Mini Project #2

Samantha Harris

March 15, 2020

Project Rules

The rules for this mini-project are as follows:

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

(2) Aside from the Professor, you are not permitted to get assistance from anyone in any form whatsoever.

(3) Although you are not permitted to comnunicate with anyone, you are permitted to use any resources from books, lecture notes, or any online resources you find useful provided you abide by (1) above. If you use any specific external resource (aside from lecture notes or course materials), you *must provide a citation* to the source from which you obtained the information. The citation should also appear as a reference in the bibliography of your submission.

(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 assignment 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 completing this assignment.

Name: Samantha Harris

UF-ID: 11428864

Signature: Samantha Harris

Date: 3/15/2026

Task 1-1 Matlab Function to Compute True Anomaly

rootFinder.m

```
function nu = rootFinder(f,t0,nu0,t,p,e,mu,N)
% ------------------------------------------------------------------%
% ------------------------ ROOTFINDER.M ----------------------------%
% ------------------------------------------------------------------%
% Find the root of a function of the form                           %
%           G(nu) = F(nu) - (t-t0) = 0                              %
% where F(nu) is given as                                           %
%                                                                   %
%                                 /nu                               %
%                                /                                  %
%                                |                                  %
%                      F(nu) = | f(q)dq                            %
%                                |                                  %
%                                /                                  %
%                               /nu0                                %
%                                                                   %
% The inputs and outputs of this function are as follows:           %
% Inputs:                                                           %
%       f: integrand of function                                    %
%     nu0: true anomaly at current time                             %
%      t0: current time                                             %
%       t: terminal time                                            %
%       p: semi-latus rectum                                        %
%       e: eccentricity                                             %
%      mu: gravitational parameter                                  %
%       N: number of Legendre-Gauss points                          %
% Output:                                                           %
%      nu: true anomaly at time t                                   %
% ------------------------------------------------------------------%
nuItr = 20; % go through 20 iterations
nu = nu0 + 5*pi/180;  %step up 5 degrees

for m = 1:nuItr %go through each iteration
    nuprev = nu;
    F = timeChangeIntegral(f, nu0, nu, p, e, mu, N);
    Gnu = F - (t - t0); %function
    GnuDeriv = f(nu, p, e, mu); %derivative of function
    nu = nu - Gnu/GnuDeriv; %newton's method
end
```

Task 1-2: Computation of True Anomaly at a Sequence of Times on Orbit

computeNu.m

```
function nuValues = computeNu(tValues,rv0,vv0,mu)

% ------------------------------------------------------------------%
% ------------------------ COMPUTENU.M -----------------------------%
% ------------------------------------------------------------------%
% Compute the values of the true anomaly at the times given in  %
% the column matrix TVALUES using the initial position given in %
% RV0 and the initial inertial velocity given in VV0.           %
% Inputs:                                                       %
%     tValues: column matrix of values of time                  %
%         rv0: initial position expressed in                    %
%              planet-centered inertial Cartesian coordinates   %
%         vv0: initial inertial velocity expressed in           %
%              planet-centered inertial Cartesian coordinates   %
%          mu: gravitational parameter of planet                %
% Output:                                                       %
%    nuValues: column matrix of values of true anomaly that is  %
%    of the same length as the column matrix TVALUES            %
% ------------------------------------------------------------------%
N = 20;
%orbital elements
oe = rv2oe_Harris_Samantha(rv0,vv0,mu);

a     = oe(1);           %semi-major axis.
e     = oe(2);           %eccentricity
Omega = oe(3);           %longitude of the ascending node (rad)
inc   = oe(4);           %inclination (rad)
omega = oe(5);           %argument of the periapsis (rad)
nu0   = oe(6);           %true anomaly (rad)
hv    = cross(rv0,vv0);  %angular momentum vector
h     = norm(hv);        %magnitude of angular momentum
p     = h^2/mu;          %semi-latus rectum

nuValues = zeros(size(tValues));
nuValues(1) = nu0;

for j = 2:length(tValues)
    nuValues(j) = rootFinder(@timeChangeIntegrand,tValues(j-1),nuValues(j-1),tValues(j),p,e,mu,N);
end
```

Task 2: Determination of Position and Velocity

computeRandV.m

```
function [rvValues,vvValues] = computeRandV(nuValues,a,e,Omega,inc,omega,mu)

% ------------------------------------------------------------%
% ----------------------- COMPUTERANDV.M ----------------------%
% ------------------------------------------------------------%
% Compute the values of the true anomaly at the times given in  %
% the column matrix TVALUES using the initial position given in %
% RV0 and the initial inertial velocity given in VV0.           %
% Inputs:                                                       %
%       nuValues: column matrix of values of true anomaly       %
%              a: semi-major axis                               %
%              e: eccentricity                                  %
%          Omega: longitude of the ascending node               %
%            inc: orbital inclination                           %
%          omega: argument of the periapsis                     %
%             mu: gravitational parameter of planet             %
% Outputs:                                                      %
%     rvValues: column matrix of values of position,            %
%               where each row in the matrix RVVALUES is        %
%               expressed in planet-centered inertial Cartesian %
%               coordinates                                     %
%     vvValues: column matrix of values of inertial velocity,   %
%               where each row in the matrix VVVALUES is        %
%               expressed in planet-centered inertial Cartesian %
%               coordinates                                     %
% ------------------------------------------------------------%
% NOTE: the number of ROWS in RVVALUES and VVVALUES should be   %
% the same as the number of ROWS in NUVALUES                    %
% ------------------------------------------------------------%
rvValues = zeros(length(nuValues),3); %create empty matrix for position
vvValues = zeros(length(nuValues),3); %create empty matrix for velocity

for j = 1:length(nuValues)
    oe = [a, e, Omega, inc, omega, nuValues(j)]; %use orbital elements
    [rv,vv]  = oe2rv_Harris_Samantha(oe,mu); %get position and velocity
    rvValues(j,1:3) = rv;
    vvValues(j,1:3) = vv;
end
```

Task 3-1: Transformation from ECI to ECEF Coordinates

dcmI2E.m

```
function TI2E = dcmI2E(t,OmegaE)
% ------------------------------------------------------------%
% ------------------------- DCMI2E.M -------------------------%
% ------------------------------------------------------------%
% Compute the transformation matrix from Earth-centered inertial%
% Cartesian coordinates to Earth-centered Earth-fixed Cartesian %
% coordinates. The transformation matrix is of size 3 by 3 and  %
% represents the transformation matrix at the time t            %
% Inputs:                                                       %
%           t: the time at which the transformation is desired  %
%      OmegaE: the rotation rate of the Earth                   %
% Outputs:                                                      %
%        TI2E: 3 by 3 matrix that transforms components of a    %
%              vector expressed in Earth-centered inertial      %
%              Cartesian coordinates to Earth-centered          %
%              Earth-fixed Cartesian coordinates                %
% ------------------------------------------------------------%
% NOTE: the units of the inputs T and OMEGAE must be consistent %
% and the angular rate must be in RADIANS PER TIME UNIT         %
% ------------------------------------------------------------%

TEI = [cos(OmegaE*t) -sin(OmegaE*t) 0; sin(OmegaE*t) cos(OmegaE*t) 0; 0 0 1];
TI2E = TEI';
```

Task 3-2: Computation of Position in ECEF Coordinates

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
```

Task 3-3: Determination of Earth-Relative Longitude and Latitude

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
```

Task 4-1: Visualization of the Orbit in Three-Dimensions

See figure 1.

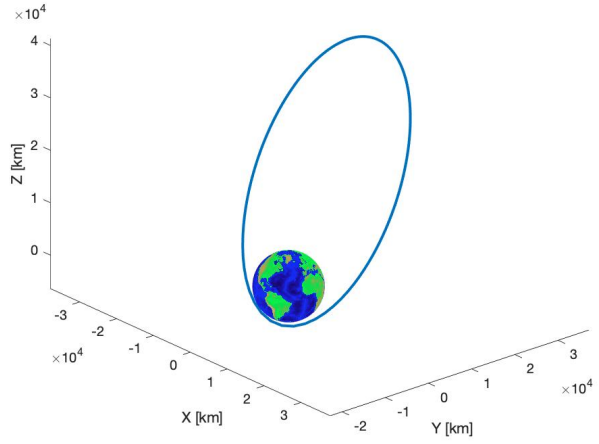

Figure 1: Visualization of the Orbit in Three-Dimensions

Task 4-2: Visualization of Groundtrack

See figure 2.

Task 4-3: Analysis of Particular Segments of the Orbit

```
------------------------------------------------------------
------------------------------------------------------------
The stimated amount of time spent by the spacecraft when the
Earth-relative longitude is between [80,100] degrees and
[-100,-80] degrees = 40199.999535 seconds or 11.166667 hours
------------------------------------------------------------
------------------------------------------------------------
```

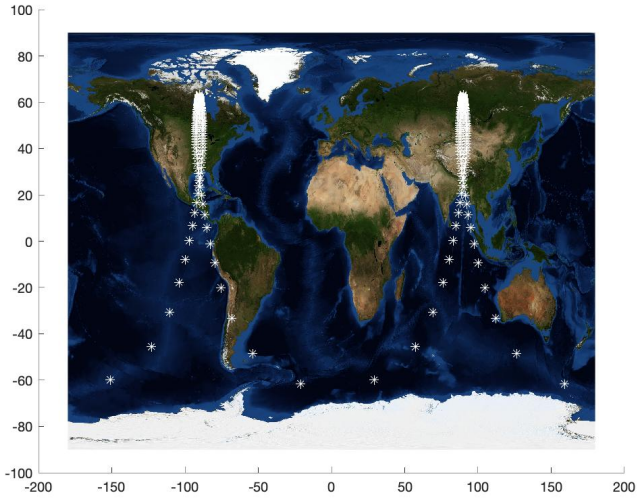

Figure 2: Visualization of the Groundtrack

```
                        Part (a):
These segments lie near apoapsis.
------------------------------------------------------------
------------------------------------------------------------
                        Part (b):
The maximum latitude was 63.399096 degrees and
the corresponding Earth-relative longitudes were
-89.998556 degrees and 89.998556 degrees.
------------------------------------------------------------
------------------------------------------------------------
                        Part (c):
The two countries and two cities within the countries that
correspond to the local maxima are Chesterfield Inlet, Canada
and Krasnoyarsk Krai, Russia
------------------------------------------------------------
------------------------------------------------------------
                        Part (d):
Looking at the direction of the eccentricity vector, the
size and shape of the orbit, and the orientation of the orbit
in space, it is clear that the spacecraft would spend the
amount of time it does in the Earth-relative longitude ranges
in order to gather information about the US and then send
that information to Russia.
------------------------------------------------------------
```

Script

MiniProject2Main.m

```
clc
clear all
close all

mu     = 398600;            %gravitational parameter
N      = 20;                %number of iterations
OmegaE = 2*pi/(24*60*60);   %earth rotation rate
rv0 = [-1217.39430415697 -3091.41210822807 -6173.40732877317]; %position
vv0 = [9.88635815507896 -0.446121737099303 -0.890884522967222]; %velocity

oe    = rv2oe_Harris_Samantha(rv0,vv0,mu);  %orbital elements
a     = oe(1);      %semi-major axis
e     = oe(2);      %eccentricity
Omega = oe(3);      %longitude of the ascending node (rad)
inc   = oe(4);      %inclination (rad)
omega = oe(5);      %argument of the periapsis (rad)
nu0   = oe(6);      %true anomaly (rad)
p     = a*(1-e^2);  %semi-latus rectum

tau = 2*pi*sqrt(a^3/mu);    %orbital period
t0 = timeChangeIntegral(@timeChangeIntegrand, pi, nu0, p, e, mu, N); %initial time
tf = t0 + 2*tau; %final time
tValues = [t0:5*60:tf]'; %time values
nuValues = computeNu(tValues,rv0,vv0,mu); %nu values of each time value

%position and velocity in ECI
[rvValues,vvValues] = computeRandV(nuValues,a,e,Omega,inc,omega,mu);
rvValuesECI = rvValues;
%position and velocity in ECEF
rvValuesECEF = eci2ecef(tValues,rvValuesECI,OmegaE);
%longitude and latitude
[lonE,lat] = ecef2LonLat(rvValuesECEF);

%visualization of the orbit in three dimensions
earthSphere
hold on
plot3(rvValues(:,1),rvValues(:,2), rvValues(:,3),'LineWidth',2)
view(49.5,22.8)
```

```
%visualization of groundtrack
mercatorDisplay(lonE,lat)

%find the indices where the longitudes are between 80 and 100
jvalues = find(lonE < 100*pi/180 & lonE > 80*pi/180);

%associate those positions to their corresponding values for rvValuesECEF
%use these values to convert position and velocity to orbital equations
oefornu1 = rv2oe_Harris_Samantha(rvValues(jvalues(1),1:3),vvValues(jvalues(1),1:3),mu);
oefornu2 = rv2oe_Harris_Samantha(rvValues(jvalues(end),1:3),vvValues(jvalues(end),1:3),mu);

%time spent by aircraft when longitude is [80,100] and [-100,-80]
deltatsec = timeChangeIntegral(@timeChangeIntegrand, oefornu1(6), oefornu2(6),p,e,mu, N);

% NEAR APOAPSIS
goingfrom = oefornu1(6);
to = oefornu2(6);

% LATITUDE LOCAL MAXIMA
maxlat = max(lat);
maxlatIndex = find(lat == maxlat);
longitudeUSA = lonE(maxlatIndex);


fprintf('-------------------------------------------------------------\n');
fprintf('-------------------------------------------------------------\n');
fprintf('The stimated amount of time spent by the spacecraft when the\n');
fprintf('Earth-relative longitude is between [80,100] degrees and \n');
fprintf('[-100,-80] degrees = %8.6f seconds or %8.6f hours\n', deltatsec, deltatsec/60/60);
fprintf('-------------------------------------------------------------\n');
fprintf('-------------------------------------------------------------\n');
fprintf('                        Part (a):                          \n');
fprintf('These segments lie near apoapsis. \n');
fprintf('-------------------------------------------------------------\n');
fprintf('-------------------------------------------------------------\n');
fprintf('                        Part (b):                          \n');
fprintf('The maximum latitude was %8.6f degrees and\n',maxlat*180/pi);
fprintf('the corresponding Earth-relative longitudes were\n');
fprintf('%8.6f degrees and %8.6f degrees.\n',longitudeUSA*180/pi,abs(longitudeUSA*180/pi));
fprintf('-------------------------------------------------------------\n');
fprintf('-------------------------------------------------------------\n');
fprintf('                        Part (c):                          \n');
fprintf('The two countries and two cities within the countries that\n');
fprintf('correspond to the local maxima are Chesterfield Inlet, Canada \n');
fprintf('and Krasnoyarsk Krai, Russia \n');
fprintf('-------------------------------------------------------------\n');
```

```
fprintf('-------------------------------------------------------------\n');
fprintf('                        Part (d):                          \n');
fprintf('Looking at the direction of the eccentricity vector, the\n');
fprintf('size and shape of the orbit, and the orientation of the orbit\n');
fprintf('in space, it is clear that the spacecraft would spend the\n');
fprintf('amount of time it does in the Earth-relative longitude ranges\n');
fprintf('in order to gather information about the US and then send\n');
fprintf('that information to Russia.\n');
fprintf('-------------------------------------------------------------\n');
fprintf('-------------------------------------------------------------\n');
```

mercatorDisplay.m

```
function mercatorDisplay(lonE,lat)
earth = imread('earth.jpg');
% Open a new figure, then run the command "clf"
% Example:
figure(2);
clf
% Then run the code below.
image('CData',earth,'XData',[-180 180],'YData',[90 -90])
hold on

plot(lonE*180/pi,lat*180/pi,'w*');
```