UCLouvain

École polytechnique de Louvain

CubeSats in formation for Earth observation

Author: **Ugo-Paul DURAFFOURD**
Supervisors: **Christophe CRAEYE, Paul FISETTE**
Readers: **Véronique DEHANT, Maxime DROUGUET**
Academic year 2021–2022
Master [120] in Electro-mechanical Engineering

Acknowledgments

First of all, I would like to thank my thesis promoter, Prof. Craeye, who has been of immense help throughout the realization of this thesis. I could not have done it without his precious advice at all hours of the day and night, without his encouragement, without his unfailing availability, without his thoughts that always pushed me to go further. I am extremely grateful to have had the chance to carry out this research under his direction.
I would also like to thank Prof. Fisette and Mr. Drouguet who also accompanied and supervised me during part of my work. Their availability and benevolence towards me were dear to me.

Abstract

Earth observation is one of the main sectors of space industry. The objective of this observation is to collect information on the systems of our planet.
This market is growing rapidly and opens up opportunities for many players in the sector, from satellite manufacturers to those who put them into orbit. It also opens up a world of opportunity for companies that can use data to help solve today's critical problems.
This thesis is intended to illustrate our ambition to take part in this growing sector. It also serves to reflect the first part of a long study whose ultimate goal is to launch its own satellites to observe the Earth in an innovative way.
The innovation behind this method is the use of a CubeSat array to observe the Earth. This paper will study different array configurations and determine the one that best fits our objectives.

Contents

1 Introduction

Earth observation is of the main sectors of space industry. The purpose of this observation is to collect information on the physical, chemical and biological systems of our planet. Using this information, we can monitor and evaluate the state and changes of Earth's natural and man-made environment.

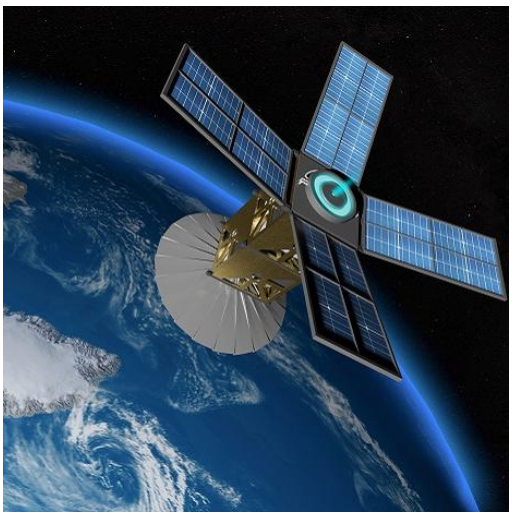


Figure 1.1: [13]

Global environmental concerns now pose major threats to humanity. Therefore, using satellite data to monitor the climate could become instrumental to better prepare for the decisions that will determine our future. This market opens up opportunities for many players in the sector, from satellite manufacturers to those who launch them into orbit. This opens up a world of possibilities for businesses that have the ability to use data to help solve today's critical problems.
This thesis aims to illustrate our ambition to take part in this growing industry. Indeed, this work is the first part of a long study whose final objective is to launch its own satellites to observe the Earth. However, we do not only want to participate, we want to innovate at the same time by offering new technologies, new methods in a more accessible way in order to better observe our globe.

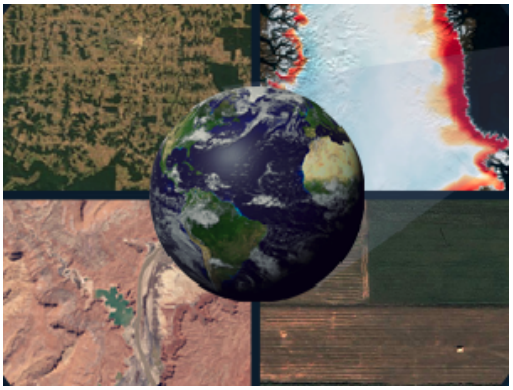


Figure 1.2: [1]

In this text, we will explain our methods to observe the Earth. First, we will introduce the unconventional satellites that will be used: the CubeSats.
Second, we will explain the GNSS-R method used to observe the Earth. Third, we will detail the approach and the model to represent the Earth's shape in our simulations.
Fourth, we will describe our geometric calculations to locate the specular point, between our CubeSats and a GNSS satellite, on Earth.
Fifth, we will examine antennas. Indeed, in this section we will study the characteristics of these antennas, antenna arrays, the configurations of these arrays, the choice of a particular array for our application and the beam steering method.
Finally, we will end with a conclusion which will summarize our main preliminary results and which will also serve as an introduction to the sequel of the project.

2 Choice of satellites: CubeSats

Figure 2.1: [2]

It is well known that the space industry is a very expensive field and requires large funds. Previously, only space companies with the necessary budget could afford sending satellites into space. But in 1999, California Polytechnic University and Stanford University defined a format of nano-satellites called CubeSat. The aim was to reduce the launch costs of very small satellites and thus allow universities to develop and place their own spacecraft in orbit.
This corresponds exactly to our situation. Indeed, as mentionned above, the University of Louvain aims to place its own satellites in orbit within the next 10-15 years in the most affordable way.
The CubeSats , hence their names, are cubes weighing about 2kg and measuring 10 cm on each side. They use solar cells to convert solar energy into electricity which is then stored in rechargeable lithium-ion batteries which provide power during eclipses as well as during peak periods. One of the roles of this CubeSats is to be equipped with one or more antennas which will be used as a radar to observe the Earth. Due to their small size, the CubeSats are not able to emit their own electromagnetic signal. These antennas will thus be only receiving antennas which will be used in a bi-static radar system configuration. This is explained in the next section.

3 Method for Earth observation: GNSS-R

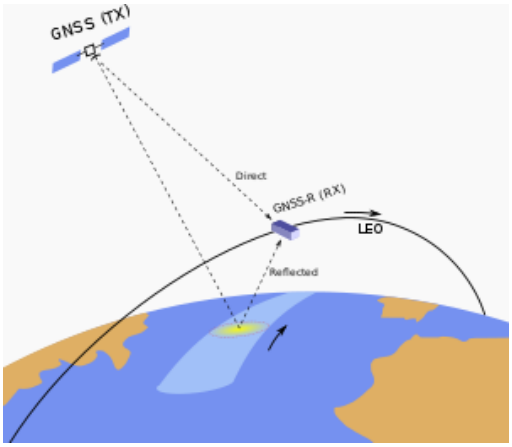


Figure 3.1: [8]

Some satellites use radars to observe the Earth. Indeed, today's satellites carry various radar sensors such as a synthetic aperture radar (SAR) for high-resolution imaging, a radar altimeter, to measure the topography of the oceans or a wind scatterer to measure wind speed and direction. For example, RADARSAT is a Canadian remote sensing satellite for Earth observation that uses a synthetic aperture radar to obtain terrain and land cover information.
The operation of a radar is based on the emission of an electromagnetic signal, its reflection on an object and then the reception of this modified (or not) signal which will be processed and analyzed. In the so-called mono-static radar systems, the transmission and reception of the signal takes place on the same device. Because of the small size of the CubeSats, it is impossible for them to emit their own electromagnetic signal. Indeed, SAR satellites in space emit an RF power of about 1 kW using electricity generated by sunlight and large solar panels. That is why in our application we will use a bi-static system. This means that our CubeSats are only equipped with receiving antennas. In our case we will use the GNSS reflectometry[8] (or GNSS-R) method. This method consists in making measurements from the reflections on the Earth of navigation signals from global navigation satellite systems such as GPS. The research applications of GNSS-R can be found in the following fields: altimetry, oceanography or soil moisture monitoring.
GNSS reflectometry is a passive sensing method that exploits and relies on separate active sources: the satellites generating the navigation signals (GPS). For this purpose, the GNSS receiver measures the delay of the signal from the satellite and the rate of change of the distance between the satellite and the observer (Doppler measurement). The surface of the reflected GNSS signal also provides the two parameters of delay and frequency variation. With these measurements and some additional calculations, we can deduce the geophysical information of the reflected area. The GNSS signal spectra is between 1.1 and 1.6 GHz, we will set the frequency to 1.5 GHz for our study.
Using a bi-static configuration complicates things a bit. In contrast to a mono-static radar, it is not sufficient to point the antennas to the area of interest. Indeed, the beams from the GNSS satellites are reflected all over the planet and the objective of our CubeSats is to come into low orbit and intercept these reflected waves. The more laborious task is to calculate the location where the CubeSats should be positioned to intercept the signal that will have been reflected on the desired point. This point is called the specular point. The process and methods used to find these specular points are explained in Section 5.

4 Earth Shape Modeling

To perform the calculations and simulations to find the specular points between the GNSS satellites and the CubeSats it was necessary to conceptualize the Earth numerically. First of all, we started from a spherical Earth on which we could create a simplified algorithm to locate these points. Then to make our results a little more real we decided to slightly complicate the task and to apply our algorithm to an ellipsoidal model of the Earth. Finally, we took the challenge to apply our calculations to a geoid model of the Earth. This greatly complicated our calculations as well as the computation time but it allowed us to convince ourselves of the effectiveness of our methods.

4.1 Earth Shape Modeling: Sphere

The equation of the points of a sphere are quite trivial but we will recall them to compare the different methods used to design Earth.
We used the following system of coordinates, derived from the conventions used by geographers. We call the coordinates ρ, θ, δ:

- ρ is the distance from the point to the center of the frame, which in our case is the average radius of the Earth ($\rho = 6371$ km);
- θ is the longitude, measured from the x axis, usually between $-180°$ and $180°$ ($-\pi \leq \theta \leq \pi$);
- δ is the latitude, the angle from the equatorial plane, between $-90°$ and $90°$ ($-\pi/2 \leq \delta \leq \pi/2$);

The exchange between Cartesian coordinates and spherical coordinates [10] is then done by the formulas :

$$\begin{cases} x = \rho\cos(\delta)\sin(\theta) \\ y = \rho\cos(\delta)\cos(\theta) \\ z = \rho\sin(\delta) \end{cases}$$

It is easy to switch from one system to another because latitude and colatitude are linked by :

$$\delta = 90° - \phi \text{ (or, in radians, } \delta = \frac{\pi}{2} - \phi)$$

where, ϕ is the colatitude.

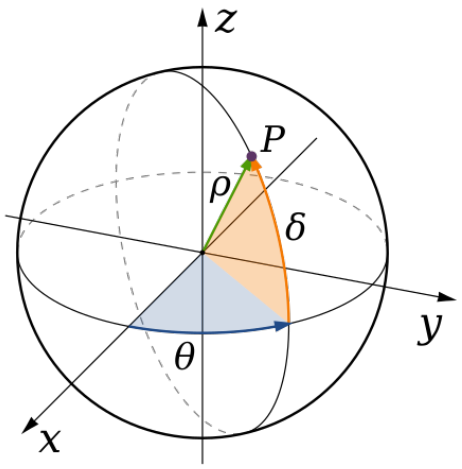


Figure 4.1: Spherical & Cartesian coordinates

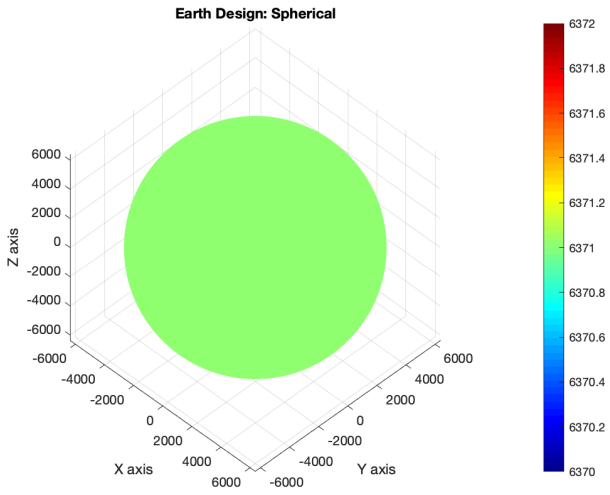


Figure 4.2: Spherical Earth design on Matlab

4.2 Earth Shape Modeling : Ellipsoid

In order to make our calculations a little more realistic, we then used the ellipsoidal [7] model of the Earth. An ellipsoid is a surface that can be obtained from a sphere by deforming it with directional scaling. An ellipsoid has three perpendicular axes of symmetry in pairs that intersect at a center of symmetry, called the center of the ellipsoid. If two of the axes have the same length, the ellipsoid is then an ellipsoid of revolution, also called a spheroid.
We therefore used the spheroid model to design the Earth.

The ellipsoid of revolution is parameterized in the Cartesian space as follows:

$$\begin{cases} x = q\sin(\phi)\sin(\theta) \\ y = q\sin(\phi)\cos(\theta) \\ z = p\cos(\phi) \end{cases}$$

Here, we call the coordinates q , p, ϕ, θ:

- q and p correspond to the semi-major axis. q represents the radius of the Earth at the equator (6378 km) and p the radius of the Earth at the poles (6357 km);
- θ is the longitude, measured from the x axis, usually from 0° to 360° ($0 \leq \theta \leq 2\pi$);
- ϕ is the colatitude, the angle from the pole axis (z-axis), from 0° to 180° ($0 \leq \phi \leq \pi$);

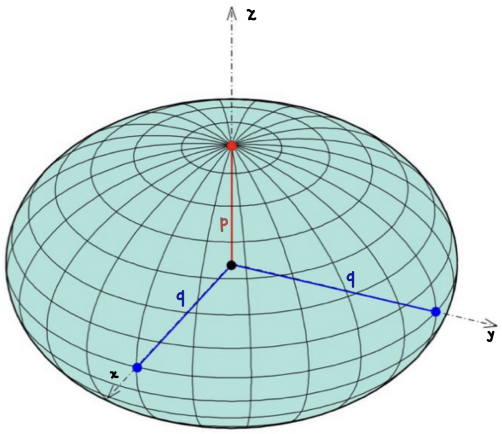


Figure 4.3: Ellipsoid parameters

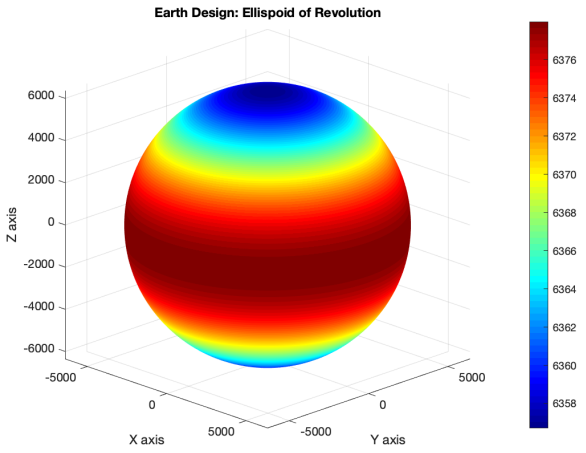


Figure 4.4: Ellipsoidal Earth design on Matlab

4.3 Earth Shape Modeling : Geoid

In reality the Earth is not exactly an ellipsoid of revolution. Indeed, the Earth's altitude can vary from -10 km to almost +9 km. Today the most accurate figure of the Earth is the geoid. The description of the geoid is traditionally done by the decomposition of the gravity field into spherical harmonics over the entire surface of the Globe. We proceeded in a different way, we decided to directly decompose the topography of the planet into spherical harmonics [9][3] in order to approach this geoid.
To do so, we had to find a database [12] which gave us access to the topography of the planet. We finally found one that was accurate to 5 arc minutes.

The objective of this design is to create a function that returns Earth's radius, R, as a function of latitude and longitude.

$$R = f(\delta, \theta)$$

We decided to approximate the data of the globe's topography by a sequence of spherical harmonics. Spherical harmonics are an equivalent of Fourier series for functions defined on a sphere.

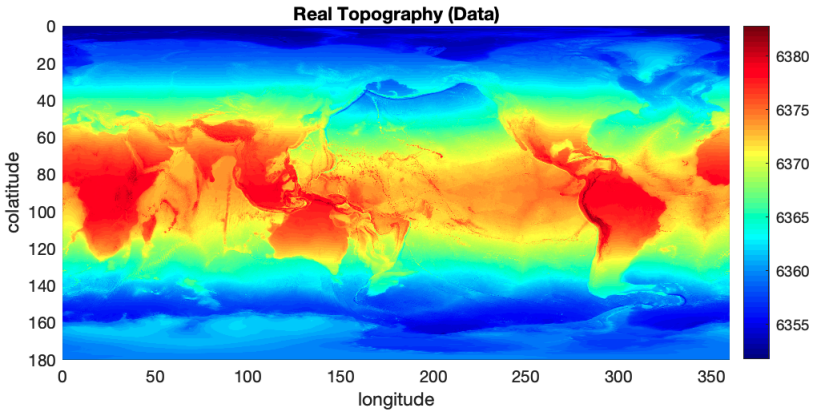


Figure 4.5: Real Earth topography

Let ρ, ϕ, θ the spherical coordinates : ρ is the radius, ϕ the colatitude, θ the longitude. The spherical harmonics, denoted $Y_l^m(\phi, \theta)$, are functions of the two angular coordinates. The index l and the exponent m are two integers called the degree and the order of the harmonic. They take the values $l = 0, 1, 2, ..., \infty$ and $m = -l, ..., 0, ..., l$.

The $Y_l^m(\phi, \theta)$ functions are defined as follows:

$$Y_l^m(\phi, \theta) = k \; P_{l,m}(cos(\phi)) \; e^{i \, m \, \theta}$$

where,

- k is a constant, which is fixed by normalization: $k = (-1)^{\frac{1}{2}(m+|m|)} \sqrt{\frac{(2l+1)}{4\pi} \frac{(l-|m|)!}{(l+|m|)!}}$
- $P_{l,m}$ are the associated Legendre polynomials: $P_{l,m}(x) = \frac{(-1)^m}{2^l l!} [1 - x^2]^{m/2} \frac{d^{l+m}}{dx^{l+m}} [x^2 - 1]^l$

The generalized spherical harmonics are defined on the S_3 sphere. The normalization of the spherical harmonics leads to the fully developed expression :

$$Y_l^m(\phi, \theta) = (-1)^{\frac{1}{2}(m+|m|)} \sqrt{\frac{(2l+1)}{4\pi} \frac{(l-|m|)!}{(l+|m|)!}} \; P_{l,|m|}(cos(\phi)) \; e^{i \, m \, \theta}$$

These functions are illustrated in Figure 4.6

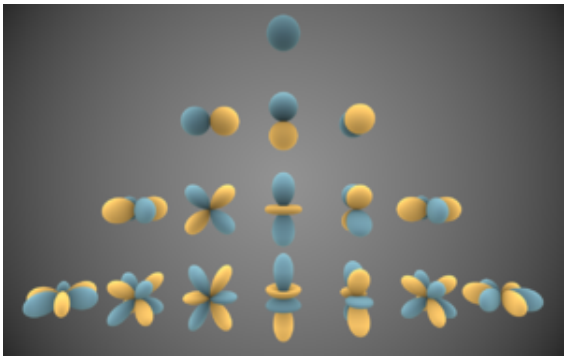


Figure 4.6: Representations of the first "real" spherical harmonics.

Using these harmonic functions we can create our desired ray function. Indeed, any sufficiently regular function $f(\phi, \theta)$ admits a series development :

$$f(\phi,\theta) = \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} a_{l,m} Y_{l,m}(\phi,\theta)$$

Here the function f represents a mapping function that fetch in the table of data the radius of the earth as a function of the longitude and latitude.
In order to construct our function radius R, we had to determine the value of the coefficients $a_{l,m}$. Given the orthogonality of the spherical harmonics $Y_{l,m}$, these coefficients can be calculated as:

$$a_{l,m} = \iint_{S_2} d\Omega(\phi,\theta)\ \overline{Y}^*_{l,m}(\phi,\theta)\ f(\phi,\theta)$$

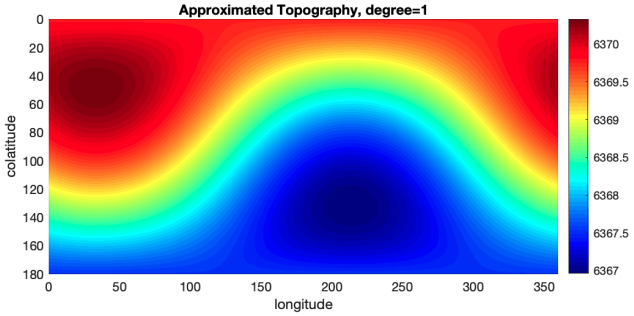
where,

- $d\Omega(\phi,\theta) = sin(\phi) d\phi d\theta$
- $\overline{Y}^*_{l,m}$ represents the conjugate of $Y^*_{l,m}$

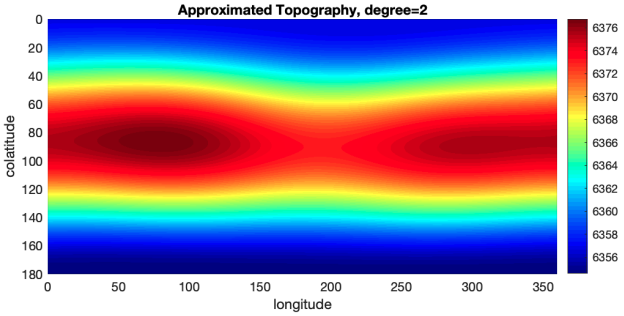
By determining these coefficients we can therefore design the topography of the Earth to a certain degree, d, using the radius function $R^d(\phi,\theta)$,

$$R^d(\phi,\theta) = \sum_{l=0}^{d} \sum_{m=-l}^{+l} a_{l,m} Y_{l,m}(\phi,\theta)$$

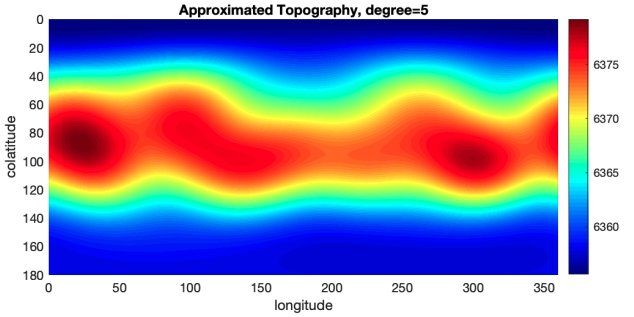
Here is the topography of the Earth broken down into a series of spherical harmonics for different orders:



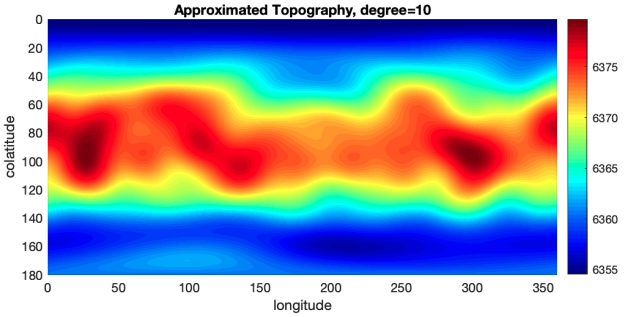
(a) Degree = 1



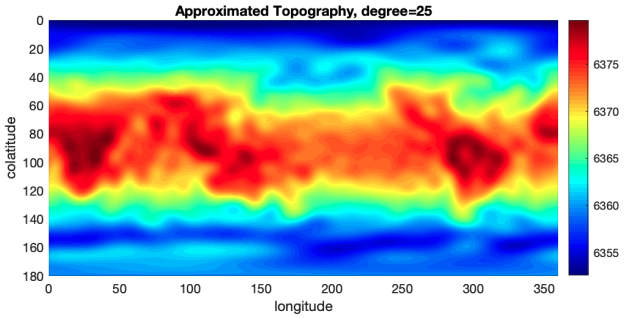
(b) Degree = 2



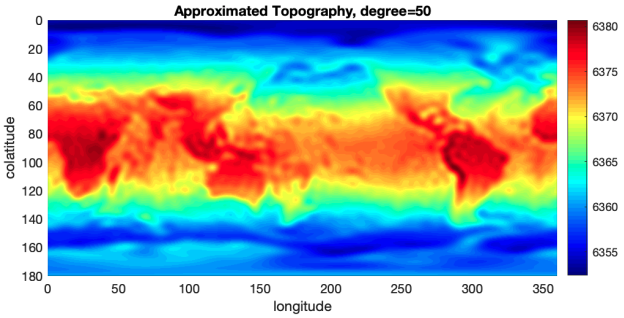
(c) Degree = 5



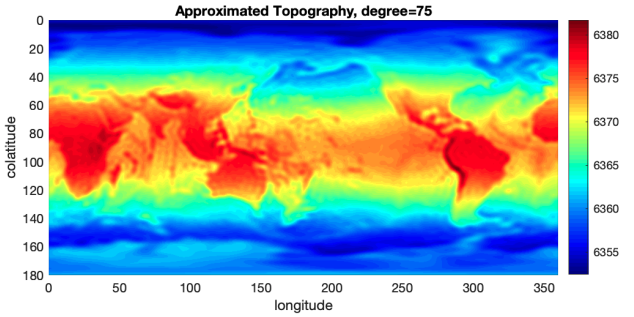
(d) Degree = 10



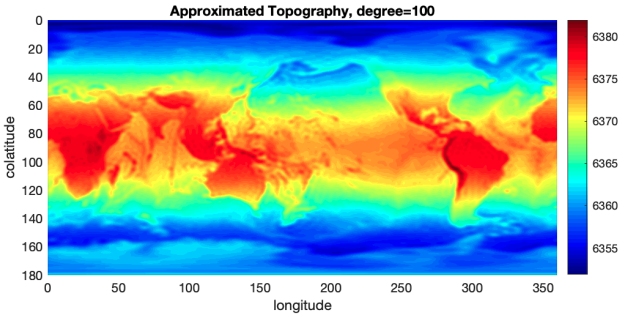
(e) Degree = 25



(f) Degree = 50



(g) Degree = 75



(h) Degree = 100

Figure 4.7: Earth's topography decomposed into a series of spherical harmonics for different orders

5 Specular points

Once the Earth's shape is established, we can start our calculations in order to locate the specular points, i.e. the point on the ground's surface where the reflection of the GNSS or Galileo satellite signal takes place. It's actually more of a zone than a point. Indeed, the roughness of the surface causes the diffusion of a glistening zone around the point of specular reflection. The scattering component, also known as the incoherent or diffuse component, consists of power scattered randomly in all directions, and its magnitude is less than that of the coherent component.
The method used for a spherical Earth is almost identical as the one for an ellipsoidal Earth. However, for the geoid model, the algorithm used is a bit more complex and requires additional calculations.

5.1 Specular points on Sphere

The specular point has the property of corresponding to the path of shortest time propagation between the transmitter, the surface and the receiver, and its incident angle is equal to the reflected one.

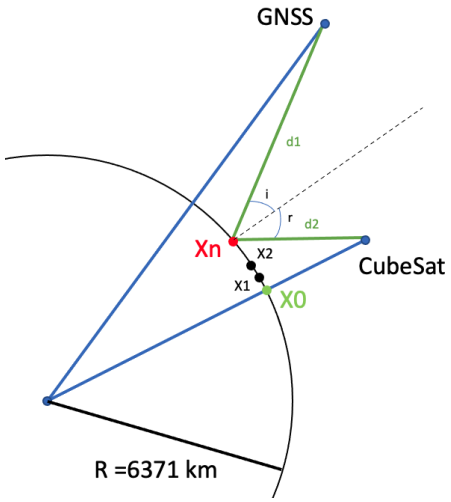


Figure 5.1: Specular point searching

It is therefore a function minimization. The initial parameters to which we have access are the coordinates of the GNSS satellite location, the coordinates of the CubeSat location and the fact that the potential specular point is located on the surface of the Earth. The first candidate for the location of the specular point is the projection of the CubeSat on the Earth, i.e. the point x_0 on Figure 5.1. This initial point is then reintroduced in the optimization function until the point corresponding to the minimum path $(d_1 + d_2)$ is found. Hence, x_n is the specular point which is approximately in the center of the glistening zone on the surface of the Earth and the incident angle, i, is equal to the reflected angle, r.

A step prior to the algorithm is a test to verify that the GNSS satellite is in the field of view of the CubeSat satellite. Indeed in some configuration, the Earth can be between the two satellites and therefore the specular point does not exist. It is therefore useless to launch the search algorithm for it.
To verify this, we draw a line linking the two satellites. Then we draw a perpendicular to this line which intercepts the center of the Earth i.e the red line r on Figure 5.2. If this line smaller than the average radius of the spherical Earth ($R = 6371$ km), then it is possible that the two satellites are not in the same field of view.
However, this condition is necessary but not sufficient to establish that they are not in the same field of vision. Indeed in a special case, it is possible that this line is not greater than the radius of the Earth but where the satellites see each other. For this we must verify two new conditions. It must be verified that the length of the line linking the two satellites i.e. the line of length d_3 on Figure 5.3 is smaller than the line linking the projection of the center of the Earth to the GNSS satellite i.e the line of length d_1. It must also be verified that d_3 is smaller than the line of length d_1 i.e. the line linking the projection of the center of the Earth to the CubeSat.

In summary if $r < 6371$ then it is possible that the two satellites do not see each other. But if $d_1 > d_2$ and $d_1 > d_3$ then they see each other otherwise they do not.

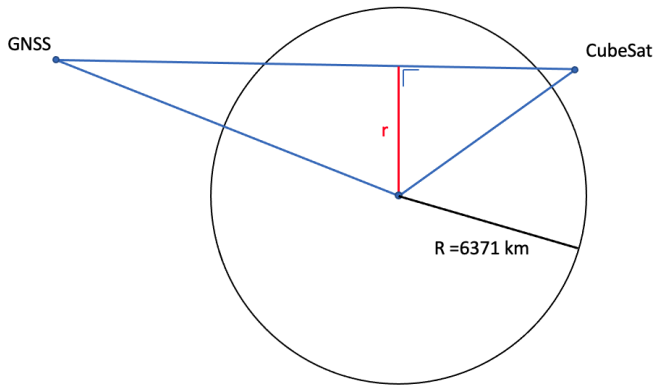


Figure 5.2: First case: satellites are not in the same field of vision

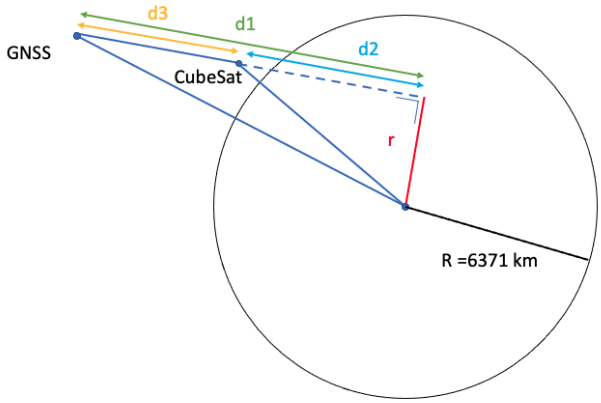


Figure 5.3: Second case: satellites are in the same field of vision

5.2 Specular points on Ellipsoid

For the ellipsoidal model of the Earth we use the same algorithm. The only difference is the shape of the Earth. Indeed, we know that the potential candidates of the specular point are located on the surface of the Earth. In this case, the Earth is ellipsoidal, thus, the norm of the vector linking the center of the Earth to the candidate in question will no longer be constant and equal to 6371 km. This norm will vary with the longitude and latitude as defined in Section 4.2. We recall that in this model of the Earth, the radius at the equator is maximum and the radius at the poles is minimum.

Another difference is in the verification step. The difference is in the first condition. In the case of the ellipsoid, we check that the length of r is smaller than the maximum radius of the Earth, that is to say the radius at the equator and not the average radius as before. It is useless to make it more complicated than that because in some cases the satellites will be in the same field of view but the transmitted and reflected rays will be grazing and therefore the resulting signal will not be exploitable.

5.3 Specular points on Geoid

In the case of the geoid model of the Earth, we proceed in 3 steps, meaning that we have an additional step compared to the ellipse model. The first step is the verification step. The second step is looking for the specular point on the ellipsoidal Earth. Then, it is necessary to add an additional step because it is possible that the solution found in step 2 is not the specular point because of the local topography, this case is illustrated on Figure 5.4. Because of the relief, the shortest signal path condition no longer holds. Indeed, the reflection point that corresponds to the minimum distance trajectory potentially no longer corresponds to a point where the incident angle is equal to the reflected angle.

In this additional step, we make a new optimization with as initial candidate the solution of step 2, s_1 on the Figure. This allows approaching the solution but limiting the computations at the same time. In this new optimization, the specular point must be on the surface of the Geoid, that is to say that the vector linking the specular point to the center of the Earth has a norm equal to R, where R is the Earth radius decomposed into a series of spherical harmonics (see Section 4.3). With this definition of the radius, we take into account the local relief that can affect the location of the specular point. In this third step, we seek to minimize the tilt between the local normal to the Earth's surface and the bisector of the vectors linking the specular point to the respective satellites. By doing this we can get as close as possible to reality and find the real specular point, s_2.

We can change the accuracy of these results by changing the accuracy of the earth radius approximation. We soon realize that a high degree of accuracy is not necessary because radars are not actually that accurate. Indeed, in reality we are looking for a specular area and not a specular point. This is why the use of the 3rd step may not be necessary because it is too accurate. We can make the approximation that the Earth has a regular enough surface to not need this last step. Moreover, we notice that the results of the ellipsoidal model are quite close to the results of the geoid. Indeed, we have performed several tests with our algorithm in different conditions. And even in extreme cases where the specular point was in mountainous areas (near Everest) and where the satellites

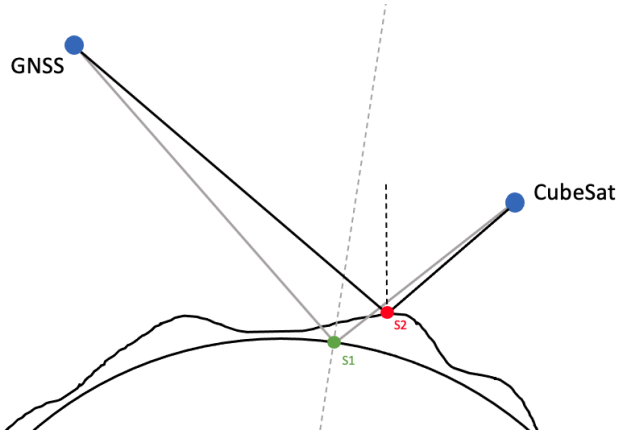


Figure 5.4: Use of additional step to locate specular point

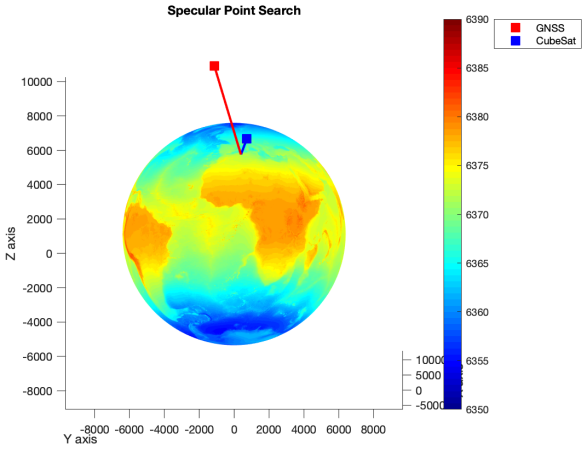


Figure 5.5: Specular search with Matlab algorithm

were quite far (difference of 100 degrees in latitude), we still obtained an accuracy of around 10 km. This is the reason why in Section 6.3 we are interested in an 10x10 km^2 area to study the gain of the antennas projected on the ground.

6 Antennas

As said before, we want to observe the Earth with a bistatic radar system working at around 1.5 GHz. Our CubeSats will therefore be equipped with antennas. Indeed, an antenna is a device allowing to radiate (transmitter) or to receive (receiver) electromagnetic waves. In this section we will remind several characteristics of antennas.
First of all, we will look at the transmission and reception characteristics of these antennas.
Then, we will determine an efficient layout of these antennas for our application.
After that, we will study the observable ground area and the reception resolution of the reflected signal.
Finally, we will define the method of beam steering and the reason for using it.

6.1 Characteristics

[11] Using the principle of reciprocity in electromagnetism, which allows us to relate the properties of transmission and reception of an antenna, we will show how an antenna is characterized. This principle states globally that the emission and the reception characteristics are equivalent. This means that in our application we will study the behavior of the CubeSat antennas as if they were transmitting.

6.1.1 Radiation pattern

An isotropic antenna, i.e. radiating in the same way in all directions, is a theoretical model that cannot be realized in practice. In reality, the energy radiated by an antenna is distributed unevenly in space, with certain directions being favored: these are the "radiation lobes". A radiation or emission pattern [6] is the graphical representation of the angular distribution of a quantity characterizing the radiation of a radio antenna, and, by extension, it corresponds to the distribution itself. The radiation pattern of an antenna can therefore visualize these emission lobes.

There is a link between the radiation pattern and the electric field. Indeed, we note that the expression of the field $\vec{E}(\vec{r})$ is the product of a function of distance and a function depending on the direction of observation $\hat{u}$. This second factor is therefore the radiation pattern $\vec{F}(\hat{u})$ so that the expression of the electric field radiated at great distance R can be written:

$$\vec{E}(\vec{r}) = \frac{e^{-jkR}}{R}\vec{F}(\hat{u})$$

Where k is the wave number, $k = \frac{2\pi}{\lambda}$, where λ is the wavelength which equals around 20 cm in our application because the frequency of the GNSS signal is equal to about 1.5 GHz.

6.1.2 Directivity and Gain

The directivity of the antenna in the horizontal plane is an important feature in the selection of an antenna. It has one or a few lobes which are clearly more important than the others, the so-called "main lobes". The narrower the main lobe, the more directional it is. As for the gain, it is defined as the increase in power emitted or received in the main lobe as compared to a hypothetical isotropic antenna. It is due to the fact that the energy is focused in one direction.

An antenna is thus characterized by its directivity $D(\hat{u})$ and its gain $G(\hat{u})$ in an arbitrary direction $\hat{u}$. Let P_r be the total radiated power and $S(\hat{u})$ the norm of the Poynting vector at large distance R from the antenna (S is real and equals to $|E|^2/2\eta$). The directivity is the ratio of the Poynting vector $S(\hat{u})$ at distance R to the Poynting vector that would be radiated at the same distance by an isotropic source for the same radiated power P_r.

$$D(\hat{u}) \triangleq \frac{S(\hat{u})}{P_r/4\pi R^2}$$

where,

$$P_r = R^2 \iint S(\hat{u}) d\Omega$$

We can therefore express the directivity as a function of the radiation pattern $\vec{F}(\hat{u})$. Since the Poynting vector is proportional to the square of the modulus of the electric field, we obtain:

$$D(\hat{u}) = 4\pi \frac{|\vec{F}(\hat{u})|^2}{\iint |\vec{F}(\hat{u})|^2 d\Omega}$$

The gain is related to the directivity through the following relationship:

$$G(\hat{u}) = \nu D(\hat{u})$$

where ν is the radiation efficiency. The latter is the ratio between radiated power and power delivered to the antenna (the difference between these quantities corresponds to the power lost through Joule effects in the antenna).

6.1.3 Array of antennas

Several sources can be grouped to form an array. The resulting radiation will depend on the individual characteristics of the sources on one hand, and the geometry of the array on the other. We will assume here that the sources are similar.

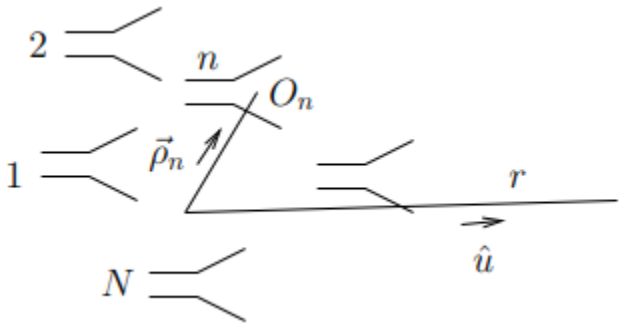


Figure 6.1: Array of similar antennas

Let a set of similar antennas (cf. Figure 6.1), i.e. such that placed at the same point O, their radiation patterns are of the form

$$\vec{F}_{no}(\vec{u}) = a_n \, \vec{F}_o(\vec{u})$$

where a_n is a complex number. In other words, the radiation patterns of the antennas differ only in amplitude and phase; the relative spatial distribution and polarization are identical.

The field radiated by the antenna n placed in O_n is then by the principle of translation

$$\vec{E}_n = \vec{E}_o \ \exp(jk \ \vec{\rho_n} \cdot \vec{u})$$

i.e. the radiation pattern is

$$\vec{F}_n(\vec{u}) = \vec{F}_{no}(\vec{u}) \ \exp(jk \ \vec{\rho_n} \cdot \vec{u})$$

The field radiated by the array is then by the superposition principle

$$\vec{E} = \sum_n \vec{E}_n = \frac{\exp(-jkr)}{r} \ \vec{F}(\vec{u})$$

with a radiation diagram

$$\vec{F}(\vec{u}) = \sum_n \vec{F}_n(\vec{u}) = \sum_n \vec{F}_{no}(\vec{u}) \exp(jk \ \vec{\rho_n} \cdot \vec{u})$$

where

$$\vec{F}(\vec{u}) = \vec{F}_o(\vec{u}) \sum_n a_n \exp(jk \ \vec{\rho_n} \cdot \vec{u})$$

If we define an array factor such that

$$R(\vec{u}) = \sum_n a_n \exp(jk \ \vec{\rho_n} \cdot \vec{u})$$

we obtain

$$\vec{F}(\vec{u}) = \vec{F}_o(\vec{u}) \ R(\vec{u})$$

The resulting radiation pattern is therefore equal to the product of the normalized radiation pattern of an element (radiation pattern of the antenna n normalized to a_n) and the array factor. If we set $|\vec{F}_o(\vec{u})| = 1$, we see that $|\vec{F}(\vec{u})| = R(\vec{u})$, so that the array factor is also the radiation pattern of an array obtained by replacing each antenna with an isotropic antenna of complex amplitude a_n. Although an isotropic antenna is physically impractical, we conclude that the characteristics obtained by forming an array can be analyzed independently of the radiation pattern of the antennas by analyzing the properties of the array factor.

If we set this complex amplitude a_n to 1 we notice that we get a very simple formula for the radiation pattern of an antenna array:

$$\vec{F}(\vec{u}) = \sum_n \vec{F}_n(\hat{\mathrm{u}}) = \sum_n \exp(jk \ \vec{\rho_n} \cdot \hat{\mathrm{u}})$$

We will therefore use this definition to plot the radiation patterns numerically for different layouts of the array (see next section).

6.2 Array Layout

There are several antenna array geometries. Linear arrays have their elements arranged along a line. Planar arrays are two dimensional groupings whose elements are arranged in a plane. Conformal arrays have their elements on a surface with an imposed shape. Spherical gratings can be interesting when one wishes to scan a half-sphere with a uniform beam, presenting the same gain in all directions.

The position and number of elements in a array influences the radiation pattern of the array. We will see here what their impact is.

6.2.1 Effect on the number

It is worth remembering that there are no direct formulas for finding the minimum number of sources necessary to form a given radiation pattern. The number N is a parameter rarely optimized and yet fundamental because of the cost, or for technical reasons where it is more convenient to use the least number of sources possible. Usually we take several arrays with a different number of elements N in order to find the one that best approaches the fixed radiation constraints or the previously imposed template.

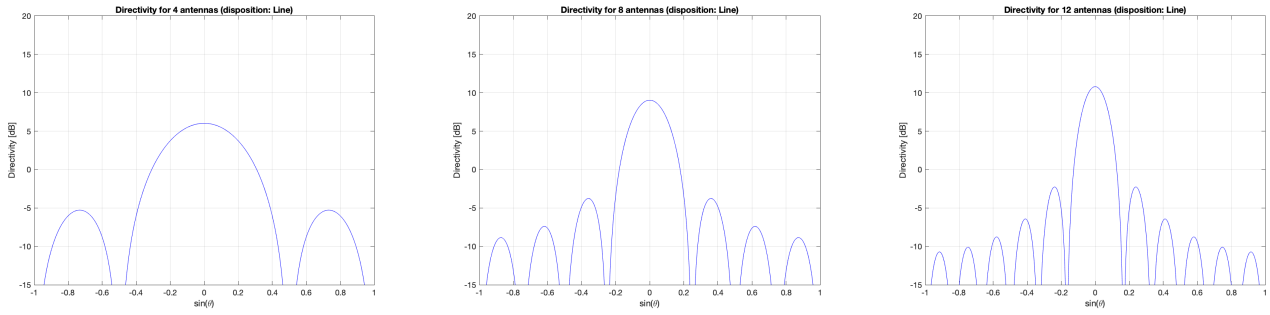


Figure 6.2: Directivity of an inline antenna array with $\lambda/2$ distance between antennas

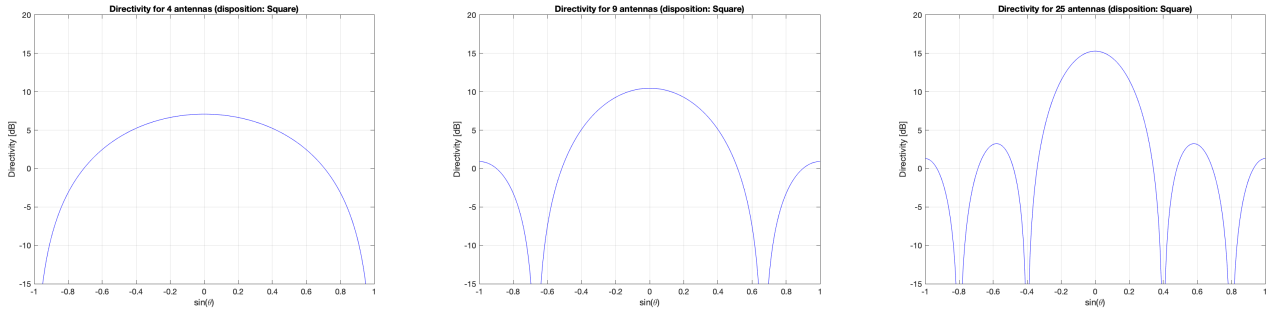


Figure 6.3: Directivity of a square antenna array with $\lambda/2$ distance between antennas

We notice that the increase of N increases the directivity and the number of secondary lobes, so the array is more directive if it contains more elements. However the level of the secondary lobes is too high, this forces us to look for a compromise or to manipulate other parameters. We can see that for 2D arrays this effect is delayed, indeed for 25 antennas arranged in a square we have only 4 side lobes as opposed to 10 for 12 antennas in lines.

6.2.2 Effect on the source position

The number of main lobes depends on the distance d between the sources.

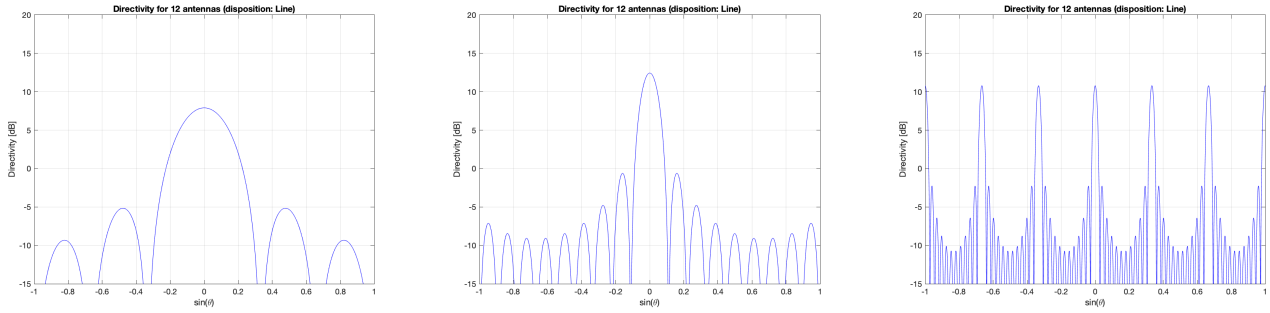


Figure 6.4: Directivity of an inline antenna array ($N = 12$) with varying distance between antennas : $d = \lambda/4\ ; 3\lambda/4\ ; 3\lambda$

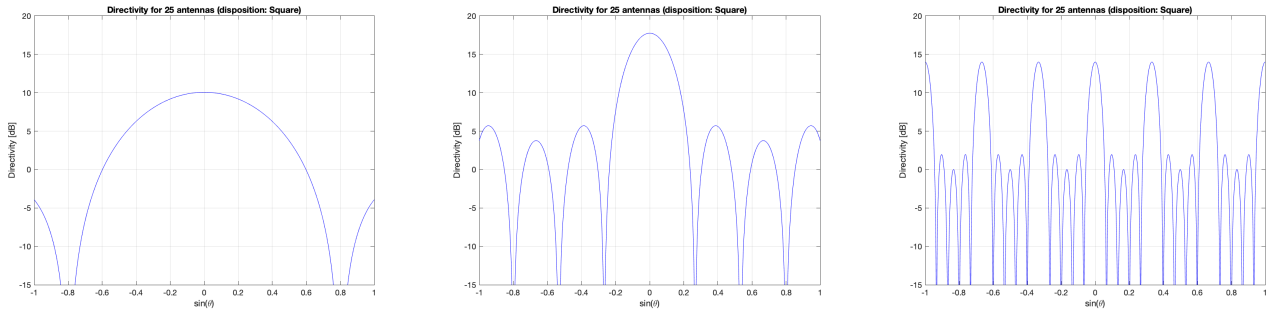


Figure 6.5: Directivity of a square antenna array ($N = 25$) with varying distance between antennas : $d = \lambda/4\ ; 3\lambda/4\ ; 3\lambda$

We can notice that the width of the main lobe as well as the secondary lobes are inversely proportional to the distance between the array elements. The greater the distance between sources, the more directional the array is and the greater the number of secondary lobes. On the other hand, when the array is regular, we also observe the repetition of the primary lobe for greater distances. These are called grating lobes and they will be explained in the next section.

6.3 Choice of a specific layout: GRS

We can notice that, for antenna arrays where the distance between the antennas is greater than half the wavelength, we have the appearance of grating lobes. A grating lobe is defined as "a lobe other than the main lobe, produced by an antenna array when the spacing between the elements is large enough to allow in-phase addition of the fields radiated in more than one direction". These grid lobes will prevent us from observing the Earth properly because we will have main lobes in directions we are not interested in and may detect interfering signals.

The goal behind the efficient layout of the array is that our antennas are separated by a minimum distance due to the size of the CubeSats, that the antennas are arranged at more or less equal distances from each other, that the main lobe is of a certain size relative to the side lobes and that we avoid the special effect of mutual coupling between the antennas caused by a regular layout.

After several tests and hints found in the literature [4] we have determined an antenna array arrangement which more or less meets our constraints even when the antennas are separated by a large distance. This layout is the GRS layout for Golden Ratio Spiral. It is inspired from the golden ratio. In geometry, a golden spiral is a logarithmic spiral whose growth factor is ϕ, the golden ratio. That is, a golden spiral expands (or moves away from its origin) by a factor ϕ for every quarter turn it makes. The radius of the spiral is thus defined as follows: $r = \phi^{(\frac{2\theta}{\pi})}$.

where,

- ϕ is the golden ratio
- θ is the angle formed by the radius

Figure 6.6 shows the shape of a golden spiral and Figure 6.7 shows the shape of a GRS antenna array made of 49 elements inside a 15m radius circle:

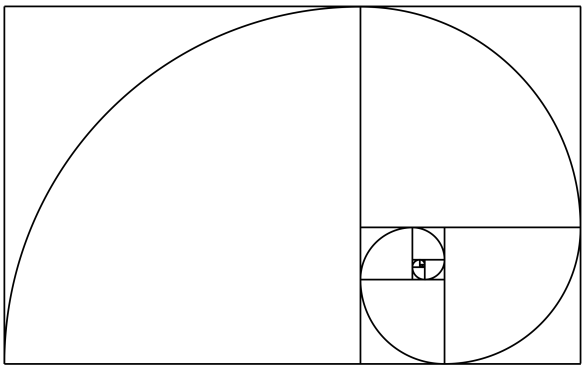


Figure 6.6: Golden spiral

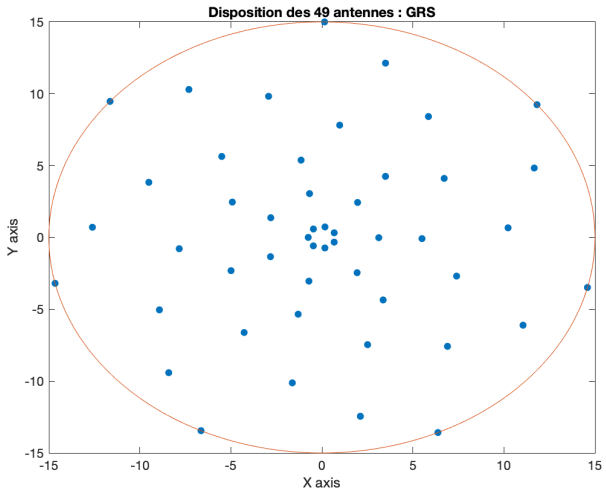
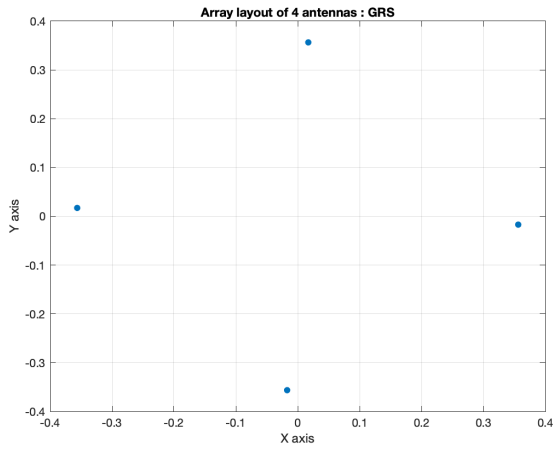


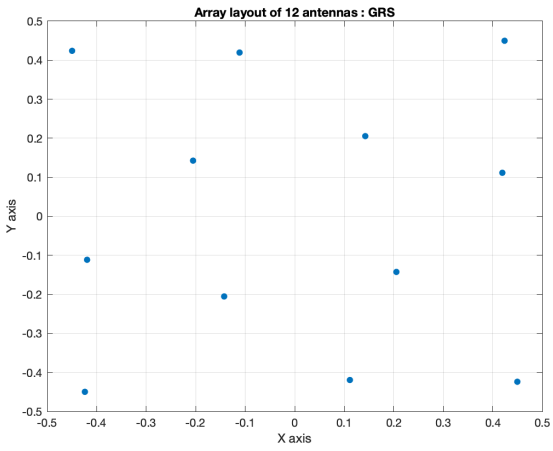
Figure 6.7: GRS antenna array ($N = 49$): 7 golden spirals of 7 antennas

The idea behind this arrangement is to create several spirals offset by a certain angle in order to have several arms containing antennas. In order to compare our results we will plot the antenna patterns for the different GRS antenna array arrangements. We will then be able to analyze the level of the side lobes as well as the size of the main lobe. We will also plot the gain of the ground spot, i.e. we will be able to analyze the intensity in decibels of the area we will be observing on the ground as well as the resolution of our antenna array. In our calculations we decided to study a zone around the specular point with an area of 100 square km i.e. a square of 10km sides. In this area we will define the glistening zone as the area where the gain intensity drops by 3 decibels compared to the maximum value.

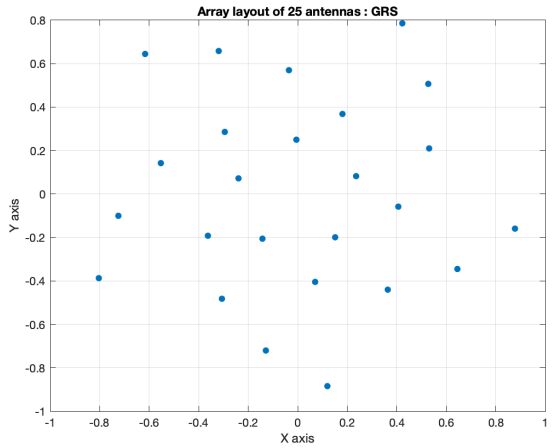
Here are the results for different antenna arrays where we vary the number of antennas (N), the density of the array (D [$\mathrm{N_{antennas}/m^2}$], the number of arms and the number of antennas per arm (N_{arms}x$N_{per\ arm}$).



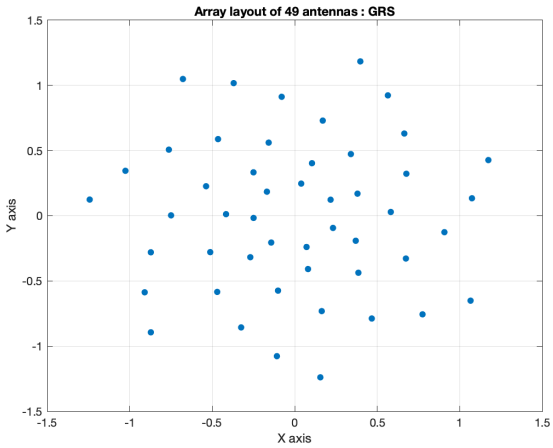
(a) N = 4; D = 10



(b) N = 12; D = 10



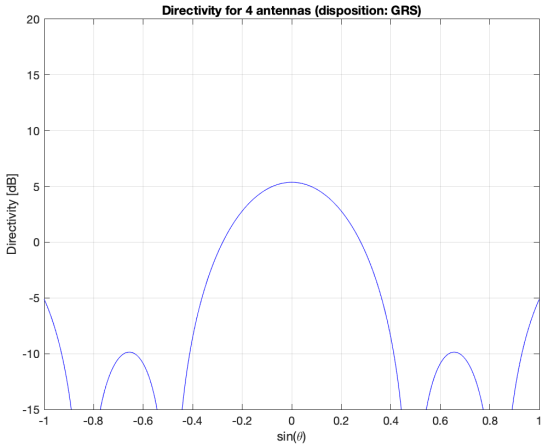
(c) N = 25; D = 10



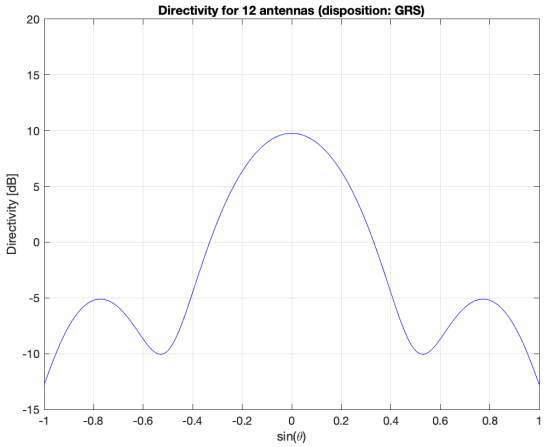
(d) N = 49; D = 10

Figure 6.8: GRS antenna array layout for a fixed density of 10 antenna per m^2

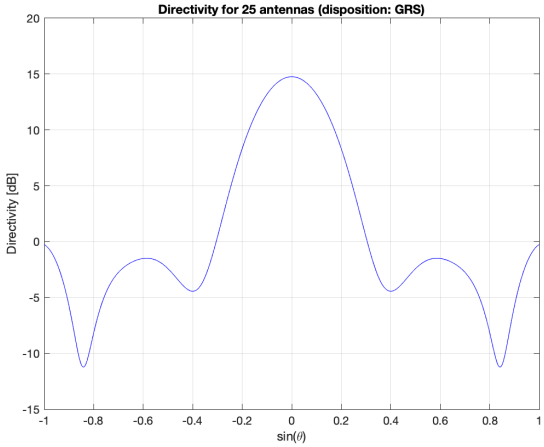
Figure 6.8 shows different layouts for GRS antenna arrays where we increase the number of antennas but we maintain the same density.



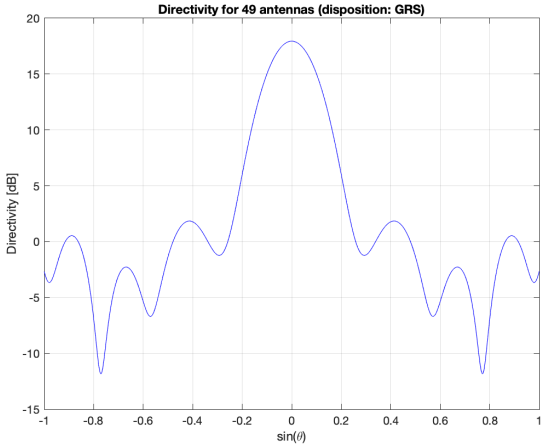
(a) N = 4; D = 50



(b) N = 12; D = 50



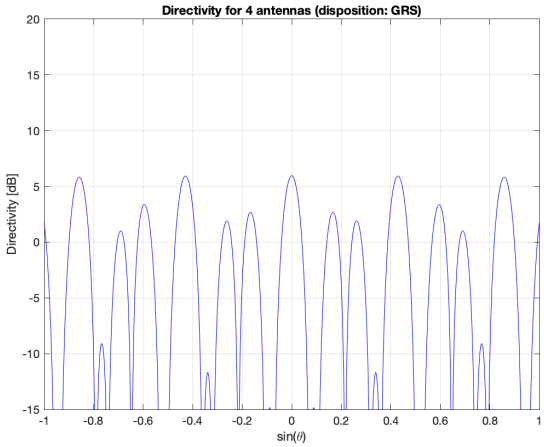
(c) N = 25; D = 50



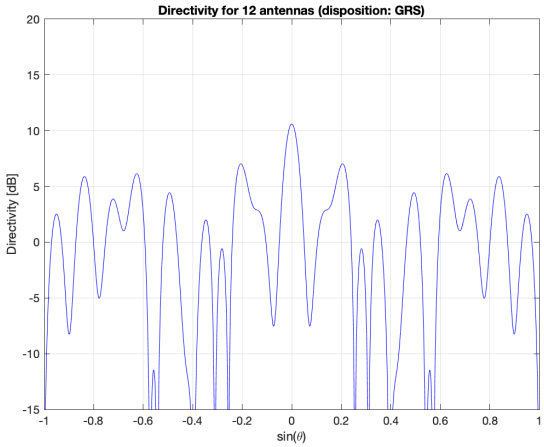
(d) N = 49; D = 50

Figure 6.9: Directivity of GRS antenna array layout for an increasing N and a fixed $D = 50$

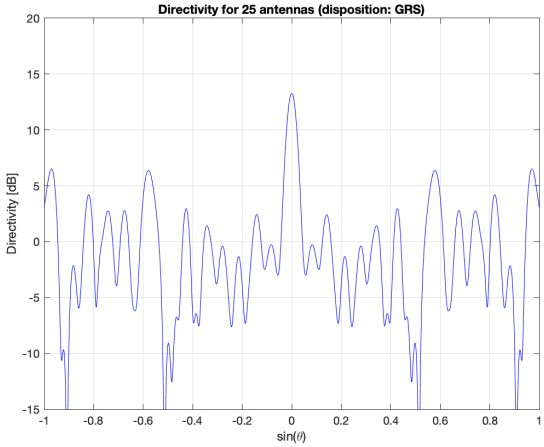
Figure 6.9 shows the directivity for GRS antenna arrays as a function of $\sin\theta$ where θ is the angle from broadside. The direction of interest is represented by the angle $\theta = 0$. Here also we increase N but D remains fixed, here it is equal to 50 [antennas/m^2].
Here we can see that the results are promising for these parameters. Indeed the more we increase the number of antennas the more the main lobe prevails over the secondary lobes. The latter become more numerous but their intensity decreases. The only disadvantage of this arrangement is the density as it is very high. It should be kept in mind that a CubeSat is about 10 cm wide and that a minimum safety distance is needed between them. Here the CubeSats do not touch each other but they do not have enough distance between them. Indeed, the distance between the CubeSats at the center of the array is of the order of 10-15 cm. In space it is difficult to place these satellites to the nearest centimeter so this arrangement is too risky as they could collide and change orbits. A trivial solution would be to place several antennas on the same satellite and thus to realize an entire or part of a GRS array on a single CubeSat.



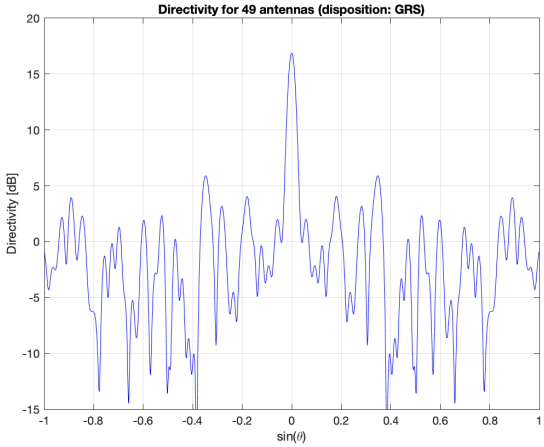
(a) N = 4; D = 1



(b) N = 12; D = 1



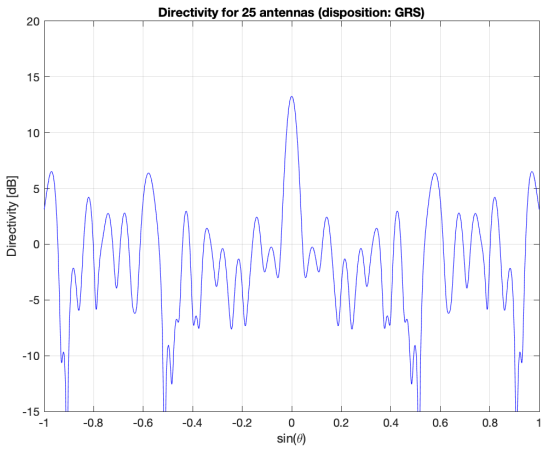
(c) N = 25; D = 1



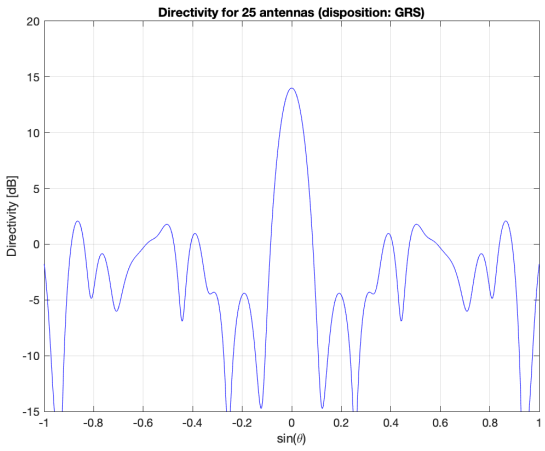
(d) N = 49; D = 1

Figure 6.10: Directivity of GRS antenna array layout for an increasing N and a fixed $D = 1$

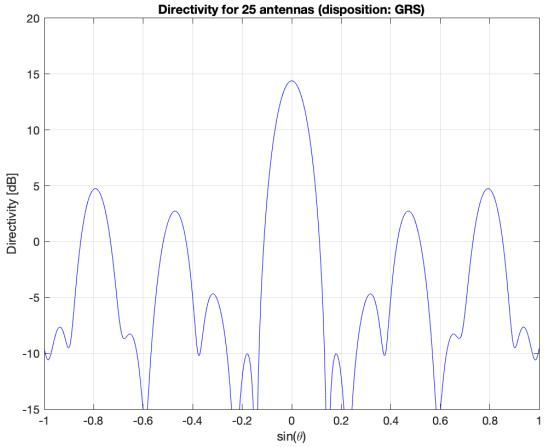
Figure 6.10 shows the directivity for GRS antenna arrays as a function of $\sin\theta$. Here also we increase N but D remains fixed, here it s equal to 1 [antennas/m^2].
Here we can see that the results are less promising than for $D = 50$ but they are more realistic. Indeed we still observe the same effect where when we increase the number of antennas the level of main lobe prevails over the level of the secondary lobes. The latter become more numerous but their intensity decreases. The disadvantage of this arrangement is the appearance of large grating lobes for low values of N. Indeed because we decrease the density we increase the distance between the different antennas and the latter exceeds the order of half the wavelength. Because of this we observe this repetition of the primary lobe.
However, what is very interesting to observe is the fact that when we increase N these lobes disappear and leave only one primary lobe. This means that this arrangement, with one antenna per satellite, is feasible in terms of safety distance but perhaps less feasible in terms of budget.



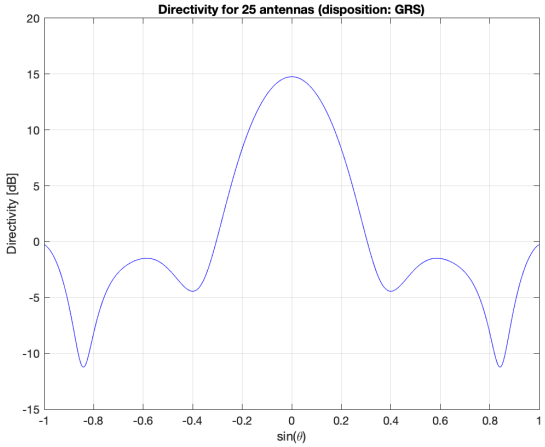
(a) N = 25; D = 1



(b) N = 25; D = 5



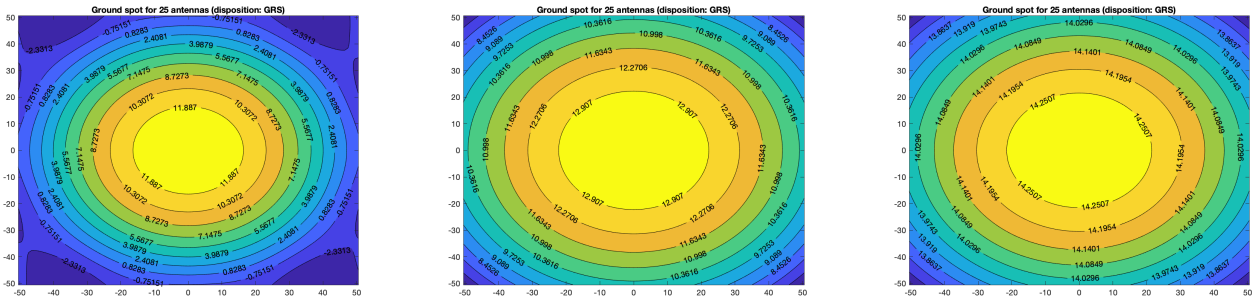
(c) N = 25; D = 10



(d) N = 25; D = 50

Figure 6.11: Directivity of GRS antenna array layout for a fixed $N = 25$ and an increasing D

Figure 6.11 shows the directivity for GRS antenna arrays as a function of $\sin\theta$. Here also we increase D but N is fixed and equal to 25 antennas (5 arms of 5 antennas).
Here we can better see the influence of the array density on its directivity. As discussed before, the array with density $D = 50$ is the most efficient but the least feasible. Let's then study other densities. We can see large sidelobes for density $D = 10$ which become smaller for lower densities. For $N = 25$, the side lobes are the weakest for $D = 5$ (without taking into account the results for $D = 50$). It is therefore at first glance the best arrangement. We will see in the next results that choosing a smaller density also has its advantages.



(a) N = 25; D = 1 (b) N = 25; D = 5 (c) N = 25; D = 50

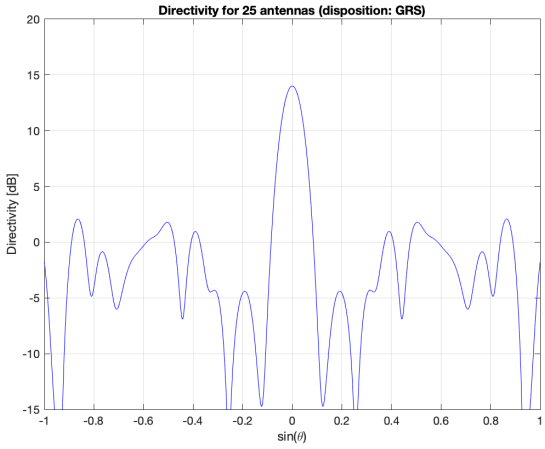
Figure 6.12: Ground spot formed by GRS antenna array layout for a fixed $N = 25$ and an increasing D

Figure 6.12 shows the intensity of the electric field over the observed area on the ground. This is the gain of the receiving antenna projected over the 10 km^2 square area. Here we see that the intensity is higher for high densities. On the other hand, the resolution is much better for density $D = 1$. Indeed we can see that the main beam is concentrated on a smaller area and this will increase the accuracy of the observation of the Earth. Instead of being accurate to the nearest 10 km for density $D = 5$ we can be accurate to the nearest 5 km. This is very important for the accuracy of the Earth observation. At first glance the difference in maximal intensity between $D = 5$ and $D = 1$ is of the order of a decibel (this represents one tenth of the maximum intensity) while the resolution almost doubles.
Hence, For $N = 25$ the density $D = 1$ is the best compromise. Furthermore if an antenna is well made the side lobes perpendicular to the main lobe are suppressed through multiplication by the element pattern which would further increase the accuracy of this arrangement because we can see that in Figure 6.11a we have the highest side lobes for $|\sin(\theta)| \approx 1$.

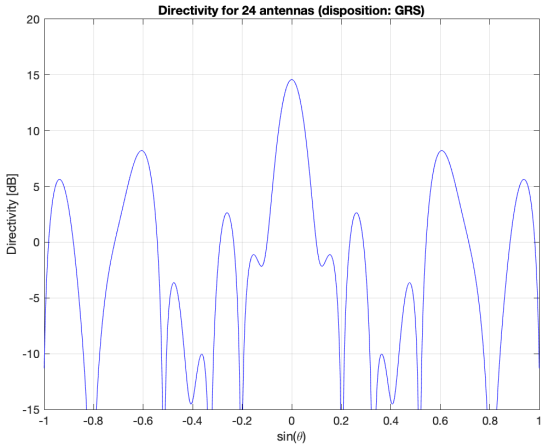
In Figure 6.13 we can observe better results when the number of arms is close to the number of elements on it. Indeed the 6x4 and 4x6 configurations are more efficient than the 8x3 or 3x8 ones and the 5x5 configuration is the best. When it is impossible to take the exact root, it is preferable to have a larger value for the number of arms than for the number of elements on each arm. Indeed on Figure 6.13, we can see that the directivity is better for the 6x4 arrangement as the secondary lobes are smaller than for the 4x6 one. This is because it allows filling a disk so that the antennas are at more or less equal distances from each other.
We can also notice that increasing the size of the array also increases the resolution of the detection of the reflected signal at the surface of the Earth. Another observation is the fact that for an equal density when we increase the number of elements of the array we can observe a increase in the level of the main lobe and a decrease in the level of the secondary lobes. Another thing we can notice is that the results for networks composed of a low number of elements are not promising. Indeed for arrays of 12 elements, it is preferable to stay at separation distances of the $\lambda/2$ order because the grating lobes are still too large and the GRS arrangement is not yet fully efficient for these cases. The best solution for this case is perhaps to realize the antenna array on a single CubeSat. This would allow us to have better results and it would make it easier to arrange and synchronize.

With all these results we can conclude that the best compromise found so far between performance and feasibility is the GRS configuration[1] consisting of 25 antennas with a density of 1 antenna per square meter. Indeed, this arrangement has a primary lobe whose intensity is 3 times greater than that of the secondary lobes (see Figure 6.11a). They have secondary lobes perpendicular to the primary which are easily removed. The low density allows a safety distance between the satellites which is more than sufficient and also allows the best resolution. The only disadvantage of this arrangement is the budget as the number of satellites is quite large. Indeed, our project aims to really send satellites into space but we had planned to only send between 4 and 12 satellites in the first instance. A solution would be to create a GRS layout where we would mount several antennas per CubeSats, this would achieve similar results to those where the density was of 50 antennas per square meter. This solution would also facilitate the synchronization between the antennas.

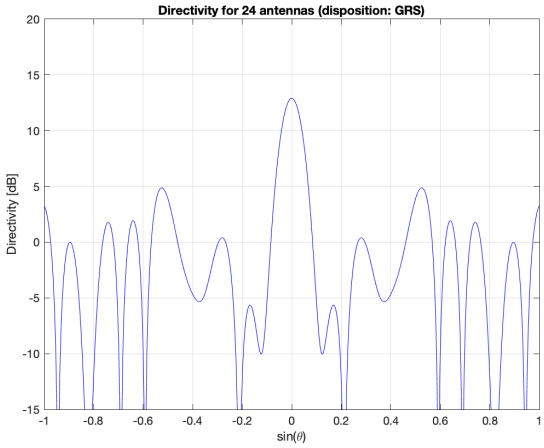
[1]The GRS layout already allows us to do a lot of things but it is not perfect and it is still experimental. However, there is an algorithm created by the ICTEAM which allows to create large arrays of optimal non-regular antennas. Indeed, this algorithm allows to have arrays with minimal distance between the antennas, with optimized main lobe and side lobe sizes. It is therefore possible to find the ideal layout for our application but this algorithm was not used in this study.



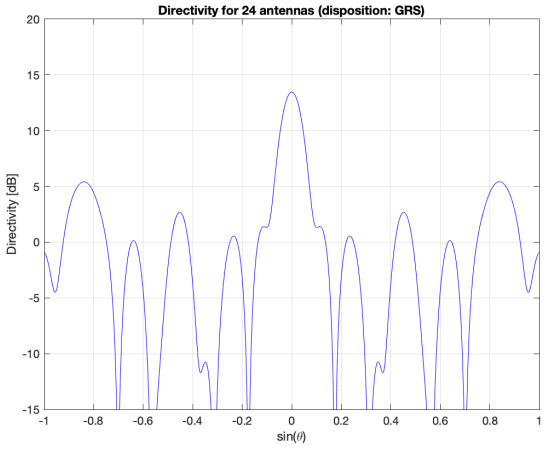
(a) N = 5x5; D = 5



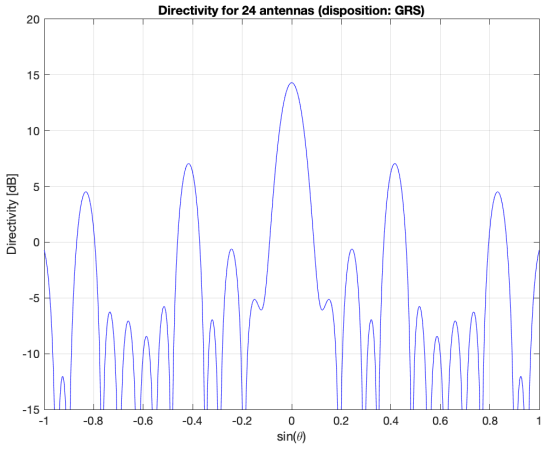
(b) N = 4x6; D = 5



(c) N = 6x4; D = 5



(d) N = 3x8; D = 5



(e) N = 8x3; D = 5

Figure 6.13: Directivity of GRS antenna array layout for a fixed $N = 24$ or $N = 25$, a fixed $D = 5$ but for varying N_{arms}

6.4 Beam steering method

One of the challenges of our application is the fact that our antenna array is in orbit, i.e. we cannot easily change its configuration depending on the situation. So we used a method to ensure that our antennas always "point" to the specular area without changing their position. This is the method of Beam steering [5]. Beam steering is a technique to change the direction of the main lobe of a radiation pattern. In radio and radar systems, beam steering can be achieved by switching antenna elements or by changing the relative phases of the RF signals that drive the elements.
Thematically, this is equivalent to introducing a phase shift between the antennas so that the main lobe is always directed towards the area of interest.
Let's remember the definition of the radiation pattern:

$$\vec{F}(\vec{u}) = \sum_n \vec{F_n}(\vec{u}) = \sum_n \exp(jk\ \vec{\rho_n} \cdot \vec{u})$$

Let u_0 be the direction we want to point to, we just need to introduce the following phase shift to steer our array to the new direction.

$$\vec{F}(\vec{u}) = \sum_n \vec{F_n}(\vec{u}) = \sum_n \exp(jk\ \vec{\rho_n} \cdot \vec{u}) * \exp(-jk\ \vec{\rho_n} \cdot \vec{u_0})$$

This will allow us to have an array of orbiting antennas whose layout is fixed but whose pointing direction can be changed as needed. Figure 6.14 illustrates the method of beam steering, we can see that the lobes of the radiation pattern are directed towards a specific direction. On this figure we can also see the symmetry with respect to the horizontal plane which gives us 2 primary lobes. One of them will essentially disappear through multiplication with the element pattern.

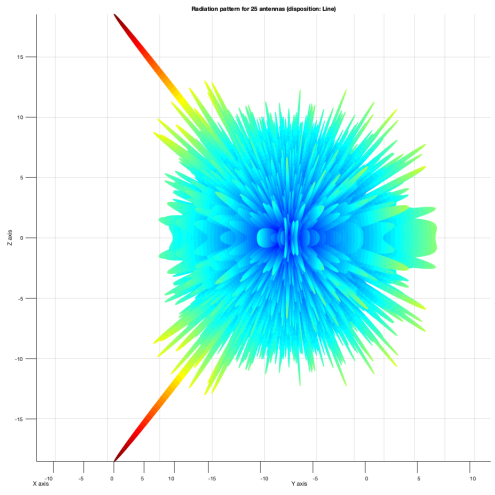


Figure 6.14: Beam steering methode: Radiation pattern

7 Conclusion

In short, the goal of this ambitious project is for the University of Louvain to send its own satellites into space within 10-15 years to observe the Earth. In this thesis, which serves as a prelude to this project, we have defined the type of satellite that will be used to observe the Earth: the CubeSat. We have also determined the method of observation: GNSS reflectrometry.

Then we used 3 different models to represent the shape of the planet in our simulations: the spherical model, the ellipsoidal model and the geoid model where we decomposed the topography of the Earth in series of spherical harmonics. On these different models we have detailed the methods used to locate the specular point, that is the point on the surface of the planet where the reflection of the GNSS signal takes place. The methods for locating this point on the sphere and ellipsoid are similar. However, for the geoid, this method is complicated and requires an additional step to allow better accuracy. This method is however too accurate compared to the accuracy of the satellite antenna signals. Indeed, because of the rough surface of the Earth, this point observable by satellite antennas is actually a zone called glistening zone. Since the results for the ellipsoidal model are quite close to the results for the geoid and that in reality we are dealing with a specular zone it may not be cost effective to implement this additional precision.

Another part of this thesis was dedicated to antennas and more particularly to the configuration of an antenna array. After having detailed the characteristics of the antennas we studied the impact of the layout of the antennas within an array. We studied the impact of the number of antennas in an array as well as the impact of the distance between the antennas to help us find an efficient configuration. We then studied one configuration in particular, the GRS (Golden Ratio Spiral) configuration inspired by the golden ratio. On this configuration we studied the influence of several parameters: the number of antennas, the density of antennas and the distribution of antennas within the array. Based on these results a particular GRS configuration was better than the others. The latter is composed of 25 antennas with a density of 1 antenna per square meter. Indeed, for a sufficient safety distance between the CubeSats, the ratio between the level of the primary lobe and the level of the secondary lobes was the best. Moreover, it had the best resolution of the image observed on the ground. The only drawback is the budget related to the large number of satellites needed. Indeed, the results for antenna arrays with small numbers of antennas were less promising because they required a very small distance between the antennas (of the order of half a wavelength $= \approx 10$ cm). A trivial solution to this problem would be for example to equip several antennas on a single CubeSat.

Finally, we explained the operation and implementation of the beam steering method that will allow us to have an orbiting antenna array whose layout is fixed but whose pointing direction can be changed as needed. This method is essential for our application because it is difficult to change the configuration of the CubeSats in space.

This concludes the introductory phase of the ambitious project at the University of Louvain. The next phase of the project consists of further research and the start of experimental work. It would now be useful to: test the different antenna configurations in real life, to build a device to deploy the antennas, to find a solution for wireless synchronization of the antennas in space and much more...

The bottom line is that this wonderful project is just starting and there is still a long way to go.

References

[1] Dragonfly aerospace. *How Earth Observations are Used To Protect our Planet*. URL: https://dragonflyaerospace.com/how-earth-observations-are-used-to-protect-our-planet-dragonfly-aerospace/. (accessed: 15.07.2022).

[2] Bridgewaytech.org. *CubeSat*. URL: https://bridgewaytech.org/index.php/2021/12/04/1026/. (accessed: 15.07.2022).

[3] Frédéric Chambat. *Les harmoniques sphériques*. URL: https://planet-terre.ens-lyon.fr/ressource/harmoniques-spheriques.xml. (accessed: 15.11.2021).

[4] Thibault Clavier et al. "A Global-Local Synthesis Approach for Large Non-Regular Arrays". In: *IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION* 62.4 (2014), pp. 1596–1606. DOI: 10.1109/TAP.2013.2284816.

[5] Wikimedia Foundation. *Beam Steering*. URL: https://en.wikipedia.org/wiki/Beam_steering. (accessed: 15.07.2022).

[6] Wikimedia Foundation. *Diagramme de Rayonnement*. URL: https://fr.wikipedia.org/wiki/Diagramme_de_rayonnement. (accessed: 12.05.2022).

[7] Wikimedia Foundation. *Ellipsoïde*. URL: https://fr.wikipedia.org/wiki/Ellipso%C3%AFde. (accessed: 30.09.2021).

[8] Wikimedia Foundation. *GNSS reflectometry*. URL: https://en.wikipedia.org/wiki/GNSS_reflectometry. (accessed: 15.10.2021).

[9] Wikimedia Foundation. *Harmonique sphérique*. URL: https://fr.wikipedia.org/wiki/Harmonique_sph%C3%A9rique. (accessed: 20.10.2021).

[10] Wikimedia Foundation. *Spherical coordinate system*. URL: https://en.wikipedia.org/wiki/Spherical_coordinate_system. (accessed: 20.09.2021).

[11] A. Guissard and C. Craeye. *Antennes et propagation, chapitre 1 chapitre 2*. (accessed: 30.03.2021). 20XX.

[12] Christian Hirt. *Earth2014 1 arcmin global topography and relief models*. URL: https://ddfe.curtin.edu.au/models/. (accessed: 21.11.2021).

[13] BUSINESS STRANDARD. *WHAT IS EARTH OBSERVATION SATELLITES?* URL: https://www.business-standard.com/about/what-is-earth-observation-satellites#collapse. (accessed: 15.07.2022).

Appendices

A Matlab code 1: Approximation of the Earth's radius using spherical harmonics

```
1
2  function bool = earth_geiod_harmonics()
3      bool = false;
4      %Data
5
6      fileID = ...
           fopen('/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Spatial/Earth2014Shape_minus_6371000m.TBI2014.5min.
           %download real Earth topography
7      Earth_topography = fread(fileID,[4320 2160],'int16','ieee-be');
8      Earth_topography = (6371000 + Earth_topography') / 1000; % radius km, transpose to get ...
           [theta, phi]
9
10     Etopo = zeros(2160,4320);
11     Etopo(:,1:2160) = Earth_topography(:,2161:4320);
12     Etopo(:,2161:4320) = Earth_topography(:,1:2160); % [-180;180] -->[0;360]
13
14     %Earth_topo = Etopo;
15     %save('/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Earth_topo.mat','Earth_topo'); %save ...
           for futur use
16
17     degree = 75; %accuracy/number of harmonics
18     MAX = max(Etopo,[],'all'); %raidus max
19
20     %Angles
21
22     t= 180-5/120:-5/60:5/120; %colatitude
23     p= 5/120:5/60:360-5/120; %longitude
24     theta = t'*pi/180;
25     phi = p*pi/180; %deg to rad
26     SINtheta = sin(theta)*ones(size(phi));
27
28     %Step
29
30     dtheta = (5/60)*pi/180;
31     dphi = (5/60)*pi/180;
32
33     %Function to approach
34
35     rayon = Etopo;
36
37     %Coefficients & function
38
39     coeff = zeros(degree+1,2*degree +1);
40
41     %%%%%%
42
43     % Load harmonic series coefficients already computed
44     %harm_coeff_struct = ...
           load('/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Spatial/harm_fun_coeff_d75.mat', ...
           'coeff_harm');
45     %coeff = harm_coeff_struct.coeff_harm;
46
47     %%%%%%
48
49     rayon_harm = zeros(size(theta,1),size(phi,2));
50
51     for i = 1:degree+1
52         for j = 1:2*i-1
53             Yij = normalised_harmonics(i-1,-1*i+j,theta, phi);
54             coeff(i,j) = sum(conj(Yij).*rayon.*SINtheta*dtheta*dphi,'all');
55             rayon_harm = rayon_harm + coeff(i,j) * Yij;
56         end
57     end
58
59     %Plot geoid (spherical to cartesian)
```

```
60
61      %%% Plot1 : difference between real and appr. map %%%
62
63      x = ones(size(t))'*p;
64      y = t'*ones(size(p));
65      z = real(rayon_harm)-Etopo;
66      pcolor(x,y,z);
67      set(gca,'ydir','reverse');
68      shading interp
69      colorbar
70
71      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
72
73      %%% Plot 2: 2D map (real and appr. map) %%%
74
75      x = ones(size(t))'*p;
76      y = t'*ones(size(p));
77      z1=Etopo;
78      z2=real(rayon_harm);
79
80      subplot(211)
81      pcolor(x,y,z1);
82      xlabel('longitude')
83      ylabel('colatitude')
84      xlim([0 360])
85      ylim([0 180])
86      title('Real Topography')
87      set(gca,'ydir','reverse');
88      shading interp
89      colormap(jet)
90      colorbar
91
92      subplot(212)
93      pcolor(x,y,z2);
94      xlabel('longitude')
95      ylabel('colatitude')
96      xlim([0 360])
97      ylim([0 180])
98      title('Approximated Topography')
99      set(gca,'ydir','reverse');
100     shading interp
101     colormap(jet)
102     colorbar
103
104     %save files
105
106     %saveas(gcf,['/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Spatial/topo/plot_d_',num2str(degree),'.png'])
107     %saveas(gcf,['/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Spatial/topo/plot_d_',num2str(degree),'.svg'])
108
109     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110
111     %%% Plot 3: 3D map (appr. map) %%%
112
113     %r = Etopo;
114
115     r = real(rayon_harm);
116     x = r.*(sin(theta)*cos(phi));
117     y = r.*(sin(theta)*sin(phi));
118     z = r.*(cos(theta)*ones(size(phi)));
119     mesh(x,y,z,r);
120     shading interp
121     axis equal
122     view(90,0)
123     colormap(jet)
124     colorbar
125
126     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
127
128     %%%%% SAVE HARMONIC FUNCTION OF DEGREE D %%%%
129
130     %coeff_harm = coeff;
```

```
131     %save('/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Spatial/harm_fun_coeff_d5.mat','coeff_harm');
132
133     %dist_geocenter_harm = real(rayon_harm);
134     %save('/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/harm_fun_d150.mat','dist_geocenter_harm');
135
136     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
137
138     %%%%% LOAD HARMONIC FUNCTION OF DEGREE D %%%%
139
140     %harm_fun_struct = load('/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/harm_fun_d150', 'a');
141     %mat_harm_fun = harm_fun_struct.a;
142
143     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
144
145
146
147     %%% Harmonics creator
148
149     function Ylm = normalised_harmonics(l,m, theta, phi)
150         norm = (-1)^(0.5*(m+abs(m)))*sqrt( (2*l+1)/(4*pi) * ...
                factorial(l-abs(m))/factorial(l+abs(m)) );
151         Pl = legendre(l,cos(theta));
152         Plm = Pl(abs(m)+1,:);
153         Ylm =  norm * Plm'*exp(1i*m*phi) ;
154     end
155
156     %%% Harmonics matrix
157
158     function Matrix_Y = mat_normalised_harmonics(degree, theta, phi)
159         Matrix_Y = zeros(degree+1,2*degree +1) ;
160         for i = 1:degree+1
161             for j = 1:2*i-1
162                 Matrix_Y(i,j) = normalised_harmonics(i-1,-1*i+j,theta, phi);
163             end
164         end
165     end
166 end
```

B Matlab code 2: Location of the specular point on a spherical harmonics approximated Earth

```
1
2  % [cart_coord6,polar_coord6] = search_specular(30, 60, 7500, 60, 50.0, 7500, 25, false); %Kazakhstan
3  % [cart_coord,polar_coord] = search_specular(40, -5, 7500, 60, 15, 7500, 5, false); %Belgium
4  [cart_coord,polar_coord] = search_specular(30, -5, 15000, 37, 7, 7500, 5, true);
5
6  function [cart_coord,polar_coord] = search_specular(lat_GNSS, long_GNSS, alt_GNSS, lat_C, ...
       long_C, alt_C, degree, plotBoolean)
7
8      %%% Data %%%
9
10     intermediaryValues = zeros(1,2);
11     intermediaryAngles = zeros(1,2);
12
13     max_radius = 6385;
14
15     % Load harmonic series coefficients
16     harm_coeff_struct = ...
           load('/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Spatial/harm_fun_coeff_d100.mat', ...
           'coeff_harm'); %load coefficients computed before hand of degree 100
17     coeff = harm_coeff_struct.coeff_harm;
18
19     %spherical --> cartesian
20
21     % GNSS vector
22     GNSS_0 = [ alt_GNSS * cosd( lat_GNSS ) * cosd( long_GNSS ), alt_GNSS * cosd( lat_GNSS ) * ...
           sind( long_GNSS ), alt_GNSS * sind( lat_GNSS ) ] ;
23
24     %Cubesat vector
25     C_0 = [ alt_C * cosd( lat_C ) * cosd( long_C ), alt_C * cosd( lat_C ) * sind( long_C ), ...
           alt_C * sind( lat_C ) ] ;
26
27     % Directing vector joigning the 2 satellites
28     v0 = GNSS_0 - C_0;
29
30     %%%%%%%%%%%%
31
32
33     %%% Test : Satellites in fiels of vision ? %%%
34
35     fhandle = @fsolve_function;
36     [S,fval,exitflag] = fsolve (fhandle, [C_0(1), C_0(2),C_0(3), 0]);
37     function F = fsolve_function (Y)
38         x1 = Y(1);
39         y1 = Y(2);
40         z1 = Y(3);
41         t = Y(4);
42         equation1 = x1 - v0(1)* t - C_0(1);
43         equation2 = y1 - v0(2)* t - C_0(2);
44         equation3 = z1 - v0(3)* t - C_0(3);;
45         equation4 = x1 * v0(1) + y1 * v0(2) + z1 * v0(3) ;
46     F = [equation1, equation2, equation3, equation4];
47     end
48
49     length1 = norm(S(1:3) - C_0);
50     length2 = norm(S(1:3) - GNSS_0);
51     % 2 conditions to test
52     if (norm(S(1:3)) ≤ max_radius) && (norm(v0) > length1 && norm(v0) > length2)
53        %f= msgbox('Not in field of vision');
54        disp(norm(S(1:3)));
55        disp(norm(v0));
56        disp(length1);
57        disp(length2);
58        fprintf('No specular point found for GNSS = (%.2f ; %.2f) and Cubesat = (%.2f ; %.2f), ...
              GNSS and Cubesats are not in vision \n',lat_GNSS, long_GNSS, lat_C, long_C);
59        %msg = 'Algorithm ended';
60        %error(msg)
```

```
61      end
62
63      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
64
65      %%%%%%% Find specular point for ellipse %%%%%%
66      % fermat's principle
67      %fmincon
68      lb = [-90.0, -180.0];
69      ub = [90.0, 180.0];
70      %function to minimise
71      fun = @(x) norm(GNSS_0 - [ ellipse_radius(x(1), x(2)) * cosd( x(1) ) * cosd( x(2) ), ...
            ellipse_radius(x(1), x(2)) * cosd( x(1) ) * sind( x(2) ), ellipse_radius(x(1), x(2)) * ...
            sind( x(1) ) ]) + norm(C_0 - [ ellipse_radius(x(1), x(2)) * cosd( x(1) ) * cosd( x(2) ), ...
            ellipse_radius(x(1), x(2)) * cosd( x(1) ) * sind( x(2) ), ellipse_radius(x(1), x(2)) * ...
            sind( x(1) ) ]) ;
72      %fun = @(x) norm(GNSS_0 - [ geoid_radius_harm(x(1), x(2)) * cosd( x(1) ) * cosd( x(2) ), ...
            geoid_radius_harm(x(1), x(2)) * cosd( x(1) ) * sind( x(2) ), geoid_radius_harm(x(1), ...
            x(2)) * sind( x(1) ) ]) + norm(C_0 - [ geoid_radius_harm(x(1), x(2)) * cosd( x(1) ) * ...
            cosd( x(2) ), geoid_radius_harm(x(1), x(2)) * cosd( x(1) ) * sind( x(2) ), ...
            geoid_radius_harm(x(1), x(2)) * sind( x(1) ) ]) ;
73      %no linear constraints
74      A = [] ;
75      b = [] ;
76      Aeq = [] ;
77      beq = [] ;
78      %r0 = ellipse_radius(lat_C, long_C);
79      x0 = [ lat_C, long_C ] ;
80      function [c,ceq] = constraint(x)
81          c = [] ;
82          ceq=[];
83      end
84      nonlcon = @constraint ;
85      %options = optimoptions('fmincon', 'Display','iter','Algorithm','sqp');
86      polar_coord_0 = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon);
87      disp(polar_coord_0);
88      %cart_coord = [ ellipse_radius(polar_coord(1), polar_coord(2)) * cosd( polar_coord(1) ) * ...
            cosd( polar_coord(2) ), ellipse_radius(polar_coord(1), polar_coord(2)) * cosd( ...
            polar_coord(1) ) * sind( polar_coord(2) ), ellipse_radius(polar_coord(1), ...
            polar_coord(2)) * sind( polar_coord(1) ) ];
89
90
91      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
92
93      %%% Find specular point with local normal %%%
94
95      %fmincon
96      % new approach: minimise the difference between the two vectors
97      % those vectors being:
98      % 1. The bissectrice between GNSS vector and Cubesat vector (linked by specular point)
99      % 2. The local normal of the earth
100     %function to minimise
101
102     vector_specular = @(x) [ geoid_radius_harm(x(1), x(2)) * cosd( x(1) ) * cosd( x(2) ), ...
            geoid_radius_harm(x(1), x(2)) * cosd( x(1) ) * sind( x(2) ), geoid_radius_harm(x(1), ...
            x(2)) * sind( x(1) ) ];
103     GNSS = @(x) GNSS_0 - vector_specular(x);
104     C = @(x) C_0 - vector_specular(x);
105     bissectrice = @(x) GNSS(x)/norm(GNSS(x)) + C(x)/norm(C(x));
106     bissectrice_norm = @(x) bissectrice(x)/norm(bissectrice(x));
107     normal_local = @(x) cross(vector_specular([x(1),x(2)+0.005]) - ...
            vector_specular([x(1),x(2)-0.005]) , vector_specular([x(1)+0.005,x(2)])- ...
            vector_specular([x(1)-0.005,x(2)]));
108     normal_local_norm = @(x) normal_local(x)/norm(normal_local(x));
109     fun = @(x) norm(bissectrice_norm(x) - normal_local_norm(x));
110
111     x0 = polar_coord_0 ;
112     options = optimset('OutputFcn', @outfunction);
113     %options = optimoptions('fmincon', 'Display','iter','Algorithm','sqp');
114     A = []; b = [];
115     Aeq = []; beq = [];
116     lb = [-90.0 -180.0]; ub = [90.0 180.0];
```

```
117     nonlcon = [];
118     [polar_coord, fval, exitflag, output] = fmincon(fun,x0, A, b, Aeq, beq, lb, ub, ...
            nonlcon,options);
119     %polar_coord = fminsearch(fun,x0);
120     cart_coord = [ geoid_radius_harm(polar_coord(1), polar_coord(2)) * cosd( polar_coord(1) ) * ...
            cosd( polar_coord(2) ), geoid_radius_harm(polar_coord(1), polar_coord(2)) * cosd( ...
            polar_coord(1) ) * sind( polar_coord(2) ), geoid_radius_harm(polar_coord(1), ...
            polar_coord(2)) * sind( polar_coord(1) ) ];
121
122
123     %%%%%%%%%%%%%%%%%%%%%%%%%%%%
124
125     %%% Plot difference in angles %%%
126
127     figure
128     iteRR = 1:1:size(intermediaryValues(:,2));
129     plot(intermediaryValues(:,2),intermediaryValues(:,1),'-o');
130     %plot(iteRR,intermediaryAngles(:,2),'-o')
131     %xlabel('longitude'), ylabel('latitude')
132     xlabel('iteration'), ylabel('angle in deg')
133     title("difference btw local normal and bissectrice for degree " + degree)
134
135     %%%%%%%%%%%%%%%%%%%%%%%%%%%%
136
137     %%% Plot difference in altitude %%%
138
139     Xlong = polar_coord_0(2)-0.1:0.01:polar_coord_0(2)+0.1;
140     Ylat = polar_coord_0(1)-0.1:0.01:polar_coord_0(1)+0.1;
141     Zalt = ones(size(Ylat,2),size(Xlong,2));
142     for i=1:size(Ylat,2)
143         for j = 1:size(Xlong,2)
144             Zalt(i,j) = geoid_radius_harm(Ylat(i), Xlong(j)) - ellipse_radius(Ylat(i), Xlong(j));
145         end
146     end
147     figure
148     pcolor(Xlong,Ylat,Zalt);
149     hold on
150     plot(intermediaryValues(:,2),intermediaryValues(:,1),'-o');
151     hold on
152     plot(intermediaryValues(end,2),intermediaryValues(end,1),'ro');
153     xlabel('longitude'), ylabel('latitude')
154     title("Altitude for degree " + degree)
155     %mesh(Xlat,Ylong,Zalt,Zalt);
156     shading interp
157     axis equal
158     colormap(jet)
159     hc=colorbar;
160     title(hc,'altitude in km');
161
162     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
163
164     %%% Plot %%%
165
166     if plotBoolean
167
168         t= 180-5/120:-5/60:5/120; %colatitude
169         p= 5/120:5/60:360-5/120; %longitude
170         theta = t'*pi/180;
171         phi = p*pi/180; %deg to rad
172
173         r_struct = ...
                load('/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Spatial/Earth_topo.mat','Earth_topo');
174         r = r_struct.Earth_topo;
175
176         x = r.*(sin(theta)*cos(phi));
177         y = r.*(sin(theta)*sin(phi));
178         z = r.*(cos(theta)*ones(size(phi)));
179
180         P = cart_coord;
181         Pol = polar_coord;
182         B = bissectrice_norm(Pol);
```

```
183         N = normal_local_norm(Pol);
184         GNSS_unit = (GNSS(Pol))/norm(GNSS(Pol));
185         C_unit = (C(Pol))/norm(C(Pol));
186 
187         mesh(x,y,z,r);
188         shading interp
189         axis equal
190         view(90,-10)
191         colormap(jet)
192         colorbar
193         caxis([6350 6390])
194         pbaspect([1 1 1])
195 
196         hold on
197         % P to GNSS vector
198         p1 = ...
                plot3(GNSS_0(1),GNSS_0(2),GNSS_0(3),'gs','MarkerSize',10,'MarkerEdgeColor','r','MarkerFaceColor','r')
199         p3 = ...
                plot3(C_0(1),C_0(2),C_0(3),'gs','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','b');
200         plot3([P(1) GNSS_0(1)],[P(2) GNSS_0(2)],[P(3) GNSS_0(3)],'r-', 'LineWidth',2);
201         hold on
202         % P to Cubesat vector
203         plot3([P(1) C_0(1)],[P(2) C_0(2)],[P(3) C_0(3)],'b-', 'LineWidth',2);
204         hold on
205         % Bissectrice
206         %plot3([P(1) P(1)+2000*B(1)],[P(2) P(2)+2000*B(2)],[P(3) P(3)+2000*B(3)],'b-^', ...
                'LineWidth',3);
207         %plot3([P(1) 1.5*P(1)],[P(2) P(2)*1.5],[P(3) P(3)*1.5],'b-^', 'LineWidth',3);
208         hold on
209         % Normal local
210         %plot3([P(1) P(1)+2000*N(1)],[P(2) P(2)+2000*N(2)],[P(3) P(3)+2000*N(3)],'m-^', ...
                'LineWidth',3);
211         %plot3([P(1) 1.5*P(1)],[P(2) P(2)*1.5],[P(3) P(3)*1.5],'b-^', 'LineWidth',3);
212         hold on
213         % Directing vector of line joigning 2 sats
214         %plot3([GNSS_0(1) C_0(1)],[GNSS_0(2) C_0(2)],[GNSS_0(3) C_0(3)],'m-^', 'LineWidth',3);
215         hold on
216 
217         xlabel('X axis'), ylabel('Y axis'), zlabel('Z axis')
218         title('Specular Point Search')
219         legend([p1 p3],{'GNSS','CubeSat'})
220         %legend('GNSS','Cubesats')
221         %legend('','GNSS','Cubesats','Point sp culaire','Normal')
222         %saveas(gcf,'/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/TFE_report/Specular_Matlab_Geiod.png')
223     end
224 
225     %%%%%%%%%%%%
226 
227     % Option function to display intermediary values of optimisation
228 
229     function stop = outfunction(x, optimValues, state)
230         stop = false;
231         hold on;
232 
233         intermediaryValues(optimValues.iteration+1,:) = x;
234 
235         po = 6356.7;
236         q = 6378;
237 
238         normal_global = @(x) 2 * [ 1/q * ellipse_radius(x(1), x(2)) * cosd( x(1) ) * cosd( x(2) ...
                ), 1/q * ellipse_radius(x(1), x(2)) * cosd( x(1) ) * sind( x(2) ), 1/po * ...
                ellipse_radius(x(1), x(2)) * sind( x(1) ) ]; %gradient to take into account
239         normal_global_norm = @(x) normal_global(x)/norm(normal_global(x));
240 
241         angle1 = @(x) acosd(dot(normal_global_norm(x), normal_local_norm(x)));
242         angle2 = @(x) acosd(dot(bissectrice_norm(x), normal_local_norm(x)));
243 
244         intermediaryAngles(optimValues.iteration+1,:) = [angle1(x),angle2(x)] ;
245 
246         fprintf('iteration : %d = (%.4f ; %.4f) \n', optimValues.iteration, x(1), x(2));
247         fprintf('difference between local and global normal = %.3f  deg \n', angle1([x(1),x(2)]));
```

```
248         fprintf('difference between local normal and bissectrice = %.3f  deg \n', ...
                angle2([x(1),x(2)]));
249         alt_specular = geoid_radius_harm(x(1),x(2)) - ellipse_radius(x(1), x(2));
250         alt_specular_1 = geoid_radius_harm(x(1)+0.01,x(2)) - ellipse_radius(x(1)+0.01, x(2));
251         alt_specular_2 = geoid_radius_harm(x(1),x(2)+0.01) - ellipse_radius(x(1), x(2)+0.01);
252         fprintf('difference in altitude = %.4f  m and %.4f  m \n',1000*abs(alt_specular - ...
                alt_specular_1), 1000*abs(alt_specular - alt_specular_2)) ;
253         %plot(optimValues.iteration, angle2([x(1),x(2)]),'o');
254         %drawnow
255
256     end
257
258
259     %%%%%%%%%%%%
260     % Earth radius in function of latitude and longitude
261     function geoid_radii_harm = geoid_radius_harm(lat, long)
262         % lat --> colat
263         colat = 90 - lat ;
264         % deg to rad
265         colat = colat *pi/180.0;
266         long = long *pi/180.0;
267         geoid_radii_harm = real(sum( coeff(1:degree+1,1:2*degree +1) .* ...
                mat_normalised_harmonics(degree, colat, long),'all'));
268     end
269
270 end
271
272
273 % Ellipse radius in function of latitude and longitude
274 function ellipse_radii = ellipse_radius(lat, long)
275     % lat --> colat
276     p = 6356.7;
277     q = 6378;
278     colat = 90 - lat ;
279     colat = colat *pi/180.0;
280     long = long *pi/180.0;
281     ellipse_radii = sqrt(q^2*(sin(colat))^2 + p^2*(cos(colat))^2);
282 end
283
284 function Matrix_Y = mat_normalised_harmonics(degree, theta, phi)
285     Matrix_Y = zeros(degree+1,2*degree +1) ;
286     for i = 1:degree+1
287         for j = 1:2*i-1
288             Matrix_Y(i,j) = normalised_harmonics(i-1,-1*i+j,theta, phi);
289         end
290     end
291
292 end
293
294 function Ylm = normalised_harmonics(l,m, theta, phi)
295     norm = (-1)^(0.5*(m+abs(m)))*sqrt( (2*l+1)/(4*pi) * factorial(l-abs(m))/factorial(l+abs(m)) );
296     Pl = legendre(l,cos(theta));
297     Plm = Pl(abs(m)+1,:);
298     Ylm =  norm * Plm'*exp(1i*m*phi) ;
299 end
```

C Matlab code 3: Calculation of the radiation pattern, directivity, gain, ground spot, etc.

```
1
2  %Data
3
4  center_array_sats=[7500*cosd(90)*sind(0),7500*cosd(90)*cosd(0),7500*sind(90)]; % coordinates of ...
       the center of the antenna array
5  ratio_angle = 7500 * pi /180; %km per degree of lat/long
6  ratio_dist = 1 /ratio_angle; %degree of lat_long equivalent to 1km
7  specular_cart_coord=[6371*cosd(90)*sind(0),6371*cosd(90)*cosd(0),6371*sind(90)]; % cartesian ...
       coordinates of the specular point
8  u0= (center_array_sats - specular_cart_coord)/norm(center_array_sats - specular_cart_coord); % ...
       direction of interest for beam steering method
9
10 % Sats layout 1 : Simple dipole antenna
11 dipole_pos = zeros(2,3);
12 dipole_pos(1,:) = [0,0.19/4,7500];
13 dipole_pos(2,:) = [0,-0.19/4,7500];
14
15 % Sats layout 2 : Very large array telescope
16 VeryLargeArray = zeros(12,3);
17 VeryLargeArray(1,:) = [1.5*cosd(30),1.5*sind(30),0];
18 VeryLargeArray(2,:) = [1.5*cosd(150),1.5*sind(150),0];
19 VeryLargeArray(3,:) = [1.5*cosd(270),1.5*sind(270),0];
20 VeryLargeArray(4,:) = [3.0*cosd(30),3.0*sind(30),0];
21 VeryLargeArray(5,:) = [3.0*cosd(150),3.0*sind(150),0];
22 VeryLargeArray(6,:) = [3.0*cosd(270),3.0*sind(270),0];
23 VeryLargeArray(7,:) = [4.5*cosd(30),4.5*sind(30),0];
24 VeryLargeArray(8,:) = [4.5*cosd(150),4.5*sind(150),0];
25 VeryLargeArray(9,:) = [4.5*cosd(270),4.5*sind(270),0];
26 VeryLargeArray(10,:) = [6.0*cosd(30),6.0*sind(30),0];
27 VeryLargeArray(11,:) = [6.0*cosd(150),6.0*sind(150),0];
28 VeryLargeArray(12,:) = [6.0*cosd(270),6.0*sind(270),0];
29
30 % Sats layout 3 : X antenna array
31 X_pos = zeros(12,3);
32 X_pos(1,:) = [3.0*cosd(45),3.0*sind(45),0];
33 X_pos(2,:) = [3.0*cosd(135),3.0*sind(135),0];
34 X_pos(3,:) = [3.0*cosd(225),3.0*sind(225),0];
35 X_pos(4,:) = [3.0*cosd(315),3.0*sind(315),0];
36 X_pos(5,:) = [9.0*cosd(45),9.0*sind(45),0];
37 X_pos(6,:) = [9.0*cosd(135),9.0*sind(135),0];
38 X_pos(7,:) = [9.0*cosd(225),9.0*sind(225),0];
39 X_pos(8,:) = [9.0*cosd(315),9.0*sind(315),0];
40 X_pos(9,:) = [15.0*cosd(45),15.0*sind(45),0];
41 X_pos(10,:) = [15.0*cosd(135),15.0*sind(135),0];
42 X_pos(11,:) = [15.0*cosd(225),15.0*sind(225),0];
43 X_pos(12,:) = [15.0*cosd(315),15.0*sind(315),0];
44
45 % Sats layout 4 : Inline antenna array
46 Line_pos = zeros(12,3);
47 for i=1:12
48     Line_pos(i,:) = [(-6.5+i)*(1.0*0.19),0,0];
49 end
50 for ii=1:12
51     newLine(ii,:)=Line_pos(ii,:)+center_array_sats; %Real layout in space
52 end
53
54 % Sats layout 5 : Square antenna array
55 side=5;
56 Square_pos = zeros(side^2,3);
57 for i=1:side
58     Square_pos(i,:) = [(-3+i)*(0.19/2),2*0.19/2,0];
59     Square_pos(i+side,:) = [(-3+i)*(0.19/2),0.19/2,0];
60     Square_pos(i+2*side,:) = [(-3+i)*(0.19/2),0,0];
61     Square_pos(i+3*side,:) = [(-3+i)*(0.19/2),-1*0.19/2,0];
62     Square_pos(i+4*side,:) = [(-3+i)*(0.19/2),-2*0.19/2,0];
63 end
```

```
for ii=1:side^2
    newSquare(ii,:)=Square_pos(ii,:)+center_array_sats;%Real layout in space
end

% Sats layout 6 : Random antenna array
Rand_pos = zeros(12,3);
for i=1:12
    Random1 = rand;
    Random2 = rand;
    Rand_pos(i,:) = [20.0*Random1*cosd(360*Random2),20.0*Random1*sind(360*Random2),0];
end

% Sats layout 7 : GRS antenna array
%Rmax=5; % choose value of the radius of the disk containing all the satellites
Density=5;
N_arm=4;
N_per_arm=4;
Rmax = sqrt(N_arm*N_per_arm/(pi*Density)); %Radius in function of Density and N_sats
theta = zeros(N_arm,N_per_arm);
radii=linspace(0.25 ,Rmax,N_per_arm); % distance of antenna from the center of the disks, allows ...
    equal distribution
theta(1,:)=(pi/2)*log(radii); % theta in function of radius
Δ=zeros(N_per_arm-1,1);

for i=2:N_arm
        theta(i,:)=theta(1,:) + (i-1)*2*pi/N_arm; %phase shift to have several arms equally ...
            distributed
end

r=exp(2*theta(1,:)/pi); % equation of the golden spiral

%compute distance between antenna of a same arm
for ii=1:(N_per_arm-1)
    Δ...
        (ii)=((r(ii+1)*cos(theta(1,ii+1))-r(ii)*cos(theta(1,ii)))^2+(r(ii+1)*sin(theta(1,ii+1))-r(ii)*sin(theta(1
end

GRS=[reshape(r'.*cos(theta'),[],1), reshape(r'.*sin(theta'),[],1),zeros(N_arm*(N_per_arm),1)]; ...
    %from 2D to 3D

newGRS= GRS;
for ii=1:N_arm*(N_per_arm)
    newGRS(ii,:)=GRS(ii,:)+center_array_sats; %Real layout in space
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sats_pos =  newGRS; % choice of layout for the computations
number_sat = size(sats_pos,1);

t = 0.1:0.1:180; %elevation
p = 0.1:0.1:360; %azimuth
theta = t*pi/180;
phi = p'*pi/180; %deg to rad

width=10; %km width of the specular zone

[F,D,G,int] = ray_diagram(sats_pos); % radiation pattern, directivity, gain and integral ...
    computation based on satellites layout

% ground spot and radiation pattern computation based on specular point coord., antenna array ...
    location, satellites layout, width of specular zone and integral value
[P,Fe] = ground_spot(specular_cart_coord, center_array_sats, sats_pos, width,int);
p = 10*log10(P); %in dB

% Plot 1: Ground spot
x = linspace(-(width*10+1)/2,(width*10+1)/2,width*10+1);
y = linspace(-(width*10+1)/2,(width*10+1)/2,width*10+1);
[X,Y] = meshgrid(x,y);
figure
contourf(X,Y,p,10,'ShowText','on')
```

```
title(['Ground spot for ',num2str(number_sat),' antennas (disposition: GRS)'])
%saveas(gcf,['/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Antenna/GRS_report/ground_spot_',num2str(number_sat)

% Plot 2: Radiation pattern in 3D
% Directivity in dB
D = 10*log10(D);

x = F.*(cos(phi)*sin(theta));
y = F.*(sin(phi)*sin(theta));
z = F.*(ones(size(phi))*cos(theta));

figure
mesh(x,y,z,F);
shading interp
axis equal
view(0,0)
colormap(jet)
%colorbar
xlabel('X axis'), ylabel('Y axis'), zlabel('Z axis')
title(['Radiation pattern for ', num2str(number_sat), ' antennas (disposition: Line)'])
%saveas(gcf,['/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Antenna/Results_report_Line_Square_vary_lambda/ray_d

% Plot 3: Radiation pattern in 2D
x2D = F(1800,1:900).*cos(theta(1:900));
y2D = F(1800,1:900).*sin(theta(1:900));

figure
plot(y2D,x2D,'b');
hold on
plot(-y2D,x2D,'b');
axis equal
grid on
xlabel('X axis'), ylabel('Y axis'),
title(['Radiation pattern 2D for ', num2str(number_sat), ' antennas (disposition: Line)'])
%saveas(gcf,['/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Antenna/Results_report_Line_Square_vary_lambda/ray_d

% Plot 4: Directivity in function of the broadside angle to analyse secondary lobes
figure
plot(sin(theta(1:1800)),D(1800,1:1800),'b')
hold on
plot(-sin(theta(1:1800)),D(1800,1:1800),'b')
grid on
xlabel('sin(\theta)'), ylabel('Directivity [dB]'),
title(['Directivity for ', num2str(number_sat), ' antennas (disposition: GRS)'])
%saveas(gcf,['/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Antenna/GRS_report/directivity_2D_',num2str(number_s

% Plot 5: Radiation pattern in 2D in a polar plot
figure
polarplot(pi/2-theta, F(1800,:),'b')
hold on
polarplot(pi/2-theta, -F(1800,:),'b')
ax = gca;
ax.RTick = 0:2.5:25;
%colorbar
%xlabel('X axis'), ylabel('Y axis'),
title(['Radiation pattern 2D for ', num2str(number_sat), ' antennas (disposition: Line)'])
%saveas(gcf,['/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Antenna/Results_report_Line_Square_vary_lambda/ray_d

% Plot 6: Array layout in 2D
figure
plot(sats_pos(:,1),sats_pos(:,2),'.','MarkerSize',15)
%hold on
%plot(Rmax*cos(2*theta),Rmax*sin(2*theta),'MarkerSize',5)
xlabel('X axis'), ylabel('Y axis')
grid on
title(['Array layout of ', num2str(number_sat), ' antennas : GRS'])
%saveas(gcf,['/Users/ugopaulduraffourd/Desktop/TFE_2021_2022/Antenna/GRS_report/array_layout_',num2str(number_sat

% radiation pattern, directivity, gain and integral computation based on satellites layout
function [F,D,G,int] = ray_diagram(antenna_pos)

```

```
200     %center_array_sats=[7500*cosd(90)*sind(0),7500*cosd(90)*cosd(0),7500*sind(90)];
201     %specular_cart_coord=[6371*cosd(90)*sind(0),6371*cosd(90)*cosd(0),6371*sind(90)];
202     %u0= (center_array_sats - specular_cart_coord)/norm(center_array_sats - ...
            specular_cart_coord); % to display beam steering method
203
204     j = sqrt(-1);
205     lambda = 0.19; %longueur d'onde
206     k = 2*pi/lambda; %nombre d'onde
207
208     u = zeros(3600,1800,3);
209     F = zeros(3600,1800);
210     for i = 1:3600
211         for l= 1:1800
212             u(i,l,:) = ...
                    [sin(l*pi/1800)*cos(i*pi/1800),sin(l*pi/1800)*sin(i*pi/1800),cos(l*pi/1800)]; ...
                    %unit director vector u
213             for n = 1:size(antenna_pos,1)
214                 % uncomment for beam steering method
215                 % F = array factor = radiation pattern
216                 F(i,l) = F(i,l) + exp(j*k*dot([u(i,l,1), u(i,l,2), ...
                        u(i,l,3)],antenna_pos(n,:)));%*exp(-1*j*k*dot([u0(1), u0(2), ...
                        u0(3)],antenna_pos(n,:)));
217             end
218         end
219     end
220     t= pi/1800:pi/1800:pi; %elevation
221     p= pi/1800:pi/1800:2*pi; %azimuth
222     sin_theta = (ones(size(p'))*sin(t));
223     F = abs(F);
224     integral = trapz(p,trapz(t,F.^2.*sin_theta,2)); %double integral on radiation pattern
225     D = (4*pi/integral)*F.^2; %directivity
226     G = 0.9*D; %gain
227     int=integral;
228 end
229
230 % ground spot and radiation pattern computation based on specular point coord., antenna array ...
        location, satellites layout, width of specular zone and integral value
231 function [G,F] = ground_spot(specular_coord, sat_center_coord, sat_disp, zone_area, integral)
232
233     width = zone_area*10 ;% precision = 100m
234     F = zeros(width+1,width+1); %radiation pattern
235     j = sqrt(-1);
236     lambda = 0.19; %longueur d'onde
237     k = 2*pi/lambda; %nombre d'onde
238
239     u0= (sat_center_coord - specular_coord)/norm(sat_center_coord - specular_coord); %beam ...
            steering method, u0: vector pointing from the center of the array to the specular point
240     u0=-1*u0; %change direction
241     ground_coord = zeros(width+1,width+1);
242     u = zeros(width+1,width+1);
243
244     for i = 1:width+1
245         for l= 1:width +1
246
247             ground_coord(i,l,1) = specular_coord(1)+(i-(width/2+1));
248             ground_coord(i,l,2) = specular_coord(2)+(l-(width/2+1));
249             ground_coord(i,l,3) = specular_coord(3)+ 0;
250
251             u(i,l,1) = ground_coord(i,l,1) - sat_center_coord(1);
252             u(i,l,2) = ground_coord(i,l,2) - sat_center_coord(2);
253             u(i,l,3) = ground_coord(i,l,3) - sat_center_coord(3);
254
255             for n = 1:size(sat_disp,1)
256                 rho = sat_disp(n,:) - sat_center_coord;
257                 F(i,l) = F(i,l) + exp(j*k*dot([u(i,l,1), u(i,l,2), u(i,l,3)]/norm([u(i,l,1), ...
                        u(i,l,2), u(i,l,3)]),rho))*exp(-1*j*k*dot([u0(1), u0(2), u0(3)],rho)); ...
                        %beamsteering method
258             end
259         end
260     end
261
```

```
262       F=abs(F);
263       D = (4*pi/integral)*F.^2; %directivity
264       G = 0.9*D; %gain
265
266   end
```

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**