Walker-Delta Satellite Constellation for Earth Observation: SPE 510 Midterm Report

Kelvin Loh (20114534)

Room 3333, Department Of Aerospace Engineering

KAIST

kklloh@kaist.ac.kr

March 31, 2012

Abstract

A suite of MATLAB functions are developed to analyze the Walker-Delta Constellation. Since this is one of the more popular constellations, two cases are also being thought out to determine if the constellation pattern can be a solution for the cases that require extensive Earth observation. It is shown that the constellation is sensitive to altitude changes.

1 Introduction

A constellation is a set of satellites distributed over space intended to work together to achieve common objectives. The Walker-Delta Constellation is one such popular constellation design since it is the most symmetric. In order to study the performance of this type of constellation, a set of computational tools have to be developed. The theoretical foundations are important as it directly affects the accuracy of the analysis. The functions are tested and the results of the tests are also reported in this report. Additional models are also included in the code such as low-thrust maneuvers tangential to the spacecraft flight path, but they are only documented in the comments of the MATLAB functions.

2 Theoretical Models

This section covers the theoretical basis for the codes used to evaluate the performances of the Walker-Delta constellation to cover a target on Earth for observation. Proofs of the models used are not provided in this report.

2.1 Equations of relative motion

The two-body equation of motion, namely, Newton's second law and his universal law of gravitation form the starting point for any study of astrodynamics. Equation 1 describes the motion of a body (satellite) of negligible mass with respect to another body (Earth) under the influence of a gravitational force. This then assumes that an inertial frame of reference can be centered at the barycenter of the massive body.

$$\ddot{\vec{r}} = -\frac{\mu}{r^2}\,\frac{\vec{r}}{r} \tag{1}$$

In Newton's law of gravitation, the gravitational force can be expressed in terms of the gradient ∇ of a scalar gravitational potential function ϕ of a body. For a spherical body

$$\vec{F} = \nabla\phi = \nabla\left(\frac{\mu}{r}\right) \tag{2}$$

For a non-spherical body, ϕ can contain additional terms which can be used to determine $\vec{F}$, and subsequently $\ddot{\vec{r}}$. As the inertial reference frame is centered at the center of the massive body, Equation 1 is also known as the fundamental equation of relative two-body motion. The MATLAB function orbit.m uses this equation to determine the satellite motion.

2.2 Keplerian Orbit Propagation

Kepler's laws are derived from Equation 1. The orbital position as a function of time can be decribed by the mean anomaly, M_e. The propagation routines used in the codes ground_track2.m and satellite_constellation.m are based on solving the Kepler's equation (Equation 3) for each time step.

$$M_e = E - e\sin(E) \tag{3}$$

where E is the eccentric anomaly, and e is the eccentricity.

For elliptical orbits, it can be shown [1] that

$$M_e = \frac{\mu^2}{h^3}\left(1 - e^2\right)^{\frac{3}{2}} t \tag{4}$$

So, Equation 4 becomes

$$M_e = \frac{2\pi}{T} t \tag{5}$$

With this, it can be seen that numerically solving the Kepler's equation for each time step and directly using an Euler forward first-order time-stepping scheme is faster than directly integrating Equation 1 using a Runge-Kutta (4th-order or higher) fixed time-step scheme. Therefore, this method is primarily used to propagate the satellite constellation for this project.

2.3 Orbit Perturbations

Equation 1 assumes that the revolving body experiences forces only from the central body. However, this assumption does not hold for actual orbits. In reality, there are many perturbations to the orbital motion. The equation can then be generalized to include all the other perturbation forces acting on the body as described by Equation 6.

$$\ddot{\vec{r}} = -\frac{\mu}{r^2}\,\frac{\vec{r}}{r} + \vec{a}_p \tag{6}$$

As the Earth is not a perfect sphere, the gravitational potential will deviate from that of a perfect sphere. Due to this effect, the orbital motion will encounter perturbations. This project considers the effects of Earth Gravity Harmonics mainly, the J_2 perturbation on the secular rates of the right ascension of ascending node (Ω), the argument of perigee (ω_p), and a small correction to the mean motion of the orbit ($\dot{M}_e$). The rates are computed by the following equations:

$$\dot{\Omega} = -\frac{3}{2}\frac{J_2 R_E^2}{p^2}\bar{n}\cos i \tag{7}$$

$$\dot{\omega}_p = \frac{3}{2}\frac{J_2 R_E^2}{p^2}\bar{n}\left(2 - \frac{5}{2}\sin^2 i\right) \tag{8}$$

$$\dot{M}_e = \bar{n} = \sqrt{\frac{\mu}{a_0^3}}\left[1 + \frac{3}{2}\frac{J_2 R_E^2}{p^2}\left(1 - \frac{3}{2}\sin^2 i\right)\sqrt{1 - e^2}\right] \tag{9}$$

with the symbols taking their usual notations. Atmospheric drag perturbation effects was not included in the simulations since it is assumed that the main propulsion system onboard a satellite will perform the necessary station-keeping maneuvers during the mission.

For the special perturbation method employed in orbit.m, Cowell's method [2] was used. From Equation 2, it is seen that the gravitational force acting on the revolving body can be described by the gradient of a potential function. From this, the potential theory provides a clear picture of the gravity harmonics of the Earth. The potential function for the primary body due to the J_2 perturbation is described by Equation 10 in the inertial Cartesian coordinate frame (ECI).

$$\phi = \frac{\mu}{r}\frac{1}{2}J_2\left(\frac{R_E}{r}\right)^2\left(1-3\left(\frac{z}{r}\right)^2\right) \tag{10}$$

where $r = \sqrt{x^2+y^2+z^2}$

It can then be derived from Equation 10 and Equation 2 that

$$\vec{a}_{J_2} = -\frac{3}{2}\frac{\mu J_2 R_E^2}{r^4}\left[\frac{x}{r}\left(1-5\left(\frac{z}{r}\right)^2\right), \frac{y}{r}\left(1-5\left(\frac{z}{r}\right)^2\right), \frac{z}{r}\left(3-5\left(\frac{z}{r}\right)^2\right)\right]^\top \tag{11}$$

The acceleration $\vec{a}_{J_2}$ is then added to the right-hand side of Equation 6 as part of the perturbation acceleration.

2.4 Earth Coverage

This section will only reference figures from the text used [3] to compute the Earth target coverage by a satellite. Also, the programs assume that for coverage purposes, the Earth is a perfect sphere. An area access coordinate frame on the Earth's surface is defined corresponding to the nadir coordinate frame of the spacecraft. Figure 9-2 of the text [3] shows the definition of the access area coordinate frame. The transformation between latitude and longitude and access area coordinates is relatively straightforward.

For the programs, the maximum Earth Central Angle is given by,

$$\lambda_0 = \frac{R_E}{R_E+H} \tag{12}$$

However, to account for the optical sensor resolution, the Earth central angle is limited by the resolution requirements (Figure 9-3 of the text [3]). From the Rayleigh criterion and cosine rule,

$$\sin\eta = \frac{x_{res}}{2h'} = 1.220\frac{WL}{D_a} \tag{13}$$

$$\cos\lambda = 1 - \frac{(h')^2 - h^2}{2R_E(R_E + H)} \tag{14}$$

The transformation from the latitude and longitude of the target (P) and sub-satellite point (SSP) (Figure 9-4 of the text [3])are as follows:

$$Lat'_P = 90^\circ - Lat_P\ , Lat'_{SSP} = 90^\circ - Lat_{SSP} \tag{15}$$

$$\Delta L = Long_{SSP} - Long_P \tag{16}$$

$$\lambda_{r_P} = \cos^{-1}\left[\cos Lat'_P \cos Lat'_{SSP} + \sin Lat'_P \sin Lat'_{SSP} \cos\Delta L\right] \tag{17}$$

The target will be observable from the spacecraft optical payload if $\lambda_{r_P} \leq \min(\lambda_0, \lambda)$

2.5 Walker-Delta constellation

The most symmetric of the satellite patterns is the Walker Constellation. The Walker-Delta Pattern contains a total of T satellites with S satellites evenly distributed in each of P orbit planes. All of the orbit planes are assumed to be at the same inclination, i, relative to the equator. The ascending nodes of the P orbit planes in Walker patterns are uniformly distributed around the equator at intervals of $\frac{360^\circ}{P}$. Within each orbital plane, the S satellites are uniformly distributed at intervals of $\frac{360^\circ}{S}$. The phase difference, $\Delta\phi$, in a constellation is defined as the angle in the direction of motion from the ascending node to the nearest satellite at a time when a satellite in the next most westerly plane is at its ascending node. $\Delta\phi$ must be an integral multiple, F, of $\frac{360^\circ}{T}$, where $0 < F \leq P - 1$.

Figure 1 shows an example of a Walker-Delta pattern generated by the program "ground_track2.m" with the satellites at their initial positions. The green cube shows the seed/first satellite.

Figure 2 shows the classical orbit elements for the satellites in the 45:4/2/1 Walker-Delta constellation as generated by the "ground_track2.m" program.

3 General Description of the Codes

Several MATLAB functions (mainly, optim_main.m, ground_track2.m, orbit.m, and test_t.m) were written to analyze the behaviour of the Walker-Delta constellation.

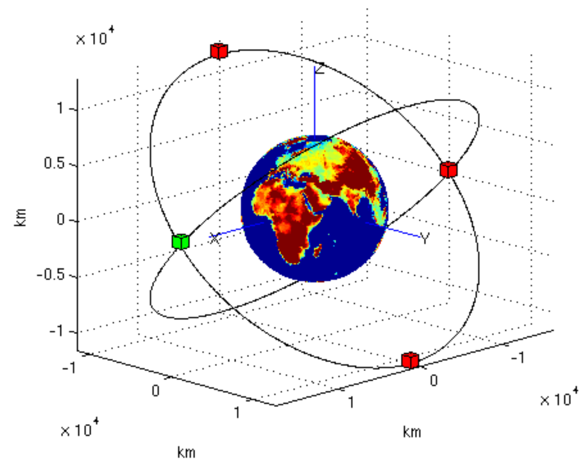

Figure 1: Walker-Delta Pattern 45:4/2/1 at 10,000 km altitude

```
Sat # = 1, h = 10000 km, i = 45 deg, Wo = 0 deg, wpo = 0 deg, TAo = 0 deg, Mo = 0 deg

Sat # = 2, h = 10000 km, i = 45 deg, Wo = 0 deg, wpo = 0 deg, TAo = 180 deg, Mo = 180 deg

Sat # = 3, h = 10000 km, i = 45 deg, Wo = 180 deg, wpo = 0 deg, TAo = 90 deg, Mo = 90 deg

Sat # = 4, h = 10000 km, i = 45 deg, Wo = 180 deg, wpo = 0 deg, TAo = 270 deg, Mo = -90 deg
```

Figure 2: The initial Walker-Delta Pattern in Classical Orbit Elements

The ground_track2.m function was written as a tool to test the orbit propagator (satellite_constellation.m) for a perturbed Keplerian orbit used in optim_main.m as well as to visualize the final design parameter chosen for the observation of KAIST.

In general the basic structure of the codes are similar to each other. The first section will be the inputs of the simulation, second function will initialize the spacecraft state vectors in the ECI (SV) or in the Keplerian orbit elements (COE). After that each satellite in the constellation are propagated along the same rates, hence, for the constellation, only the initial condition is important. After the SVs or COEs are known at each time step from the propagator, the position of the satellites are then converted into the Earth latitude and longitude coordinate frame. In the "find_ra_and_dec" function, this is being done. This makes it easier to obtain the subsatellite point latitudes and longitudes for coverage determination.

The appendix lists the important codes. The optim_main.m code is a wrapper code which can brute force calculate potentially all the design variables of the Walker-Delta constellation. It uses the MATLAB parallel toolbox for independent design variable computations.

4 Validation cases

There currently are no test cases to be compared against. However, the author believes that if the individual functions are working and tested, the results after the functions are combined can be reliable.

4.1 Initialization and propagation

The first test is the satellite initialization subroutine. The output in Figure 2 shows that the initialization was done correctly. The orbit was also propagated correctly with all the outputs showing that the satellites were uniformly propagated. The final output is shown in Figure 3. Table 1 presents the results of a single satellite using different propagator models. One is run using orbit.m and the other using ground_track2.m. As can be seen, the results show not a large difference between the RK5 and Kepler propagator model, with the exception that the Kepler model is faster to run but the disadvantage is that it assumes an elliptical orbit.

```
Sat # = 1, h = 10000 km, i = 45 deg, Wo = -0.0626967 deg, wpo = 0.0664999 deg, TAo = 0.0319325 deg, Mo = 360.032 deg
Sat # = 2, h = 10000 km, i = 45 deg, Wo = -0.0626967 deg, wpo = 0.0664999 deg, TAo = -179.968 deg, Mo = 540.032 deg
Sat # = 3, h = 10000 km, i = 45 deg, Wo = 179.937 deg, wpo = 0.0664999 deg, TAo = 90.0319 deg, Mo = 450.032 deg
Sat # = 4, h = 10000 km, i = 45 deg, Wo = 179.937 deg, wpo = 0.0664999 deg, TAo = -89.9681 deg, Mo = 270.032 deg
```

Figure 3: Final output

Table 1: Results of FOM for single satellite using different propagator models

Method of propagation	Average altitude (km)	% Coverage	Mean response time (s)
RK5 - Newton	392.75	0.255	27987
Kepler	400	0.301	27667

4.2 Earth coverage

Unfortunately, there is no known standard test case to be run for the calculation of the earth coverage.

4.3 Figure-of-Merit

The calculation of the coverage FOM by the subroutine test_t.m is particularly important to establish the performance of the design of the constellation. Hence the percentage coverage and mean response time calculations must be validated. For the test case, the observation patterns for case A is reproduced with varying number of timesteps for the same length. The first is that shown by the text [3] with dt = 1 time unit. The second case gives a dt = 0.1 time unit. The third case gives dt = 0.01 time unit.

The percent coverage and mean response time matches well with the FOM given out by the text (Page 485 in [3]). When the time steps are refined, however, the mean response time differs by quite a significant value. The second case A with a finer time step of 0.1 time unit, gives a mean response time of 0.32 time unit instead of 0.5 as quoted by the text. Figures 4 to 6 show the case A observation histogram with increasingly finer timesteps.

The third case A with the finest time step of 0.01 time unit gave a mean response time of 0.302 time unit with a percent coverage of 60%. Figure 7 shows the results from the test_t.m for the mean response time as a function of dt. It shows that the actual mean response time as dt gets closer to 0 is 0.3 time unit instead of the given 0.5 by the text. However, after reading up on the text and checking the code

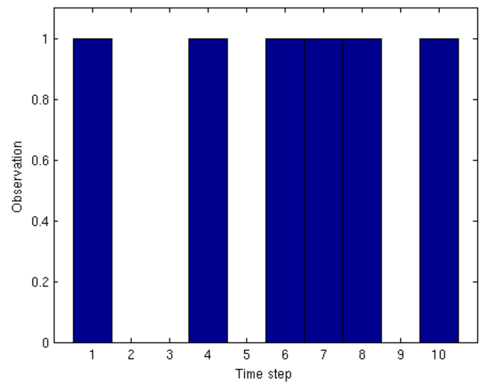

Figure 4: First case A (dt = 1)

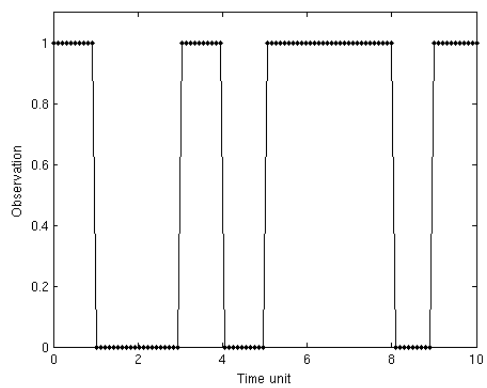

Figure 5: Second case A (dt = 0.1)

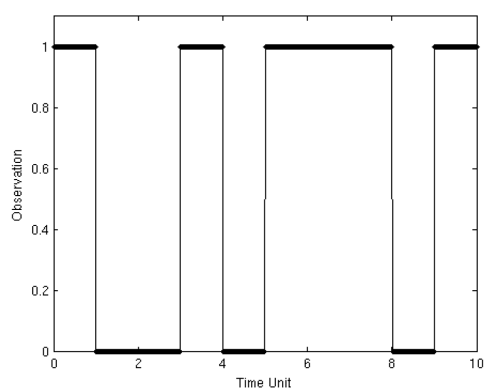

Figure 6: Third case A (dt = 0.01)

several times, the author believes that the way to calculate the mean response time by the subroutine test_t.m was correct and attributes the difference in the results by temporal refinement issues.

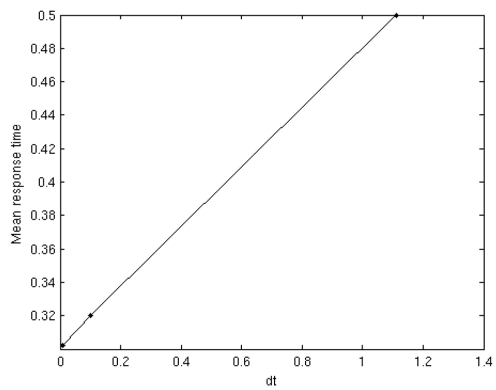

Figure 7: Mean response time as a function of dt

5 Results and Discussion

5.1 Case 1 - Observation of the city of Patna

Patna is the capital of the Indian state Bihar. It is situated at 25.611 N, and 85.144 E. It is a growing state in India therefore, there is a need for remote sensing from space to help city planners in building the city. As such, this case serves as a first test for both the code and also the idea of having remote sensing satellites to aid in city planning. The requirements are laid out such that a ground resolution is no more than 5 m. Also, the number of satellites should be no more than 10.

The design chosen to test is a Walker-Delta constellation of 26:10/5/1. The altitude is given at 1000 km. The percent coverage obtained for a 24 hour simulation was 48.3% and the mean response time was shown to be 33.1 s with a maximum response time of 419.9 s. Figures 8 to 10 shows the constellation configuration along with the orbit, ground track, and the observation histogram for a single revolution.

5.2 Case 2 - Observation of KAIST

KAIST is South Korea's first research oriented science and engineering institution. Naturally, as South Korea's foremost center for strategic research and development,

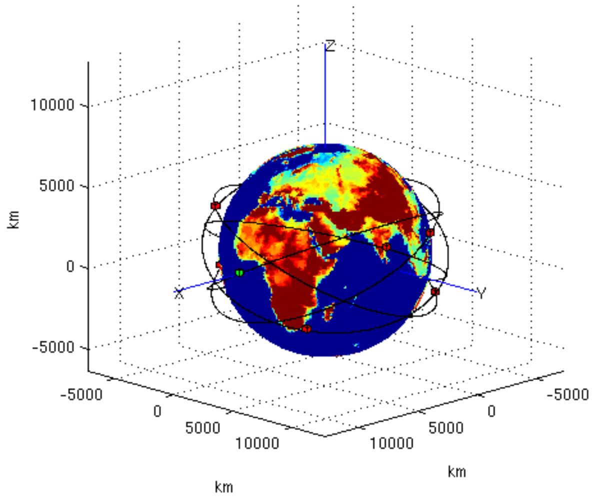

Figure 8: Case 1 - Initial constellation configuration

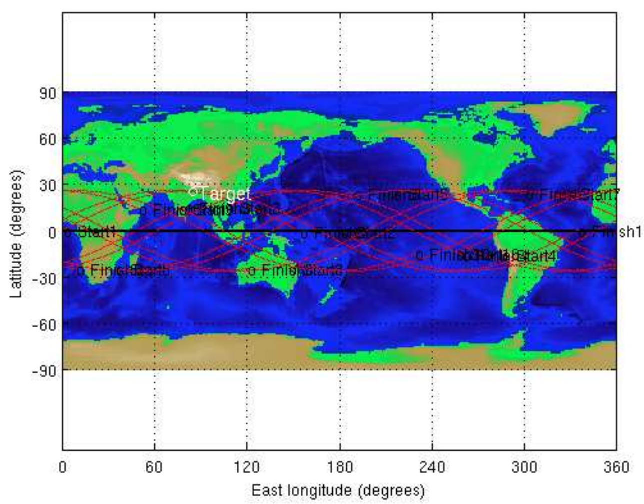

Figure 9: Case 1 - Ground track for 1 revolution

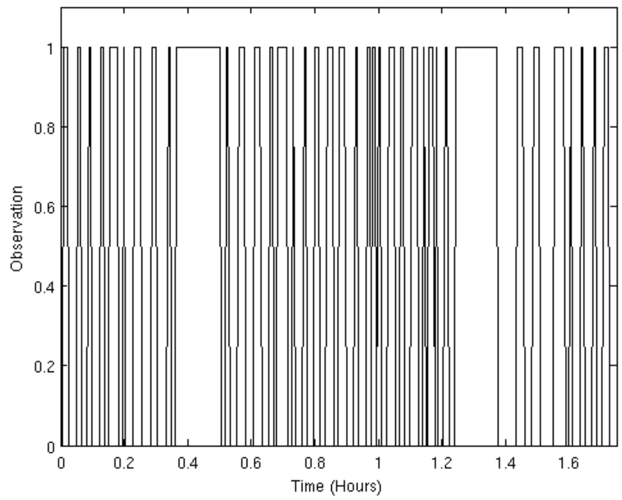

Figure 10: Case 1 - Observation histogram for 1 revolution

the safety of the institute is of paramount importance. In order to help safeguard this institution, it is required that a constellation of remote sensing satellites be placed in orbit with a ground resolution of 1.1 m, and the camera payload aperture diameter at 1 m. The inclination of the orbit has to be 60° and the total number of satellites must not be more than 30 due to cost considerations. The percent coverage is of utmost priority and has to be maximized for a duration of 1 day.

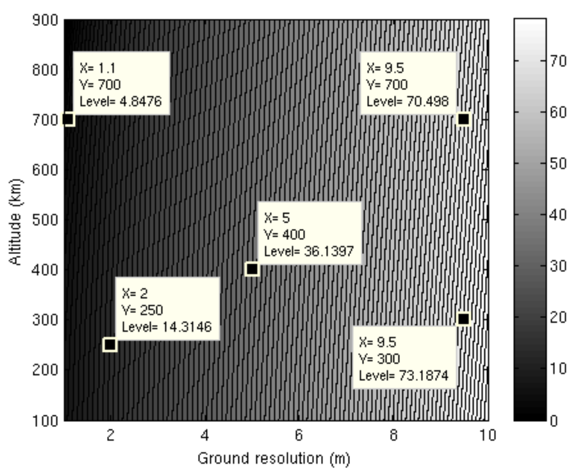

Figure 11: λ as a function of altitude, h, and ground resolution, x_{res}

Figure 11 shows the contour plot of the Earth Central Angle limitation imposed by Rayleigh's Criterion as a function of altitude and ground resolution requirements. For the aperture diameter of 1 m, the limited Earth central angle is at 5° with an altitude of 700 km. This also imposes an altitude limitation in which the satellite altitudes must not go above 1000 km.

For the case of observing KAIST, the optim_main.m code was used to determine the design variables which gives the maximum percent coverage. The brute force searching technique was used. Figure 12 provides an indication of the percent coverage variation with respect to the total number of satellites (T) parameter and the altitude. For this case, the number of planes (P) is set at the maximum which is equal to T. The discontinuous looking pattern is very likely due to the discrete nature of the variable T, and also, the simulation for observing the ground, the time step might be too large in that it might have skipped a few points on the ground, thereby causing the percent coverage plot to be rather irregular. However, a pattern can be seen, which shows that the percent coverage increases with decreasing altitude. This is probably due to the increasing limited Earth Central Angle imposed by the Rayleigh criterion. Of particular interest is shown in Figure 13 that increasing or decreasing the number of planes with a maximum number of satellites (T = 30) does not change the percent coverage by a significant amount. What is clear from the plot is that the largest difference comes again yet from changing altitude. This can only be explained by performance plateaus which is affected by altitude and inclination as described by the text [3].

As can be seen from Figure 14, changing the initial east longitude of the first satellite makes no difference to the percent coverage FOM.

Using the information gained, the constellation 60:30/5/1 is chosen as the final design point to maximize the percent coverage to observe KAIST. The constellation will have an altitude of 400 km and the initial longitude of the first satellite at t=0 is at 0°. The estimated performance of this constellation design is a percentage coverage of 8.54%, a mean response time of 200.44 s, and a maximum response time of 1019.76 s. Figures 15 to 17 shows the configuration, ground track, and observation histogram respectively for the satellite constellation for a time length of 1 orbital period.

The results show that the ground resolution, and inclination requirements have to be relaxed to attain a more feasible solution.

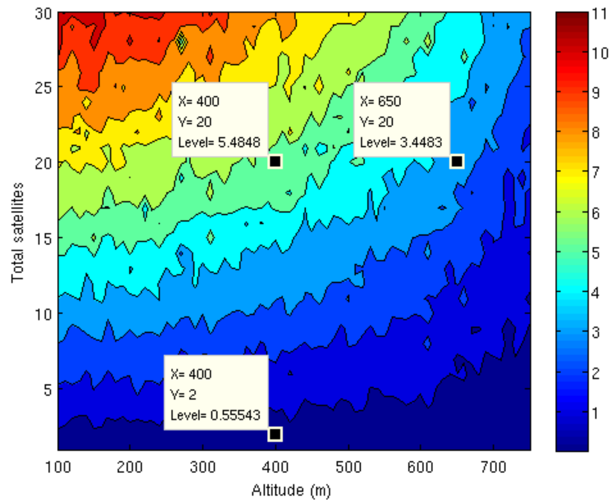

Figure 12: Contour map of percent coverage as a function of altitude and number of satellites

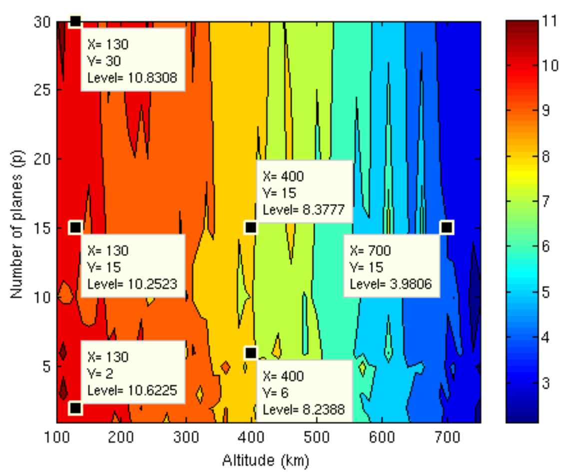

Figure 13: Contour map of percent coverage as a function of altitude and number of planes

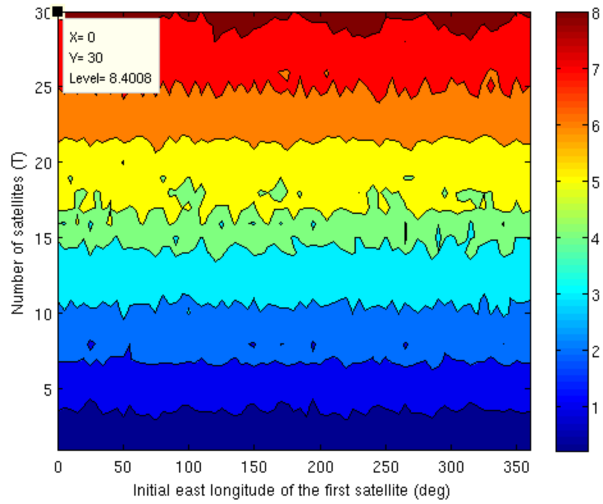

Figure 14: Percent coverage as a function of initial longitude of seed/first satellite and number of satellites

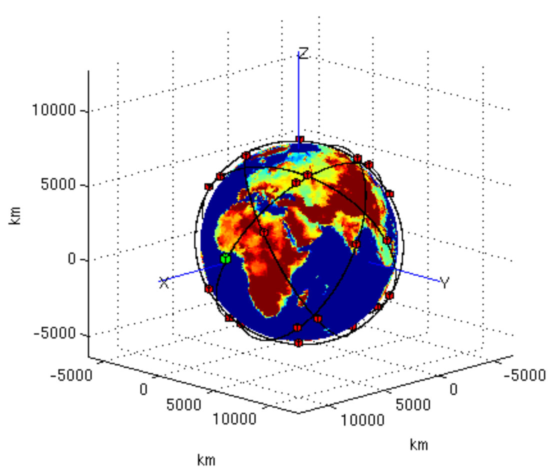

Figure 15: Initial satellite constellation (60:30/5/1 configuration)

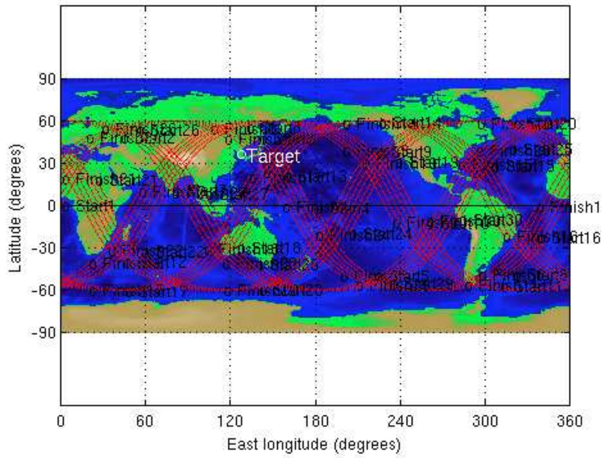

Figure 16: Ground track for the constellation after 1 period

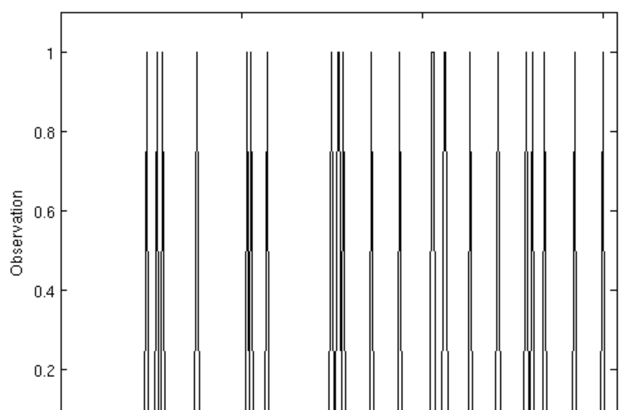

Figure 17: Observation histogram for the constellation observing KAIST in 1 period

6 Summary

A suite of MATLAB functions was developed to analyze the behaviour of Walker-Delta Constellations. Two propagator models were compared, and both models considered the oblateness of the Earth (J_2 perturbation effects). The functions were validated and the functions were used to analyze the performance of such constellations in observing primarily, KAIST, and Patna. The results show that the design requirements are quite tight for the KAIST case while the imaginary Patna case seem to be feasible.

Acknowledgments

The author is grateful to Prof. Ahn Jaemyung for his lectures and out-of-class technical discussion which were necessary to complete this project. The author acknowledges the use of the algorithms as structured by one of the text used [1].

References

[1] Curtis, H. *Orbital Mechanics for Engineering Students, 2nd Ed.* (Elsevier Ltd, Oxford, UK, 2010).

[2] Chobotov, V. *Orbital Mechanics, 2nd Ed.* (AIAA Education Series, Reston, VA, USA 1996).

[3] Wertz, J. *Mission Geometry: Orbit and Constellation Design and Management, 1st Ed.* (Space Technology Library, Microcosm Press, CA, USA, 2001).

A Listing of code - "optim_main.m"

```
clear all; close all; clc;
matlabpool(3); %Comment out if no parallel toolbox installed
deg = pi/180;

mu = 398600;
J2 = 0.00108263;
Re = 6378;
we = (2*pi + 2*pi/365.26)/(24*3600);
%Walker constellation design variables
```

```
inclw = 60*deg; %Inclination of the planes
Nsat = 1:30; %t total number of satellites
Nplane = Nsat; %p total number of planes
f = 1; %f phase multiplier
%...Seed/First satellite parameters
Alt = 400;
hP1 = Alt;
hA1 = Alt;
TAo1 = 0*deg;
Woi = (0:5:360)*deg;
wpo1 = 0*deg;

%Case Simulation length and timesteps
dt = 20;
to = 0;
tf = 1*24*3600; % Ts in assignment paper

%Target latitude and longitude
latT = 36.372*deg; %36.372*deg;
longT = 127.363*deg; %127.363*deg;

%Satellite payload parameters
Da = 1;
WL = 5e-7;
x_res = 1.1;
KA = 20626.4806; % for IAA area in deg^2

for k = 1:length(Nsat)
    parfor i = 1:length(Woi) %Change parfor to for if no parallel
        toolbox is installed
        [t SEE] = satellite_constellation(mu,J2,Re,we,inclw,Nsat(k),
            Nplane(k),f,hP1,hA1,TAo1,Woi(i),wpo1,dt,to,tf,latT,longT,Da,
            WL,x_res,KA);
        [p_cover(k,i) mean_response(k,i) max_response(k,i)] = test_t(SEE
            ,t);
        % fprintf('\n Altitude = %g km | Percentage coverage = %g
            percent | Mean response time = %g seconds | Max response
            time = %g seconds \n', Alt(i), p_cover*100, mean_response,
            max_response)
        drawnow;
    end
end
```

```
matlabpool close; %Comment out if no parallel toolbox installed

contourf(Woi,Nsat,p_cover*100,'-k');
```

B Listing of code - "satellite_constellation.m"

```
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
function [t X] = satellite_constellation(mu,J2,Re,we,inclw,Nsat,Nplane,f
    ,hP1,hA1,TAo1,Woi,wpo1,dt,to,tf,latT,longT,Da,WL,x_res,KA)
% ~~~~~~~~~~~~~

%clear all; close all; clc
global ra dec n_curves RA Dec
%matlabpool(4) %Only used if parallel toolbox is installed
deg = pi/180;
hours = 3600;
%Calculate Pattern Unit for Walker constellation
PU = 360*deg/Nsat;
%Calculate Number of satellites in a plane
NSsat = Nsat/Nplane;
%Angle division between satellites in the plane
beta = PU*Nplane;
%RAAN spacing
dRAAN = PU*NSsat;
%...Seed/First satellite parameters
hP(1:Nsat) = hP1;
hA(1:Nsat) = hA1;
rP1 = hP1 + Re; rP(1:Nsat) = rP1;
rA1 = hA1 + Re; rA(1:Nsat) = rA1;
incl1 = inclw; incl(1:Nsat) =incl1;
TAo(1:Nsat) = TAo1;
Wo1(1:Nsat) = Woi;
wp(1:Nsat) = wpo1;

fprintf('Initial satellite constellation properties \n')
for i=1:Nsat
    pindex(i) = ceil(i/NSsat);
    Spindex(i) = i - (pindex(i) - 1)*NSsat - 1;
    fsatindex(i) = abs(1-sign(Spindex(i)));
    TAo(i) = TAo1 + Spindex(i)*beta + (pindex(i)-1)*f*PU;
```

```
    W(i) = Wo1(i) + (pindex(i) - 1)*dRAAN;
end

%Calculate Earth central angle from payload parameters
h_prime = Da*x_res*1e-3/(2.440*WL); %From Rayleigh optical resolution (km)
%...End data declaration

%...Compute the rates of node regression and perigee advance
a = (rA + rP)/2; %1:Nsat
T = 2*pi/sqrt(mu).*a.^(3/2); %1:Nsat
e = (rA - rP)./(rA + rP); %1:Nsat
h = sqrt(mu.*a.*(1 - e.^2)); %1:Nsat
Eo = 2*atan(tan(TAo/2).*sqrt((1-e)./(1+e))); %1:Nsat
M = Eo - e.*sin(Eo); %1:Nsat
P = a.*(1-e.^2);
fac = 3/2*J2*(Re./P).^2;
Mdot = (2*pi./T).*(1 + fac.*sqrt(1-e.^2).*(1-(3/2).*sin(incl).^2)); %From Chobotov
Wdot = -Mdot.*fac.*cos(incl);
wpdot = fac.*Mdot.*(-5/2*sin(incl).^2 + 2);


% for i=1:Nsat
%           fprintf('\n Sat # = %d, h = %g km, i = %g deg, Wo = %g deg, wpo = %g deg, TAo = %g deg, Mo = %g deg \n', i, hP(i), incl(i)/deg, W(i)/deg, wp(i)/deg, TAo(i)/deg, M(i)/deg);
% end
times=to:dt:tf;
ra(1:length(times),1:Nsat) = 0;
dec(1:length(times),1:Nsat) = 0;
theta = 0;
r_rel_mag(1:length(times),1:Nsat) = 0;
for i = 1:length(times)
   t(i) = times(i);
   for ii = 1:Nsat
  M(ii) = M(ii) + Mdot(ii)*dt;
  E(ii) = kepler_E(e(ii),M(ii));
  TA(ii) = 2*atan(tan(E(ii)/2)*sqrt((1+e(ii))/(1-e(ii))));
   r = h(ii)^2/mu/(1 + e(ii)*cos(TA(ii)))*[cos(TA(ii)) sin(TA(ii)) 0]';

  W(ii) = W(ii) + Wdot(ii)*dt;
```

```
    wp(ii) = wp(ii) + wpdot(ii)*dt;
    Rone = R3(W(ii));
    Rtwo = R1(incl(ii));
    Rthree = R3(wp(ii));
    QxX = (Rthree*Rtwo*Rone)';
    R = QxX*r;

    theta = we*(t(i) - to);
    Q = R3(theta);
    r_rel = Q*R;
    r_rel_mag(i,ii) = norm(r_rel);
    [alpha delta] = ra_and_dec_from_r(r_rel);

    ra(i,ii) = alpha;
    dec(i,ii) = delta;
    end
end

SEE(1:length(times),1:Nsat+1) = 0;
for i = 1:length(times)
    ti = times(i);
    for ii = 1:Nsat
        alt = r_rel_mag(i,ii) - Re;
        clambda = 1-(h_prime^2 - alt^2)/(2*Re*(Re + alt)); %Cos lambda
        IAA = KA.*(1 - clambda);
        latP_prime = pi/2 - latT; %Transformation from lat long to access area coordinates (Section 9.1 - Mission Geometry (Wertz))
        latSSP_prime = pi/2 - dec(i,ii);
        delta_L = ra(i,ii)*deg - longT;
        lambda(i,ii) = acosd(clambda);
        lambda0(i,ii) = acosd(Re/(Re + alt));
        rlambda(i,ii) = acosd(cos(latP_prime)*cos(latSSP_prime) + sin(latP_prime)*sin(latSSP_prime)*cos(delta_L)); %Angular Distance from target to subsatellite point, Ditto Eqn 9-9
        if (rlambda(i,ii) > 180)
            rlambda(i,ii) = 360 - rlambda(i,ii);
        end
        if (rlambda(i,ii) <= min(lambda(i,ii),lambda0(i,ii)))
            SEE(i,ii) = 1; %If target is within the minimum of either lambda or lambda0, then, it is covered by the ii-th satellite
```

```
            end
        end
        SEE(i,Nsat+1) = sign(sum(SEE(i,1:Nsat))); %As long as there is one satellite covered, then the target is covered by the constellation
end
% form_separate_curves
% plot_ground_track
% print_orbital_data
% figure(3)
% plot(t/hours,SEE(:,Nsat+1),'-k');
X = SEE(:,Nsat+1);
%[p_cover mean_response max_response] = test_t(SEE(:,Nsat+1),t);

% figure(4)
% plot(t/hours,alt,'-k');
%return

end %ground_track
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

C Listing of code - "ground_track2.m"

```
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
function ground_track2
% ~~~~~~~~~~~~~

clear all; close all; clc
global ra dec n_curves RA Dec
%matlabpool(4) %Only used if parallel toolbox is installed
deg = pi/180;
hours = 3600;
mu = 398600;
J2 = 0.00108263;
Re = 6378;
we = (2*pi + 2*pi/365.26)/(24*3600);
%Walker constellation design variables
inclw = 60*deg; %Inclination of the planes
Nsat = 30; %t total number of satellites
Nplane = 5; %p total number of planes
f = 1; %f phase multiplier
```

```
%Calculate Pattern Unit for Walker constellation
PU = 360*deg/Nsat;
%Calculate Number of satellites in a plane
NSsat = Nsat/Nplane;
%Angle division between satellites in the plane
beta = PU*Nplane;
%RAAN spacing
dRAAN = PU*NSsat;
%Altitude (in km)
H = 400;
%Longitude of the first satellite in the first plane at t=0
Lon0 = 0;
%...Seed/First satellite parameters
hP1 = H; hP(1:Nsat) = hP1;
hA1 = H; hA(1:Nsat) = hA1;
rP1 = hP1 + Re; rP(1:Nsat) = rP1;
rA1 = hA1 + Re; rA(1:Nsat) = rA1;
incl1 = inclw; incl(1:Nsat) =incl1;
TAo1 = 0*deg; TAo(1:Nsat) = TAo1;
Wo1 = Lon0*deg; Wo1(1:Nsat) = Wo1;
wpo1 = 0*deg; wp(1:Nsat) = wpo1;
initialize_walker
dt = 20;
%Target latitude and longitude (on the ground)
latT = 36.372*deg;%25.611*deg;%
longT = 127.363*deg;%85.144*deg;%
%Satellite payload parameters
Da = 1;
WL = 5e-7;
x_res = 1.1;
KA = 20626.4806; % for IAA area in deg^2
%Calculate Earth central angle from payload parameters
h_prime = Da*x_res*1e-3/(2.440*WL); %From Rayleigh optical resolution (
    km)
%...End data declaration

%...Compute the rates of node regression and perigee advance
a = (rA + rP)/2; %1:Nsat
T = 2*pi/sqrt(mu).*a.^(3/2); %1:Nsat
e = (rA - rP)./(rA + rP); %1:Nsat
h = sqrt(mu.*a.*(1 - e.^2)); %1:Nsat
Eo = 2*atan(tan(TAo/2).*sqrt((1-e)./(1+e))); %1:Nsat
```

```
M = Eo - e.*sin(Eo); %1:Nsat
coe0 = [h' e' W'/deg incl'/deg wp'/deg TAo'/deg];
to = 0;
tf = to + T;%1*24*3600;
times=to:dt:tf;
P = a.*(1-e.^2);
fac = 3/2*J2*(Re./P).^2;
Mdot = (2*pi./T).*(1 + fac.*sqrt(1-e.^2).*(1-(3/2).*sin(incl).^2));
Wdot = -Mdot.*fac.*cos(incl);
wpdot = fac.*Mdot.*(-5/2*sin(incl).^2 + 2);

for i=1:Nsat
    [R0 V0] = sv_from_coe(coe0(i,:),mu);
    Rplot(1,:,i) = [R0 V0];
    fprintf('\n Sat # = %d, h = %g km, i = %g deg, Wo = %g deg, wpo = %g deg, TAo = %g deg, Mo = %g deg \n', i, hP(i), incl(i)/deg, W(i)/deg, wp(i)/deg, TAo(i)/deg, M(i)/deg);
end
find_ra_and_dec
figure(2)
ground_map
for ii = 1:Nsat
form_separate_curves
plot_ground_track
end
% print_orbital_data
figure(1)
output
figure(3)
plot(t/hours,SEE(:,Nsat+1),'-k');
axis([t(1)/hours t(end)/hours 0 1.1]);
xlabel('Time (Hours)'); ylabel('Observation');
fprintf('Final orbital elements');
for i=1:Nsat
    fprintf('\n Sat # = %d, h = %g km, i = %g deg, Wo = %g deg, wpo = %g deg, TAo = %g deg, Mo = %g deg \n', i, hP(i), incl(i)/deg, W(i)/deg, wp(i)/deg, TA(i)/deg, M(i)/deg);
end
[p_cover mean_response max_response] = test_t(SEE(:,Nsat+1),t);
fprintf('\n Percentage coverage = %g percent | Mean response time = %g seconds | Max response time = %g seconds \n', p_cover*100, mean_response, max_response)
```

```
% figure(4)
% plot(t/hours,alt,'-k');
return

function initialize_walker
    fprintf('Initial satellite constellation properties \n')
    for i=1:Nsat
        pindex(i) = ceil(i/NSsat);
        Spindex(i) = i - (pindex(i) - 1)*NSsat - 1;
        fsatindex(i) = abs(1-sign(Spindex(i)));
        TAo(i) = TAo1 + Spindex(i)*beta + (pindex(i)-1)*f*PU;
        W(i) = Wo1(i) + (pindex(i) - 1)*dRAAN;
    end
end

function find_ra_and_dec %Get subsatellite point and also determine if satellite can see the target
ra(1:length(times),1:Nsat) = 0;
dec(1:length(times),1:Nsat) = 0;
theta = 0;
r_rel_mag(1:length(times),1:Nsat) = 0;
for i = 1:length(times)
  t(i) = times(i);
  for ii = 1:Nsat
  M(ii) = M(ii) + Mdot(ii)*dt;
  E(ii) = kepler_E(e(ii),M(ii));
  TA(ii) = 2*atan(tan(E(ii)/2)*sqrt((1+e(ii))/(1-e(ii))));
  r = h(ii)^2/mu/(1 + e(ii)*cos(TA(ii)))*[cos(TA(ii)) sin(TA(ii)) 0]';

  W(ii) = W(ii) + Wdot(ii)*dt;
  wp(ii) = wp(ii) + wpdot(ii)*dt;
  coe0 = [h(ii) e(ii) W(ii) incl(ii) wp(ii) TA(ii)];
  [R0 V0] = sv_from_coe(coe0,mu); %Obtain R, V for plotting
  Rplot(i,:,ii) = [R0 V0];

  Rone = R3(W(ii));
  Rtwo = R1(incl(ii));
  Rthree = R3(wp(ii));
  QxX = (Rthree*Rtwo*Rone)';
  R = QxX*r;

  theta = we*(t(i) - to);
```

```
   Q = R3(theta);
   r_rel = Q*R;
   r_rel_mag(i,ii) = norm(r_rel);
   [alpha delta] = ra_and_dec_from_r(r_rel);

   ra(i,ii) = alpha;
   dec(i,ii) = delta;
   end
end

SEE(1:length(times),1:Nsat+1) = 0;
for i = 1:length(times)
    ti = times(i);
    for ii = 1:Nsat
        alt = r_rel_mag(i,ii) - Re;
        clambda = 1-(h_prime^2 - alt^2)/(2*Re*(Re + alt)); %Cos lambda
        IAA = KA.*(1 - clambda);
        latP_prime = pi/2 - latT; %Transformation from lat long to access area coordinates (Section 9.1 - Mission Geometry (Wertz)
        latSSP_prime = pi/2 - dec(i,ii);
        delta_L = ra(i,ii)*deg - longT;
        lambda(i,ii) = acosd(clambda);
        lambda0(i,ii) = acosd(Re/(Re + alt));
        rlambda(i,ii) = acosd(cos(latP_prime)*cos(latSSP_prime) + sin(latP_prime)*sin(latSSP_prime)*cos(delta_L)); %Angular Distance from target to subsatellite point, Ditto Eqn 9-9
        if (rlambda(i,ii) > 180)
            rlambda(i,ii) = 360 - rlambda(i,ii);
        end
        if (rlambda(i,ii) <= min(lambda(i,ii),lambda0(i,ii)))
            SEE(i,ii) = 1; %If target is within the minimum of either lambda or lambda0, then, it is covered by the ii-th satellite
        end
    end
    SEE(i,Nsat+1) = sign(sum(SEE(i,1:Nsat))); %As long as there is one satellite covered, then the target is covered by the constellation
end

end %find_ra_and_dec
```

```

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
function form_separate_curves
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Breaks the ground track up into separate curves which start
% and terminate at right ascensions in the range [0,360 deg].
% ---------------------------------------------------------------
tol = 100;
curve_no = 1;
n_curves = 1;
k = 0;
ra_prev = ra(1,ii);
for i = 1:length(ra)
if abs(ra(i) - ra_prev) > tol
curve_no = curve_no + 1;
n_curves = n_curves + 1;
k = 0;
end
k = k + 1;
RA{curve_no}(k) = ra(i,ii);
Dec{curve_no}(k) = dec(i,ii);
ra_prev = ra(i,ii);
end
end %form_separate_curves

% ~~~~~~~~~~~~~~~~~~~~~~~~~
function plot_ground_track
% ~~~~~~~~~~~~~~~~~~~~~~~~~
for i = 1:n_curves
plot(RA{i}, Dec{i},'.r','MarkerSize',4)
end
plot(longT/deg,latT/deg,'ow');
% axis ([0 360 -90 90])
text('Position', [ra(1,ii) dec(1,ii)], 'String', ['o Start' num2str(ii)])
text('Position', [ra(end,ii) dec(end,ii)], 'String', ['o Finish' num2str(ii)])
text('Position',[longT/deg,latT/deg], 'String', ' Target', 'FontSize', 12, 'Color', 'white')
line([min(ra(:,ii)) max(ra(:,ii))],[0 0], 'Color','k') %the equator
end %plot_ground_track

```

```
function ground_map
load('topo.mat','topo','topomap1');
contour(0:359,-89:90,topo,[0 0],'b')
axis equal
box on
set(gca,'XLim',[0 360],'YLim',[-90 90], ...
'XTick',[0 60 120 180 240 300 360], ...
'Ytick',[-90 -60 -30 0 30 60 90]);
hold on
image([0 360],[-90 90],topo,'CDataMapping', 'scaled');
colormap(topomap1);
xlabel('East longitude (degrees)')
ylabel('Latitude (degrees)')
axis equal
grid on
end %ground_map

function output
  %...Plot the results:
  % Draw the planet
  load('topo.mat','topo','topomap1');
  [xx, yy, zz] = sphere(100);
  cla reset
  props.AmbientStrength = 0.1;
  props.DiffuseStrength = 1;
  props.SpecularColorReflectance = .5;
  props.SpecularExponent = 20;
  props.SpecularStrength = 1;
  props.FaceColor= 'texture';
  props.EdgeColor = 'none';
  props.FaceLighting = 'phong';
  props.Cdata = topo;
  sf = surface(Re*xx,Re*yy,Re*zz,props);%'facecolor','texturemap','cdata',topo);
  for rt = 1:100 %Re-orient topo map to longitude east 0 at GMT
      rotate(sf,[0,0,1],45);
  end
%    sf;
  %colormap(light_gray)
  caxis([-Re/10 Re/10])
  %shading interp
  % Draw and label the X, Y and Z axes
```

```
   line([0 2*Re], [0 0], [0 0]); text(2*Re, 0, 0, 'X')
   line( [0 0], [0 2*Re], [0 0]); text( 0, 2*Re, 0, 'Y')
   line( [0 0], [0 0], [0 2*Re]); text( 0, 0, 2*Re, 'Z')
   % Plot the orbit, draw a radial to the starting point
   % and label the starting point (o) and the final point (f)
   hold on
   for ii =1:Nsat
   plot3( Rplot(:,1,ii), Rplot(:,2,ii), Rplot(:,3,ii),'k')
%    line([0 r0(1)], [0 r0(2)], [0 r0(3)])
   if(ii==1)
        draw_sat(Rplot(1,1:3,ii),[350 350 350],'g',1);
   else
        draw_sat(Rplot(1,1:3,ii),[350 350 350],'r',1);
   end
   end
   % Select a view direction (a vector directed outward from the origin)
   view([1,1,.4])
   % Specify some properties of the graph
   grid on
   axis equal
   xlabel('km')
   ylabel('km')
   zlabel('km')
end %output

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~
function print_orbital_data
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~
coe = [h e Wo incl wpo TAo];
[ro, vo] = sv_from_coe(coe, mu);
fprintf('\n ------------------------------------------------------------\n')
fprintf('\n Angular momentum = %g km^2/s' , h)
fprintf('\n Eccentricity = %g', e)
fprintf('\n Semimajor axis = %g km', a)
fprintf('\n Perigee radius = %g km', rP)
fprintf('\n Apogee radius = %g km', rA)
fprintf('\n Period = %g hours' , T/3600)
fprintf('\n Inclination = %g deg', incl/deg)
fprintf('\n Initial true anomaly = %g deg', TAo/deg)
fprintf('\n Initial RA = %g deg', Wo/deg)
fprintf('\n RA_dot = %g deg/day' , Wdot/deg*(tf-to))
fprintf('\n Initial wp = %g deg', wpo/deg)
```

```
fprintf('\n wp_dot = %g deg/period' , wpdot/deg*T)
fprintf('\n')
fprintf('\n r0 = [%12g, %12g, %12g] (km)', ro(1), ro(2), ro(3))
fprintf('\n magnitude = %g km\n', norm(ro))
fprintf('\n v0 = [%12g, %12g, %12g] (km)', vo(1), vo(2), vo(3))
fprintf('\n magnitude = %g km/s\n', norm(vo))
fprintf('\n -----------------------------------------------------\n')

end %print_orbital_data
end %ground_track
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

D Listing of code - "orbit.m"

```
function orbit %similar to groundtrack.m except that state vector is
    propagated instead using Newton's law
  clc; close all; clear all
  global ra dec n_curves RA Dec
  hours = 3600;
  deg = pi/180;
  %...Input data:
  % Earth:
  R = 6378;
  mu = 398600;
  we = (2*pi + 2*pi/365.26)/(24*3600);
  %Physical model input
  J2_inc = 1; %Include J2 perturbation effects? 1 - yes, 0 for no
  low_thrust = 0; %Include low thrust tangential to orbit path? 1 - yes,
      0 - no
  at_thrust = 5e-6;
  Atmos_drag = 0; %Include atmospheric drag? 1 - yes, 0 - no
  ball_coeff = 150; % Ballistic coefficient Used for atmospheric drag
      calculations
  %Simulation Parameters
  t0 = 0;
  tf = 1*24*hours;
  dt = 20; %in seconds for ode4 routine
  %Initial Orbit parameters
  hP = 400;
  hA = 400;
  TAo = 0*deg;
```

```
  Wo = 0*deg;
  incl = 51.43*deg;
  wpo = 0*deg;
  %Target latitude and longitude
  latT = 25.611*deg; %36.372*deg;
  longT = 85.144*deg;%127.363*deg;
  %Satellite payload parameters
  Da = 1;
  WL = 1.11e-2;
  x_res = 1.1;
  KA = 20626.4806; % for IAA area in deg^2
  %...End input data
  %Calculate Earth central angle from payload parameters
  h_prime = Da*x_res*1e-3/(2.440*WL); %From Rayleigh optical resolution
      (km)

  %Numerical conditions
  rkr45_int = 0;
  % Obtain R0 and V0 vectors from coe data
  rP = hP + R; rA = hA + R;
  a = (rA + rP)/2;
  T = 2*pi/sqrt(mu)*a^(3/2);
%   tf = 0.5*T;
  e = (rA - rP)/(rA + rP);
  h = sqrt(mu*a*(1 - e^2));

  coe = [h e Wo incl wpo TAo];
  coe0 = coe;
  [r0 v0] = sv_from_coe(coe,mu);
  %...Numerical integration:
%   mu = G*(m1 + m2);
  theta = 0; %Earth revolution
  y0 = [r0 v0 theta]';
  if(rkr45_int == 1)
    [t,y] = rkf45(@rates, [t0 tf], y0);
  else
    t = t0:dt:tf;
    [y] = ode5(@rates, t, y0);
  end
   coe1 = coe_from_sv(y(end,1:3),y(end,4:6),mu);
  %...Output the results:
  figure(1)
```

```
    output
    find_ra_and_dec
    figure(2)
    form_separate_curves
    plot_ground_track
    print_orbital_data
    figure(3)
    plot(t/hours,SEE,'-k');
    figure(4)
    plot(t/hours,alt,'-k');

[p_cover mean_response max_response] = test_t(SEE,t);
mean_alt = mean(alt);
fprintf('\n Altitude = %g km | Percentage coverage = %g percent | Mean response time = %g seconds | Max response time = %g seconds \n', mean_alt, p_cover*100, mean_response, max_response)
%    axis([0 tf/hours 0 1.1]);
return

function dydt = rates(t,f)
    x = f(1);
    y = f(2);
    z = f(3);
    vx = f(4);
    vy = f(5);
    vz = f(6);
%    we = 0; %(2*pi + 2*pi/365.26)/(24*3600);
    r = norm([x y z]);
    if((J2_inc == 1) && (incl ~= 0))
        %To calculate J2 orbit perturbation from Chobotov Question 10.5 J2
        %perturbation in Cartesian coordinates
        J2 = 0.00108263;
        fac = -(mu/r^2)*(3/2)*J2*(R/r)^2;
        px = fac*(x/r)*(1-5*(z/r)^2);
        py = fac*(y/r)*(1-5*(z/r)^2);
        pz = fac*(z/r)*(3-5*(z/r)^2);
    else
        P(1:3) = 0;
        px = 0;
        py = 0;
        pz = 0;
    end
```

```

  if(low_thrust == 1)
      T_vec = [vx vy vx]/norm([vx vy vz]);
      H_vec = cross([x y z],[vx vy vz]);
      W_vec = H_vec/norm(H_vec);
      N_vec = cross(T_vec,W_vec);
      Qmat = [N_vec' T_vec' W_vec'];
      at = at_thrust;
      AT = at*[0;1;0];
      AT = Qmat*AT;
  else
      AT(1:3) = 0;
  end

  ax = -mu*x/r^3 + px + AT(1);
  ay = -mu*y/r^3 + py + AT(2);
  az = -mu*z/r^3 + pz + AT(3);
  dydt = [vx vy vz ax ay az we]';
end %rates

function find_ra_and_dec %Get subsatellite point and also determine if
    satellite can see the target
times=t;
ra = [];
dec = [];
theta = 0;
for i = 1:length(times)
  ti = times(i);
  r = [y(i,1) y(i,2) y(i,3)]';
  v = [y(i,4) y(i,5) y(i,6)]';
  coe1 = coe_from_sv(r,v,mu);
  Rr = r;
  Q = R3(y(i,7));
  r_rel = Q*Rr;
  [alpha delta] = ra_and_dec_from_r(r_rel);

  ra = [ra; alpha];
  dec = [dec; delta];
end
SEE(1:length(times)) = 0;
for ii = 1:length(times)
    ti = times(ii);
```

```
    r = [y(ii,1) y(ii,2) y(ii,3)]';
    alt(ii) = norm(r) - R;
    clambda(ii) = 1-(h_prime^2 - alt(ii)^2)/(2*R*(R + alt(ii)));
    IAA = KA.*(1 - clambda);
    latP_prime = pi/2 - latT;
    latSSP_prime = pi/2 - dec(ii);
    delta_L = ra(ii)*deg - longT;
    lambda(ii) = acosd(clambda(ii));
    lambda0(ii) = acosd(R/(R + hP));
    rlambda(ii) = acosd(cos(latP_prime)*cos(latSSP_prime) + sin(
        latP_prime)*sin(latSSP_prime)*cos(delta_L));
    if (rlambda(ii) > 180)
        rlambda(ii) = 360 - rlambda(ii);
    end
    if (rlambda(ii) <= min(lambda(ii),lambda0(ii)))
        SEE(ii) = 1;
    end
end

end %find_ra_and_dec

function output
  for i = 1:length(t)
    r(i) = norm([y(i,1) y(i,2) y(i,3)]);
  end
  [rmax imax] = max(r);
  [rmin imin] = min(r);
  v_at_rmax = norm([y(imax,4) y(imax,5) y(imax,6)]);
  v_at_rmin = norm([y(imin,4) y(imin,5) y(imin,6)]);
  fprintf('\n\n--------------------------------------------------------------\
    n')
  fprintf('\n Earth Orbit\n')
  fprintf(' %s\n', datestr(now))
  fprintf('\n The initial position is [%g, %g, %g] (km).',...
                                                   r0(1), r0(2),
                                                     r0(3))
  fprintf('\n Magnitude = %g km\n', norm(r0))
  fprintf('\n The initial velocity is [%g, %g, %g] (km/s).',...
                                                   v0(1), v0(2),
                                                     v0(3))
  fprintf('\n Magnitude = %g km/s\n', norm(v0))
  fprintf('\n Initial time = %g h.\n Final time = %g h.\n',0,tf/hours)
```

```
fprintf('\n The minimum altitude is %g km at time = %g h.',...
    rmin-R, t(imin)/hours)
fprintf('\n The speed at that point is %g km/s.\n', v_at_rmin)
fprintf('\n The maximum altitude is %g km at time = %g h.',...
    rmax-R, t(imax)/hours)
fprintf('\n The speed at that point is %g km/s\n', v_at_rmax)
fprintf('\n--------------------------------------------------------------\n\n')
%...Plot the results:
% Draw the planet
load('topo.mat','topo','topomap1');
[xx, yy, zz] = sphere(100);
cla reset
props.AmbientStrength = 0.1;
props.DiffuseStrength = 1;
props.SpecularColorReflectance = .5;
props.SpecularExponent = 20;
props.SpecularStrength = 1;
props.FaceColor= 'texture';
props.EdgeColor = 'none';
props.FaceLighting = 'phong';
props.Cdata = topo;
sf = surface(R*xx,R*yy,R*zz,props);%'facecolor','texturemap','cdata',topo);
for rt = 1:100 %Re-orient topo map to longitude east 0 at GMT
    rotate(sf,[0,0,1],45);
end
caxis([-R/5 R/5])
%shading interp
% Draw and label the X, Y and Z axes
line([0 2*R], [0 0], [0 0]); text(2*R, 0, 0, 'X')
line( [0 0], [0 2*R], [0 0]); text( 0, 2*R, 0, 'Y')
line( [0 0], [0 0], [0 2*R]); text( 0, 0, 2*R, 'Z')
% Plot the orbit, draw a radial to the starting point
% and label the starting point (o) and the final point (f)
hold on
plot3( y(:,1), y(:,2), y(:,3),'k')
draw_sat([y(1,1) y(1,2) y(1,3)],[1000 1000 1000],'g',1);
line([0 r0(1)], [0 r0(2)], [0 r0(3)])
text( y(1,1), y(1,2), y(1,3), 'o')
```

```
    text( y(end,1), y(end,2), y(end,3), 'f')
    % Select a view direction (a vector directed outward from the origin)
    view([0,0,1])
    % Specify some properties of the graph
    grid on
    axis equal
    xlabel('km')
    ylabel('km')
    zlabel('km')
% ~~~~~~~~~~~~~~~~~~~~~~~
end %output
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~
function form_separate_curves
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Breaks the ground track up into separate curves which start
% and terminate at right ascensions in the range [0,360 deg].
% ---------------------------
tol = 100;
curve_no = 1;
n_curves = 1;
k = 0;
ra_prev = ra(1);
for i = 1:length(ra)
if abs(ra(i) - ra_prev) > tol
curve_no = curve_no + 1;
n_curves = n_curves + 1;
k = 0;
end
k = k + 1;
RA{curve_no}(k) = ra(i);
Dec{curve_no}(k) = dec(i);
ra_prev = ra(i);
end
end %form_separate_curves

% ~~~~~~~~~~~~~~~~~~~~~~~~
function plot_ground_track
% ~~~~~~~~~~~~~~~~~~~~~~~~
load('topo.mat','topo','topomap1');
contour(0:359,-89:90,topo,[0 0],'b')
```

```
axis equal
box on
set(gca,'XLim',[0 360],'YLim',[-90 90], ...
    'XTick',[0 60 120 180 240 300 360], ...
    'Ytick',[-90 -60 -30 0 30 60 90]);
hold on
image([0 360],[-90 90],topo,'CDataMapping', 'scaled');
colormap(topomap1);
xlabel('East longitude (degrees)')
ylabel('Latitude (degrees)')
axis equal
grid on
for i = 1:n_curves
plot(RA{i}, Dec{i},'-r')
end
plot(longT/deg,latT/deg,'or');
% axis ([0 360 -90 90])
text( ra(1), dec(1), 'o Start')
text(ra(end), dec(end), 'o Finish')
text(longT/deg,latT/deg, ' Target')
line([min(ra) max(ra)],[0 0], 'Color','k') %the equator
end %plot_ground_track

% ~~~~~~~~~~~~~~~~~~~~~~~~~~
function print_orbital_data
% ~~~~~~~~~~~~~~~~~~~~~~~~~~
coe = [h e Wo incl wpo TAo];
[ro, vo] = sv_from_coe(coe, mu);
fprintf('\n -----------------------------------------------------\n')
fprintf('\n Angular momentum = %g km^2/s' , h)
fprintf('\n Eccentricity = %g', e)
fprintf('\n Semimajor axis = %g km', a)
fprintf('\n Perigee radius = %g km', rP)
fprintf('\n Apogee radius = %g km', rA)
fprintf('\n Period = %g hours' , T/3600)
fprintf('\n Inclination = %g deg', incl/deg)
fprintf('\n Initial true anomaly = %g deg', TAo/deg)
% fprintf('\n Time since perigee = %g hours' , to/3600)
fprintf('\n Initial RA = %g deg', Wo/deg)
% fprintf('\n RA_dot = %g deg/period' , Wdot/deg*T)
fprintf('\n Initial wp = %g deg', wpo/deg)
% fprintf('\n wp_dot = %g deg/period' , wpdot/deg*T)
```

```
fprintf('\n')
fprintf('\n r0 = [%12g, %12g, %12g] (km)', ro(1), ro(2), ro(3))
fprintf('\n magnitude = %g km\n', norm(ro))
fprintf('\n v0 = [%12g, %12g, %12g] (km)', vo(1), vo(2), vo(3))
fprintf('\n magnitude = %g km\n', norm(vo))
fprintf('\n -----------------------------------------------------------\n')

end %print_orbital_data
end %orbit
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

E Listing of code - ”test_t.m”

```
function [p_cover mean_response max_response min_response] = test_t(x,t)
dt = (t(end) - t(1))/(length(t) );
p_cover = sum(x)/length(x);
t_response(1:length(t)) = 0;
for i=1:length(t)
    if(x(i) > 0)
        t_response(i) = 0;
    elseif (i~=length(t))
        ii = i;
        while ((x(ii) <= x(ii+1))&&(x(ii) < 1)) %End of Gap reached when x(ii) == 1
            t_response(i) = t_response(i) + dt;
            ii = ii + 1;
            if(ii == length(t))
                break
            end
        end
    else
        t_response(i) = t_response(i) + dt; %Already at the end, and it is a gap, so, add dt
    end
end

nt_response = nonzeros(t_response);
total_response = sum(t_response);
max_response = max(nt_response);
min_response = min(nt_response);
mean_response = total_response/length(t);
```