inertial Cartesian coordinates to         %
% Earth-centered Earth-fixed Cartesian coordinates.             %
% represents the transformation matrix at the time t            %
% Inputs:                                                       %
%        tValues: column matrix of values of time at which the  %
%                 transformation is to be performed             %
%    rvValuesECI: matrix of values of the position expressed    %
%                 in Earth-centered inertial Cartesian          %
%                 coordinates and stored ROW-WISE               %
%         OmegaE: the rotation rate of the Earth                %
% Outputs:                                                      %
%    rvValuesECEF: matrix of values of the position expressed   %
%                  in Earth-centered Earth-fixed Cartesian      %
%                  and stored ROW-WISE                          %
% ------------------------------------------------------------%

rvValuesECEF = zeros(size(rvValuesECI)); %start with zero matrix
for j = 1:length(tValues)
    TI2E = dcmI2E(tValues(j),OmegaE); %get transformation matrix
    rvValuesECEF(j,1:3) = rvValuesECI(j,1:3)*TI2E';%transform
end
```

ecef2LonLat.m

```
function [lonE,lat] = ecef2LonLat(rvValuesECEF)

% ------------------------------------------------------------%
% ----------------------- ECEF2LONLAT.M ----------------------%
% ------------------------------------------------------------%
% Compute the Earth-relative longitude and geocentric latitude  %
% from the Earth-centered Earth-fixed position.                 %
% Input:                                                        %
%    rvValuesECEF: matrix of values of the position expressed   %
%                  in Earth-centered Earth-fixed Cartesian      %
%                  and stored ROW-WISE                          %
% Outputs:                                                      %
%    lonE: a column matrix containing the values of the         %
%          Earth-relative longitude (radians) .                 %
%     lat: a column matrix containing the values of the         %
%          geocentric latitude (radians)                        %
% ------------------------------------------------------------%
for j = 1:length(rvValuesECEF)
    lonE(j) = atan2(rvValuesECEF(j,2), rvValuesECEF(j,1));
```

```
    lat(j) = atan2(rvValuesECEF(j,3), sqrt(rvValuesECEF(j,1)^2 + rvValuesECEF(j,2)^2));
end
```

mercatorDisplay5.m

```
function mercatorDisplay5(lonE,lat)
earth = imread('earth.jpg');
% Open a new figure, then run the command "clf"
% Example:
figure;
clf
% Then run the code below.

image('CData',earth,'XData',[-180 180],'YData',[90 -90])
hold on

plot(lonE(1,:)*180/pi,lat(1,:)*180/pi,'o');
plot(lonE(2,:)*180/pi,lat(2,:)*180/pi,'s');
plot(lonE(3,:)*180/pi,lat(3,:)*180/pi,'d');
plot(lonE(4,:)*180/pi,lat(4,:)*180/pi,'v');
plot(lonE(5,:)*180/pi,lat(5,:)*180/pi,'>');
```

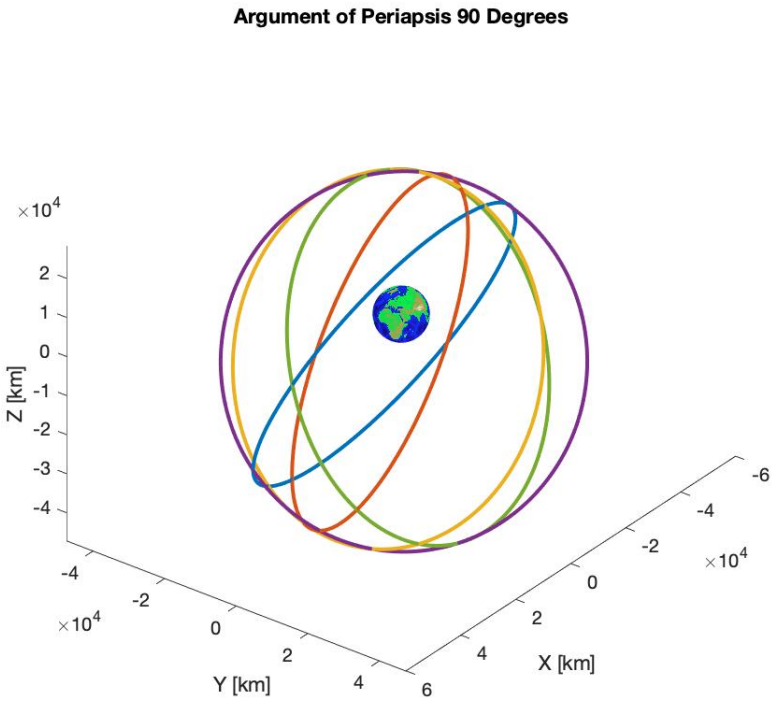

Figure 1: Different orbits when the argument of periapsis = 90 degrees

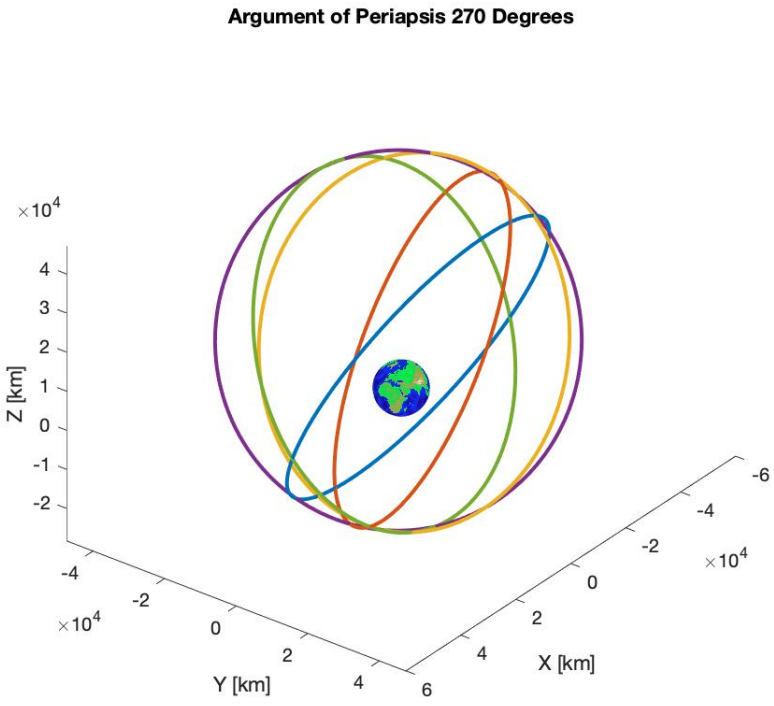

Figure 2: Different orbits when the argument of periapsis = 270 degrees

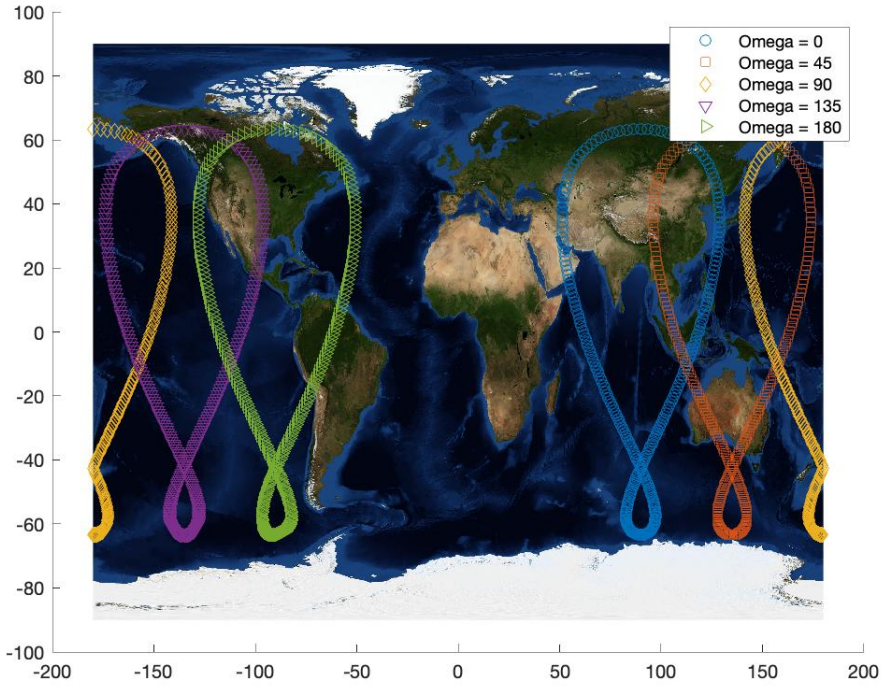

Figure 3: Argument of periapsis = 90 degrees.

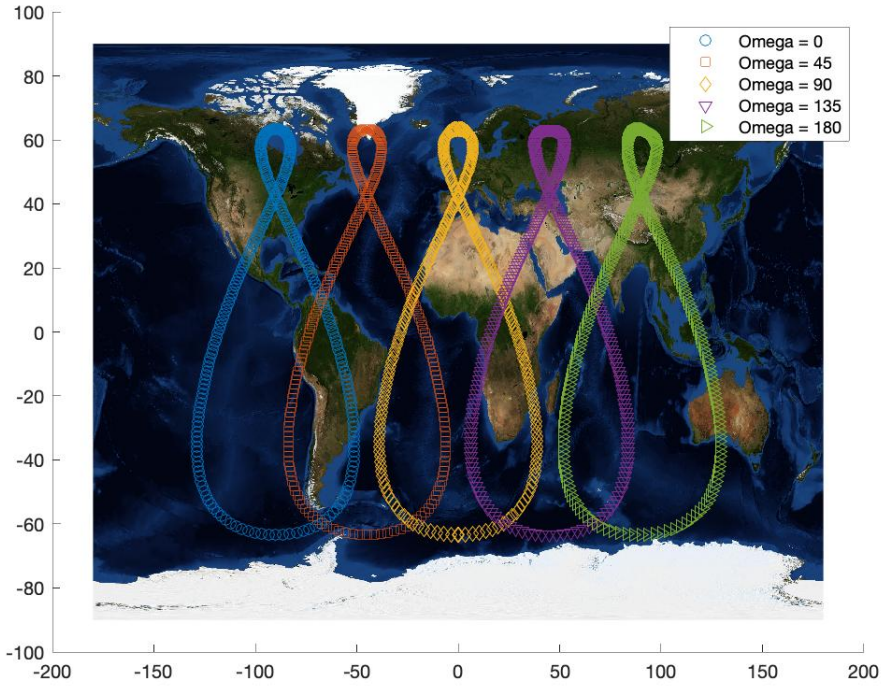

Figure 4: Argument of periapsis = 270 degrees.