



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	ii
List of Figures	iii
1 Matlab Codes	1
1.1 Satellite Footprint	1
1.2 Minimum Plane Inclination	3
1.3 Satellite Number Computation for Polar Orbits	5
1.4 Orbit Parameters	8
1.5 Walker Delta Testing	13
1.6 Orbit Plotter	14
1.7 Perturbations	21
1.8 Orbit Decay	27
1.9 Performance Evaluator	31
1.10 Thrust	35
1.11 Satellites Datasheet	40
1.12 Ground Station Localization	42
2 Bibliography	46

List of Tables

List of Figures

Chapter 1

Matlab Codes

1.1 Satellite Footprint

```
1  %----ASTREA CONSTELLATION----
2  %PROJECTS - 220028
3  %Aerospace Engineering Barchelor's Degree
4  %ESEIAAT - UPC
5  %Autumn 2016-2017
6
7  % ORBIT DESIGN TEAM
8  % COMPUTATION OF A SATELLITE FOOTPRINT
9
10 clc
11 clear all
12 close all
13
14 %% PHISICAL CONSTANTS AND PARAMETERS
15
16 Re = 6378;           %Earth's Radius [Km]
17 Se = 4*pi*Re^2;      %Earth's Surface [Km^2]
18 h = 500:100:1000;   %Satellte height [Km]
19 eo = 0:5:40;         %Elevation angle [deg]
20
21 %% SOLVER
22
23 for i = 1:length(h)
24     for j = 1:length(eo)
25
26         d(i,j) = Re*(sqrt(((h(i)+Re)/Re)^2-(cosd(eo(j)))^2))-sind(eo(j)));
27         Bo(i,j) = asind( d(i,j)*cosd(eo(j)) / (h(i)+Re) );
28
29         S(i,j) = 2 * pi * Re^2 * ( 1-cosd(Bo(i,j)) );
```

```

30
31     R(i,j) = sqrt( d(i,j)^2 - h(i)^2 );
32
33     %Satellite coverage expressed as a fraction of Earth's Area (%)
34     Cov(i,j) = ( S(i,j) / Se ) * 100;
35
36     %Number of satellites needed
37     nsat(i,j) = ( Se / S(i,j) );
38
39     end
40 end
41
42 %% PLOTS
43
44 figure
45 for k = 1:length(h)
46     plot ( eo , Cov(k,:) )
47     hold on
48 end
49
50 title('Coverage vs Elevation')
51 xlabel('Elevation [deg]') % x-axis label
52 ylabel('Coverage [%]') % y-axis label
53 legend(num2str(h(1)),num2str(h(2)),num2str(h(3)),...
54         num2str(h(4)),num2str(h(5)),num2str(h(6)))
55
56 figure
57 for k = 1:length(h)
58     plot ( eo , nsat(k,:) )
59     legend ('h(j)')
60     hold on
61 end
62
63 title('Num.Satellites vs Elevation')
64 xlabel('Elevation [deg]') % x-axis label
65 ylabel('Num.Satellites') % y-axis label
66 legend(num2str(h(1)),num2str(h(2)),num2str(h(3)),...
67         num2str(h(4)),num2str(h(5)),num2str(h(6)))
68
69 figure
70 for k = 1:length(h)
71     plot ( eo , 2*Bo(k,:) )
72     hold on
73 end
74
75 title('Angle vs Elevation')
76 xlabel('Elevation [deg]') % x-axis label
77 ylabel('Angle between satellites[deg]') % y-axis label
78 legend(num2str(h(1)),num2str(h(2)),num2str(h(3)),...
79         num2str(h(4)),num2str(h(5)),num2str(h(6)))

```

1.2 Minimum Plane Inclination

```

1  %---ASTREA CONSTELLATION---
2  %PROJECTS - 220028
3  %Aerospace Engineering Barchelor's Degree
4  %ESEIAAT - UPC
5  %Autumn 2016-2017
6
7  % ORBIT DESIGN TEAM
8  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9  % MINIMUM INCLINATION - To provide Full coverage %
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12 %PROBLEM:
13 % Given: - Minimum Elevation over the horizon to contact the satellite
14 %         - Height of the orbit
15 % Compute the minimum elevation to ensure visual contact
16 clear; clc;
17
18 %% Key variable
19 min_elevation_pole=5; %Initially set by @lfore as 32;
20
21 %% Previous Calculation: Visibility(latitude)
22
23 R = 6371;
24 N = 180;
25 deg = 180;
26 x = linspace(0,90,N);
27
28 %%% Minimum angle of elevation due to atmospheric conditions at given
29 %%% latitudes. 0 km represents the poles whereas 6371 km represents the
30 %%% Earth's equator taking as reference point the south pole.
31
32 elvlat = zeros (1,deg);
33 for i = 1:deg/5
34     elvlat(i) = min_elevation_pole-(min_elevation_pole-15)/(2*(90-75))*i;
35 end
36 for i = deg/5+1:4*deg/10
37     elvlat(i) = elvlat(deg/5);
38 end
39 for i = 4*deg/10+1:6*deg/10
40     elvlat(i) = elvlat(2*deg/5)+3/15*(i-4*deg/10);
41 end
42 for i = 6*deg/10+1:8*deg/10
43     elvlat(i) = elvlat(6*deg/10)-3/15*(i-6*deg/10);
44 end
45 for i = 8*deg/10+1:10*deg/10
46     elvlat(i) = elvlat(8*deg/10)-1/15*(i-8*deg/10);

```


Minimum Plane Inclination

```

47 end
48 elvlat=flipplr(elvlat); %Sets the first value to the value in the equator
49 %The first value matches 0 latitude
50
51 %% Minimum i computation
52
53 h = 400:100:1000; %Satellite height [Km]
54 L = 0:5:90; %Latitude angle [deg]
55 e = (interp1q(x',elvlat',L'))'; %Minimum elevations interpolation [deg]
56
57 figure(1)
58 subplot(1,2,1)
59 plot(x,elvlat)
60 subplot(1,2,2)
61 title('Interpolation Verification')
62 plot(L,e)
63
64 inc=zeros(length(h),length(L)); %Inclinations preallocation
65
66 for i = 1:length(h)
67     for j = 1:length(L)
68
69         A=cosd(e(j))/(1+h(i)/R);
70         theta=acosd(A)-e(j); %Earth Central Angle
71         inc(i,j)=L(j)-theta;
72         if inc(i,j)<0
73             inc(i,j)=0;
74         end
75     end
76 end
77
78 %% 3. Results Plotting
79
80 figure(2)
81 colors=['r','b','y','g','m','k','c'];
82 for k = 1:length(h)
83     hold on
84     plot ( L , inc(k,:),colors(k))
85 end
86
87 grid on
88 title('Minimum inclination required to fulfill global coverage')
89 xlabel('Latitude [deg]')
90 ylabel('Inclination [deg]')
91 legend(num2str(h(1)),num2str(h(2)),num2str(h(3)),...
92         num2str(h(4)),num2str(h(5)),num2str(h(6)),num2str(h(7)))
93
94 %% Bonus!!!
95
96 L=90;

```

```

97 e=5.33;
98 h=500:1:600;
99 inc=zeros(1,length(h));
100 for i = 1:length(h)
101     A=cosd(e)/(1+h(i)/R);
102     theta=acosd(A)-e; %Earth Central Angle
103     inc(i)=L-theta;
104     if inc(i)<0
105         inc(i)=0;
106     end
107 end
108
109 figure(3)
110 plot(h,inc)
111 xlabel('Heights')
112 ylabel('Inclination')
113 grid on
114 axis([min(h) max(h) 45 90])

```

1.3 Satellite Number Computation for Polar Orbits

```

1  %----ASTREA CONSTELLATION----
2  %PROJECTS - 220028
3  %Aerospace Engineering Barchelor's Degree
4  %ESEIAAT - UPC
5  %Autumn 2016-2017
6
7  % ORBIT DESIGN TEAM
8  % POLAR ORBITS Number of satellites computation
9
10 %PROBLEM:
11 % Given: - Elevation over the horizon to contact the satellite
12 %         - Height of the orbit
13 % Compute the final number of satellites
14 % Through the "Streets of coverage" method.
15
16 clear; clc;
17
18 %% 1. Input data
19 % PHYSICAL CONSTANTS AND PARAMETERS
20
21 Re = 6378;           %Earth's Radius [Km]
22 Se = 4*pi*Re^2;      %Earth's Surface [Km^2]
23 h = 500:20:600;      %Satellite height [Km]
24 eo = 20;             %Elevation angle [deg]
25

```

```

26 %% 2. Number of satellites computation
27
28 d=zeros(length(h),length(eo));
29 nsats_opt=d;
30
31 for i = 1:length(h)
32     for j = 1:length(eo)
33
34         d(i,j)= Re*(sqrt(((h(i)+Re)/Re)^2-(cosd(eo(j)))^2)-sind(eo(j)));
35         Bo = asind( d(i,j)*cosd(eo(j)) / (h(i)+Re) );
36         Sup = 2 * pi * Re^2 * ( 1-cosd(Bo) );
37
38         Nspp_min=ceil(360/(2*Bo));           %Number of satellites per plain
39         Nsatspp=Nspp_min:1:Nspp_min+50;     %Array of sats/plain to optimize
40
41         S=360./Nsatspp;
42
43         Lstreet=acosd(cosd(Bo)./cosd(S/2));
44
45         DmaxS=Lstreet+Bo;
46         DmaxO=2*Lstreet;
47
48         Nplains=ceil(1+((180-DmaxO)./DmaxS));
49         Nsats=Nplains.*Nsatspp;
50         [row,col]=find(min(Nplains));
51         Nplains_opt(i,j)=Nplains(row,col);
52         nsats_opt(i,j)=Nsats(row,col);
53
54         %Number of satellites needed
55         nsat(i,j) = ( Se / Sup );
56
57     end
58 end
59
60 %% 3. Results Plotting
61
62 figure(2)
63 colors=['r','b','y','g','m','k','c'];
64 for k = 1:length(eo)
65     subplot(1,2,1)
66     hold on
67     semilogy ( h , nsats_opt(:,k),colors(k) )
68     subplot(1,2,2)
69     hold on
70     semilogy ( h , Nplains_opt(:,k),colors(k) )
71 end
72
73 subplot(1,2,1)
74 title('Num.Sats vs Height')
75 xlabel('Height [km]') % x-axis label

```

```

76 ylabel('Num.Satellites') % y-axis label
77 %legend(num2str(eo(1)),num2str(eo(2)),num2str(eo(3)),...
78 %      num2str(eo(4)),num2str(eo(5)),num2str(eo(6)),num2str(eo(7)))
79 axis([400 900 0 1000])
80 grid on
81
82 subplot(1,2,2)
83 title('Num.Planes vs Height')
84 xlabel('Height [km]') % x-axis label
85 ylabel('Num of Orbital Planes') % y-axis label
86 % legend(num2str(eo(1)),num2str(eo(2)),num2str(eo(3)),...
87 %      num2str(eo(4)),num2str(eo(5)),num2str(eo(6)),num2str(eo(7)))
88 axis([400 900 0 30])
89 grid on
90
91 %% Single detailed analysis
92
93 figure(1)
94 plot(h,nsats_opt)
95 title('Num.Sats vs Height - Streets of coverage Method')
96 xlabel('Height [km]') % x-axis label
97 ylabel('Num.Satellites') % y-axis label
98 grid on
99
100 %% Multianalysis variation with height
101
102 figure(3)
103 colors=['r','b','y','g','m','k'];
104 for k = 1:length(h)
105     subplot(1,2,1)
106     hold on
107     plot ( eo , nsats_opt(k,:),colors(k))
108     subplot(1,2,2)
109     hold on
110     plot ( eo , Nplains_opt(k,:),colors(k))
111 end
112
113 subplot(1,2,1)
114 title('Num.Sats vs Elevation - Streets of coverage Method')
115 xlabel('Elevation [deg]') % x-axis label
116 ylabel('Num.Satellites') % y-axis label
117 legend(num2str(h(1)),num2str(h(2)),num2str(h(3)),...
118        num2str(h(4)),num2str(h(5)),num2str(h(6)))
119 axis([0 40 0 700])
120 grid on
121
122 subplot(1,2,2)
123 title('Num.Planes vs Elevation - Surfaces Method')
124 xlabel('Elevation [deg]') % x-axis label
125 ylabel('Num.Orbital Planes') % y-axis label

```

```

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

```

1.4 Orbit Parameters

```

1  %----ASTREA CONSTELLATION----
2  %PROJECTS - 220028
3  %Aerospace Engineering Barchelor's Degree
4  %ESEIAAT - UPC
5  %Autumn 2016-2017
6
7  % ORBIT DESIGN TEAM
8  % ORBIT PARAMETERS
9
10 % This routine calculates the minimum number of plans and satellites
11 % to guarantee global coverage.
12 % -----
13 % _v2 --> Computes angles with arccosine
14 % _v3 --> Also iterates in f
15
16 clc
17 clear all
18 close all
19
20 tic
21
22 %% DATA INPUT
23
24 Re = 6371;           %Earth's Radius [Km]
25 Se = 4*pi*Re^2;      %Earth's Surface [Km^2]
26 h = 500:5:600;      %Height vector [km]
27 in = 75:3:90;        %Inclination vector [deg]
28 pmin = 5; p=12;      %Minimum number of plans
29 pmax = 20;           %Maximum number of plans
30 sppmin = 10; spp=10; %Minimum number of satellites each plane
31 sppmax = 24;         %Maximum number of satellites each plane
32 typeWD = 2;          %Defines:
33                       % 1- Semi-Walker-Delta
34                       % 2- 3/4 of Walker-Delta
35                       % 3- Full Walker-Delta
36 plaunch = 5.76e6;    %Launch price
37 psat = 250e3;        %Price per satellite
38 eo = 20;             %Vison angle
39 q=0;                 %Variable used as counter

```

```

40 nplanes_win=0;
41 %Boolean, Indicates a succesful combination with that number of planes
42
43 %% COMPUTATIONS
44 %result=zeros(6,100);
45
46 for j=1:length(h)
47     fprintf('\n\nH = %g km',h(j))
48     Bo = visibleangle(h(j),eo); %Footprint angle
49
50     for k=1:length(in)
51         fprintf('\ni=%g: ',in(k))
52
53         for p=pmin:pmax
54             F=0:1:p-1; nF=length(F); anglesp_f=zeros(1,nF);
55
56             for spp=sppmin:sppmax
57                 angles = 360/spp; %Angle between satelllites of same plane
58
59                 for kf=1:nF
60                     % We look for the optimum f for this spp,npp,inc,h
61                     anglesp_f(kf)=distance_opt_f(in(k),h(j),p,spp,typeWD,F(kf));
62                     %Angle between satelllites of different planes
63                 end
64
65                 [anglesp,kf_min]=min(anglesp_f);
66
67                 if (angles ≤ 2*Bo) && (anglesp ≤ Bo)
68
69                     q=q+1;
70                     result(:,q)=[h(j) in(k) spp p spp*p F(kf_min)];
71                     fprintf(['Solution %g: h=%gkm - i=%g - '...
72                             '%gspp - %gplanes - %g sats - f=%g'],q,result(:,q))
73                     nplanes_win=1;
74                     break %Stop iteration in satelllites per plain
75
76                 end
77
78             end
79             if nplanes_win %If a combination with that num of planes
80                 nplanes_win=0; %works, then jump to the next inclination
81                 break
82             end
83         end
84     end
85 end
86
87 %% Optimum Results
88 % Minimum number of cubesats
89 numsat = result (5,1:q);

```

```

90 [minsat,index] = min(numsat);
91
92 disp(sprintf('\nMINIMUM SATELLITES\nheight: %d', result(1,index)))
93 disp(sprintf('inclination: %d',result(2,index)))
94 disp(sprintf('number of satellites in each plane: %d',result(3,index)))
95 disp(sprintf('number of planes: %d',result(4,index)))
96 disp(sprintf('minimum number of satellites: %d',result(5,index)))
97
98 % Minimum number of planes
99 numplane = result(4,1:q);
100 [minplane,index]= min(numplane);
101
102 disp(sprintf('\nMINIMUM PLANES\nheight: %d', result(1,index)))
103 disp(sprintf('inclination: %d',result(2,index)))
104 disp(sprintf('number of satellites in each plane: %d',result(3,index)))
105 disp(sprintf('number of planes: %d',result(4,index)))
106 disp(sprintf('minimum number of satellites: %d',result(5,index)))
107
108 % Price Optimization
109 TCost = plaunch*result(4,:) + psat*result(5,:); TCost = TCost/(1e6);
110 [CostOpt,index] = min(TCost);
111 height = result(1,:);
112
113 disp(sprintf('\nMINIMUM PRICE\nprice[M]: %d', TCost(index)))
114 disp(sprintf('height: %d', result(1,index)))
115 disp(sprintf('inclination: %d',result(2,index)))
116 disp(sprintf('number of satellites in each plane: %d',result(3,index)))
117 disp(sprintf('number of planes: %d',result(4,index)))
118 disp(sprintf('minimum number of satellites: %d',result(5,index)))
119
120 fprintf('\n\n MORE THAN ONE SOLUTION COULD HAVE THE SAME MIN NUMBER\n')
121
122 %% Plots
123
124 x=0:q-1;
125 plot(x,numplane,x,numsat,x,TCost,x,height)
126 strValues = strtrim(cellstr(num2str([x(:) numplane(:)],'(%d, %d)')));
127 text(x,numplane,strValues,'VerticalAlignment','bottom');
128 strValues = strtrim(cellstr(num2str([x(:) numsat(:)],'(%d, %d)')));
129 text(x,numsat,strValues,'VerticalAlignment','bottom');
130 strValues = strtrim(cellstr(num2str([x(:) TCost(:)],'(%d, %.1f)')));
131 text(x,TCost,strValues,'VerticalAlignment','bottom');
132 strValues = strtrim(cellstr(num2str([x(:) height(:)],'(%d, %d)')));
133 text(x,height,strValues,'VerticalAlignment','bottom');
134 legend('number of planes','number of satellites','Total Cost (M)')
135
136
137
138 toc

```

```

1 function [Bo] = visibleangle (h, eo)
2
3 d=zeros(length(h),length(eo));
4 nsats_opt=d;
5 Re = 6371;
6
7 for i = 1:length(h)
8     for j = 1:length(eo)
9
10         d(i,j)= Re * ( sqrt( ( (h(i)+Re)/Re )^2 - ( cosd(eo(j)) )^2 ) - sind(eo(j)) );
11         Bo = asind( d(i,j)*cosd(eo(j)) / (h(i)+Re) );
12
13     end
14 end
15
16 end

```

```

1 function [anglesp] = distance_opt_f (i,a,p,s,typeWD,F)
2 %% Data input
3
4 % i: Inclination angle [deg]
5 % a: Height of the orbit [km]
6 % p: Number of planes
7 % s: Number of satellites per plane
8
9 %% Walker Delta type of geometry
10 WD=[180 225 360];
11 degreegenerate = WD(typeWD);
12 % Walker Delta 360 deg, SemiWalker Delta 180 deg,
13 % other constellations range between 180 and 360
14
15 m = degreegenerate/180;
16 % generates Walker Delta Constellation
17 % (m=2 -> 2*180 generated constellation),
18 % Semi Walker (m=1 -> 1*180) or other 1<m<2.
19
20 % Phasing between adjacent planes
21 %f = F*p*cosd(i); % parameter defined graphically
22 f = F; % Whereas F is a number 0 < f < (p-1)
23     % According to Astrodynamics notes
24
25 REarth = 6371; % [km]
26 h = (REarth+a)/REarth; %radius of the orbit
27 % (in terms of the sphere radius) R=h*r;
28
29 %% Distribution of Coordinates of the satellites
30 Omega = zeros(1,p);
31 nu = zeros(s,p);

```



```

32 angle = zeros(1,p);
33 X = zeros(3,s,p);
34
35 for j = 1:p
36     Omega(j) = m*pi*(j-1)/p;
37     angle(j) = f*2*pi*(j-1)/(s*p); %Phasing due to f between planes
38     for k = 1:s
39         % True anomalies of the s satellites
40         nu(k,j) = 2*pi*(k-1)/s+angle(j);
41         X(:,k,j) = cartesian(h,0,i,Omega(j),0,nu(k,j));
42         % h[adim] --> X[adim]
43     end
44 end
45
46 %% Fast comprovation
47 % Does the adjacent phasing exceed pi
48 % THIS MEANS
49 % The minimum might be between
50
51
52 %% Angle between different satellites
53
54 c=zeros(s,p); %General aproach. Where is the minimum angle?
55
56 for j=2:p
57     for i=1:s-1
58
59         % We need to assess the angles between the last and first plane
60         % Specially in Semi-Walker configurations. That's why we add this
61         % auxiliar variables.
62
63         p1=j;
64         if j==1
65             p2=p;
66         else
67             p2=j-1;
68         end
69
70         % Angles a2 and a3 could be used to increase de reliability of the
71         % constellation, allowing greater possibilities of communication
72         % between planes. (not only between (i,p1) and (i,p2)
73
74         % Angle between satellite (i,p1) and satellite (i,p2)
75         a1=acosd((X(:,i,p1)'*X(:,i,p2))/(norm(X(:,i,p1))*norm(X(:,i,p2))));
76
77         % Angle between satellite (i,j) and satellite (i+1,j-1)
78         a2=acosd((X(:,i,p1)'*X(:,i+1,p2))/(norm(X(:,i,p1))*norm(X(:,i+1,p2))));
79
80         % Angle between satellite (i,j) and satellite (i-1,j-1)
81         a3=acosd((X(:,i,p1)'*X(:,i-1,p2))/(norm(X(:,i,p1))*norm(X(:,i-1,p2))));

```

```

82
83     % Among the computed angles, choose the minimum
84     [angle(i,j),c(i,j)]=min([a1 a2]);
85
86     % Why minimum? Then we know that at least it will be able to
87     % cover the angle with one of the two adjacent close nodes.
88
89     end
90 end
91
92 anglesp = max(max(angle));
93 %Maximum angle between satellites of different planes

```

1.5 Walker Delta Testing

```

1  %----ASTREA CONSTELLATION----
2  %PROJECTS - 220028
3  %Aerospace Engineering Barchelor's Degree
4  %ESEIAAT - UPC
5  %Autumn 2016-2017
6
7  % ORBIT DESIGN TEAM
8  % ORBIT SMART WALKER-DELTA
9
10 % This routine compute the WD possible configurations that give global
11 % coverage by having the distance between planes matches the planes
12 % rotation
13
14 p=[12 15 18];
15 S=360./p; S=S*pi/180;
16
17 re=6378.01e3;
18 u=3.986012e14;
19 J2=1.0826e-3;
20
21 we=2*pi/(24*3600);
22
23 syms a;
24 inc=75; inc=inc*pi/180;
25 a_sol=zeros(1,length(inc));
26
27 for i=1:length(inc)
28     dOmega=-1.5*J2*(re/a)^2*sqrt(u/a^3)*cos(inc(i));
29     P0=2*pi*sqrt(a^3/u);
30     Pn=P0*(1-1.5*J2*(re/a)^2-0.75*J2*(4-5*sin(inc(i))^2)*(re/a)^2);
31     eq=S(2)-Pn*(we-dOmega);

```

```

32     sol=vpasolve(eq,re*1.15);
33     if imag(sol)==0
34         a_sol(i)=sol;
35     end
36 end
37
38 plot(inc*180/pi, (a_sol-re)/1000)
39 h=(a_sol-re)/1000
40
41 %% Satellites computation
42
43 emin=20*pi/180;
44 Lstreet=5*pi/180;
45
46 rho=re/(re+h*1000);
47 etha=asin(sin(rho)*cos(emin));
48 Lmax=pi/2-emin-etha;
49 S=2*acos(cos(Lmax)/cos(Lstreet));
50 N=ceil(2*pi/S)

```

1.6 Orbit Plotter

```

1  %----ASTREA CONSTELLATION----
2  %PROJECTS - 220028
3  %Aerospace Engineering Barchelor's Degree
4  %ESEIAAT - UPC
5  %Autumn 2016-2017
6
7  % ORBIT DESIGN TEAM
8  % ORBIT PLOTTER - Influential phenomena computation
9
10 % This matlab routine plots the position of the satellites of a Walker
11 % Delta / SemiWalker Delta or other generated configurations.
12
13 clc, clear, close all;
14 %-----
15 %% 0. Data Input
16 %add the subfolder programs to the current path
17 addpath(genpath('./auxiliar'));
18
19 %_____Physical input_____
20
21 global h, h = 542; % [km]
22 degreegen = 225; % Walker Delta 360 deg, SemiWalker Delta 180 deg,
23 % other constellations range between 180 and 360 deg
24 in = degtorad(72); % inclination of the planes

```

```

25 s = 21;           % satellites per plane
26 p = 9;           % number of planes
27 f=8;             % f = 1.25*p*cos(i); % parameter defined graphically
28 eo=degtorad(20); % elevation angle [rad]
29
30 %___Numerical Input_____
31 r = 1;           %radius of the sphere
32 n = 50;          %number of cells of the sphere generated
33 N = 50;          %number of points used to print the orbit
34 nn = 100;        %number of points used to compute the footprint
35
36 %___Physical Constants_____
37 global Re, Re = 6371; %Earth's Radius [Km]
38 %% 1. Preprocessing
39
40 global m, m = degreeen/180;
41         % generates Walker Delta Constellation (m=2 -> 2*180)
42         % generated constellation), Semi Walker (m=1 -> 1*180)
43         % or other 1<m<2.
44
45 global a, a = (Re+h)/Re;
46         %radius of the orbit (in terms of the sphere radius) R=h*r;
47
48 %% 2. Main Process
49
50 %compute the sat RAAN and some sat. parameters (Laura)
51 Omega = zeros(1,p);
52 nu = zeros(s,p); %true anomaly of the satellites
53 angle = zeros(1,p); %angle between satellites
54 for i = 1:p
55     Omega(i) = m*pi*(i-1)/p; %orbits
56
57     angle(i) = f*2*pi*(i-1)/(s*p); %sats
58     for k = 1:s
59         nu(k,i) = 2*pi*(k-1)/s+angle(i);
60     end
61 end
62
63 % Compute the satellite coordinates (Laura)
64 X = zeros(3,s,p);
65 for i = 1:p
66     for k = 1:s
67         X(:,k,i) = cartesian(a,0,in,Omega(i),0,nu(k,i));
68     end
69 end
70 sCOOR=reshape(X,3,p*s)'; %put the data in a more friendly structure
71
72 %compute the radius of the footprint
73 %SATELLITE_FOOTPRINT_COMPUTATION.m
74 d=Re*(( (h+Re)/Re)^2-cos(eo)^2)^.5-sin(eo);

```

```

75 Bo=asin(d*cos(eo)/(h+Re));
76 S=2*pi*Re^2*(1-cos(Bo));
77 R=(S/pi)^.5;
78
79 %create the footprint of each satellite
80 SATfp=[];
81 for i=1:s*p
82     %unitary vector pointing to the sat from the earth's center
83     u=-sCOORD(i,:)/norm(sCOORD(i,:));
84     %Generate the circle coordinates (2D)
85     [satfp(:,1),satfp(:,2),satfp(:,3)]=circle3D(sCOORD(i,:),u, R/Re, nn);
86     satfp=satfp+repmat(u*h/Re,nn,1); %move the circle to be tg to Earth
87
88     for j=1:size(satfp,1)
89         v=sCOORD(i,:)-satfp(j,:); v=v/norm(v);
90         %unitary vector sat->footprint
91
92         %verify if the footprint intersect with the earth's surface
93         tol=satfp(j,1)^2+satfp(j,2)^2+satfp(j,3)^2-r^2;
94         while tol>1e-2
95             satfp(j,:)=satfp(j,:)-0.01*v;
96             tol=satfp(j,1)^2+satfp(j,2)^2+satfp(j,3)^2-r^2;
97         end
98     end
99     %store the footprint
100     SATfp=[SATfp; satfp];
101 end
102
103
104 %% 3. Plots
105
106 %3D plot
107 [COORD]=Orbitplot3D( in,r, X,n, SATfp, nn );
108 %Ground track
109 x=zeros(size(COOR,1),size(COOR,2)/3); y=x; z=x;
110 for i=1:size(COOR,2)/3
111     x(:,i)=COORD(:,3*(i-1)+1);
112     y(:,i)=COORD(:,3*(i-1)+2);
113     z(:,i)=COORD(:,3*(i-1)+3);
114 end
115 GroundTrackPlot( x, y, z, sCOORD, SATfp, nn);

```

```

1 function [x,y,z]=circle3D(center,normal,radius,n)
2 %This matlab function generates the points of a circle normal to a 3D
3 %vector
4 %In:
5 %   center: center of the circle (x, y, z)
6 %   normal: vector normal to the circle
7 %   radius: radius of the circle

```

```

8 %   n: number of points (small number can make the circle not smooth)
9 %Out:
10 %   x, y z: vectors containing the coord.
11 %-----
12
13 theta=linspace(0,2*pi,n);
14 v=null(normal);
15 points= repmat(center',1,size(theta,2))+...
16     radius*(v(:,1)*cos(theta)+v(:,2)*sin(theta));
17 x=points(1,:); y=points(2,:); z=points(3,:);
18 end

```

```

1 function [COOR] = Orbitplot3D( in,r, X,n, sCOOR, N )
2 %Orbitplot3D is a function that creates a 3D representation of a
3 %constellation and plots the satellites. It can be also configured to plot
4 %their footprint
5 %In:
6 %   in: inclinaton angle [rad]
7 %   r: sphere radius [-]
8 %   X: Coordinates of the sats (3,spp,p)
9 %   n: number of points used to draw thhe sphere
10 %   sCOOR
11 %   COOR: coordinates of the footprints
12 %   N: number of points used to compute the footprint
13 %-----
14 p=size(X,3);
15 s=size(X,2);
16 global a m;
17
18 % Earth
19 %Create the earth instance as an sphere
20 [x,y,z]=sphere(n); %Generate the sphere
21 surf(r*x,r*y,r*z,'FaceColor','interp','EdgeColor','none'); %plot the sphere
22 colormap(linspace(0,1,n)'.*[0 0.6 1]); %custom colormap (blue tones);
23 axis equal;
24 box off;
25 hold on;
26 rotate3d on;
27
28 % Orbits
29 %Generate the circle coordinates (2D)
30 coor=zeros(N+1,3);
31 for i=1:N
32     coor(i,:)=[a*r*cos(2*i*pi/N), a*r*sin(2*i*pi/N), 0];
33 end
34 coor(N+1,:)=coor(1,:);
35
36 %Rotation Matrices
37 Ry=[ cos(in) 0 sin(in)

```

```

38         0      1      0
39     -sin(in) 0 cos(in)];
40
41 Rz= @(Omega) [-sin(Omega) cos(Omega) 0
42               cos(Omega) sin(Omega) 0
43               0          0          1]; % he modificat aquesta matriu
44
45 coor=coor*Ry; %Rotate the circle in the y axis (inclination)
46
47 %rotate the plane to get the other planes
48 COOR=zeros(N+1,3*p);
49 for i=1:p
50     COOR(:,(1+(i-1)*3):(3+(i-1)*3))=coor*Rz(m*(i-1)*pi/p);
51 end
52 for i=1:p
53     plot3(COOR(:,1+(i-1)*3),COOR(:,2+(i-1)*3),COOR(:,3+(i-1)*3),...
54           'linewidth',1.5);
55 end
56
57 % Satellites (Laura)
58 for i = 1:p
59     for k = 1:s
60         scatter3(X(1,k,i),X(2,k,i),X(3,k,i),25,'square','k','filled');
61         hold on;
62     end
63 end
64
65 %Plot also the footprint if the number of input variables is bigger than 4
66 if nargin > 4
67     for i=1:p*s
68         b=1+N*(i-1); c=N+N*(i-1);
69         fill3(sCOOR(b:c,1),sCOOR(b:c,2),sCOOR(b:c,3),[1 0 0],...
70             'FaceAlpha','.3','LineStyle','none');
71     end
72 end
73 end

```

```

1 function [] = GroundTrackPlot( x, y, z, coor, fp_coor, N)
2
3 %GroundTrackPlot is used to plot the ground
4 %track of the constellation orbits
5 %given their coordinates (cartesian).It has two variants, only introducing
6 %the first 3 inputs will plot the orbits and if you introduce all the
7 %inputs it plots the satellites and their footprint.
8 %In:
9 %   x: matrix containing the x coordinate of the orbits to plot
10 %   y: matrix containing the y coordinate of the orbits to plot
11 %   z: matrix containing the z coordinate of the orbits to plot
12 %   ---

```

```

13 %   coor: satellite coordinates (nx3)
14 %   fp_coor: satellite footprint coordinates ((n*nn)x3)
15 %   N: number of points used to compute the footprint
16 %-----
17 p=size(x,2); %n of planes
18 s=size(coor,1)/size(x,2); %n of spp
19
20 %% Set the background image for the ground track plot
21 figure('name','Constellation Ground Track');
22 tit=[ 'Num. of planes: ' num2str(p) ' Num. of spp: ' num2str(s)];
23 ground='World-satellite-map.png';
24 im=imread(ground);
25 imagesc([-180,180],[-90,90],im);
26 grid on;
27 axis([-180 180 -90 90]);
28 title(tit);
29 hold on;
30 %set the plot lines color
31 c = ['r','y','m','g','w']; C=[];
32 for i=1:ceil(size(x,2)/5)
33     C=[C c];
34 end
35
36 %% Main process
37 for i=1:size(x,2)
38     %convert the cartesian coordinates to spherical
39     [lon,lat]=cart2sph(x(:,i),y(:,i),z(:,i));
40     %get the coordinates of the orbit in spherical coord.
41     lon=radtodeg(lon); %pass to degrees
42     lat=radtodeg(lat);
43     Lon=[lon(end); lon(1:end-1)];
44
45     %Plot the sat ground track
46     [j,j]=max(abs(lon-Lon));
47     plot(lon(1:j-1),lat(1:j-1),C(i),'linewidth',1.2);
48     plot(lon(j:end),lat(j:end),C(i),'linewidth',1.2);
49 end
50
51 if nargin > 3
52     [slon,slat]=cart2sph(coor(:,1),coor(:,2),coor(:,3));
53     %get the coordinates of the sats in spherical coord.
54     slon=radtodeg(slon); %pass to degrees
55     slat=radtodeg(slat);
56
57     %plot the sats
58     scatter(slon,slat,30,'square','w','filled');
59
60     %get the coordinates of the sats footprint in spherical coord.
61     [fplon,fplat]=cart2sph(fp_coor(:,1),fp_coor(:,2),fp_coor(:,3));
62     fplon=radtodeg(fplon); %pass to degrees

```



```

63     fplat=radtodeg(fplat);
64
65     %Plot their footprint
66     for i=1:p*s
67         b=1+N*(i-1); c=N+N*(i-1);
68         fpLon=[fplon(c); fplon(b:c-1)];
69
70         %get the maximum values (plotting purposes)
71         [sorted, I]=sort(abs(fplon(b:c)-fpLon));
72         M=sorted(end-1:end);
73         j=I(end-1:end); j(2)=j(2)-1;
74
75         if M(1) > 300
76             if j(1) < j(2)
77                 auxlon=fplon(b+j(1):b+j(2)-1); %negative lon. part
78                 auxlat=fplat(b+j(1):b+j(2)-1);
79
80                 auxlon2=[fplon(b+j(2):c); fplon(b:b+j(1)-2)]; %+ lon. part
81                 auxlat2=[fplat(b+j(2):c); fplat(b:b+j(1)-2)];
82             else
83                 auxlon2=fplon(b+j(2):b+j(1)-2); %negative lon. part
84                 auxlat2=fplat(b+j(2):b+j(1)-2);
85
86                 auxlon=[fplon(b+j(1):c); fplon(b:b+j(2)-1)];
87                 %positive lon. part
88                 auxlat=[fplat(b+j(1):c); fplat(b:b+j(2)-1)];
89             end
90
91             fill(auxlon,auxlat,[1 0 0],'FaceAlpha','.2','LineStyle','none');
92             fill(auxlon2,auxlat2,[1 0 0],'FaceAlpha','.2','LineStyle','none');
93         else
94             fill(fplon(b:c),fplat(b:c),[1 0 0],...
95                 'FaceAlpha','.3','LineStyle','none');
96
97         %%%%%%%%%%%
98         pause(.05); %
99         %%%%%%%%%%%
100     end
101
102
103     end
104 end
105 end

```

```

1 % This function converts kepler orbit elements to cartesian coordinates
2
3 % a: semimajor axis
4 % e: eccentricity
5 % i: inclination [rad]

```

```

6 % Omega: longitude of the ascending node [rad]
7 % w: Argument of periapsis [rad]
8 % nu: True anomaly [rad]
9
10 function [X] = cartesian(a,e,i,Omega,w,nu)
11
12 % Position in cylindrical coordinates
13 r = a*(1-e^2)/(1+e*cos(nu));
14
15 % Position components
16 X = [r*(cos(Omega)*cos(w+nu)-sin(Omega)*sin(w+nu)*cos(i));
17      r*(sin(Omega)*cos(w+nu)+cos(Omega)*sin(w+nu)*cos(i));
18      r*sin(i)*sin(w+nu)];
19
20 end

```

1.7 Perturbations

```

1 %----ASTREA CONSTELLATION----
2 %PROJECTS - 220028
3 %Aerospace Engineering Barchelor's Degree
4 %ESEIAAT - UPC
5 %Autumn 2016-2017
6
7 % ORBIT DESIGN TEAM
8 % ORBIT PERTURBATIONS - Influential phenomena computation
9
10 %PROBLEM:
11 % Given: - Initial orbital parameters
12 % Compute the final orbital parameters after each orbit
13 %clear; clc;
14
15 %% 1.A Input data physical constants
16 % PHYSICAL CONSTANTS AND PARAMETERS
17 %Physical data
18 RE=6.378e6; %Earth Radius [m]
19 u=3.986e14; %GM Earth
20 J2=1082.6e-5; %J2 coefficient;
21
22 %% 1. Input data Orbit parameters
23 % SPACECRAFT DATA
24 m=4; % Satellite mass [kg]
25 Cd=2.2; % Drag Coefficient
26 A=0.1; % Satellite surface [m^2] --> 3U pointing to Earth
27 Bc=m/(Cd*A); % Ballistic coefficient of the satellite;
28

```

```

29 % ORBIT DATA
30 H0=542e3;      % Altura inicial [m]
31 e0=0.0001;     % Excentricity
32 i0=72*pi/180;  % Inclination [deg]
33 a0=RE+H0;
34
35 % INITIAL PARAMETERS
36 P=2*pi*(a0^3/u)^.5; % Orbit Period [s]
37 v=sqrt(u/a0);     %satellite velocity [m/s]
38 n=(2*pi/P)*86400; %number of revolutions/day
39
40
41 % SIMULATION PARAMETERS
42 N=10000000;
43
44 %% 2. Perturbations propagation
45 %Outputs de la funcio Perturbation:
46 %   Pert: matrix (4x5) with the perturbations causes at the rows (J2,Drag,
47 %   Moon,Sun) and the orbital elements in the columns (a,e,i,omega,Omega)
48 %       a  e  i  w  Omega
49 %   [                ]J2
50 %   [                ]Drag
51 %   [                ]Moon
52 %   [                ]Sun as 3rd Body
53
54
55
56 t=zeros(1,N); t(1)=0;
57 a=t; a(1)=a0;
58 e=t; e(1)=e0;
59 i=t; i(1)=i0;
60
61 w_sun=zeros(1,N-1); w_moon=w_sun; w_J2=w_sun;
62 Om_sun=w_sun; Om_moon=w_sun; Om_J2=w_sun;
63
64 w0=0; Omega0=0;
65 w=t; w(1)=w0;
66 Omega=t; Omega(1)=Omega0;
67
68 tic
69 for n=2:N
70
71     a0=a(n-1);
72     e0=e(n-1);
73     i0=i(n-1);
74     w0=w(n-1);
75     Omega0=Omega(n-1);
76
77     P=2*pi*(a0^3/u)^.5; % Orbit Period [s]
78

```

```

79     [Pert_i rho]=Perturbation(a0,e0,i0,Bc);
80
81     a(n)=a0+sum(Pert_i(:,1));
82     e(n)=e0+sum(Pert_i(:,2));
83     i(n)=i0+sum(Pert_i(:,3));
84     w(n)=w0+sum(Pert_i(:,4));
85     Omega(n)=Omega0+sum(Pert_i(:,5));
86
87     w_sun(n-1)=Pert_i(4,4);
88     w_moon(n-1)=Pert_i(3,4);
89     w_J2(n-1)=Pert_i(1,4);
90
91     Om_sun(n-1)=Pert_i(4,5);
92     Om_moon(n-1)=Pert_i(3,5);
93     Om_J2(n-1)=Pert_i(1,5);
94
95     % We don't want angles bigger than 360!
96     w(n)=w(n)-360*floor(w(n)/360);
97     Omega(n)=Omega(n)-360*floor(Omega(n)/360);
98
99     if Omega(n)>2*pi
100         Omega(n)=Omega(n)-2*pi;
101     end
102
103     t(n)=t(n-1)+P;
104
105     if a(n)<(RE+180e3)
106         fprintf('Your satellite succesfully burned in the atmosphere.\n\n')
107         break
108     end
109
110 end
111
112 %% Post processing and results plotting
113
114 time=toc;
115 fprintf('Time to do %g iterations = %g s.\n ',n,time)
116
117 figure(1)
118 plot(t(1:n)/(3600*24),(a(1:n)-RE)/1000)
119 ylim([100 500])
120 grid on
121 ylabel('Orbit height [km]')
122 xlabel('Time [days]')
123 tfin=floor(t(n)/(3600*24));
124 days = num2str(floor(tfin/365));
125 titulaso=['Orbit decay in ' num2str(tfin) ' days = ' days 'years'];
126 title(titulaso)
127
128

```

Perturbations

```

129 figure(2)
130 subplot(1,2,1)
131 plot(t(1:n)/(3600*24),w(1:n))
132 grid on
133 ylabel('Perigee Argument [deg]')
134 xlabel('Time [days]')
135 title('Perigee Argument deviation in 100 days')
136 axis([0 100 0 360])
137
138 subplot(1,2,2)
139 plot(t(1:n)/(3600*24),Omega(1:n))
140 grid on
141 ylabel('Ascenent Node Argument [deg]')
142 xlabel('Time [days]')
143 tfin=t(n)/(3600*24);
144 title('Ascendent Node deviation in 100 days')
145 axis([0 100 0 360])
146
147 %% Modulus Analysis
148
149 figure(3)
150
151 subplot(1,2,1)
152 semilogy(t(2:n+1)/(3600*24),abs(w_sun(1:n)),'b','LineWidth',2);
153 hold on
154 semilogy(t(2:n+1)/(3600*24),abs(w_moon(1:n)),'r','LineWidth',2)
155 semilogy(t(2:n+1)/(3600*24),abs(w_J2(1:n)),'k','LineWidth',2)
156 legend('Sun','Moon','J2')
157 xlabel('Time (days)');
158 ylabel('Modulus of Perturbation []');
159 title('Effects on Argument of the Perigee');
160 grid on
161 axis([0 t(n)/(3600*24) 1e-8 1])
162
163 subplot(1,2,2)
164 semilogy(t(2:n+1)/(3600*24),abs(Om_sun(1:n)),'b','LineWidth',2);
165 hold on
166 semilogy(t(2:n+1)/(3600*24),abs(Om_moon(1:n)),'r','LineWidth',2)
167 semilogy(t(2:n+1)/(3600*24),abs(Om_J2(1:n)),'k','LineWidth',2)
168 legend('Sun','Moon','J2')
169 xlabel('Time (days)');
170 ylabel('Modulus of Perturbation []');
171 title('Effects on Argument of Ascendent Node');
172 axis([0 t(n)/(3600*24) 1e-8 1])
173 grid on
174
175 print -depsc ModulusAngulars

```

```

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

```

Perturbations

```

2 %% Perturbations
3 % This matlab is used to compute the perturbations in the classical orbital
4 % elements due to:
5 % J2 (non spherical earth)
6 % Atmospheric drag
7 % Third body (Moon and Sun)
8 %Inputs:
9 % a: semi-major axis [m]
10 % e: eccentricity [-]
11 % i: inclination [rad]
12 % m: mass of the satellite [kg]
13 % Cd: drag coefficient [-]
14 % A: Area of the satellite [m^2]
15 %Outputs:
16 % Pert: matrix (4x5) with the perturbations causes at the rows (J2,Drag,
17 % Moon,Sun) and the orbital elements in the columns (a,e,i,omega,Omega)
18 % incP/revolution
19 % incv/revolution
20 %TO DO:
21 %Les perturbacions degudes a J2 i 3rd body no se si son en pert/dia o no,
22 %cal mirar-ho per tal de tenir totes les pertrubacions escalades igual
23 % ~~~~~
24 %% Previous calculations and PreAllocation
25 %Physical data
26 RE=6.378e6; %Earth Radius [m]
27 u=3.986e14; %GM Earth
28 J2=1082.6e-5; %J2 coefficient;
29
30 P=2*pi*(a^3/u)^.5; % Orbit Period [s]
31 v=sqrt(u/a); %satellite velocity [m/s]
32 n=(2*pi/P)*86400; %number of revolutions/day
33
34
35 % ATMOSPHERIC MODEL
36 %get the density from MSISE model (extracted as a list from
37 %http://omniweb.gsfc.nasa.gov/vitmo/msis_vitmo.html
38 % h[km] rho[kg/m^3]
39 % filename1='msis_26371.lst'; filename2='msis_2226.lst';
40 % formatSpec='%f %f';
41 % size=[2 Inf];
42 % fileID=fopen(filename1,'r');
43 % rho_h=fscanf(fileID,formatSpec,size);
44 % fclose(fileID);
45 % fileID=fopen(filename2,'r');
46 % rho_h=[rho_h fscanf(fileID,formatSpec,size)];
47 % fclose(fileID);
48 % rho=interp1(rho_h(1,:),rho_h(2,:), (a-RE)/1000);
49
50 H = (a - RE)*1e-3;
51 % Compute exospheric temperature [K]

```

Perturbations

```

52     T = 900 + 2.5*(120-70);
53     % Compute effective atmospheric molecular mass [km/K], valid 180<H<500
54     M = 27 - 0.012 * (H - 200);
55     % Compute atmospheric scale height [km]
56     SH = T / M;
57     % Compute atmospheric density [kg/m3]
58     rho = 6E-10 * exp(-(H - 175) / SH);
59
60
61     pert=zeros(4,5);
62
63     %% Computation
64
65     %J2 perturbation_____ PER REVOLUCIO
66     j=1;
67     pert(j,5)=-1.5*n*J2*(RE/a)^2*cos(i)*(1-e^2)^-2;
68     pert(j,4)=0.75*n*J2*(RE/a)^2*(4-5*(sin(i))^2)*(1-e^2)^-2;
69
70     %Atmospheric Drag_____ PER REVOLUCIO
71     %Perturbations/revolution
72     j=2;
73     pert(j,1)=-2*pi*rho*a^2/Bc;
74
75     % incP=-6*pi^2*a^2*rho/(v*Bc);
76     % incv=pi*a*rho*v/Bc;
77
78     %Moon Perturbation_____ /DIA --> /REV
79     j=3;
80     pert(j,5)=-0.00338*cos(i)/n;
81     pert(j,4)=0.00169*(4-5*(sin(i))^2)/n;
82
83     % Fins aqui son /dia,
84     % Canviem unitats + apliquem periode actual
85     pert(j,5)=pert(j,5)*P/(24*3600);
86     pert(j,4)=pert(j,4)*P/(24*3600);
87
88     %Sun Perturbation_____ /DIA --> /REV
89     j=4;
90     pert(j,5)=-0.00154*cos(i)/n;
91     pert(j,4)=0.00077*(4-5*(sin(i))^2)/n;
92
93     % Fins aqui son /dia,
94     % Canviem unitats + apliquem periode actual
95     pert(j,5)=pert(j,5)*P/(24*3600);
96     pert(j,4)=pert(j,4)*P/(24*3600);
97
98
99     end

```

1.8 Orbit Decay

```

1  %----ASTREA CONSTELLATION----
2  %PROJECTS - 220028
3  %Aerospace Engineering Barchelor's Degree
4  %ESEIAAT - UPC
5  %Autumn 2016-2017
6
7  % ORBIT DESIGN TEAM
8  % ORBIT DECAY
9
10 % This routine computes the orbital decay of an spacecraft with time
11 % using the cowell's method.
12 % The differential equations are integrated using "ode45"(Runge-Kutta 4th,
13 % 5th order)
14
15
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17 %%
18 clear all;
19 clc;
20 close all;
21
22 %% Initialization %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
23 global mu mmu smu re rs rm omega J2;
24
25 %_____Physical Variables_____
26 mu=3.986004418e14; % earth gravitational constant [m^3/s^2]
27 mmu=4902.800076e9; %moon gravitational constant [m^3/s^2]
28 smu=132712440040.944e9; %sun gravitational constant [m^3/s^2]
29
30 re=6378136.3; %earth radius [m]
31 rs=696e6; %sun radius [m]
32 rm=1730e3; %moon radius [m]
33
34 omega=7.292115e-5; %earth angular velocity [rad/s]
35 J2=0.001081874; %earth oblateness gravity coeff []
36
37 au=149597870691; %astronomical unit [m]
38 c=2.99792458e8; %[m/s] speed of light
39
40 %_____Numerical variables_____
41 hlim=180; %minimum height for considering the decay [km]
42 dt=15*60; %timestep of the simulation [s]
43 t0=0; %initial time of the sim
44 tf=dt;
45
46 %% User Input %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```



```

47  %__Orbital elements_____
48  a=re+542e3; %[m] semimajor axis
49  e=0.0001; % eccentricity
50  i=deg2rad(72); %[rad] inclination
51  w=0; %[rad] argument of periapsis
52  Omega=0; %[rad] RAAN
53  nu=0; %true anomaly
54
55  date=datetime(2017,2,4); %datetime array [Y, M, D]
56
57  %__Oblateness parameters_____
58  znls=0; %zonals [0-18]
59  tssrls=0; %tesserals [0-18]
60
61  %__Drag perturbation inputs_____
62  Cd=2.2; %Drag coefficient (Typical for sats, various ref.)
63  Adrag=.09; %[m^2] Wet area for the drag
64  m=4.1; %[kg] Cubesat mass
65
66  Bc=Adrag*Cd/m; %Ballistic coeff.
67
68  %__SRP perturbation inputs_____
69  Cr=2; %reflectivity ct.
70  Asrp=.5; %[m^2] Area for the SRP
71
72  % Ws (@Dap) = 1361/(1+0.0334*cos(2*pi*Dap/365));
73  Ws=1361; %W/m^2 approximation for the solar irradiance
74
75  srp=Ws/c*Cr*Asrp/m*au^2; %Solar radiation pressure ct.
76
77  %% Main Process %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
78  display('Computing...');
79  f=@(q) q*((3+3*q+q^2)/(1+(1+q)^3/2)); %richard Battin's function
80
81  aobl=zeros(3,1);
82  adrag=aobl; asun=aobl; amoon=aobl; asrp=aobl;
83
84
85  y=cartesian2(a,e,i,Omega,w,nu); %obtain the state vector (r,V)
86  jdate0=juliandate(date); %compute the julian date
87
88  tic
89  iter=0; tt=[]; xx=[];
90
91  h=(norm(y(1:3))-re)*1e-3;
92  while h>hlim
93      jdate=jdate0+tt/86400;
94      %      GMST=JD2GMST(jdate);
95      %
96      %      %Acceleration due to oblateness

```

```

97 %      aobl=Oblat((t0+tf)/2,y,znls,tssrls,GMST);
98 %
99 %      %Acceleration due to the 3rd body pert.
100 %      rsun=planetEphemeris(jdate,'Earth','Sun');
101 %      %geocentric pos of the Sun
102 %      rmoon=planetEphemeris(jdate,'Earth','Moon');
103 %      %geocentric pos of the Moon
104 %      rsun=rsun'; rmoon=rmoon';
105 %
106 %      rs2s=y(1:3)-rsun; %vector sun-sat
107 %      rm2s=y(1:3)-rmoon; %vector moon-sat
108 %
109 %      qs=dot(y(1:3),(y(1:3)-2*rsun))/dot(rsun,rsun);
110 %      qm=dot(y(1:3),(y(1:3)-2*rmoon))/dot(rmoon,rmoon);
111 %
112 %      asun=-smu/norm(rs2s)^3*(y(1:3)+f(qs)*rsun);
113 %      amoon=-mmu/norm(rm2s)^3*(y(1:3)+f(qm)*rmoon);
114 %
115 %      %Acceleration due to the solar radiation presure.
116 %      usun=rsun/norm(rsun); %unit vector of the pos of the sun
117 %      us=y(1:3)/norm(y(1:3)); %unit vector of the pos of the sat
118 %
119 %      shadow=random('Normal',0.5,.1); %shadow factor
120 %
121 %      asrp=srp*rs2s/norm(rs2s)^3;
122
123 %Acceleration due to the drag force
124 % Compute exospheric temperature [K]
125 T = 900 + 2.5*(100-70);
126 % Compute effective atmospheric molecular mass [km/K], valid 180<H<500
127 M = 27 - 0.012 * (h - 200);
128 % Compute atmospheric scale height [km]
129 SH = T / M;
130 % Compute atmospheric density [kg/m3]
131 rho = 6E-10 * exp(-(h - 175) / SH);
132
133 atmosV=y(4:6)+omega*[y(2); -y(1); 0];
134 %relative vel between sat and atmos
135
136 adrag=-.5*rho*Bc*atmosV*norm(atmosV);
137
138 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
139 acc=aobl+asun+amoon+asrp+adrag;
140
141 ode=@(t,y)[y(4);
142             y(5);
143             y(6);
144             acc(1)-mu/norm(y(1:3))^3*y(1);
145             acc(2)-mu/norm(y(1:3))^3*y(2);
146             acc(3)-mu/norm(y(1:3))^3*y(3);

```

```

147         ]; %function to integrate (6eqn)
148
149         [t,x]=ode45(ode,[t0 tf], y);
150
151         y=x(size(x,1),:);
152         %save one value each 12 hours
153         if mod(t(length(t)),12*3600)==0
154             tt=[tt; t(length(t))]; xx=[xx; y];
155         end
156         y=y';
157
158         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
159         %End of the timestep
160         t0=tf;
161         tf=tf+dt;
162
163         h=(norm(y(1:3))-re)*1e-3;
164
165         iter=iter+1;
166         if mod(iter,300)==0
167             fprintf('Days elapsed: %0.1f \nH=%0.3f km\n',(jdate-jdate0),h);
168         end
169     end
170     toc
171     %% Post Process %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
172     fprintf('The satellite decays in %0.0f days',tf/86400);
173
174     h=zeros(size(xx,1),1);
175     for i=1:size(xx,1)
176         h(i)=(norm(xx(i,1:3))-re)*1e-3;
177     end
178
179     plot(tt/86400,h);

```

```

1  % This function converts kepler orbit elements to cartesian coordinates
2
3  % a: semimajor axis
4  % e: eccentricity
5  % i: inclination [rad]
6  % Omega: longitude of the ascending node [rad]
7  % w: Argument of periapsis [rad]
8  % nu: True anomaly [rad]
9
10 function [y] = cartesian2(a,e,i,Omega,w,nu)
11 global mu;
12 % Position in cylindrical coordinates
13 r = a*(1-e^2)/(1+e*cos(nu));
14
15 % Position components

```

```

16 X = [r*(cos(Omega)*cos(w+nu)-sin(Omega)*sin(w+nu)*cos(i));
17       r*(sin(Omega)*cos(w+nu)+cos(Omega)*sin(w+nu)*cos(i));
18       r*sin(i)*sin(w+nu)];
19
20 A = sin(Omega)*cos(i)*(e*cos(w)+cos(w+nu));
21 v(1)=-sqrt(mu/(a*(1-e^2)))*(cos(Omega)*(e*sin(w)+sin(w+nu))+A);
22
23 B = cos(Omega)*cos(i)*(e*cos(w)+cos(w+nu));
24 v(2)=-sqrt(mu/(a*(1-e^2)))*(sin(Omega)*(e*sin(w)+sin(w+nu))-B);
25
26 v(3)=sqrt(mu/(a*(1-e^2)))*(e*cos(w)+cos(w+nu))*sin(i);
27
28 y=[X;v'];
29 end

```

1.9 Performance Evaluator

```

1  %----ASTREA CONSTELLATION----
2  %PROJECTS - 220028
3  %Aerospace Engineering Barchelor's Degree
4  %ESEIAAT - UPC
5  %Autumn 2016-2017
6
7  % ORBIT DESIGN TEAM
8  % PERFORMANCE EVALUATOR
9
10 clear all;
11
12 %% Input Data
13
14 a=6378.01e3;
15 u=3.986012e14;
16 h=(a+546.5101e3)/a;
17 ap=h*a;
18 J2=1.0826e-3;% J2=0;
19
20 ws=sqrt(u/ap^3); % Satellite mean motion
21 we=2*pi/(24*3600); % Earth mean motion
22 dOmega=-1.5*J2*(a/ap)^2*sqrt(u/ap^3)*cosd(75);
23
24 At=1/6; % [min] % Every 10 seconds
25 t=0:At:60*24; t=t*60; % Time array in seconds
26
27 i=72*pi/180;
28
29 p=9;

```

```

30 spp=21;
31 N=p*spp;
32 %f=floor(0.25*spp);
33 f=0;
34 fprintf('\ni=%gdeg, p=%g, spp=%g, N=%g\n',i*180/pi,p,spp,N);
35
36 latGS=57.5*pi/180;
37 longGS=(17.73-60)*pi/180; longGS=0;
38
39 fprintf('GS Coordinates: Lat=%g, Long=%g \n',latGS*180/pi, longGS*180/pi)
40
41 f1=pi/2-latGS; f2=longGS;
42 R1=[cos(f1) 0 -sin(f1);0 1 0;sin(f1) 0 cos(f1)];
43 R2=[cos(f2) sin(f2) 0; -sin(f2) cos(f2) 0; 0 0 1];
44 RotGS=R1*R2;
45
46 % Rotation Matrix:
47 % From ECEF - To SEZ
48 % Where:
49 % ECEF = Earth Centered Earth Fix - Coordinate System
50 % SEZ = Topocentric Horizon - Coordinate System
51
52 emin=20*pi/180;
53
54 %% Le simulasion
55 %Omega = 0:30:240; Omega=Omega*pi/180;
56 Omega = 0:225/(p-1):225; Omega=Omega*pi/180;
57 %Omega=0:360/p:360-360/p; Omega=Omega*pi/180;
58
59 nu = zeros(spp,p);
60 angle = zeros(1,p);
61 X = zeros(3,spp*p);
62
63 flight_time=zeros(1,1000); % Length of the flyby of a satellite
64 links_at_time=zeros(1,1000); % Number of links when one sat has finished
65 time_end_flyby=zeros(1,1000); % When did this flyby end?
66
67 contact=zeros(1,length(t)); % Number of links
68
69 quality_time=zeros(1,length(t));
70 %If a flyby lasts longer than 3 minutes,
71 % then the previous 3 minutes were successfully covered.
72
73 time_record=zeros(1,N); % Accumulates Timesteps being on the GS
74 before_tracking=zeros(1,N);
75 % Boolean to know which ones are already passing by
76 now_tracking=zeros(1,N); % Boolean to know which ones are now passing by
77 Nflyby=0; % Number of flight paths computed
78
79 for n=1:length(t)

```

```

80
81     % J2 deviation
82     Omegat=Omega+dOmega*t(n);
83
84     % Ground Station Coordinates
85     f1=we*t(n)+longGS;
86     f2=latGS;
87     RGS=cos(f2);
88     XGS=[RGS*cos(f1);RGS*sin(f1);sin(f2)];
89
90     R2=[cos(f1) sin(f1) 0; -sin(f1) cos(f1) 0; 0 0 1];
91     RotGS=R1*R2;
92
93     % Constellation Coordinates
94     nu_t=ws*t(n);    % True anomaly due to time passing by
95     for j = 1:p
96         angle(j) = f*2*pi*(j-1)/(spp*p); %Phasing due to f between planes
97         for k = 1:spp
98             % True anomalies of the s satellites
99             A = spp*(j-1)+k; % Number of the satellite
100            nu(k,j) = 2*pi*(k-1)/spp+angle(j);
101            X(:,A) = cartesian(h,0,i,Omegat(j),0,nu(k,j)+nu_t);
102        end
103    end
104
105    % Contact Evaluation
106    win=0;    % Counter to know number of links
107    now_tracking=zeros(1,N);
108    for sat=1:N
109        X_hor=RotGS*(X(:,sat)-XGS);
110        % fprintf('%g,%g,%g --> |r|=%g\n',X_hor(1),X_hor(2),X_hor(3),norm(X_hor))
111        ang=asin(X_hor(3)/norm(X_hor));
112        if ang >= emin
113            win=win+1;
114            time_record(sat)=time_record(sat)+1;
115            now_tracking(sat)=1;
116        end
117
118        % Flight time computation and reset of this satellite flyby
119        if before_tracking(sat)==1 && now_tracking(sat)==0
120            Nflyby=Nflyby+1;
121            flight_time(Nflyby)=At*time_record(sat);
122
123            links_at_time(Nflyby)=contact(n-1);
124            time_end_flyby(Nflyby)=t(n);
125
126            % THE KEY QUESTION
127            % Was this flyby useful?
128            if flight_time(Nflyby)>3
129                start=n-time_record(sat);

```

```

130             finish=n;
131             quality_time(start:finish)=quality_time(start:finish)+1;
132         end
133         time_record(sat)=0;
134     end
135 end
136 before_tracking=now_tracking;
137 contact(n)=win;
138 end
139
140 %% Post-Processing
141 % PLOT 1: NUMER OF LINKS
142 figure(1)
143 title('Links vs Time')
144 plot(t/(3600),contact)
145 axis([min(t)/(3600) max(t)/(3600) 0 max(contact)+1]);
146 fails=length(find(contact==0));
147 ratio=fails/length(contact);
148 fprintf('Links on GS           : %g percent of the time\n',(1-ratio)*100)
149
150 % PLOT 2: LENGTH OF THE LINKS
151 figure(2)
152 plot(flight_time(1:Nflyby));
153 titola=['Length of the ' num2str(Nflyby) ' flyby s. Mean time = '...
154         num2str(mean(flight_time(1:Nflyby))) 'min'];
155 title(titola)
156 ylabel('Length (minutes)')
157 xlabel('Contact')
158 %xlim([1 Nflyby])
159
160 % PLOT 3: ANALYSIS OF THE FLYBYS
161 figure(3)
162 index=1:Nflyby; % X variable to the following plots
163 plot(time_end_flyby(index)/3600,flight_time(index),...
164       time_end_flyby(index)/3600,links_at_time(index),...
165       t/3600,quality_time);
166
167 titola='Flybys Analysis';
168 title(titola)
169
170 legend('Length of flybys','Links by end of flyby',...
171        'Num of sats @flybys longer than 3 min')
172 xlabel('Time (h)')
173
174 epic_wins=length(find(quality_time>=1));
175 ratio_covered=epic_wins/length(t);
176 fprintf('Quality flybys on GS: %g percent of the time\n',ratio_covered*100)
177 fprintf('Mean flyby time = %g minutes\n',mean(flight_time(1:Nflyby)))
178
179

```

Thrust

```

180 %% MAXIMUM GAP SEARCH
181 gap=0;
182 at_gap=0;
183 gap_record=zeros(1,100);
184 ngap=0;
185
186 for n=1:length(t)
187     if quality_time(n)==0
188         at_gap=1;
189         gap=gap+1;
190     end
191
192     if quality_time(n)>0 && at_gap==1
193         at_gap=0;
194         ngap=ngap+1;
195         gap_record(ngap)=gap*At;
196         gap=0;
197     end
198 end
199
200 max_gap=max(gap_record);
201 fprintf('Maximum gap = %g minutes\n',max_gap)
202 fprintf('Number of gaps = %g\n\n',ngap)

```

1.10 Thrust

```

1  %----ASTREA CONSTELLATION----
2  %PROJECTS - 220028
3  %Aerospace Engineering Barchelor's Degree
4  %ESEIAAT - UPC
5  %Autumn 2016-2017
6
7  % ORBIT DESIGN TEAM
8  % THRUST
9
10 % This function computes the  $\Delta V$  and propellant mass necessary to
11 % maintain an orbit between to heights
12
13 %function [mp, $\Delta V$ ,tHoh] = thrust(hmax,hmin,ms,Isp,Thr)
14
15 % This function computes the  $\Delta V$  and propellant mass necessary to
16 % maintain an orbit between to heights
17
18 % Input variables:
19 % - hmax: maximum height [m]
20 % - hmin: minimum height [m]

```



```

21 % - ms: dry mass of the spacecraft [kg]
22 % - Isp: specific impulse of the spacecraft [s]
23 % - Thr: thrust of the spacecraft [N]
24
25 % Output variables:
26 % - mp: array of propellant mass necessary for every Hohmann transfer [kg]
27 % - ΔV: array of ΔVs necessary for every Hohmann transfer [m/s]
28 % - tHoh: array of time necessary to do a Hohmann transfer [s]
29
30 % Proposed values:
31 hmax = 550e3;
32 hmin = hmax-8;
33 ms = 3.95;           % Dry mass [kg]
34 Isp = 2150;
35 Thr = 100e-6;
36
37 %% Data
38
39 % PHYSICAL CONSTANTS AND PARAMETERS
40 RE = 6.378e6; %Earth Radius [m]
41 mu = 3.986e14; %GM Earth
42 g0 = 9.81;
43
44 % SPACECRAFT DATA
45 m = 4;           % Satellite mass [kg]
46 Cd = 2.2;       % Drag Coefficient
47 A = 0.1*0.3;    % Satellite surface [m^2] --> 3U pointing to Earth
48
49 % ORBIT DATA
50 H0 = hmax;      % Altura inicial [m]
51 E0 = 0.01;     % Excentricity
52 IO = 80*pi/180; % Inclination [deg]
53 A0 = RE+H0;
54
55 % SIMULATION PARAMETERS
56 N=100000;
57 M = 10000;
58
59 %% 2. Perturbations propagation
60
61 % Creation of the matrices
62 temp=zeros(M,N);
63 temp(1,1)=0;
64 H = zeros(M,N);
65 H(1,1) = H0;
66 % ΔV = zeros(2,N);
67
68 % Assign initial values
69 W0=0; OMEGA0=0;
70 a=zeros(1,N); a(1)=A0;

```

```

71 e=a; e(1)=E0;
72 i=a; i(1)=I0;
73 t=a; t(1)=0;
74 w=a; w(1)=W0;
75 Omega=a; Omega(1)=OMEGA0;
76
77 for j = 1:M
78
79     n = 1;
80     while a(n) ≥ (RE+hmin)
81
82         n = n+1;
83
84         a0=a(n-1);
85         e0=e(n-1);
86         i0=i(n-1);
87         w0=w(n-1);
88         Omega0=Omega(n-1);
89
90         P=2*pi*(a0^3/mu)^.5; % Orbit Period [s]
91
92         Bc = m/(Cd*A); % Ballistic coefficient of the satellite
93         Perti=Perturbation(a0,e0,i0,Bc);
94
95         a(n)=a0+sum(Perti(:,1));
96         e(n)=e0+sum(Perti(:,2));
97         i(n)=i0+sum(Perti(:,3));
98         w(n)=w0+sum(Perti(:,4));
99         Omega(n)=Omega0+sum(Perti(:,5));
100
101         % We don't want angles bigger than 360!
102         w(n)=w(n)-360*floor(w(n)/360);
103         Omega(n)=Omega(n)-360*floor(Omega(n)/360);
104
105         if Omega(n)>2*pi
106             Omega(n)=Omega(n)-2*pi;
107         end
108
109         t(n)=t(n-1)+P;
110
111     end
112
113     % Hohmann
114     [ΔV1,ΔV2,mpit,tHohit] = Hohmann(a(n)-RE,hmax,m,Isp);
115     m = m-mpit;
116
117     if m≤ms
118         break
119     end
120

```

Thrust

```

121     ΔV(1,j) = ΔV1; % first row of the column -> ΔV1
122     ΔV(2,j) = ΔV2; % second row of the column -> ΔV2
123     mp(j) = mpit;
124     tHoh(j) = tHohit;
125
126     a(1)=A0;
127     e(1)=E0;
128     i(1)=I0;
129     w(1)=W0;
130     Omega(1)=OMEGA0;
131     altura = a-RE;
132
133     H(j,:) = altura;
134     temp(j,:) = t;
135
136 end
137
138 %% Post process
139
140 mfr = Thr/(g0*Isp);
141
142 for i = 1:N
143     if H(1,i) ≤ 0
144         break
145     end
146 end
147 for j = 1:M
148     if H(j,1) ≤ 0
149         break
150     end
151 end
152 H = H(1:(j-1),1:(i-1));
153
154 for i = 2:N
155     if temp(1,i) ≤ 0
156         break
157     end
158 end
159 for j = 2:M
160     if temp(j,2) ≤ 0
161         break
162     end
163 end
164 temp = temp(1:(j-1),1:(i-1));
165 for j = 2:size(temp,1)
166     temp(j,:) = temp(j,:)+tHoh(j-1)+temp(j-1,size(temp,2));
167 end
168
169 % Convert matrix to vector
170 H = reshape(H.',1,size(H,1)*size(H,2));

```

Thrust

```

171 temp = reshape(temp.',1,size(temp,1)*size(temp,2));
172
173 figure(1)
174 plot(temp/(3600*24),H/1000)
175 grid on
176 ylabel('Orbit height [km]')
177 xlabel('Time [days]')
178 tfin=floor(temp(length(temp))/(3600*24));
179 days=num2str(floor(tfin/365));
180 titulaso=['Orbit decay in ' num2str(tfin) ' days = ' days ' years'];
181 title(titulaso)
182
183 %% Final calculations
184
185 % % Perimeter of the ellipse
186 % r1 = RE+hmax; % semimajor axis
187 % r2 = RE+hmin; % semiminor axis
188 % h = (r1-r2)^2/(r1+r2)^2;
189 % % Ramanujan approximation
190 % C = pi*(r1+r2)*(1+3*h/(10+sqrt(4-3*h)));
191 %
192 % % Check if ΔV is possible for the given thruster
193 % OK = true;
194 % for j = 1:length(mp)
195 %     for i = 1:2
196 %         mfrnecessary = ΔV(i,j)*mp(j)/(C/2);
197 %         if mfrnecessary>mfr
198 %             OK = false;
199 %         end
200 %     end
201 % end
202 %
203 % if OK ==true
204 %     fprintf('The trajectory is possible\n\n');
205 % else
206 %     fprintf('Trajectory not possible\n\n');
207 % end
208 if length(mp)<M
209     fprintf('There is still propellant left ._. \n\n')
210 end

```

```

1 function [ΔV1,ΔV2,mp,t] = Hohmann(hinicial,hfinal,m,Isp)
2
3 % Hohmann transfer orbit between two circular orbits
4 % - hinicial: height of the first orbit [m]
5 % - hfinal: height of the second orbit [m]
6 % - m: mass of the satellite (total) [kg]
7 % - Isp: Specific impulse [s]
8 % - mp: fuel mass [kg]

```

```

9 % - t: time needed to do the maneuver [s]
10
11 mu = 3.986004418e14; % Standard gravitational parameter (Earth)
12 REarth = 6.371e6; % [m]
13 g0 = 9.81; % Earth's gravity [m/s^2]
14
15 % First orbit
16 r1 = REarth+hinicial; % [m]
17 v1 = sqrt(mu/r1); % [m/s]
18
19 % Second orbit
20 r2 = REarth+hfinal; % [m]
21 v2 = sqrt(mu/r2); % [m/s]
22
23 % Transfer orbit (ellipse)
24 vp = sqrt(2*mu*r2/(r1*(r1+r2))); % [m/s]
25 va = sqrt(2*mu*r1/(r2*(r1+r2))); % [m/s]
26 a = (r1+r2)/2;
27 T = sqrt(2*pi^2*a^3/mu);
28
29 % ΔV 1 -> p(transfer orbit)
30 ΔV1 = vp-v1;
31 % ΔV a(transfer orbit) -> 2
32 ΔV2 = v2-v1;
33 % Total
34 ΔV = ΔV1+ΔV2;
35
36 % Fuel mass
37 mp = m*(1-exp(-ΔV/(g0*Isp)));
38
39 % Time
40 t = T/2;
41
42 end

```

1.11 Satellites Datasheet

```

1 %----ASTREA CONSTELLATION----
2 %PROJECTS - 220028
3 %Aerospace Engineering Barchelor's Degree
4 %ESEIAAT - UPC
5 %Autumn 2016-2017
6
7 % ORBIT DESIGN TEAM
8 % SATELLITES DATASHHET
9

```

```

10 % Writing the constellation characteristics
11
12 a=6378.01e3;
13 u=3.986012e14;
14 h=(a+542e3)/a;
15 ap=h*a;
16 J2=1.0826e-3;% J2=0;
17
18 ws=sqrt(u/ap^3); % Satellite mean motion
19 i=72*pi/180;
20
21 p=9;
22 spp=21;
23 N=p*spp;
24 %f=floor(0.25*spp);
25 f=0;
26 fprintf('\ni=%gdeg, p=%g, spp=%g, N=%g\n',i*180/pi,p,spp,N);
27
28
29 %Omega = 0:30:240; Omega=Omega*pi/180;
30 Omega = 0:225/(p-1):225; %Omega=Omega*pi/180;
31 %Omega=0:360/p:360-360/p; Omega=Omega*pi/180;
32
33 P=2*pi/ws/60;
34
35 %% Excel IMPORT
36
37 D=zeros(N,9);
38 for sat=1:N
39
40     Om=ceil(sat/spp); ID_p=sat-(Om-1)*spp;
41     ph=(ID_p-1)*360/spp;
42     %[ID Plane h P i e Omega phase per]
43     D(sat,1:9)=[sat Om (ap-a)/1000 P i*180/pi 0 Omega(Om) ph 0];
44
45 end
46 NameCells=['B2:I' num2str(N)];
47 xlswrite('SatsDatasheet.xls',D,NameCells)
48
49 %% Latex IMPORT
50
51 for sat=1:N
52
53     Om=ceil(sat/spp); ID_p=sat-(Om-1)*spp;
54     ph=(ID_p-1)*360/spp;
55     %[ID Plane h P i e Omega phase per]
56     D(sat,1:9)=[sat Om (ap-a)/1000 P i*180/pi 0 Omega(Om) ph 0];
57
58 end
59

```

```

60 [rows,cols]=size(D);
61 file=fopen('Autotable.txt','w');
62
63 for i=1:rows
64     Text=['AstreaSAT ' num2str(i) ' $ '];
65     for j=2:cols
66         if j≠cols
67             Text=[Text num2str(D(i,j-1)) ' $ '];
68         else
69             Text=[Text num2str(D(i,j-1)) ' \\\ '];
70         end
71     end
72     fprintf(file,[Text '\n']);
73 end
74
75 fclose(file);

```

1.12 Ground Station Localization

```

1  %----ASTREA CONSTELLATION----
2  %PROJECTS - 220028
3  %Aerospace Engineering Barchelor's Degree
4  %ESEIAAT - UPC
5  %Autumn 2016-2017
6
7  % COMMUNICATION TEAM
8  % GROUND STATIONS LOCALIZATION
9
10 clc; clear;
11
12 %% Input Data
13 N_sat=21;    %number of sats in a plane
14 N_planes=9; %number of orbital planes
15 h=542;      %high of the sats in km
16 I=72;       %inclination of the orbits in degrees
17 phase=210/(N_planes-1); %Angle between planes in the equator
18 At=0.1;     %time step in minutes
19 T=48;       %time to simulate in hours
20 t=0:At:T*60; %array of time in minutes
21 e_min=7.5;  %minumum elevation in degrees
22 lambda=57.5; %latitudes to simulate in degrees
23 lat=length(lambda);
24 mu=0;       %longitudes to simulate in degrees
25 long=length(mu);
26 l=length(t);
27

```

Ground Station Localization

```

28 %% Solver
29
30 Xs=Orbitalposition(N_sat,N_planes,h,I,phase,t);
31 L=zeros(lat,1);
32
33 for i=1:lat
34     Xg=Groundposition(lambda(i),mu,t);
35     [L(i,:)]=Links(Xs,Xg,e_min);
36 end
37
38 %% Plots
39
40 figure;plot(t/60,L);legend 0 30 60 90;xlabel('time (h)');ylabel('links');

```

```

1 function [X] = Orbital_position(N_sat,N_planes,h,I,mu,t)
2 %N_sat: number of sats per plane
3 %N_planes: number of planes
4 %h: Orbit high in km
5 %I: inclination of the orbital plane in degrees
6 %mu: Angle between two adjacent orbital planes at the equatorial plane in
7 %degrees
8 %t: array of time in minutes
9
10 I=I*pi/180; %I in rad
11 mu=mu*pi/180; %mu in rad
12 Omega=0:mu:(N_planes-1)*mu;
13 %longitude of the ascending node respect the inertial X axe [rad]
14
15 %for defining the orbits are defined a system of
16 %local axes X_o in which the y_o is the rotation axe and x_o is
17 %contained in the equatorial plane. The transformation for the inertial
18 %system X_I to the local system 2 rotations are defined.
19 %FIRST ROTATION: it defines a intermediate system X_1. The inertial system
20 %rotates around z_I an Omega angle.
21 %SECOND ROTATION: The X_1 system is rotated (90deg-I) around x_1
22
23 L_1o=[1 0 0; 0 sin(I) cos(I); 0 -cos(I) sin(I)]; %1st rotation matrix.
24 %It transform a vector from local coordinates X_o to intermediate
25 %coordinates X_1. This matrix is the same for every orbital plane
26 %since all orbits have the same inclination
27
28 Re=6371; %Earth radius km.
29 %The distances are going to be refered to the earth radius
30 h=h/Re; %high in earth radius
31 GM=6.474e-11*5.972e24*3600/Re^3;
32 %Gravitational constant*earth mass [(Re^3)/min^2]
33 Ro=1+h; %orbital radius in earth radius
34
35 w=sqrt(GM/(Ro^3)); %sat's angular speed [rad/min]

```



```

36 phi=0:2*pi/N_sat:(N_sat-1)*2*pi/N_sat;
37 %relative angle between every sat of a plane respect the first one
38 f=8;
39 %factor to module the relative desfase of the first sat of one plane
40 %to the first sat of the next plane
41 l=length(t);
42
43 for p=1:N_planes    %going through each plane
44
45     phi_0(p)=f*2*pi*(p-1)/(N_planes*N_sat);
46     %relative desfase of the first sat of one plane to the
47     %first sat of the next plane
48
49     One = cos(Omega(p));
50     Two = sin(Omega(p));
51     L_I1=[One -Two 0; Two One 0; 0 0 1];
52     %2nd rotation matrix. It transforms a vector from
53     %intermediate coordinates X_1
54     %to inertial coordinates X_I. This matrix is characteristic of every
55     %plane since every plane has a different longitude of the ascending node
56     %(Omega)
57
58     L_eo=L_I1*L_1o;
59     %Global rotation matrix from local coordinates to inertial coordinates
60
61     for s=1:N_sat    %going through every sat of the plane
62
63         for k=1:l    %going through every time step
64
65             First = Ro*cos(w*t(k)+phi(s)+phi_0(p));
66             Second = -Ro*sin(w*t(k)+phi(s)+phi_0(p));
67             X_loc=[First;0;Second];
68             %Coordinates of the sat at the given time in
69             %local coordinates X_o.
70             %It defines a circumference at the plane x_o-z_o with an angular
71             %velocity w, an initial phase respect the first sat and an
72             %initial phase respect the first sat of the previous orbit
73             %first sat.
74
75             X(:,(p-1)*N_sat+s,k)=L_eo*X_loc;
76             %Coordinates of the sat in inertial system
77             %(Coordinate [x y z], Sat number, instant(time))
78         end
79     end
80 end
81 end

```

```

1 function [X] = Ground_position(lambda,long,t)
2 %Ground_position calculates the position of a place of the earth, with a

```

Ground Station Localization

```

3 %wiven longitude and latitude, in an inertial system of coordiantes X_I in
4 %a given period of time
5
6 %lamda: is the latitude of the place in degrees[-90,90]
7 %long: is the longitude of the place iin degrees [0,360)
8 %t: is a array of time in minutes
9
10 long=long*pi/180;    %longin radians
11 lambda=lambda*pi/180;    %lambda in radians
12
13 Rg=cos(lambda);
14 %distance between the Earth point to the rotation axis of the Earth [0,1]
15 hg=sin(lambda);
16 %distance between th Earth point to the equatorial plane [0,1]
17 %this 2 distances are measured in earth radius.
18
19 w=2*pi/(24*60); %angular velocity of the Earth [rad/min]
20
21 X=[Rg*cos(w*t+long);Rg*sin(w*t+long);hg*ones(1,length(t))];
22 %position of the point in X_I
23 %for the given time. The point describes a cicumference
24 %with a Rg radius in the horizontal
25 %plane x_I-y_I with an angular velocity w and a initial phase long. The
26 %coordinate z_I is allways hg. It is expresed in earth radius.
27 %X(cooridinales, instant (time)
28
29 end

```

Chapter 2

Bibliography