Departament de Projectes d'Enginyeria
UNIVERSITAT POLITÈCNICA DE CATALUNYA

**Secció de Terrassa**

**Degree:** Aerospace Engineering
**Course:** Engineering Projects

**Title and acronym of the project:**
Cubesat Constellation
Astrea

**Contents:** ANNEX VI: Matlab Codes

**Group:** G4 EA-T2016

**Delivery date:** 22-12-2016

**Students:**

Cebrián Galán, Joan
Foreman Campins, Lluís
Fuentes Muñoz, Óscar
Herrán Albelda, Fernando
Martínez Viol, Víctor
Pla Olea, Laura
Puig Ruiz, Josep
Tarroc Gil, Sergi
Urbano González, Eva María
Fontanes Molina, Pol
Fraixedas Lucea, Roger
González García, Sílvia
Kaloyanov Naydenov, Boyan
Morata Carranza, David
Pons Daza, Marina
Serra Moncunill, Josep Maria
Tió Malo, Xavier

**Customer:** Pérez Llera, Luís Manuel

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Matlab Codes

## 1.1 Satellite Footprint

```
%----ASTREA CONSTELLATION----
%PROJECTS - 220028
%Aerospace Engineering Barchelor's Degree
%ESEIAAT - UPC
%Autumn 2016-2017

% ORBIT DESIGN TEAM
% COMPUTATION OF A SATELLITE FOOTPRINT

clc
clear all
close all

%% PHISICAL CONSTANTS AND PARAMETERS

Re = 6378;           %Earth's Radius [Km]
Se = 4*pi*Re^2;      %Earth's Surface [Km^2]
h = 500:100:1000;    %Satellte height [Km]
eo = 0:5:40;         %Elevation angle [deg]

%% SOLVER

for i = 1:length(h)
    for j = 1:length(eo)

        d(i,j)= Re*(sqrt(((h(i)+Re)/Re)^2-(cosd(eo(j)))^2 )-sind(eo(j)));
        Bo(i,j) = asind( d(i,j)*cosd(eo(j)) / (h(i)+Re) );

        S(i,j) = 2 * pi * Re^2 * ( 1-cosd(Bo(i,j)) );
```

```
        R(i,j) = sqrt( d(i,j)^2 - h(i)^2 );

        %Satellite coverage expressed as a fraction of Earth's Area (%)
        Cov(i,j) = ( S(i,j) / Se )*100;

        %Number of satellites needed
        nsat(i,j) = ( Se / S(i,j) );

    end
end

%% PLOTS

figure
for k = 1:length(h)
    plot ( eo , Cov(k,:) )
    hold on
end

title('Coverage vs Elevation')
xlabel('Elevation [deg]') % x-axis label
ylabel('Coverage [%]') % y-axis label
legend(num2str(h(1)),num2str(h(2)),num2str(h(3)),...
    num2str(h(4)),num2str(h(5)),num2str(h(6)))

figure
for k = 1:length(h)
    plot ( eo , nsat(k,:) )
    legend ('h(j)')
    hold on
end

title('Num.Satellites vs Elevation')
xlabel('Elevation [deg]') % x-axis label
ylabel('Num.Satellites') % y-axis label
legend(num2str(h(1)),num2str(h(2)),num2str(h(3)),...
    num2str(h(4)),num2str(h(5)),num2str(h(6)))

figure
for k = 1:length(h)
    plot ( eo , 2*Bo(k,:) )
    hold on
end

title('Angle vs Elevation')
xlabel('Elevation [deg]') % x-axis label
ylabel('Angle between satellites[deg]') % y-axis label
legend(num2str(h(1)),num2str(h(2)),num2str(h(3)),...
    num2str(h(4)),num2str(h(5)),num2str(h(6)))
```

## 1.2 Minimum Plane Inclination

```
%----ASTREA CONSTELLATION----
%PROJECTS - 220028
%Aerospace Engineering Barchelor's Degree
%ESEIAAT - UPC
%Autumn 2016-2017

% ORBIT DESIGN TEAM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MINIMUM INCLINATION - To provide Full coverage %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%PROBLEM:
% Given: - Minimum Elevation over the horizon to contact the satellite
%        - Height of the orbit
% Compute the minimum elevation to ensure visual contact
clear; clc;

%% Key variable
min_elevation_pole=5; %Initially set by @lfore as 32;

%% Previous Calculation: Visibility(latitude)

R = 6371;
N = 180;
deg = 180;
x = linspace(0,90,N);

%%%% Minimum angle of elevation due to atmospheric conditions at given
%%%% latitudes. 0 km represents the poles whereas 6371 km represents the
%%%% Earth's equator taking as reference point the south pole.

elvlat = zeros (1,deg);
for i = 1:deg/5
    elvlat(i) = min_elevation_pole-(min_elevation_pole-15)/(2*(90-75))*i;
end
for i = deg/5+1:4*deg/10
    elvlat(i) = elvlat(deg/5);
end
for i = 4*deg/10+1:6*deg/10
    elvlat(i) = elvlat(2*deg/5)+3/15*(i-4*deg/10);
end
for i = 6*deg/10+1:8*deg/10
    elvlat(i) = elvlat(6*deg/10)-3/15*(i-6*deg/10);
end
for i = 8*deg/10+1:10*deg/10
    elvlat(i) = elvlat(8*deg/10)-1/15*(i-8*deg/10);
```

```
end
elvlat=fliplr(elvlat); %Sets the first value to the value in the equator
                        %The first value matches 0 latitude

%% Minimum i computation

h = 400:100:1000;   %Satellte height [Km]
L = 0:5:90;         %Latitude angle [deg]
e = (interp1q(x',elvlat',L'))'; %Minimum elevations interpolation [deg]

figure(1)
subplot(1,2,1)
plot(x,elvlat)
subplot(1,2,2)
title('Interpolation Verification')
plot(L,e)

inc=zeros(length(h),length(L)); %Inclinations preallocation

for i = 1:length(h)
    for j = 1:length(L)

        A=cosd(e(j))/(1+h(i)/R);
        theta=acosd(A)-e(j); %Earth Central Angle
        inc(i,j)=L(j)-theta;
        if inc(i,j)<0
            inc(i,j)=0;
        end
    end
end

%% 3. Results Plotting

figure(2)
colors=['r','b','y','g','m','k','c'];
for k = 1:length(h)
    hold on
    plot ( L , inc(k,:),colors(k))
end

grid on
title('Minimum inclination required to fulfill global coverage')
xlabel('Latitude [deg]')
ylabel('Inclination [deg]')
legend(num2str(h(1)),num2str(h(2)),num2str(h(3)),...
    num2str(h(4)),num2str(h(5)),num2str(h(6)),num2str(h(7)))

%% Bonus!!!

L=90;
```

```
e=5.33;
h=500:1:600;
inc=zeros(1,length(h));
for i = 1:length(h)
        A=cosd(e)/(1+h(i)/R);
        theta=acosd(A)-e; %Earth Central Angle
        inc(i)=L-theta;
        if inc(i)<0
            inc(i)=0;
        end
end

figure(3)
plot(h,inc)
xlabel('Heights')
ylabel('Inclination')
grid on
axis([min(h) max(h) 45 90])
```

## 1.3 Satellite Number Computation for Polar Orbits

```
%----ASTREA CONSTELLATION----
%PROJECTS - 220028
%Aerospace Engineering Barchelor's Degree
%ESEIAAT - UPC
%Autumn 2016-2017

% ORBIT DESIGN TEAM
% POLAR ORBITS Number of satellites computation

%PROBLEM:
% Given: - Elevation over the horizon to contact the satellite
%        - Height of the orbit
% Compute the final number of satellites
% Through the "Streets of coverage" method.

clear; clc;

%% 1. Input data
% PHISICAL CONSTANTS AND PARAMETERS

Re = 6378;          %Earth's Radius [Km]
Se = 4*pi*Re^2;     %Earth's Surface [Km^2]
h = 500:20:600;   %Satellte height [Km]
eo = 20;        %Elevation angle [deg]

```

```
%% 2. Number of satellites computation

d=zeros(length(h),length(eo));
nsats_opt=d;

for i = 1:length(h)
    for j = 1:length(eo)

        d(i,j)= Re*(sqrt(((h(i)+Re)/Re)^2-(cosd(eo(j)))^2)-sind(eo(j)));
        Bo = asind( d(i,j)*cosd(eo(j)) / (h(i)+Re) );
        Sup = 2 * pi * Re^2 * ( 1-cosd(Bo) );

        Nspp_min=ceil(360/(2*Bo));        %Number of satellites per plain
        Nsatspp=Nspp_min:1:Nspp_min+50;   %Array of sats/plain to optimize

        S=360./Nsatspp;

        Lstreet=acosd(cosd(Bo)./cosd(S/2));

        DmaxS=Lstreet+Bo;
        DmaxO=2*Lstreet;

        Nplains=ceil(1+((180-DmaxO)./DmaxS));
        Nsats=Nplains.*Nsatspp;
        [row,col]=find(min(Nplains));
        Nplains_opt(i,j)=Nplains(row,col);
        nsats_opt(i,j)=Nsats(row,col);

        %Number of satellites needed
        nsat(i,j) = ( Se / Sup );

    end
end

%% 3. Results Plotting

figure(2)
colors=['r','b','y','g','m','k','c'];
for k = 1:length(eo)
    subplot(1,2,1)
    hold on
    semilogy ( h , nsats_opt(:,k),colors(k))
    subplot(1,2,2)
    hold on
    semilogy ( h , Nplains_opt(:,k),colors(k))
end

subplot(1,2,1)
title('Num.Sats vs Height')
xlabel('Height [km]') % x-axis label
```

```
ylabel('Num.Satellites') % y-axis label
%legend(num2str(eo(1)),num2str(eo(2)),num2str(eo(3)),...
%    num2str(eo(4)),num2str(eo(5)),num2str(eo(6)),num2str(eo(7)))
axis([400 900 0 1000])
grid on

subplot(1,2,2)
title('Num.Planes vs Height')
xlabel('Height [km]') % x-axis label
ylabel('Num of Orbital Planes') % y-axis label
% legend(num2str(eo(1)),num2str(eo(2)),num2str(eo(3)),...
%     num2str(eo(4)),num2str(eo(5)),num2str(eo(6)),num2str(eo(7)))
axis([400 900 0 30])
grid on

%% Single detailed analysis

figure(1)
plot(h,nsats_opt)
title('Num.Sats vs Height - Streets of coverage Method')
xlabel('Height [km]') % x-axis label
ylabel('Num.Satellites') % y-axis label
grid on

%% Multianalysis variation with height

figure(3)
colors=['r','b','y','g','m','k'];
for k = 1:length(h)
    subplot(1,2,1)
    hold on
    plot ( eo , nsats_opt(k,:),colors(k))
    subplot(1,2,2)
    hold on
    plot ( eo , Nplains_opt(k,:),colors(k))
end

subplot(1,2,1)
title('Num.Sats vs Elevation - Streets of coverage Method')
xlabel('Elevation [deg]') % x-axis label
ylabel('Num.Satellites') % y-axis label
legend(num2str(h(1)),num2str(h(2)),num2str(h(3)),...
    num2str(h(4)),num2str(h(5)),num2str(h(6)))
axis([0 40 0 700])
grid on

subplot(1,2,2)
title('Num.Planes vs Elevation - Surfaces Method')
xlabel('Elevation [deg]') % x-axis label
ylabel('Num.Orbital Planes') % y-axis label
```

```
legend(num2str(h(1)),num2str(h(2)),num2str(h(3)),...
    num2str(h(4)),num2str(h(5)),num2str(h(6)))
axis([0 40 0 30])
grid on
```

## 1.4 Orbit Parameters

```
%----ASTREA CONSTELLATION----
%PROJECTS - 220028
%Aerospace Engineering Barchelor's Degree
%ESEIAAT - UPC
%Autumn 2016-2017

% ORBIT DESIGN TEAM
% ORBIT PARAMETERS

% This routine calculates the minimum number of plans and satellites
% to guarantee global coverage.
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% _v2 --> Computes angles with arccosine
% _v3 --> Also iterates in f

clc
clear all
close all

tic

%% DATA INPUT

Re = 6371;                  %Earth's Radius [Km]
Se = 4*pi*Re^2;             %Earth's Surface [Km^2]
h = 500:5:600;              %Height vector [km]
in = 75:3:90;               %Inclination vector [deg]
pmin = 5; p=12;             %Minimum number of plans
pmax = 20;                  %Maximum number of plans
sppmin = 10; spp=10;        %Minimum number of satellites each plane
sppmax = 24;                %Maximum number of satellites each plane
typeWD = 2;                 %Defines:
                            % 1- Semi-Walker-Delta
                            % 2- 3/4 of Walker-Delta
                            % 3- Full Walker-Delta
plaunch = 5.76e6;           %Launch price
psat = 250e3;               %Price per satellite
eo = 20;                    %Vison angle
q=0;                        %Variable used as counter
```

```
nplanes_win=0;
%Boolean, Indicates a succesful combination with that number of planes

%% COMPUTATIONS
%result=zeros(6,100);

for j=1:length(h)
    fprintf('\n\nH = %g km',h(j))
    Bo = visibleangle(h(j),eo);  %Footprint angle

    for k=1:length(in)
        fprintf('\ni=%g:  ',in(k))

        for p=pmin:pmax
            F=0:1:p-1; nF=length(F); anglesp_f=zeros(1,nF);

            for spp=sppmin:sppmax
                angles = 360/spp; %Angle between satellites of same plane

              for kf=1:nF
               % We look for the optimum f for this spp,npp,inc,h
               anglesp_f(kf)=distance_opt_f(in(k),h(j),p,spp,typeWD,F(kf));
               %Angle between satellites of different planes
              end

                [anglesp,kf_min]=min(anglesp_f);

                 if (angles ≤ 2*Bo) && (anglesp ≤ Bo)

                    q=q+1;
                    result(:,q)=[h(j) in(k) spp p spp*p F(kf_min)];
                    fprintf(['Solution %g: h=%gkm - i=%g - '...
                       '%gspp - %gplanes - %g sats - f=%g'],q,result(:,q))
                    nplanes_win=1;
                    break %Stop iteration in satellites per plain

                 end

            end
            if nplanes_win     %If a combination with that num of planes
                nplanes_win=0; %works, then jump to the next inclination
                break
            end
        end
    end
end

%% Optimum Results
% Minimum number of cubesats
numsat = result (5,1:q);
```

```
[minsat,index] = min(numsat);

disp(sprintf('\nMINIMUM SATELLITES\nheight: %d', result(1,index)))
disp(sprintf('inclination: %d',result(2,index)))
disp(sprintf('number of satellites in each plane: %d',result(3,index)))
disp(sprintf('number of planes: %d',result(4,index)))
disp(sprintf('minimum number of satellites: %d',result(5,index)))

% Minimum number of planes
numplane = result(4,1:q);
[minplane,index]= min(numplane);

disp(sprintf('\nMINIMUM PLANES\nheight: %d', result(1,index)))
disp(sprintf('inclination: %d',result(2,index)))
disp(sprintf('number of satellites in each plane: %d',result(3,index)))
disp(sprintf('number of planes: %d',result(4,index)))
disp(sprintf('minimum number of satellites: %d',result(5,index)))

% Price Optimization
TCost = plaunch*result(4,:) + psat*result(5,:); TCost = TCost/(1e6);
[CostOpt,index] = min(TCost);
height = result(1,:);

disp(sprintf('\nMINIMUM PRICE\nprice[M]: %d', TCost(index)))
disp(sprintf('height: %d', result(1,index)))
disp(sprintf('inclination: %d',result(2,index)))
disp(sprintf('number of satellites in each plane: %d',result(3,index)))
disp(sprintf('number of planes: %d',result(4,index)))
disp(sprintf('minimum number of satellites: %d',result(5,index)))

fprintf('\n\n MORE THAN ONE SOLUTION COULD HAVE THE SAME MIN NUMBER\n')

%% Plots

x=0:q-1;
plot(x,numplane,x,numsat,x,TCost,x,height)
strValues = strtrim(cellstr(num2str([x(:) numplane(:)],'(%d, %d)')));
text(x,numplane,strValues,'VerticalAlignment','bottom');
strValues = strtrim(cellstr(num2str([x(:) numsat(:)],'(%d, %d)')));
text(x,numsat,strValues,'VerticalAlignment','bottom');
strValues = strtrim(cellstr(num2str([x(:) TCost(:)],'(%d, %.1f)')));
text(x,TCost,strValues,'VerticalAlignment','bottom');
strValues = strtrim(cellstr(num2str([x(:) height(:)],'(%d, %d)')));
text(x,height,strValues,'VerticalAlignment','bottom');
legend('number of planes','number of satellites','Total Cost (M)')



toc
```

```
function [Bo] = visibleangle (h,eo)

d=zeros(length(h),length(eo));
nsats_opt=d;
Re = 6371;

for i = 1:length(h)
    for j = 1:length(eo)

        d(i,j)= Re * ( sqrt( ( (h(i)+Re)/Re )^2 - ( cosd(eo(j)) )^2 ) - sind(eo(j)) );
        Bo = asind( d(i,j)*cosd(eo(j)) / (h(i)+Re) );

    end
end

end
```

```
function [anglesp] = distance_opt_f (i,a,p,s,typeWD,F)
%% Data input

% i: Inclination angle [deg]
% a: Height of the orbit [km]
% p: Number of planes
% s: Number of satelites per plane

%% Walker Delta type of geometry
WD=[180 225 360];
degreegenerate = WD(typeWD);
% Walker Delta 360 deg, SemiWalker Delta 180 deg,
% other constellations range between 180 and 360

m = degreegenerate/180;
% generates Walker Delta Constellation
% (m=2 -> 2*180 generated constellation),
% Semi Walker (m=1 -> 1*180) or other 1<m<2.

% Phasing between adjacent planes
%f = F*p*cosd(i); % parameter defined graphically
 f = F; % Whereas F is a number 0 < f < (p-1)
        % According to Astrodynamics notes

REarth = 6371; % [km]
h = (REarth+a)/REarth; %radius of the orbit
%(in terms of the sphere radius) R=h*r;

%% Distribution of Coordinates of the satellites
Omega = zeros(1,p);
nu = zeros(s,p);
```

```
angle = zeros(1,p);
X = zeros(3,s,p);

for j = 1:p
    Omega(j) = m*pi*(j-1)/p;
    angle(j) = f*2*pi*(j-1)/(s*p); %Phasing due to f between planes
    for k = 1:s
        % True anomalies of the s satellites
        nu(k,j) = 2*pi*(k-1)/s+angle(j);
        X(:,k,j) = cartesian(h,0,i,Omega(j),0,nu(k,j));
        % h[adim] --> X[adim]
    end
end

%% Fast comprovation
% Does the adjacent phasing exceed pi
% THIS MEANS
% The minimum might be between


%% Angle between different satellites

c=zeros(s,p); %General aproach. Where is the minimum angle?)

for j=2:p
  for i=1:s-1

    % We need to assess the angles between the last and first plane
    % Specially in Semi-Walker configurations. That's why we add this
    % auxiliar variables.

      p1=j;
      if j==1
          p2=p;
      else
          p2=j-1;
      end

    % Angles a2 and a3 could be used to increase de reliability of the
    % constellation, allowing greater possibilities of communication
    % between planes. (not only between (i,p1) and (i,p2)

    % Angle between satellite (i,p1) and satellite (i,p2)
    a1=acosd((X(:,i,p1)'*X(:,i,p2))/(norm(X(:,i,p1))*norm(X(:,i,p2))));

    % Angle between satellite (i,j) and satellite (i+1,j-1)
    a2=acosd((X(:,i,p1)'*X(:,i+1,p2))/(norm(X(:,i,p1))*norm(X(:,i+1,p2))));

    % Angle between satellite (i,j) and satellite (i-1,j-1)
    a3=acosd((X(:,i,p1)'*X(:,i-1,p2))/(norm(X(:,i,p1))*norm(X(:,i-1,p2))));
```

```
    % Among the computed angles, choose the minimum
    [angle(i,j),c(i,j)]=min([a1 a2]);

    % Why minimum? Then we know that at least it will be able to
    % cover the angle with one of the two adjacent close nodes.

  end
end

anglesp = max(max(angle));
%Maximum angle between satellites of different planes
```

## 1.5 Walker Delta Testing

```
%----ASTREA CONSTELLATION----
%PROJECTS - 220028
%Aerospace Engineering Barchelor's Degree
%ESEIAAT - UPC
%Autumn 2016-2017

% ORBIT DESIGN TEAM
% ORBIT SMART WALKER-DELTA

% This routine compute the WD possible configurations that give global
% coverage by having the distance between planes matches the planes
% rotation

p=[12 15 18];
S=360./p; S=S*pi/180;

re=6378.01e3;
u=3.986012e14;
J2=1.0826e-3;

we=2*pi/(24*3600);

syms a;
inc=75; inc=inc*pi/180;
a_sol=zeros(1,length(inc));

for i=1:length(inc)
    dOmega=-1.5*J2*(re/a)^2*sqrt(u/a^3)*cos(inc(i));
    P0=2*pi*sqrt(a^3/u);
    Pn=P0*(1-1.5*J2*(re/a)^2-0.75*J2*(4-5*sin(inc(i))^2)*(re/a)^2);
    eq=S(2)-Pn*(we-dOmega);
```

```
    sol=vpasolve(eq,re*1.15);
    if imag(sol)==0
        a_sol(i)=sol;
    end
end

plot(inc*180/pi,(a_sol-re)/1000)
h=(a_sol-re)/1000

%% Satellites computation

emin=20*pi/180;
Lstreet=5*pi/180;

rho=re/(re+h*1000);
etha=asin(sin(rho)*cos(emin));
Lmax=pi/2-emin-etha;
S=2*acos(cos(Lmax)/cos(Lstreet));
N=ceil(2*pi/S)
```

## 1.6 Orbit Plotter

```
%----ASTREA CONSTELLATION----
%PROJECTS - 220028
%Aerospace Engineering Barchelor's Degree
%ESEIAAT - UPC
%Autumn 2016-2017

% ORBIT DESIGN TEAM
% ORBIT PLOTTER - Influential phenomena computation

% This matlab routine plots the position of the satellites of a Walker
% Delta / SemiWalker Delta or other generated configurations.

clc, clear, close all;
%-------------------------------------------------------------------------
%% 0. Data Input
%add the subfolder programs to the current path
addpath(genpath('./auxiliar'));

%_____Physical input__________

global h, h = 542;  % [km]
degreegen = 225;    % Walker Delta 360 deg, SemiWalker Delta 180 deg,
                    % other constellations range between 180 and 360 deg
in = degtorad(72);  % inclination of the planes
```

```
s = 21;               % satellites per plane
p = 9;                % number of planes
f=8;                  % f = 1.25*p*cos(i); % parameter defined graphically
eo=degtorad(20);      % elevation angle [rad]

%____Numerical Input_________
r = 1;     %radius of the sphere
n = 50;    %number of cells of the sphere generated
N = 50;    %number of points used to print the orbit
nn = 100; %number of points used to compute the footprint

%____Physical Constants_____
global Re, Re = 6371;  %Earth's Radius [Km]
%% 1. Preprocessing

global m, m = degreegen/180;
            % generates Walker Delta Constellation (m=2 -> 2*180)
            % generated constellation), Semi Walker (m=1 -> 1*180)
            % or other 1<m<2.

global a, a = (Re+h)/Re;
            %radius of the orbit (in terms of the sphere radius) R=h*r;

%% 2. Main Process

%compute the sat RAAN and some sat. parameters (Laura)
Omega = zeros(1,p);
nu = zeros(s,p);      %true anomaly of the satellites
angle = zeros(1,p);   %angle between satellites
for i = 1:p
    Omega(i) = m*pi*(i-1)/p;    %orbits

    angle(i) = f*2*pi*(i-1)/(s*p); %sats
    for k = 1:s
        nu(k,i) = 2*pi*(k-1)/s+angle(i);
    end
end

% Compute the satellite coordinates (Laura)
X = zeros(3,s,p);
for i = 1:p
    for k = 1:s
        X(:,k,i) = cartesian(a,0,in,Omega(i),0,nu(k,i));
    end
end
sCOOR=reshape(X,3,p*s)'; %put the data in a more friendly structure

%compute the radius of the footprint
%SATELLITE_FOOTPRINT_COMPUTATION.m
d=Re*((((h+Re)/Re)^2-cos(eo)^2)^.5-sin(eo));
```

```
Bo=asin(d*cos(eo)/(h+Re));
S=2*pi*Re^2*(1-cos(Bo));
R=(S/pi)^.5;

%create the footprint of each satellite
SATfp=[];
for i=1:s*p
    %unitary vector pointing to the sat from the earth's center
    u=-sCOOR(i,:)/norm(sCOOR(i,:));
    %Generate the circle coordinates (2D)
    [satfp(:,1),satfp(:,2),satfp(:,3)]=circle3D(sCOOR(i,:),u, R/Re, nn);
    satfp=satfp+repmat(u*h/Re,nn,1); %move the circle to be tg to Earth

    for j=1:size(satfp,1)
        v=sCOOR(i,:)-satfp(j,:); v=v/norm(v);
        %unitary vector sat->footprint

        %verify if the footprint intersect with the earth's surface
        tol=satfp(j,1)^2+satfp(j,2)^2+satfp(j,3)^2-r^2;
        while tol>1e-2
            satfp(j,:)=satfp(j,:)-0.01*v;
            tol=satfp(j,1)^2+satfp(j,2)^2+satfp(j,3)^2-r^2;
        end
    end
    %store the footprint
    SATfp=[SATfp; satfp];
end


%% 3. Plots

%3D plot
[COOR]=Orbitplot3D( in,r, X,n, SATfp, nn );
%Ground track
x=zeros(size(COOR,1),size(COOR,2)/3); y=x; z=x;
for i=1:size(COOR,2)/3
    x(:,i)=COOR(:,3*(i-1)+1);
    y(:,i)=COOR(:,3*(i-1)+2);
    z(:,i)=COOR(:,3*(i-1)+3);
end
GroundTrackPlot( x, y, z, sCOOR, SATfp, nn);
```

```
function [x,y,z]=circle3D(center,normal,radius,n)
%This matlab function generates the points of a circle normal to a 3D
%vector
%In:
%   center: center of the circle (x, y, z)
%   normal: vector normal to the circle
%   radius: radius of the circle
```

```
%   n: number of points (small number can make the circle not smooth)
%Out:
%   x, y z: vectors cintaining the coord.
%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬

theta=linspace(0,2*pi,n);
v=null(normal);
points=repmat(center',1,size(theta,2))+...
    radius*(v(:,1)*cos(theta)+v(:,2)*sin(theta));
x=points(1,:); y=points(2,:); z=points(3,:);
end
```

```
function [COOR] = Orbitplot3D( in,r, X,n, sCOOR, N )
%Orbitplot3D is a function that creates a 3D representation of a
%constellation and plots the satellites. It can be also configured to plot
%their footprint
%In:
%   in: inclinaton angle [rad]
%   r: sphere radius [-]
%   X: Coordinates of the sats (3,spp,p)
%   n: number of points used to draw thhe sphere
%   ¬¬o¬¬
%   COOR: coordinates of the footprints
%   N: number of points used to compute the footprint
%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
p=size(X,3);
s=size(X,2);
global a m;

% Earth
%Create the earth instance as an sphere
[x,y,z]=sphere(n); %Generate the sphere
surf(r*x,r*y,r*z,'FaceColor','interp','EdgeColor','none'); %plot the sphere
colormap(linspace(0,1,n)'*[0 0.6 1]); %custom colormap (blue tones);
axis equal;
box off;
hold on;
rotate3d on;

% Orbits
%Generate the circle coordinates (2D)
coor=zeros(N+1,3);
for i=1:N
    coor(i,:)=[a*r*cos(2*i*pi/N), a*r*sin(2*i*pi/N), 0];
end
coor(N+1,:)=coor(1,:);

%Rotation Matrices
Ry=[ cos(in) 0 sin(in)
```

```
          0     1    0
     -sin(in)  0  cos(in)];

Rz= @(Omega)  [-sin(Omega)  cos(Omega)  0
                cos(Omega)  sin(Omega)  0
                    0           0      1]; % he modificat aquesta matriu

coor=coor*Ry; %Rotate the circle in the y axis (inclination)

%rotate the plane to get the other planes
COOR=zeros(N+1,3*p);
for i=1:p
    COOR(:,(1+(i-1)*3):(3+(i-1)*3))=coor*Rz(m*(i-1)*pi/p);
end
for i=1:p
    plot3(COOR(:,1+(i-1)*3),COOR(:,2+(i-1)*3),COOR(:,3+(i-1)*3),...
        'linewidth',1.5);
end

% Satellites (Laura)
for i = 1:p
    for k = 1:s
      scatter3(X(1,k,i),X(2,k,i),X(3,k,i),25,'square','k','filled');
      hold on;
    end
end

%Plot also the footprint if the number of input variables is bigger than 4
if nargin > 4
    for i=1:p*s
        b=1+N*(i-1); c=N+N*(i-1);
        fill3(sCOOR(b:c,1),sCOOR(b:c,2),sCOOR(b:c,3),[1 0 0],...
            'FaceAlpha','.3','LineStyle','none');
    end
end
end
```

```
function [] = GroundTrackPlot( x, y, z, coor, fp_coor, N)

%GroundTrackPlot is used to plot the ground
%track of the constellation orbits
%given their coordinates (cartesian).It has two variants, only introducing
%the first 3 inputs will plot the orbits and if you introduce all the
%inputs it plots the satellites and their footprint.
%In:
%   x: matrix containing the x coordinate of the orbits to plot
%   y: matrix containing the y coordinate of the orbits to plot
%   z: matrix containing the z coordinate of the orbits to plot
%   ¬¬o¬¬
```

```
%   coor: satellite coordinates (nx3)
%   fp_coor: satellite footprint coordinates ((n*nn)x3)
%   N: number of points used to compute the footprint
%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
p=size(x,2); %n of planes
s=size(coor,1)/size(x,2); %n of spp

%% Set the background image for the ground track plot
figure('name','Constellation Ground Track');
tit=['Num. of planes: ' num2str(p) ' Num. of spp: ' num2str(s)];
ground='World-satellite-map.png';
im=imread(ground);
imagesc([-180,180],[-90,90],im);
grid on;
axis([-180 180 -90 90]);
title(tit);
hold on;
%set the plot lines color
c = ['r','y','m','g','w']; C=[];
for i=1:ceil(size(x,2)/5)
    C=[C c];
end

%% Main process
for i=1:size(x,2)
    %convert the cartesian coordinates to spherical
    [lon,lat]=cart2sph(x(:,i),y(:,i),z(:,i));
    %get the coordinates of the orbit in spherical coord.
    lon=radtodeg(lon); %pass to degrees
    lat=radtodeg(lat);
    Lon=[lon(end); lon(1:end-1)];

    %Plot the sat ground track
    [¬,j]=max(abs(lon-Lon));
    plot(lon(1:j-1),lat(1:j-1),C(i),'linewidth',1.2);
    plot(lon(j:end),lat(j:end),C(i),'linewidth',1.2);
end

if nargin > 3
    [slon,slat]=cart2sph(coor(:,1),coor(:,2),coor(:,3));
    %get the coordinates of the sats in spherical coord.
    slon=radtodeg(slon); %pass to degrees
    slat=radtodeg(slat);

    %plot the sats
    scatter(slon,slat,30,'square','w','filled');

    %get the coordinates of the sats footprint in spherical coord.
    [fplon,fplat]=cart2sph(fp_coor(:,1),fp_coor(:,2),fp_coor(:,3));
    fplon=radtodeg(fplon); %pass to degrees
```

```
    fplat=radtodeg(fplat);

    %Plot their footprint
    for i=1:p*s
        b=1+N*(i-1); c=N+N*(i-1);
        fpLon=[fplon(c); fplon(b:c-1)];

        %get the maximum values (plotting purposes)
        [sorted, I]=sort(abs(fplon(b:c)-fpLon));
        M=sorted(end-1:end);
        j=I(end-1:end); j(2)=j(2)-1;

        if M(1) > 300
            if j(1) < j(2)
                auxlon=fplon(b+j(1):b+j(2)-1); %negative lon. part
                auxlat=fplat(b+j(1):b+j(2)-1);

                auxlon2=[fplon(b+j(2):c); fplon(b:b+j(1)-2)]; %+ lon. part
                auxlat2=[fplat(b+j(2):c); fplat(b:b+j(1)-2)];
            else
                auxlon2=fplon(b+j(2):b+j(1)-2); %negative lon. part
                auxlat2=fplat(b+j(2):b+j(1)-2);

                auxlon=[fplon(b+j(1):c); fplon(b:b+j(2)-1)];
                %positive lon. part
                auxlat=[fplat(b+j(1):c); fplat(b:b+j(2)-1)];
            end

         fill(auxlon,auxlat,[1 0 0],'FaceAlpha','.2','LineStyle','none');
         fill(auxlon2,auxlat2,[1 0 0],'FaceAlpha','.2','LineStyle','none');
        else
            fill(fplon(b:c),fplat(b:c),[1 0 0],...
                'FaceAlpha','.3','LineStyle','none');

        %%%%%%%%%%%%
        pause(.05); %
        %%%%%%%%%%%%
        end


    end
end
end
```

```
% This function converts kepler orbit elements to cartesian coordinates

% a: semimajor axis
% e: eccentricity
% i: inclination [rad]
```

```
% Omega: longitude of the ascending node [rad]
% w: Argument of periapsis [rad]
% nu: True anomaly [rad]

function [X] = cartesian(a,e,i,Omega,w,nu)

% Position in cylindrical coordinates
r = a*(1-e^2)/(1+e*cos(nu));

% Position components
X = [r*(cos(Omega)*cos(w+nu)-sin(Omega)*sin(w+nu)*cos(i));
    r*(sin(Omega)*cos(w+nu)+cos(Omega)*sin(w+nu)*cos(i));
    r*sin(i)*sin(w+nu)];

end
```

## 1.7 Perturbations

```
%----ASTREA CONSTELLATION----
%PROJECTS - 220028
%Aerospace Engineering Barchelor's Degree
%ESEIAAT - UPC
%Autumn 2016-2017

% ORBIT DESIGN TEAM
% ORBIT PERTURBATIONS - Influential phenomena computation

%PROBLEM:
% Given: - Initial orbital parameters
% Compute the final orbtial parameters after each orbit
%clear; clc;

%% 1.A Input data physical constants
% PHISICAL CONSTANTS AND PARAMETERS
%Physical data
RE=6.378e6; %Earth Radius [m]
u=3.986e14; %GM Earth
J2=1082.6e-5; %J2 coefficient;

%% 1. Input data Orbit parameters
% SPACECRAFT DATA
m=4;          % Satellite mass [kg]
Cd=2.2;       % Drag Coefficient
A=0.1;   % Satellite surface [m^2] --> 3U pointing to Earth
Bc=m/(Cd*A);% Ballistic coefficient of the satellite;
```

```
% ORBIT DATA
H0=542e3;      % Altura inicial [m]
e0=0.0001;      % Excentricity
i0=72*pi/180;        % Inclination [deg]
a0=RE+H0;

% INITIAL PARAMETERS
P=2*pi*(a0^3/u)^.5; % Orbit Period [s]
v=sqrt(u/a0);        %satellite velocity [m/s]
n=(2*pi/P)*86400;  %number of revolutions/day


% SIMULATION PARAMETERS
N=10000000;

%% 2. Perturbations propagation
%Outputs de la funcio Perturbation:
%   Pert: matrix (4x5) with the perturbations causes at the rows (J2,Drag,
%   Moon,Sun) and the orbital elements in the columns (a,e,i,omega,Omega)
%     a  e  i  w  Omega
%   [                   ]J2
%   [                   ]Drag
%   [                   ]Moon
%   [                   ]Sun as 3rd Body



t=zeros(1,N); t(1)=0;
a=t; a(1)=a0;
e=t; e(1)=e0;
i=t; i(1)=i0;

w_sun=zeros(1,N-1); w_moon=w_sun; w_J2=w_sun;
Om_sun=w_sun; Om_moon=w_sun; Om_J2=w_sun;

w0=0; Omega0=0;
w=t; w(1)=w0;
Omega=t; Omega(1)=Omega0;

tic
for n=2:N

    a0=a(n-1);
    e0=e(n-1);
    i0=i(n-1);
    w0=w(n-1);
    Omega0=Omega(n-1);

    P=2*pi*(a0^3/u)^.5; % Orbit Period [s]

```

```
    [Perti rho]=Perturbation(a0,e0,i0,Bc);

    a(n)=a0+sum(Perti(:,1));
    e(n)=e0+sum(Perti(:,2));
    i(n)=i0+sum(Perti(:,3));
    w(n)=w0+sum(Perti(:,4));
    Omega(n)=Omega0+sum(Perti(:,5));

    w_sun(n-1)=Perti(4,4);
    w_moon(n-1)=Perti(3,4);
    w_J2(n-1)=Perti(1,4);

    Om_sun(n-1)=Perti(4,5);
    Om_moon(n-1)=Perti(3,5);
    Om_J2(n-1)=Perti(1,5);

    % We don't want angles bigger than 360!
    w(n)=w(n)-360*floor(w(n)/360);
    Omega(n)=Omega(n)-360*floor(Omega(n)/360);

    if Omega(n)>2*pi
        Omega(n)=Omega(n)-2*pi;
    end

    t(n)=t(n-1)+P;

    if a(n)<(RE+180e3)
     fprintf('Your satellite succesfully burned in the atmosphere.\n\n')
      break
    end

end

%% Post processing and results plotting

time=toc;
fprintf('Time to do %g iterations = %g s.\n ',n,time)

figure(1)
plot(t(1:n)/(3600*24),(a(1:n)-RE)/1000)
ylim([100 500])
grid on
ylabel('Orbit height [km]')
xlabel('Time [days]')
tfin=floor(t(n)/(3600*24));
days = num2str(floor(tfin/365));
titulaso=['Orbit decay in ' num2str(tfin) ' days = ' days 'years'];
title(titulaso)
```

```
figure(2)
subplot(1,2,1)
plot(t(1:n)/(3600*24),w(1:n))
grid on
ylabel('Perigee Argument [deg]')
xlabel('Time [days]')
title('Perigee Argument deviation in 100 days')
axis([0 100 0 360])

subplot(1,2,2)
plot(t(1:n)/(3600*24),Omega(1:n))
grid on
ylabel('Ascenent Node Argument [deg]')
xlabel('Time [days]')
tfin=t(n)/(3600*24);
title('Ascendent Node deviation in 100 days')
axis([0 100 0 360])

%% Modulus Analysis

figure(3)

subplot(1,2,1)
semilogy(t(2:n+1)/(3600*24),abs(w_sun(1:n)),'b','LineWidth',2);
hold on
semilogy(t(2:n+1)/(3600*24),abs(w_moon(1:n)),'r','LineWidth',2)
semilogy(t(2:n+1)/(3600*24),abs(w_J2(1:n)),'k','LineWidth',2)
legend('Sun','Moon','J2')
xlabel('Time (days)');
ylabel('Modulus of Perturbation []');
title('Effects on Argument of the Perigee');
grid on
axis([0 t(n)/(3600*24) 1e-8 1])

subplot(1,2,2)
semilogy(t(2:n+1)/(3600*24),abs(Om_sun(1:n)),'b','LineWidth',2);
hold on
semilogy(t(2:n+1)/(3600*24),abs(Om_moon(1:n)),'r','LineWidth',2)
semilogy(t(2:n+1)/(3600*24),abs(Om_J2(1:n)),'k','LineWidth',2)
legend('Sun','Moon','J2')
xlabel('Time (days)');
ylabel('Modulus of Perturbation []');
title('Effects on Argument of Ascendent Node');
axis([0 t(n)/(3600*24) 1e-8 1])
grid on

print -depsc ModulusAngulars
```

```
function [pert rho] = Perturbation( a,e,i,Bc)
```

```
%% Perturbations
% This matlab is used to compte the perturbaons in the classical orbital
% elements due to:
%  J2 (non spherical earth)
%  Atmospheric drag
%  Third body (Moon and Sun)
%Inputs:
%   a: semi-major axis [m]
%   e: eccentricity [-]
%   i: inclination [rad]
%   m: mass of the satellite [kg]
%   Cd: drag coefficient [-]
%   A: Area of the satellite  [m^2]
%Outputs:
%   Pert: matrix (4x5) with the perturbations causes at the rows (J2,Drag,
%   Moon,Sun) and the orbital elements in the columns (a,e,i,omega,Omega)
%   incP/revolution
%   incv/revolution
%TO DO:
%Les perturbacions degudes a J2 i 3rd body no se si son en pert/dia o no,
%cal mirar-ho per tal de tenir totes les pertrubacions escalades igual
% ---------------------------------------------------------------------------
%% Previous calculations and PreAllocation
%Physical data
RE=6.378e6; %Earth Radius [m]
u=3.986e14; %GM Earth
J2=1082.6e-5; %J2 coefficient;

P=2*pi*(a^3/u)^.5; % Orbit Period [s]
v=sqrt(u/a); %satellite velocity [m/s]
n=(2*pi/P)*86400;  %number of revolutions/day


% ATMOSPHERIC MODEL
%get the density from MSISE model (extracted as a list from
%http://omniweb.gsfc.nasa.gov/vitmo/msis_vitmo.html
%    h[km]    rho[kg/m^3]
% filename1='msis_26371.lst'; filename2='msis_2226.lst';
% formatSpec='%f %f';
% size=[2 Inf];
% fileID=fopen(filename1,'r');
% rho_h=fscanf(fileID,formatSpec,size);
% fclose(fileID);
% fileID=fopen(filename2,'r');
% rho_h=[rho_h fscanf(fileID,formatSpec,size)];
% fclose(fileID);
% rho=interp1(rho_h(1,:),rho_h(2,:)',(a-RE)/1000);

    H = (a - RE)*1e-3;
    % Compute exospheric temperature [K]
```

```
    T = 900 + 2.5*(120-70);
    % Compute effective atmospheric molecular mass [km/K], valid 180<H<500
    M = 27 - 0.012 * (H - 200);
    % Compute atmospheric scale height [km]
    SH = T / M;
    % Compute atmospheric density [kg/m3]
    rho = 6E-10 * exp(-(H - 175) / SH);


pert=zeros(4,5);

%% Computation

%J2 pertubation________________ PER REVOLUCIO
j=1;
pert(j,5)=-1.5*n*J2*(RE/a)^2*cos(i)*(1-e^2)^-2;
pert(j,4)=0.75*n*J2*(RE/a)^2*(4-5*(sin(i))^2)*(1-e^2)^-2;

%Atmospheric Drag______________ PER REVOLUCIO
%Perturbations/revolution
j=2;
pert(j,1)=-2*pi*rho*a^2/Bc;

% incP=-6*pi^2*a^2*rho/(v*Bc);
% incv=pi*a*rho*v/Bc;

%Moon Perturbation_____________ /DIA --> /REV
j=3;
pert(j,5)=-0.00338*cos(i)/n;
pert(j,4)=0.00169*(4-5*(sin(i))^2)/n;

% Fins aqui son /dia,
% Canviem unitats + apliquem periode actual
pert(j,5)=pert(j,5)*P/(24*3600);
pert(j,4)=pert(j,4)*P/(24*3600);

%Sun Perturbation_____________ /DIA --> /REV
j=4;
pert(j,5)=-0.00154*cos(i)/n;
pert(j,4)=0.00077*(4-5*(sin(i))^2)/n;

% Fins aqui son /dia,
% Canviem unitats + apliquem periode actual
pert(j,5)=pert(j,5)*P/(24*3600);
pert(j,4)=pert(j,4)*P/(24*3600);


end
```

## 1.8 Orbit Decay

```
%----ASTREA CONSTELLATION----
%PROJECTS - 220028
%Aerospace Engineering Barchelor's Degree
%ESEIAAT - UPC
%Autumn 2016-2017

% ORBIT DESIGN TEAM
% ORBIT DECAY

% This routine computes the orbital decay of an spacecraft with time
% using the cowell's method.
% The differential equations are integrated using "ode45"(Runge-Kutta 4th,
% 5th order)


%------------------------------------------------------------------------
%%
clear all;
clc;
close all;

%% Initialization %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global mu mmu smu re rs rm omega J2;

%_______Physical Variables____________________________
mu=3.986004418e14; % earth gravitational constant [m^3/s^2]
mmu=4902.800076e9; %moon gravitational constant [m^3/s^2]
smu=132712440040.944e9; %sun gravitational constant [m^3/s^2]

re=6378136.3; %earth radius [m]
rs=696e6; %sun radius [m]
rm=1730e3; %moon radius [m]

omega=7.292115e-5; %earth angular velocity [rad/s]
J2=0.001081874; %earth oblateness gravity coeff []

au=149597870691; %astronomical unit [m]
c=2.99792458e8; %[m/s] speed of light

%____Numerical variables_______
hlim=180; %minimum height for considering the decay [km]
dt=15*60; %timestep of the simulation [s]
t0=0; %initial time of the sim
tf=dt;

%% User Input %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%__Orbital elements______________
a=re+542e3; %[m] semimajor axis
e=0.0001; % eccentricity
i=degtorad(72); %[rad] inclination
w=0; %[rad] argumetn f periapsis
Omega=0; %[rad] RAAN
nu=0; %true anomaly

date=datetime(2017,2,4); %datetime array [Y, M, D]

%__Oblateness parameters_________
znls=0; %zonals [0-18]
tssrls=0; %tesserals [0-18]

%__Drag perturbation inputs______
Cd=2.2; %Drag coefficient (Typical for sats, various ref.)
Adrag=.09; %[m^2] Wet area for the drag
m=4.1; %[kg] Cubesat mass

Bc=Adrag*Cd/m; %Ballistic coeff.

%__SRP perturbation inputs______
Cr=2; %reflectivity ct.
Asrp=.5; %[m^2] Area for the SRP

% Ws (@Dap) = 1361/(1+0.0334*cos(2*pi*Dap/365);
Ws=1361; %W/m^2 approximation for the solar irradiance

srp=Ws/c*Cr*Asrp/m*au^2; %Solar radiation pressure ct.

%% Main Process %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
display('Computing...');
f=@(q) q*((3+3*q+q^2)/(1+(1+q)^3/2)); %richard Battin's function

aobl=zeros(3,1);
adrag=aobl; asun=aobl; amoon=aobl; asrp=aobl;


y=cartesian2(a,e,i,Omega,w,nu); %obtain the state vector (r,V)
jdate0=juliandate(date); %compute the julian date

tic
iter=0; tt=[]; xx=[];

h=(norm(y(1:3))-re)*1e-3;
while h>hlim
    jdate=jdate0+t0/86400;
%     GMST=JD2GMST(jdate);
%
%     %Acceleration due to oblateness
```

```
%     aobl=Oblat((t0+tf)/2,y,znls,tssrls,GMST);
%
%     %Acceleration due to the 3rd body pert.
%     rsun=planetEphemeris(jdate,'Earth','Sun');
      %geocentric pos of the Sun
%     rmoon=planetEphemeris(jdate,'Earth','Moon');
      %geocentric pos of the Moon
%     rsun=rsun'; rmoon=rmoon';
%
%     rs2s=y(1:3)-rsun; %vector sun-sat
%     rm2s=y(1:3)-rmoon; %vector moon-sat
%
%     qs=dot(y(1:3),(y(1:3)-2*rsun))/dot(rsun,rsun);
%     qm=dot(y(1:3),(y(1:3)-2*rmoon))/dot(rmoon,rmoon);
%
%     asun=-smu/norm(rs2s)^3*(y(1:3)+f(qs)*rsun);
%     amoon=-mmu/norm(rm2s)^3*(y(1:3)+f(qm)*rmoon);
%
%     %Acceleration due to the solar radiation presure.
%     usun=rsun/norm(rsun); %unit vector of the pos of the sun
%     us=y(1:3)/norm(y(1:3)); %unit vector of the pos of the sat
%
%     shadow=random('Normal',0.5,.1); %shadow factor
%
%     asrp=srp*rs2s/norm(rs2s)^3;

    %Acceleration due to the drag force
    % Compute exospheric temperature [K]
    T = 900 + 2.5*(100-70);
    % Compute effective atmospheric molecular mass [km/K], valid 180<H<500
    M = 27 - 0.012 * (h - 200);
    % Compute atmospheric scale height [km]
    SH = T / M;
    % Compute atmospheric density [kg/m3]
    rho = 6E-10 * exp(-(h - 175) / SH);

    atmosV=y(4:6)+omega*[y(2); -y(1); 0];
    %relative vel between sat and atmos

    adrag=-.5*rho*Bc*atmosV*norm(atmosV);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    acc=aobl+asun+amoon+asrp+adrag;

    ode=@(t,y)[y(4);
            y(5);
            y(6);
            acc(1)-mu/norm(y(1:3))^3*y(1);
            acc(2)-mu/norm(y(1:3))^3*y(2);
            acc(3)-mu/norm(y(1:3))^3*y(3)
```

```
            ]; %function to integrate (6eqn)

    [t,x]=ode45(ode,[t0 tf], y);

    y=x(size(x,1),:);
    %save one value each 12 hours
    if mod(t(length(t)),12*3600)==0
        tt=[tt; t(length(t))]; xx=[xx; y];
    end
    y=y';

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %End of the timestep
    t0=tf;
    tf=tf+dt;

    h=(norm(y(1:3))-re)*1e-3;

    iter=iter+1;
    if mod(iter,300)==0
        fprintf('Days elapsed: %0.1f \nH=%0.3f km\n',(jdate-jdate0),h);
    end
end
toc
%% Post Process %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('The satellite decays in %0.0f days',tf/86400);

h=zeros(size(xx,1),1);
for i=1:size(xx,1)
    h(i)=(norm(xx(i,1:3))-re)*1e-3;
end

plot(tt/86400,h);
```

```
% This function converts kepler orbit elements to cartesian coordinates

% a: semimajor axis
% e: eccentricity
% i: inclination [rad]
% Omega: longitude of the ascending node [rad]
% w: Argument of periapsis [rad]
% nu: True anomaly [rad]

function [y] = cartesian2(a,e,i,Omega,w,nu)
global mu;
% Position in cylindrical coordinates
r = a*(1-e^2)/(1+e*cos(nu));

% Position components
```

```
X = [r*(cos(Omega)*cos(w+nu)-sin(Omega)*sin(w+nu)*cos(i));
    r*(sin(Omega)*cos(w+nu)+cos(Omega)*sin(w+nu)*cos(i));
    r*sin(i)*sin(w+nu)];

A = sin(Omega)*cos(i)*(e*cos(w)+cos(w+nu));
v(1)=-sqrt(mu/( a*(1-e^2)))*(cos(Omega)*(e*sin(w)+sin(w+nu))+A);

B = cos(Omega)*cos(i)*(e*cos(w)+cos(w+nu));
v(2)=-sqrt(mu/( a*(1-e^2)))*(sin(Omega)*(e*sin(w)+sin(w+nu))-B);

v(3)=sqrt(mu/( a*(1-e^2)))*(e*cos(w)+cos(w+nu))*sin(i);

y=[X;v'];
end
```

## 1.9 Performance Evaluator

```
%----ASTREA CONSTELLATION----
%PROJECTS - 220028
%Aerospace Engineering Barchelor's Degree
%ESEIAAT - UPC
%Autumn 2016-2017

% ORBIT DESIGN TEAM
% PERFORMANCE EVALUATOR

clear all;

%% Input Data

a=6378.01e3;
u=3.986012e14;
h=(a+546.5101e3)/a;
ap=h*a;
J2=1.0826e-3;% J2=0;

ws=sqrt(u/ap^3); % Satellite mean motion
we=2*pi/(24*3600); % Earth mean motion
dOmega=-1.5*J2*(a/ap)^2*sqrt(u/ap^3)*cosd(75);

At=1/6; % [min]          % Every 10 seconds
t=0:At:60*24; t=t*60; % Time array in seconds

i=72*pi/180;

p=9;
```

```
spp=21;
N=p*spp;
%f=floor(0.25*spp);
f=0;
fprintf('\ni=%gdeg, p=%g, spp=%g, N=%g\n',i*180/pi,p,spp,N);

latGS=57.5*pi/180;
longGS=(17.73-60)*pi/180; longGS=0;

fprintf('GS Coordinates: Lat=%g, Long=%g \n',latGS*180/pi, longGS*180/pi)

f1=pi/2-latGS; f2=longGS;
R1=[cos(f1) 0 -sin(f1);0 1 0;sin(f1) 0 cos(f1)];
R2=[cos(f2) sin(f2) 0; -sin(f2) cos(f2) 0; 0 0 1];
RotGS=R1*R2;

% Rotation Matrix:
% From ECEF - To SEZ
% Where:
% ECEF = Earth Centered Earth Fix - Coordinate System
% SEZ = Topocentric Horizon - Coordinate System

emin=20*pi/180;

%% Le simulasion
%Omega = 0:30:240; Omega=Omega*pi/180;
Omega = 0:225/(p-1):225; Omega=Omega*pi/180;
%Omega=0:360/p:360-360/p; Omega=Omega*pi/180;

nu = zeros(spp,p);
angle = zeros(1,p);
X = zeros(3,spp*p);

flight_time=zeros(1,1000);     % Length of the flyby of a satellite
links_at_time=zeros(1,1000);   % Number of links when one sat has finished
time_end_flyby=zeros(1,1000); % When did this flyby end?

contact=zeros(1,length(t)); % Number of links

quality_time=zeros(1,length(t));
%If a flyby lasts longer than 3 minutes,
% then the previous 3 minutes were successfuly covered.

time_record=zeros(1,N);      % Accumulates Timesteps being on the GS
before_tracking=zeros(1,N);
% Boolean to know which ones are already passing by
now_tracking=zeros(1,N);    % Boolean to know which ones are now passing by
Nflyby=0;                   % Number of flight paths computed

for n=1:length(t)
```

```

    % J2 deviation
    Omegat=Omega+dOmega*t(n);

    % Ground Station Coordinates
    f1=we*t(n)+longGS;
    f2=latGS;
    RGS=cos(f2);
    XGS=[RGS*cos(f1);RGS*sin(f1);sin(f2)];

    R2=[cos(f1) sin(f1) 0; -sin(f1) cos(f1) 0; 0 0 1];
    RotGS=R1*R2;

    % Constellation Coordinates
    nu_t=ws*t(n);   % True anomaly due to time passing by
    for j = 1:p
        angle(j) = f*2*pi*(j-1)/(spp*p); %Phasing due to f between planes
        for k = 1:spp
            % True anomalies of the s satellites
            A = spp*(j-1)+k; % Number of the satellite
            nu(k,j) = 2*pi*(k-1)/spp+angle(j);
            X(:,A) = cartesian(h,0,i,Omegat(j),0,nu(k,j)+nu_t);
        end
    end

    % Contact Evaluation
    win=0;  % Counter to know number of links
    now_tracking=zeros(1,N);
    for sat=1:N
        X_hor=RotGS*(X(:,sat)-XGS);
%   fprintf('%g,%g,%g --> |r|=%g\n',X_hor(1),X_hor(2),X_hor(3),norm(X_hor))
        ang=asin(X_hor(3)/norm(X_hor));
        if ang ≥ emin
            win=win+1;
            time_record(sat)=time_record(sat)+1;
            now_tracking(sat)=1;
        end

        % Flight time computation and reset of this satellite flyby
        if before_tracking(sat)==1 && now_tracking(sat)==0
            Nflyby=Nflyby+1;
            flight_time(Nflyby)=At*time_record(sat);

            links_at_time(Nflyby)=contact(n-1);
            time_end_flyby(Nflyby)=t(n);

            % THE KEY QUESTION
            % Was this flyby useful?
            if flight_time(Nflyby)>3
                start=n-time_record(sat);
```

```
                finish=n;
                quality_time(start:finish)=quality_time(start:finish)+1;
            end
            time_record(sat)=0;
        end
    end
    before_tracking=now_tracking;
    contact(n)=win;
end

%% Post-Processing
% PLOT 1: NUMER OF LINKS
figure(1)
title('Links vs Time')
plot(t/(3600),contact)
axis([min(t)/(3600) max(t)/(3600) 0 max(contact)+1]);
 fails=length(find(contact==0));
 ratio=fails/length(contact);
 fprintf('Links on GS         : %g percent of the time\n',(1-ratio)*100)

% PLOT 2: LENGTH OF THE LINKS
figure(2)
plot(flight_time(1:Nflyby));
titola=['Length of the ' num2str(Nflyby) ' flyby s. Mean time = '...
    num2str(mean(flight_time(1:Nflyby))) 'min'];
title(titola)
ylabel('Length (minutes)')
xlabel('Contact')
%xlim([1 Nflyby])

% PLOT 3: ANALYSIS OF THE FLYBYS
figure(3)
index=1:Nflyby; % X variable to the following plots
plot(time_end_flyby(index)/3600,flight_time(index),...
     time_end_flyby(index)/3600,links_at_time(index),...
     t/3600,quality_time);

titola='Flybys Analysis';
title(titola)

legend('Length of flybys','Links by end of flyby',...
    'Num of sats @flybys longer than 3 min')
xlabel('Time (h)')

epic_wins=length(find(quality_time≥1));
ratio_covered=epic_wins/length(t);
fprintf('Quality flybys on GS: %g percent of the time\n',ratio_covered*100)
fprintf('Mean flyby time = %g minutes\n',mean(flight_time(1:Nflyby)))
```

```
%% MAXIMUM GAP SEARCH
gap=0;
at_gap=0;
gap_record=zeros(1,100);
ngap=0;

for n=1:length(t)
    if quality_time(n)==0
        at_gap=1;
        gap=gap+1;
    end

    if quality_time(n)>0 && at_gap==1
        at_gap=0;
        ngap=ngap+1;
        gap_record(ngap)=gap*At;
        gap=0;
    end
end

max_gap=max(gap_record);
fprintf('Maximum gap = %g minutes\n',max_gap)
fprintf('Number of gaps = %g\n\n',ngap)
```

## 1.10 Thrust

```
%----ASTREA CONSTELLATION----
%PROJECTS - 220028
%Aerospace Engineering Barchelor's Degree
%ESEIAAT - UPC
%Autumn 2016-2017

% ORBIT DESIGN TEAM
% THRUST

% This function computes the ΔV and propellant mass necessary to
% maintain an orbit between to heights

%function [mp,ΔV,tHoh] = thrust(hmax,hmin,ms,Isp,Thr)

% This function computes the ΔV and propellant mass necessary to
% maintain an orbit between to heights

% Input variables:
% - hmax: maximum height [m]
% - hmin: minimum height [m]
```

```
% - ms: dry mass of the spacecraft [kg]
% - Isp: specific impulse of the spacecraft [s]
% - Thr: thrust of the spacecraft [N]

% Output variables:
% - mp: array of propellant mass necessary for every Hohmann transfer [kg]
% - ΔV: array of ΔVs necessary for every Hohmann transfer [m/s]
% - tHoh: array of time necessary to do a Hohmann transfer [s]

% Proposed values:
 hmax = 550e3;
 hmin = hmax-8;
 ms = 3.95;        % Dry mass [kg]
 Isp = 2150;
 Thr = 100e-6;

%% Data

% PHYSICAL CONSTANTS AND PARAMETERS
RE = 6.378e6; %Earth Radius [m]
mu = 3.986e14; %GM Earth
g0 = 9.81;

% SPACECRAFT DATA
m = 4;          % Satellite mass [kg]
Cd = 2.2;       % Drag Coefficient
A = 0.1*0.3;    % Satellite surface [m^2] --> 3U pointing to Earth

% ORBIT DATA
H0 = hmax;      % Altura inicial [m]
E0 = 0.01;      % Excentricity
I0 = 80*pi/180;        % Inclination [deg]
A0 = RE+H0;

% SIMULATION PARAMETERS
N=100000;
M = 10000;

%% 2. Perturbations propagation

% Creation of the matrices
temp=zeros(M,N);
temp(1,1)=0;
H = zeros(M,N);
H(1,1) = H0;
% ΔV = zeros(2,N);

% Asign initial values
W0=0; OMEGA0=0;
a=zeros(1,N); a(1)=A0;
```

```
e=a; e(1)=E0;
i=a; i(1)=I0;
t=a; t(1)=0;
w=a; w(1)=W0;
Omega=a; Omega(1)=OMEGA0;

for j = 1:M

    n = 1;
    while a(n)≥(RE+hmin)

        n = n+1;

        a0=a(n-1);
        e0=e(n-1);
        i0=i(n-1);
        w0=w(n-1);
        Omega0=Omega(n-1);

        P=2*pi*(a0^3/mu)^.5; % Orbit Period [s]

        Bc = m/(Cd*A);% Ballistic coefficient of the satellite
        Perti=Perturbation(a0,e0,i0,Bc);

        a(n)=a0+sum(Perti(:,1));
        e(n)=e0+sum(Perti(:,2));
        i(n)=i0+sum(Perti(:,3));
        w(n)=w0+sum(Perti(:,4));
        Omega(n)=Omega0+sum(Perti(:,5));

        % We don't want angles bigger than 360!
        w(n)=w(n)-360*floor(w(n)/360);
        Omega(n)=Omega(n)-360*floor(Omega(n)/360);

        if Omega(n)>2*pi
            Omega(n)=Omega(n)-2*pi;
        end

        t(n)=t(n-1)+P;

    end

    % Hohmann
    [ΔV1,ΔV2,mpit,tHohit] = Hohmann(a(n)-RE,hmax,m,Isp);
    m = m-mpit;

    if m≤ms
        break
    end

```

```
    ∆V(1,j) = ∆V1; % first row of the column -> ∆V1
    ∆V(2,j) = ∆V2; % second row of the column -> ∆V2
    mp(j) = mpit;
    tHoh(j) = tHohit;

    a(1)=A0;
    e(1)=E0;
    i(1)=I0;
    w(1)=W0;
    Omega(1)=OMEGA0;
    altura = a-RE;

    H(j,:) = altura;
    temp(j,:) = t;

end

%% Post process

mfr = Thr/(g0*Isp);

for i = 1:N
    if H(1,i)≤0
        break
    end
end
for j = 1:M
    if H(j,1)≤0
        break
    end
end
H = H(1:(j-1),1:(i-1));

for i = 2:N
    if temp(1,i)≤0
        break
    end
end
for j = 2:M
    if temp(j,2)≤0
        break
    end
end
temp = temp(1:(j-1),1:(i-1));
for j = 2:size(temp,1)
    temp(j,:) = temp(j,:)+tHoh(j-1)+temp(j-1,size(temp,2));
end

% Convert matrix to vector
H = reshape(H.',1,size(H,1)*size(H,2));
```

```
temp = reshape(temp.',1,size(temp,1)*size(temp,2));

figure(1)
plot(temp/(3600*24),H/1000)
grid on
ylabel('Orbit height [km]')
xlabel('Time [days]')
tfin=floor(temp(length(temp))/(3600*24));
days=num2str(floor(tfin/365));
titulaso=['Orbit decay in ' num2str(tfin) ' days = ' days ' years'];
title(titulaso)

%% Final calculations

% % Perimeter of the ellipse
% r1 = RE+hmax; % semimajor axis
% r2 = RE+hmin; % semiminor axis
% h = (r1-r2)^2/(r1+r2)^2;
% % Ramanujan approximation
% C = pi*(r1+r2)*(1+3*h/(10+sqrt(4-3*h)));
%
% % Check if ΔV is possible for the given thruster
% OK = true;
% for j = 1:length(mp)
%     for i = 1:2
%         mfrnecessary = ΔV(i,j)*mp(j)/(C/2);
%         if mfrnecessary>mfr
%             OK = false;
%         end
%     end
% end
%
% if OK ==true
%     fprintf('The trajectory is possible\n\n');
% else
%     fprintf('Trajectory not possible\n\n');
% end
if length(mp)<M
    fprintf('There is still propellant left ._.\n\n')
end
```

```
function [ΔV1,ΔV2,mp,t] = Hohmann(hinicial,hfinal,m,Isp)

% Hohmann transfer orbit between two circular orbits
% - hinicial: height of the first orbit [m]
% - hfinal: height of the second orbit [m]
% - m: mass of the satellite (total) [kg]
% - Isp: Specific impulse [s]
% - mp: fuel mass [kg]
```

```
% - t: time needed to do the maneuver [s]

mu = 3.986004418e14; % Standard gravitational parameter (Earth)
REarth = 6.371e6; % [m]
g0 = 9.81; % Earth's gravity [m/s^2]

% First orbit
r1 = REarth+hinicial; % [m]
v1 = sqrt(mu/r1); % [m/s]

% Second orbit
r2 = REarth+hfinal; % [m]
v2 = sqrt(mu/r2); % [m/s]

% Transfer orbit (ellipse)
vp = sqrt(2*mu*r2/(r1*(r1+r2))); % [m/s]
va = sqrt(2*mu*r1/(r2*(r1+r2))); % [m/s]
a = (r1+r2)/2;
T = sqrt(2*pi^2*a^3/mu);

% ΔV 1 -> p(transfer orbit)
ΔV1 = vp-v1;
% ΔV a(transfer orbit) -> 2
ΔV2 = v2-va;
% Total
ΔV = ΔV1+ΔV2;

% Fuel mass
mp = m*(1-exp(-ΔV/(g0*Isp)));

% Time
t = T/2;

end
```

## 1.11 Satellites Datasheet

```
%----ASTREA CONSTELLATION----
%PROJECTS - 220028
%Aerospace Engineering Barchelor's Degree
%ESEIAAT - UPC
%Autumn 2016-2017

% ORBIT DESIGN TEAM
% SATELLITES DATASHHET

```

```
% Writing the constellation caracteristics

a=6378.01e3;
u=3.986012e14;
h=(a+542e3)/a;
ap=h*a;
J2=1.0826e-3;% J2=0;

ws=sqrt(u/ap^3); % Satellite mean motion
i=72*pi/180;

p=9;
spp=21;
N=p*spp;
%f=floor(0.25*spp);
f=0;
fprintf('\ni=%gdeg, p=%g, spp=%g, N=%g\n',i*180/pi,p,spp,N);


%Omega = 0:30:240; Omega=Omega*pi/180;
Omega = 0:225/(p-1):225; %Omega=Omega*pi/180;
%Omega=0:360/p:360-360/p; Omega=Omega*pi/180;

P=2*pi/ws/60;

%% Excel IMPORT

D=zeros(N,9);
for sat=1:N

    Om=ceil(sat/spp); ID_p=sat-(Om-1)*spp;
    ph=(ID_p-1)*360/spp;
   %[ID Plane h P i e Omega phase per]
    D(sat,1:9)=[sat Om (ap-a)/1000 P i*180/pi 0 Omega(Om) ph 0];

end
NameCells=['B2:I' num2str(N)];
xlswrite('SatsDatasheet.xls',D,NameCells)

%% Latex IMPORT

for sat=1:N

    Om=ceil(sat/spp); ID_p=sat-(Om-1)*spp;
    ph=(ID_p-1)*360/spp;
   %[ID Plane h P i e Omega phase per]
    D(sat,1:9)=[sat Om (ap-a)/1000 P i*180/pi 0 Omega(Om) ph 0];

end

```

```
[rows,cols]=size(D);
file=fopen('Autotable.txt','w');

for i=1:rows
    Text=['AstreaSAT ' num2str(i) ' $ '];
    for j=2:cols
        if j≠cols
            Text=[Text num2str(D(i,j-1)) ' $ '];
        else
            Text=[Text num2str(D(i,j-1)) ' \\ '];
        end
    end
    fprintf(file,[Text '\n']);
end

fclose(file);
```

## 1.12 Ground Station Localization

```
%----ASTREA CONSTELLATION----
%PROJECTS - 220028
%Aerospace Engineering Barchelor's Degree
%ESEIAAT - UPC
%Autumn 2016-2017

% COMMUNICATION TEAM
% GROUND STATIONS LOCALIZATION

clc; clear;

%% Input Data
N_sat=21;   %number of sats in a plane
N_planes=9; %number of orbital planes
h=542;      %high of the sats in km
I=72;       %inclination of the orbits in degrees
phase=210/(N_planes-1); %Angle between planes in the equator
At=0.1;       %time step in minutes
T=48;       %time to simulate in hours
t=0:At:T*60;    %array of time in minutes
e_min=7.5;        %minumum elevation in degrees
lambda=57.5; %latitudes to simulate in degrees
lat=length(lambda);
mu=0;           %longitudes to simulate in degrees
long=length(mu);
l=length(t);

```

```
%% Solver

Xs=Orbitalposition(N_sat,N_planes,h,I,phase,t);
L=zeros(lat,l);

for i=1:lat
    Xg=Groundposition(lambda(i),mu,t);
    [L(i,:)]=Links(Xs,Xg,e_min);
end

%% Plots

figure;plot(t/60,L);legend 0 30 60 90;xlabel('time (h)');ylabel('links');
```

```
function [X] = Orbital_position(N_sat,N_planes,h,I,mu,t)
%N_sat: number of sats per plane
%N_planes: number of planes
%h: Orbit high in km
%I: inclination of the orbital plane in degrees
%mu: Angle between two adjacent orbital planes at the equatiorial plane in
%degrees
%t: array of time in minutes

I=I*pi/180;    %I in rad
mu=mu*pi/180; %mu in rad
Omega=0:mu:(N_planes-1)*mu;
%longitude of the ascending node respect the inertial X axe [rad]

%for defining the orbits are defined a system of
%local axes X_o in which the y_o is the rotation axe and x_o is
%contained in the equatiorial plane. Thetransformation for the inertial
%system X_I to the local system 2 rotations are defined.
%FIRST ROTATION: it defines a intermediate system X_1. The inertial system
%rotates arround z_I an Omega angle.
%SECOND ROTAtion: The X_1 system is rotated (90deg-I) arround x_1

L_1o=[1 0 0; 0 sin(I) cos(I); 0 -cos(I) sin(I)]; %1st rotation matrix.
%It transform a vector from local coordinates X_o to intemediate
%coordinates X_1. This matrix is the same for every orbital plane
%since all orbits have the same inclination

Re=6371;  %Earth radius km.
%The distances are going to be refered to the earth radius
h=h/Re;      %high in earth radius
GM=6.474e-11*5.972e24*3600/Re^3;
%Gravitational constant*earth mass [(Re^3)/min^2]
Ro=1+h;      %orbital radius in earth radius

w=sqrt(GM/(Ro^3));  %sat's angular speed [rad/min]
```

```
phi=0:2*pi/N_sat:(N_sat-1)*2*pi/N_sat;
%relative angle between every sat of a plane respect the first one
f=8;
%factor to module the relative desfase of the first sat of one plane
%to the first sat of the next plane
l=length(t);

for p=1:N_planes    %going through each plane

    phi_0(p)=f*2*pi*(p-1)/(N_planes*N_sat);
    %relative desfase of the first sat of one plane to the
    %first sat of the next plane

    One = cos(Omega(p));
    Two = sin(Omega(p));
    L_I1=[One -Two 0; Two One 0; 0 0 1];
    %2nd rotation matrix. It transforms a vector from
    %intermediate coordinates X_1
    %to inertial coordinates X_I. This matrix is characteristic of every
    %plane since evey plane has a diferent longitude of the ascending node
    %(Omega)

    L_eo=L_I1*L_1o;
    %Global rotation matrix form local coordinates to nertial coordinates

    for s=1:N_sat   %going through every sat of the plane

        for k=1:l   %going through every time step

            First = Ro*cos(w*t(k)+phi(s)+phi_0(p));
            Second = -Ro*sin(w*t(k)+phi(s)+phi_0(p));
            X_loc=[First;0;Second];
            %Coordinates of the sat at the given time in
            %local coordinates X_o.
            %It defines a circumference at the plane x_o-z_o with a angular
            %velocity w, a initial phase respect the first sat and a
            %initial phase respect the first sat of the previous orbit
            %first sat.

            X(:,(p-1)*N_sat+s,k)=L_eo*X_loc;
            %Coordinates of the sat in inertial system
            %(Coordinate [x y z], Sat number, instant(time)]
        end
    end
end
end
```

```
function [X] = Ground_position(lambda,long,t)
%Gorund_position calculates the positon of a place of the earth, with a
```

```
%wiven longitude and latitude, in an inertial system of coordiantes X_I in
%a given period of time

%lamda: is the latitude of the place in degrees[-90,90]
%long: is the longitude of the place iin degrees [0,360)
%t: is a array of time in minutes

long=long*pi/180;    %longin radians
lambda=lambda*pi/180;    %lambda in radians

Rg=cos(lambda);
%distance between the Earth point to the rotation axis of the Earth [0,1]
hg=sin(lambda);
%distance between th Earth point to the equatorial plane [0,1]
%this 2 distances are measured in earth radius.

w=2*pi/(24*60); %angular velocity of the Earth [rad/min]

X=[Rg*cos(w*t+long);Rg*sin(w*t+long);hg*ones(1,length(t))];
%position of the point in X_I
%for the given time. The point descrives a cicumference
%with a Rg radius in the horizontal
%plane x_I-y_I with an angular velocity w and a initial phase long. The
%coordinate z_I is allways hg. It is expresed in earth radius.
%X(cooridinates, instant (time)

end
```

# Chapter 2

# Bibliography