Mini Project #4

Samantha Harris

April 26, 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 cannot use *any* code that I have provided you.

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

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

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

(7) 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.

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: Samantha Harris **Date:** 4/26/2020

Question 1

twoImpulseHohmann.m

```
function [R, T, imp1, imp2] = twoImpulseHohmann(r1, r2, inc1, inc2, Omega1, Omega2, mu)

omega = 0; %initial argument of the periapsis
nu    = 0; %initial true anomaly
e     = 0; %eccentricity of circuluar orbits
%%
a = (r1 + r2)/2;   %semi-major axis of transfer orbit

%oe
oe1 = [r1, e, Omega1, inc1, omega, nu];
oe2 = [r2, e, Omega2, inc2, omega, nu];
[rv0,vv0] = oe2rv_Harris_Samantha(oe1, mu);
[rvf,vvf] = oe2rv_Harris_Samantha(oe2, mu);

h1 = cross(rv0, vv0);      %angular momentum of first orbit
h2 = cross(rvf, vvf);      %angular momentum of second orbit
theta = acos(dot(h1, h2)/(norm(h1)*norm(h2))); %orbit crank

%intersection between two orbits
l = cross(h1, h2)/norm(cross(h1, h2));
%% direction vectors
u1m = cross(h1, l)/(norm(h1));
w2 = h1/norm(h1);
u1p = cos(theta)*u1m + sin(theta)*w2;
u2m = u1p;
u2p = -(cross(h2, l)/norm(h2));

%% speeds
s1m = sqrt(mu/r1);
s1p = sqrt(mu*(2/r1 - 1/a));
s2m = sqrt(mu*(2/r2 - 1/a));
s2p = sqrt(mu/r2);

%% velocities
v1m = s1m*u1m;
v1p = s1p*u1p;
v2m = s2m*u2m;
v2p = s2p*u2p;

%% Impulses
%impulses vector
imp1v = v1p - v1m;
```

```
imp2v = v2p - v2m;

%impulses scalar
imp1 = norm(imp1v);
imp2 = norm(imp2v);

%% transfer orbit 1
rstart = r1*l;
halfTau = pi*sqrt(a^3/mu);
[rvt1,vvt1,E0,nu0,E,nu] = propagateKepler(0,halfTau, rstart,v1p,mu);

%% outputs
R = rvt1;
T = linspace(0, halfTau, 500)';
```

Question 2

Example Outputs for N = 2

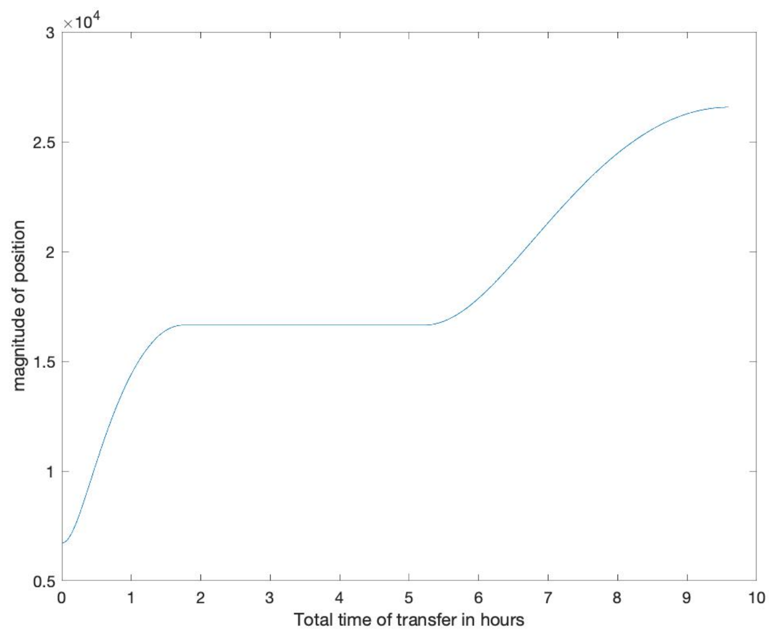

Figure 1: Graphic display of the total time required to accomplish transfer.

A graphic display of the total time required to accomplish transfer can be seen in figure 1.

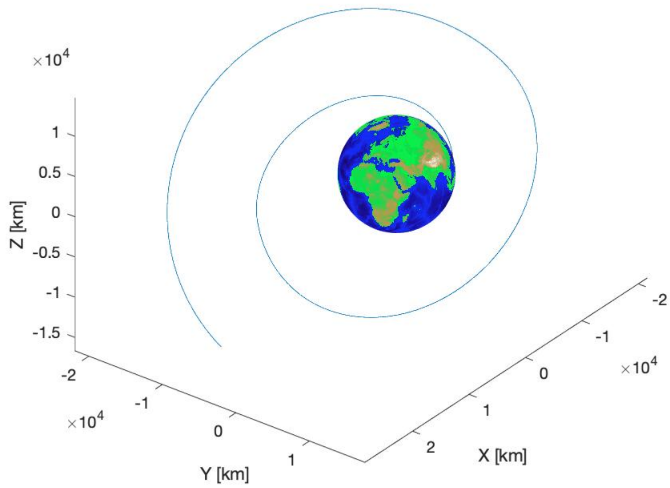

Figure 2: A graphic display of the 3-D view of the 2N impulse orbit transfer.

A graphic display of the 3-D view of the 2N impulse orbit transfer can be seen in figure 2.

A graphic display of the groundtrack of the orbit transfer can be seen in figure 3.

twoNImpulseOrbitTransfer.m

```
function [R, T, imp1, imp2, impTotal] = twoNImpulseOrbitTransfer(r0, rf, inc0, incf, Omega0, Omegaf,...
N, mu)
%% inputs
% r0     = radius of starting circular orbit
% rf     = radius of ending circular orbit
% inc0   = inclination of starting circular orbit
% incf   = inclination of ending circular orbit
% Omega0 = long of ascending node of starting circular orbit
% Omegaf = long of ascending node of ending circular orbit
% N      = number of pairs of periapsis/apoapsis impulses
% mu     = gravitational parameter of earth
%% outputs
% R       = position matrix Mx3 of transfer orbit
% T       = time points associated with R
% imp1    = column vector of length N, impulses at intermediate apoapsis
```

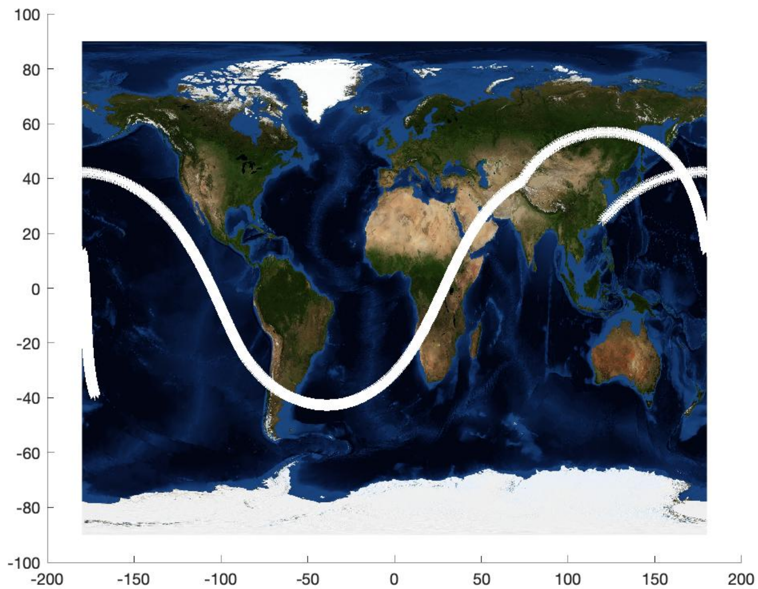

Figure 3: A graphic display of the groundtrack of the orbit transfer.

```
%  imp2     = column vector of length N, impulses at intermediate periapsis
%  impTotal = magnitude of the total impulse
%% other info
%  1/N =
%          -fraction of total required change of
%            periapsis/apoapsis radius
%          -fraction of total orbital inclination change
%%
e = 0;
omega = 0;
nu = 0;
t0 = 0;
Ncircle = 500;
OmegaE = 2*pi/(24*60*60);   %earth rotation rate
%% create orbital elements for each circle orbit
n = linspace(0, N, N+1);
for i = 1:length(n)
    radiusn(i)     = r0 + (n(i)/N)*(rf-r0);
    Omegan(i) = Omega0 + (n(i)/N)*(Omegaf-Omega0);
    incn(i)   = inc0 + (n(i)/N)*(incf-inc0);
    oeCircle(i, 1:6) = [radiusn(i) e Omegan(i) incn(i) omega nu];
    [rv(i, 1:3),vv(i, 1:3)] = oe2rv_Harris_Samantha(oeCircle(i, 1:6), mu);
end
```

```
%% create angular momentum lists
for i = 1:length(n)
    h(i,1:3) = cross(rv(i, 1:3),vv(i, 1:3));      %angular momentum of first orbit
end
%% create l lists
for i = 1:length(n)-1
    l(i,1:3) = cross(h(i, 1:3), h(i+1, 1:3))/norm(cross(h(i, 1:3), h(i+1, 1:3)));
end
%% determining how long the circle orbit will go
for i = 1:N-1
    phi(i) = acos(dot(l(i,1:3), l(i+1,1:3)));
    alpha(i) = phi(i)+pi;
    ratio(i) = alpha(i)/(2*pi);
end
%% direction of velocity vector on circle orbits
for i = 1:N
 velocityCircle = cross(h(i+1,1:3), -l(i,1:3));
 velDirection(i,1:3) = velocityCircle/norm(velocityCircle);
end
%% R and T
if N == 1
    for i = N
      %% transfer orbit n
     [transferPosition{i}, transferTime{i}, imp1(i,1), imp2(1,i,1)] = twoImpulseHohmann(radiusn(i), ra
         incn(i), incn(i+1), Omegan(i), Omegan(i+1), mu);
     t0(i) = transferTime{i}(end, 1);
    end
else
    for i = 1:N-1
     %% intermediate transfer orbit n
     [transferPosition{i},transferTime{i},imp1(i,1),imp2(i)]=twoImpulseHohmann(radiusn(i),...
     radiusn(i+1),...
         incn(i), incn(i+1), Omegan(i), Omegan(i+1), mu);
         t0(i) = transferTime{i}(end, 1);
     %% intermediate circle orbit n
     tau = ratio(i)*2*pi*sqrt(radiusn(i+1)^3/mu);
     [circleTime{i}, circlePosition{i}, vECIf, w1, w2, w3] = propogateOnCircle(transferPosition{i}(end
         norm(vv(i+1, 1:3))*velDirection(i,1:3),...
         t0(i), t0(i) + tau, mu, Ncircle);
    end
    for i = N
     %% last transfer
     [transferPosition{i}, transferTime{i}, imp1(i,1), imp2(i)] = twoImpulseHohmann(radiusn(i), radius
         incn(i), incn(i+1), Omegan(i), Omegan(i+1), mu);
         t0(i) = transferTime{i}(end, 1);
    end
end
```

```
 %% putting together all position and all time
if N == 1
      R = [transferPosition{N}];
      T = [transferTime{N}];
else
    R = [];
    for i = 1:N-1
     R = [R; transferPosition{i};circlePosition{i}];
    end
    for i = N
     R = [R; transferPosition{i}];
    end
    round = 0;
    T = [];
     for i = 1:N-1
         round = round + [transferTime{i};circleTime{i}'];
         T = [T; round];
         round = T(end,1);
     end
     for i = N
         T = [T; T(end,1)+ transferTime{i}];
     end
end
%% total impulse
 impTotal = sum(imp1) + sum(imp2);



 %% PLOTS


 %% plot the total time required to complete the orbit transfer as a function of N
for i = 1:length(T)
    radius(i) = norm(R(i, 1:3));
end
figure
plot(T/60/60,radius)
xlabel('Total time of transfer in hours')
ylabel('magnitude of position')

%% Three dimensional view of the 2N impulse orbital transfer
    figure
    earthSphere
    hold on
    plot3(R(:, 1), R(:, 2), R(:, 3));
%% groundtrack of the transfer
    %mercator plotting
    rECEF = eci2ecef(T,R,OmegaE);
    [lonE(1,:),lat(1,:)] = ecef2LonLat(rECEF);
    mercatorDisplay(lonE,lat)
```

```
end
```

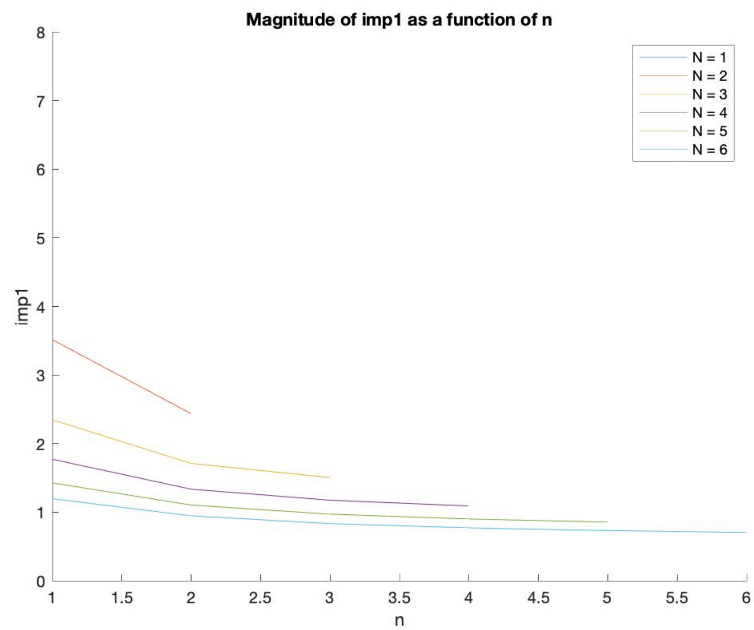

Figure 4: Magnitudes of each of the N impulses applied at the intermediate periapsis as a function of n.

Question 3

- Magnitudes of each of the N impulses applied at the intermediate periapsis as a function of n can be seen in figure 4.
- Magnitudes of each of the N impulses applied at the intermediate apoapsis as a function of n can be seen in figure 5.
- The total impulse as a function of N can be seen in figure 6.
- The time required to complete the orbit transfer as a function of N can be seen in figure 7.
- Six separate 3-D views for each of the orbit transfers N = [1 2 3 4 5 6] can be seen in figures 8 - 13
- Six separate groundtracks for each of the six orbit transfers can be seen in figures 14 - 19
- The total impulse increases as a function of N because the difference in velocities becomes greater with an increasing N.

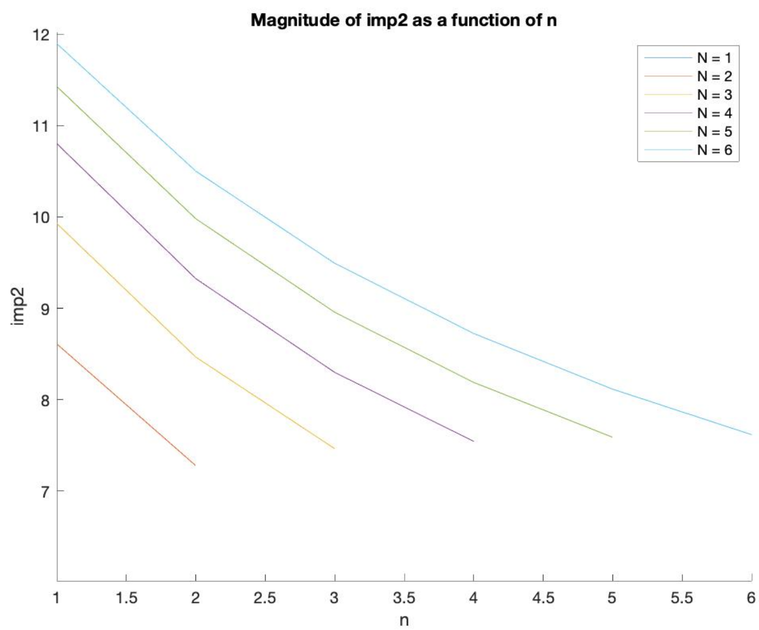

Figure 5: Magnitudes of each of the N impulses applied at the intermediate apoapsis as a function of n.

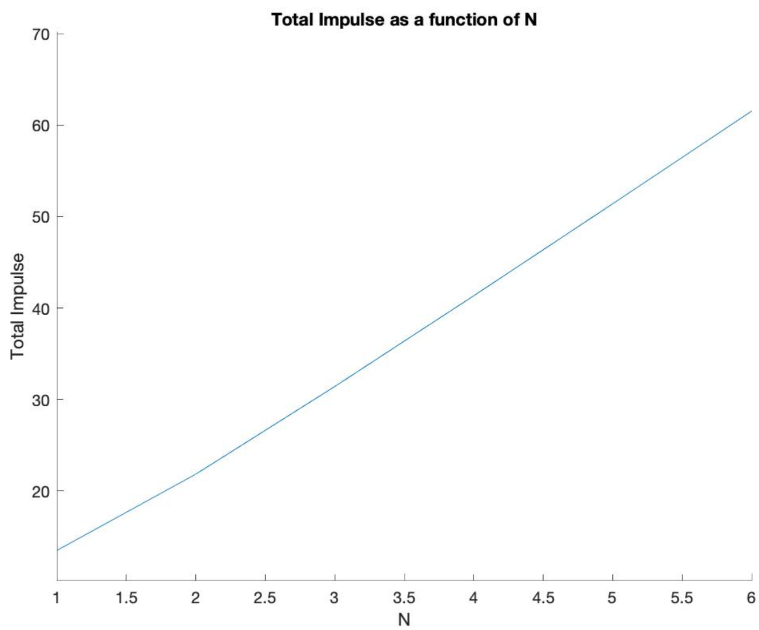

Figure 6: The total impulse as a function of N.

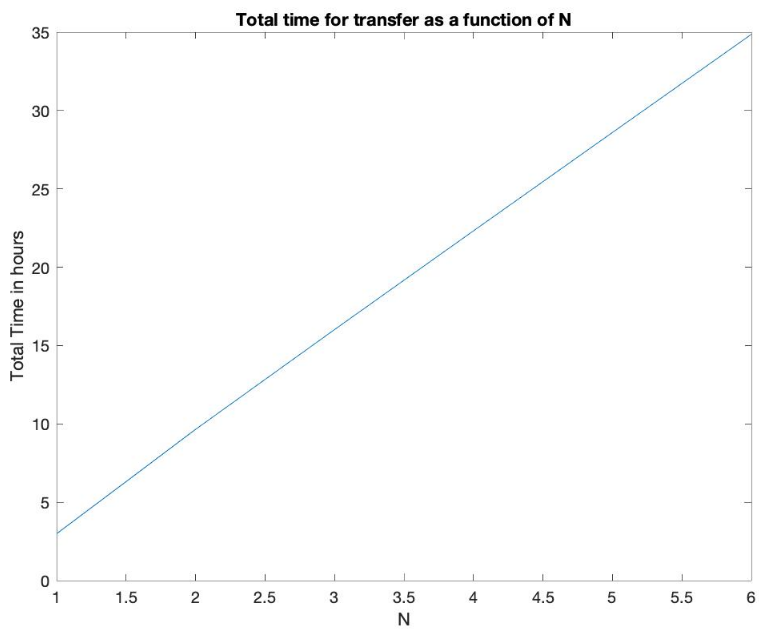

Figure 7: The time required to complete the orbit transfer as a function of N.

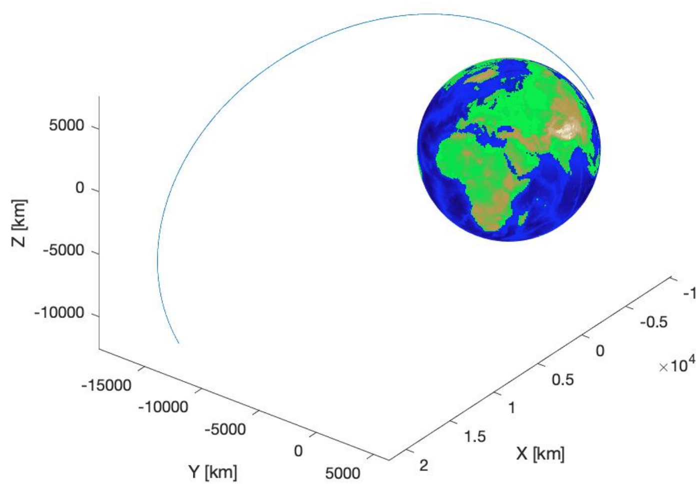

Figure 8: 3-D view for the orbit transfers when N = 1.

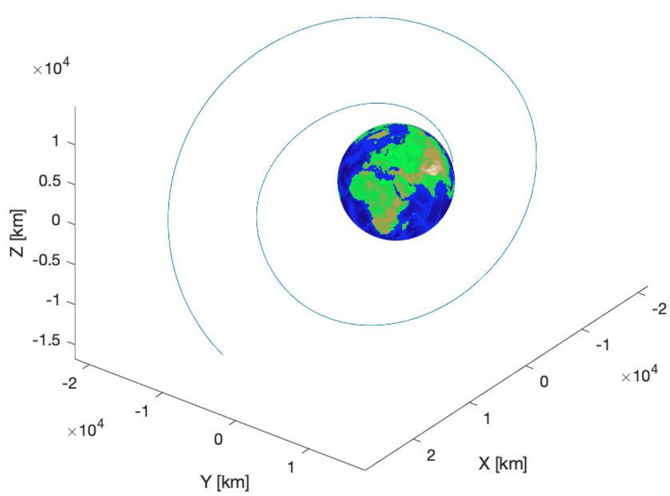

Figure 9: 3-D view for the orbit transfers when N = 2.

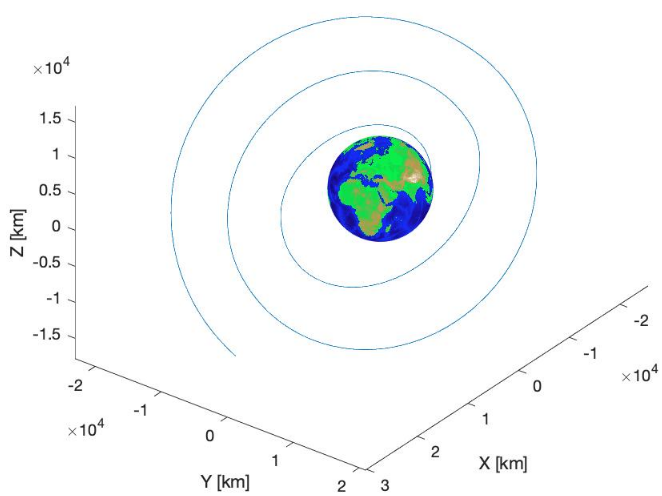

Figure 10: 3-D view for the orbit transfers when N = 3.

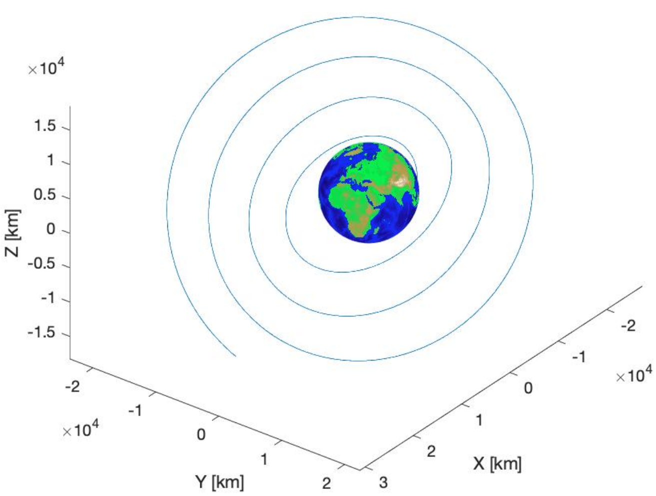

Figure 11: 3-D view for the orbit transfers when N = 4.

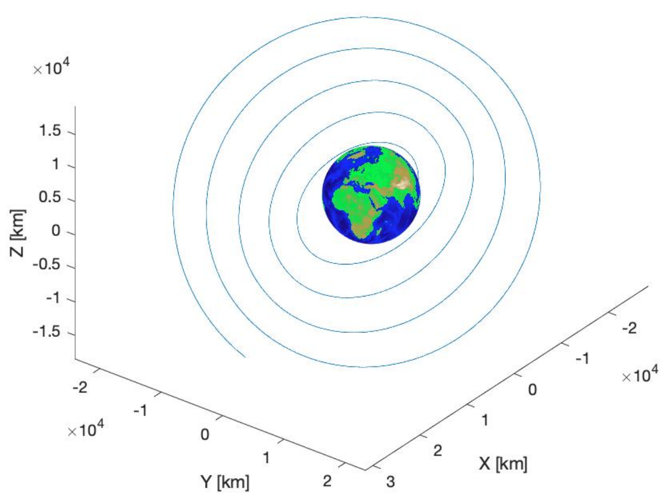

Figure 12: 3-D view for the orbit transfers when N = 5.

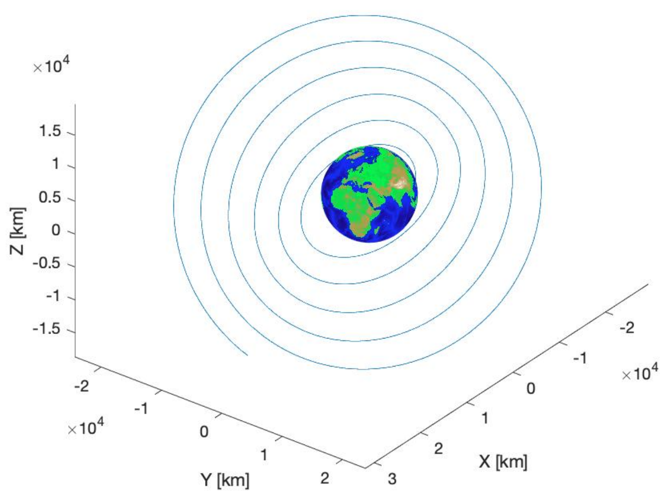

Figure 13: 3-D view for the orbit transfers when N = 6.

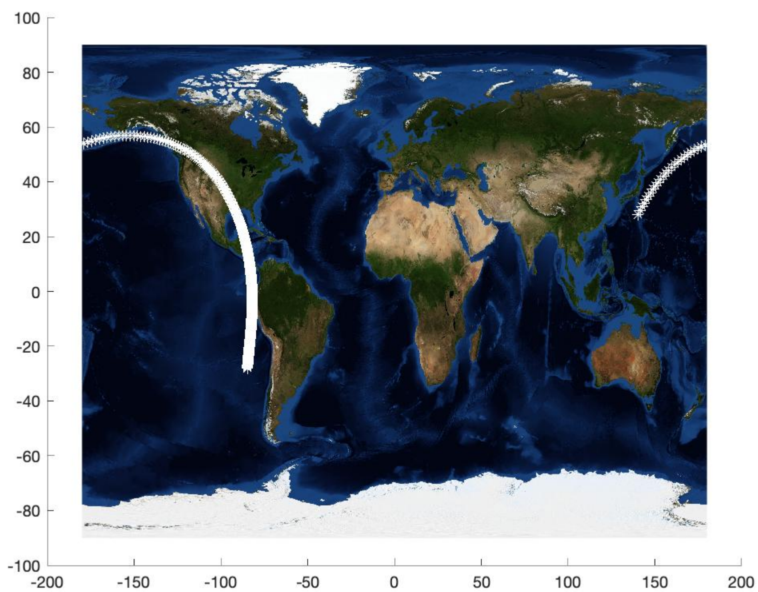

Figure 14: 3-D view for the orbit transfers when N = 1.

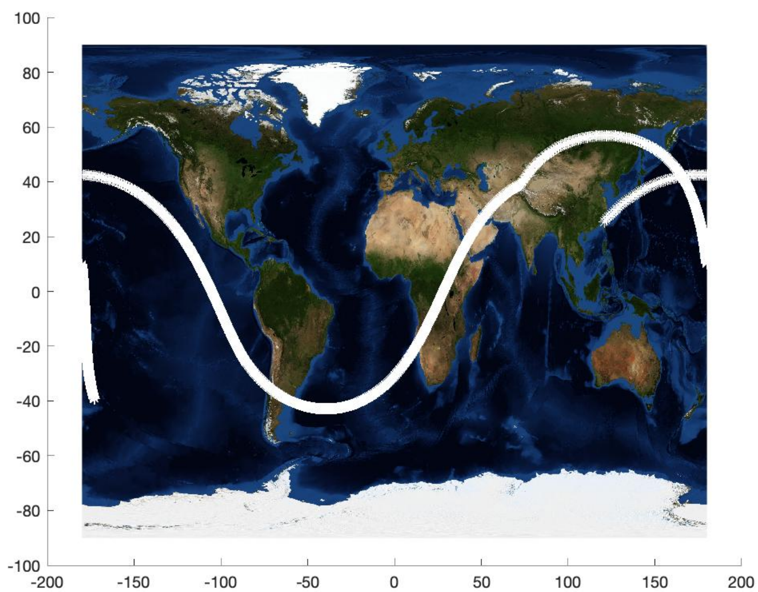

Figure 15: 3-D view for the orbit transfers when N = 2.

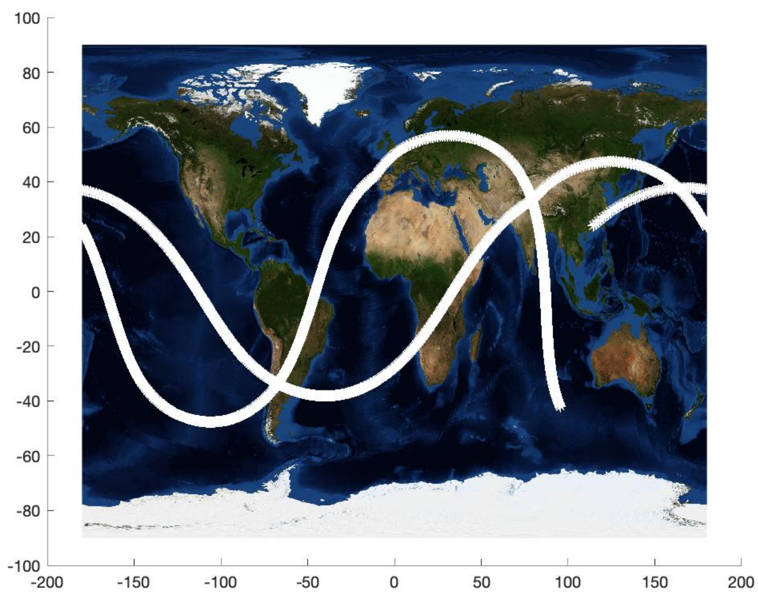

Figure 16: 3-D view for the orbit transfers when N = 3.

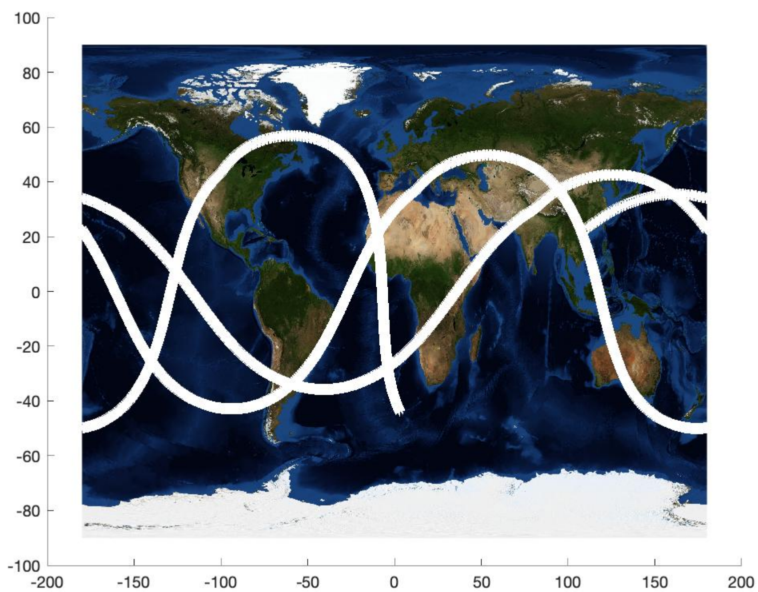

Figure 17: 3-D view for the orbit transfers when N = 4.

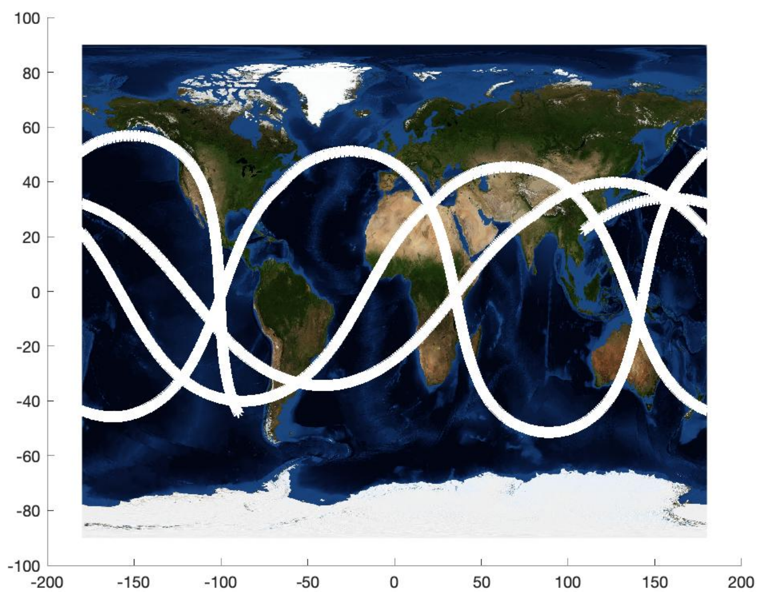

Figure 18: 3-D view for the orbit transfers when N = 5.

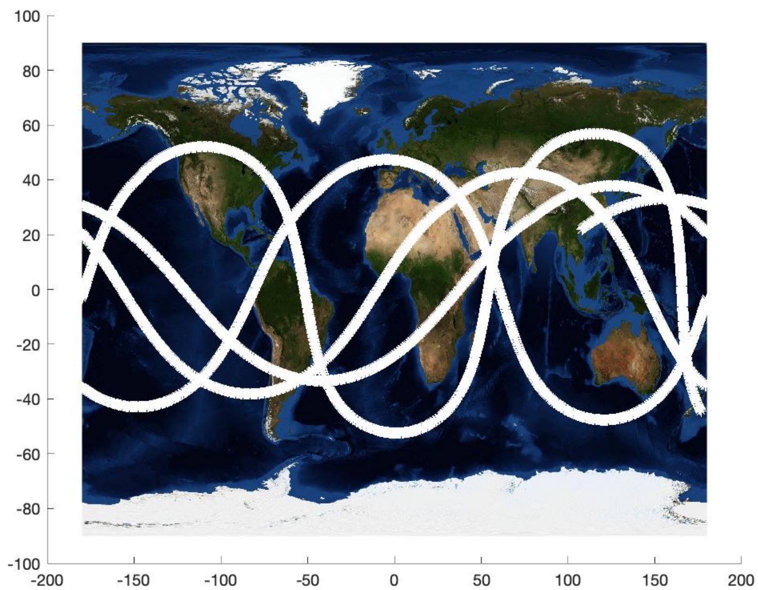

Figure 19: 3-D view for the orbit transfers when N = 6.