UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Bachelor's Thesis

New real-time GNSS algorithms for detection and measurement of potential geoeffective stellar flares

Author
David Moreno Borràs

Supervisor
Manuel Hernández-Pajares

Specialization
Computing

May 22, 2019

Abstract

Solar flares are sudden electromagnetic emissions on the Sun's surface that release large amounts of magnetic energy. These flares emit radiation that has an effect on Earth's ionosphere electron content and are detected by telescopes such as Swift or Fermi by performing gamma-ray observations from low Earth orbit. (Swift y fermi no son para solar flares son para GRB en general!!, quizas aqui deberia ir alguno que se centre en el sol especificamente)

Another approach for detecting these events is possible: the ionosphere electron content variation can be studied using data from the Global Navigation Satellite System (GNSS) (such as GPS)

Algorithms for detecting solar flares without knowing the location of the source, that is, the position of the Sun relative to Earth, as well as a study on the feasibility of the detection of such events for the challenging scenario of far-away stars, are presented, aiming to find an alternative detection method to that of the telescopes and using free, open-source data.

Keywords: Solar flares, Stellar flares, GNSS, GPS, IGS, GRB

Contents

Listings

List of Figures

Main

- Cover
- ListOfContents
- ListOfFigures???
- ListOfListings???
- Abstract/Resumen/Resum

Content

- ch1 GEP
- ch2 Background/Introduction
  - Global Navigation Satellite Systems
  - Ionosphere
  - Stellar flares
  - Gamma-Ray Bursts
- ch3 Study on the feasiblity of stellar flare detection (far-away stars)
  - Sources of data and possible candidates
  - The Neil Gehrels Swift Observatory and its data
  - Objective function
  - Obtaining the data
  - Results
- ch4 Solar flare detection
  - Data
    - GNSS Data
    - The Halloween Storm
    - Formatting
  - Vtec distribution
  - Algorithm
  - Results
- ch5 Brute force approach

  - First approach: Brute force

Hasta aqui ya esta mas o menos hecho

- ch6 BGSEES: Optimizations
  - Decrease range method
  - Least squares
  - Hill Climbing
  - OpenMP?

Annexes

- Glossary (Acronyms)
- References

Chapter 1

GEP

merda del gep

Chapter 2

Background

As the project has a large background in physics and astronomy, some of the relevant topics that are going to be studied are introduced: Global Navigation Satellite Systems, the Ionosphere, Stellar Flares and Gamma-Ray Bursts.

2.1 Global Navigation Satellite Systems

GNSS and GPS

These two terms may lead to confusion as Global Navigation Satellite Systems (GNSS) is the generic term for all satellite navigation systems. The Global Positioning System (GPS), in particular, is the United States' GNSS system, the world's most used one. Other systems, for example, are the European Galileo or Russian GLONASS [7].

Global Navigation Satellite Systems use satellites to determine the position of a given object or device in terms of latitude, longitude and height.

Positioning

In short, GNSS works as follows: out of all the GNSS satellites orbiting Earth, at least four of them are constantly visible from a specific point and transmitting information at a certain frequency. When a device receives a signal from one of them, the distance to the satellite can be calculated by means of the time required to reach it and the speed of light. As many variables might affect the speed of light such as the medium through which it is propagating, this estimation of the distance is called **pseudo-range**.

Thus, the location of the receiver can be estimated using a technique called **trilateration**. Having three spheres around each of the satellites with the pseudo-range as their radius, the intersection of these spheres yields the location of the receiver [11].

Ionospheric Pierce Points

Ionospheric Pierce Points (IPP) are going to be very relevant throughout the development of the project. These will be the locations that we are going to use for our measurements, not those of satellites or receivers. A Ionospheric Pierce Point is the point where the line between the satellite and a receiver intersect with the ionosphere, where the Total Electron Content can be estimated. [4]

The International GNSS Service

In 1998 the International GNSS Service (IGS) was created as a collaboration of several members of the scientific community: Center for Orbit Determination in Europe (CODE), (European Space Agency) ESA, Jet Propulsion Laboratory (JPL) and Polytechnic University of Catalonia (UPC). This voluntary federation has made available open access GNSS data since its creation [2] [3].

Its data is provided by more than 300 GPS receivers around the globe and is processed by the previous institutions which compute the global distribution of the Total Electron Content (TEC) [9].

GNSS is a key component to this project because, as mentioned before, many variables can affect the speed of light and therefore the time it takes for the transmitter's signal to be intercepted by the receiver. One of these variables is the electron content of the layer of the atmosphere where GNSS satellites operate: the ionosphere.

2.2 Ionosphere

The ionosphere is a layer of the Earth's atmosphere that lies 75-1000km above the surface of the planet [18].

High energy from Extreme UltraViolet (EUV) and X-ray radiation can cause its atoms to be ionized and create a layer of electrons [15]. Due to these free electrons and ionized molecules, it is capable of affecting radio wave propagation, thus having an effect on Global Navigation Satellite Systems (GNSS) technology, this phenomena allows these satellites to be used as a global scanner for the ionosphere [10].

The main physical quantity used for describing the electron content of the ionosphere is the **Total Electron Content (TEC)**, the TEC is the total number of electrons between two points $(r1, r2)$ along a cylinder of base $1m^2$. Slant TEC (STEC), in particular, can be defined as the TEC in which $r1$ and $r2$ are a satellite and a receiver's positions [17].

The unique properties the ionosphere has enable us to use the data provided by the GNSS technology to study a powerful phenomena that occurs in many stars across the universe: stellar flares.

2.3 Stellar flares

Flares from stars, in particular those that have the Sun as a source, more noticeable due to its proximity, are sudden flashes of brightness in the surface of stars which release large amounts of energy across the whole electromagnetic spectrum[1]. Flares that have the Sun as a source can increase the electron content of the ionosphere and have an effect on waves passing through it, affecting satellite communications and causing a delay. This phenomena is the key element of the project, as it enables us to study these events.

Satellites can also be harmed by this effects: the flares heat up the outer atmosphere, which in turn increases the drag on these satellites reducing their lifetime in orbit. GPS as a solar observational instrument: Real-time estimation of EUV photons flux rate during strong, medium, and weak solar flares se explica el efecto del sol sobre ionosfera

The previous phenomena applies to flares originating in the Sun, whether a flare that has a star from outside the Solar System as a source has an effect on the Earth's atmosphere or not is one of the topic that is going to be studied in this project.

SOHO and GOES are space probes used for obtaining solar flare information [8]

2.4 Gamma-Ray Bursts

Throughout the project, in particular when studying the feasibility of stellar flares detection, another type of event will be mentioned and studied as well: Gamma-Ray Bursts (GRBs):

GRBs are highly energetic explosions that occur in distant galaxies, releasing large amounts of radiation, in particular Gamma rays, hence the name of the event. These bursts, despite being millions of light years away from Earth are so powerful they might still have an impact on the ionosphere, like the aforementioned stellar flares. The main difference with flares originating from stars is that GRBs are thought to be originated from the death of massive stars, that is, supernovas.

This event, despite not being a stellar flare, the phenomena we aim to detect, is going to be studied in the following sections to test the currently working algorithms mainly for two reasons: it has been studied and cataloged by telescopes such as the Fermi Observatory and there's is available information that we can use for our study, and the large amounts of energy they emit it make GRBs a more feasible target to detect.

Fermi Observations of High-Energy Gamma-Ray Emission from GRB 080916C

Although a study on GRBs is presented in the next chapter, other cosmic events that may be detected using our proposed methods can be studied as well later on.

[1] X-rays and Extreme Ultraviolet (EUV) radiation

Chapter 3

Study on the feasibility of stellar flare detection

Flares from far-away stars and Gamma-ray Bursts, albeit more powerful than flares that have the Sun as their source, may not be possible to detect due to the large distances that separate them from our measurement tool: the ionosphere.

Before starting to adapt the algorithm for detecting solar flares to this scenario, a study was conducted parallel to its development, to see if the energy from flares originating in far-away stars could be detected using the already existing method, namely the GNSS Solar Flare Activity Indicator (GSFLAI) algorithms [8].

To study if this was feasible, the algorithms were run on certain candidates of flares and GRBs to see if they could be detected.

The project supervisor, Manuel Hernández-Pajares, who as mentioned before has previously performed several studies on the subject, provided the GSFLAI algorithm to test the candidates. The GSFLAI algorithm takes into consideration the location of the source (the Sun) to see if there's a relation between an increase in the VTEC of the ionosphere and the solar-zenith angle to determine if this increase is caused by a solar flare. [GNSS measurement of EUV photons flux rate during strong and mid solar flares]

However, its execution may take up to 2 hours, because of this the aim was to:

- Find a database for possible candidates, several online archives with information about previously recorded Gamma Ray Bursts were considered.
- Select an appropriate source of this pool of candidates by writing a quick script that yielded an ordered list of the best candidates based on certain factors, instead of selecting a random source.

3.1 Sources of data and possible candidates

The three main databases we considered for the study were:

- The GRB collection website of Jochen Greiner, scientist at the Max-Planck-Institute for extraterrestrial Physics (MPE) [13], which offers a collection of detected GRBs by different telescopes and observatories.
- The Magnetar Outburst Online Catalog (MOOC), developed by the Institute of Space Sciences (CSIC-IEEC, Barcelona) [12]. We also had the pleasure to meet one of the leaders of this project, Nanda Rea, and discuss
- The Neil Gehrels Swift Observatory website and archive by the National Aeronautics and Space Administration (NASA), Goddard Space Flight Center [6] which contains an archive of detected GRBs by the Swift observatory and is constantly updated.

Because of the layout of the website and how the data could be accessed, the option with which we started was the Swift Database, as the data could be visualized in an HTML table and was easily accessible.

3.2 The Neil Gehrels Swift Observatory and its data

The Swift Observatory is a NASA mission with international participation, designed to observe GRBs and their afterglows to study topics such as the origins of GRBs or what they can reveal about the early stages of the universe [16]. The observatory is equipped with three main instruments that work with each other to study GRBs [5] [6]:

- The **Burst Alert Telescope (BAT)**, tasked with detecting the GRBs and computing their positions. This triggers the spacecraft to point the other telescopes to the burst so it can be studied in more detail.
- The **X-ray Telescope (XRT)**, used for studying the X-ray radiation and taking images of the bursts which in turn help increase the accuracy of the location estimation.
- The **UV/Optical Telescope (UVOT)**, which serves a similar purpose to the XRT, but studies the ultraviolet band of the spectrum.

For each detected GRB, the data obtained by the different telescopes is given. The parameters that are relevant to our study and determine the fitness of each of the candidates are:

- The **name of the burst**, given by the date it was detected. For example, the GRB named 190220A was detected the 20th of February of 2019.
- The **Universal Time (UT)** of the detection, that is, hh:mm:ss of the day given by the name.
- The fluence detected by the BAT component, in units of keV. that is.
- The **UVOT magnitude**, measured by the UV Telescope.
- The location that triggered the detection, given as **Right Ascension (Ra)** and **Declination (Dec)**.

Right Ascension and Declination are two concepts similar to longitude and latitude, respectively, used to describe the location of objects in the sky, in particular in a sphere of infinite radius that with the Earth as its center called the **celestial sphere**.

Taking this into account, Right Ascension is the equivalent of longitude, expressed in degrees (or more commonly in hours, minutes and seconds) and Declination, the equivalent of latitude, is expressed in degrees between the two poles: $+90^{\text{o}}$ and -90^{o}. [19]

This reference system is used to describe the position of objects in the sky, and it is the one used by the Swift telescope to specify the location of the sources. Afterwards these concepts will play an important role when computing the angle formed between the source and the Sun.

3.3 Objective function

Our main goal in this section was to obtain a list of GRB candidates ordered from more to less probable to be detected by the algorithm, that is, their fitness. To obtain this score we had to define an objective function, taking into consideration two factors:

- The **strength** of the burst, given by the UVOT magnitude. If this value was not available (as it was the case with many of the candidates) the BAT fluence was considered as its strength. This values were already given by the archive and no additional computations were required.
- The **angle between the burst and the Sun**, this was an important factor because bursts having an effect on the night hemisphere should be more noticeable than those hitting the day one, where the Sun has a bigger influence.

Computing the angle

As mentioned before, the Swift archive gives us the Right Ascension (Ra) and Declination (Dec) where the source is thought to be located.

The location of the Sun, on the other hand, is unknown. But we do know the time when the burst was detected.

The supervisor, Manuel Hernández-Pajares, provided me an algorithm which takes date (year, month, day and UT) and a planet of the Solar System (or the Sun, our case) as the input and returns its location in the celestial sphere, that is, its Ra and Dec.

This algorithm belongs to the **Starlink Project** (Rutherford Appleton Laboratory), which provided open-source software like the one at hand to astronomical institutions. Although it was shut down in 2005, the code is still available and we could use it for our study [1].

Knowing the location of the GRB: (δ_g, α_g), declination and right ascension, respectively. And that of the Sun: (δ_s, α_s), the cosine of the angle between both can be computed and used as a parameter for the objective function.

This computation is done by performing the dot product of the two unit vectors that can be obtained from the Ra and Dec of the objects given by the following formulas: [**?**] TODO: arreglar esta cita

$$unitVectorGRB = \begin{bmatrix} \cos\delta_g * \cos\alpha_g \\ \cos\delta_g * \sin\alpha_g \\ \sin\delta_g \end{bmatrix} \tag{3.1}$$

$$unitVectorSun = \begin{bmatrix} \cos\delta_s * \cos\alpha_s \\ \cos\delta_s * \sin\alpha_s \\ \sin\delta_s \end{bmatrix} \tag{3.2}$$

$$\cos angleSunGRB = unitVectorGRB \cdot unitVectorSun \tag{3.3}$$

The code for the previous computation is shown here:

```
1 def scorePosition(sunRa, sunDec, ra, dec):
2   # If Ra and/or Dec are n/a, return 0, else, compute the
     dotProduct
3   if ra == 0 or dec == 0:
4     return 0
5
6   coordSun = [math.cos(sunDec)*math.cos(sunRa),
7   math.cos(sunDec)*math.sin(sunRa),
8   math.sin(sunDec)]
9
```

```
10  coordGRB = [math.cos(dec)*math.cos(ra),
11   math.cos(dec)*math.sin(ra),
12    math.sin(dec)]
13
14  angle = math.acos(coordSun[0]*coordGRB[0] +
15  coordSun[1]*coordGRB[1] +
16  coordSun[2]*coordGRB[2])
17
18  angle = angle*180/math.pi
19
20  return angle
```

Listing 3.1: Python function for computing the angle

3.4 Obtaining the data

Regarding the scrapping of the website to parse the data and obtain this ordered list, **Python** was chosen because the problem required a quick implementation, and Python's libraries offered a great tool to develop a simple solution as quick as possible.

In the script, the website with the table of bursts (see figure x) is scrapped using Python's **BeautifulSoup** library, which has an HTML and XML parser that allows us to easily select and obtain data from a given website.

Insertion sort was used so we could insert every considered GRB into a list of sorted candidates as we were traversing the table.

Regarding the distribution of weight between both factors, strength and angle, we ....

The best 10 candidates of the resulting sorted table (pie de pagina: bursts registered up to the 26th of February of 2019), is shown here:

We proceeded to study these bursts

3.5 Results

Chapter 4

Solar flare detection

Before developing the main solar flare detection algorithm a first program was developed that would target a powerful solar flare for which we knew the time of the event and therefore, the location of the Sun.

Although the aim of the main algorithm is detecting the solar flare without taking into consideration the position of the Sun, this first approach was done to understand how the core of the algorithm works: studying the correlation between the cosine of the solar-zenith angle and the VTEC content.

This sections also provides an introduction to the formatting and use of the Global Navigation Satellite Systems data (GPS in this case) and how the main parameters necessary for the algorithms are computed.

4.1 Data

4.1.1 GPS Data

As we have seen in previous sections, the International GNSS Service (IGS) has made available open access GNSS data since its creation. The Crustal Dynamics Data Information System (CDDIS) is a central data archive for the NASA's Crustal Dynamics Project (CDP), dedicated to archiving space geodesy data for research. This archive has been storing and providing access to the GNSS data generated by the IGS since 1992.

Figure 4.1 shows how data is stored in de CDDIS server (`ftp://cddis.nasa.gov/gps/data/hourly/`).

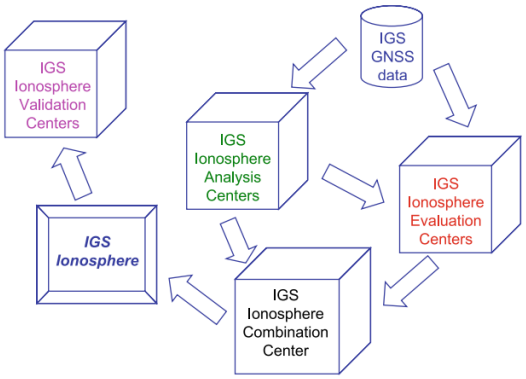
The files in this server contain raw GPS data that is then pre-processed to obtain VTEC maps in the form of **ti** files. An example diagram of this complex, several step procedure is shown in figure **??**, extracted from the paper "The IGS VTEC maps: a reliable source of ionospheric information since 1998" [9] by Manuel Hernández-Pajares, which offers a detailed explanation of this process.

Index of /gps/data/hourly/

[parent directory]

Name	Size	Date Modified
1999/		4/1/19, 6:30:00 AM
2005/		11/15/17, 1:00:00 AM
2006/		11/15/17, 1:00:00 AM
2007/		11/16/17, 1:00:00 AM
2008/		11/15/18, 12:23:00 PM
2009/		1/30/18, 1:00:00 AM
2010/		1/30/18, 1:00:00 AM
2011/		1/29/18, 1:00:00 AM
2012/		1/29/18, 1:00:00 AM
2013/		1/26/18, 1:00:00 AM
2014/		1/26/18, 1:00:00 AM
2015/		1/25/18, 1:00:00 AM
2016/		1/24/18, 1:00:00 AM
2017/		1/23/18, 1:00:00 AM
2018/		12/13/18, 1:31:00 AM
2019/		4/10/19, 2:31:00 AM
reports	0 B	8/9/17, 2:00:00 AM

(a) Files



(b) Data flow

Figure 4.1: VTEC distribution throughout the day for all IPPS (a) and for IPPs that have Vill as the receiver (b)

However, for this and the following section, only the pre-processed ti files of past dates were needed, as detecting the flares in real time is a task that will be discussed later in the project, in which this pre-processing will have to be taken into consideration. The project supervisor, Manuel Hernández-Pajares, provided me with some of data sets to use as input for the algorithms, along with information about the formatting of this files.

4.1.2 Formatting

The ti files contain several rows of pre-processed GPS data. Each row has a Receiver Id. and a Transmitter Id., therefore, for each row we have the Ionospheric Pierce Point between the Receiver and Transmitter. Each IPP has several parameters that are relevant for our computations:

- The **GPS time**
- The **Receiver Id.**
- The **Transmitter Id.**
- The **double derivate of Li**
- The **xmappingion**
- The **right ascension** and **declination** of the IPP

```
1 Field number | Example value | Description
2 [...]
3 3      0.008333333333     GPS time/hours (tsecdayobs/3600.d0)
4 4      cand               Receiver Id.
5 5      3                  Transmitter Id.
6 [...]
7 21     -0.5131586E-02     d2li
8 [...]
9 43     0.1565765332E+01     xmapping_ion
10 44    334.449              xraion
11 45    33.092               xlation
12 [...]
```

Listing 4.1: Format of the ti file

4.1.3 The Halloween Solar Storm: X17.2 flare

The data set we used was that of the so-called Halloween Storm, a powerful solar storm that took place from October to early November in the year 2003. In particular, we will try to replicate the results shown in figure 4.2, shown in the paper "GNSS measurement of EUV photons flux rate during strong and mid solar flares" for a poweful flare that took place in October 28th, 2003 [8].

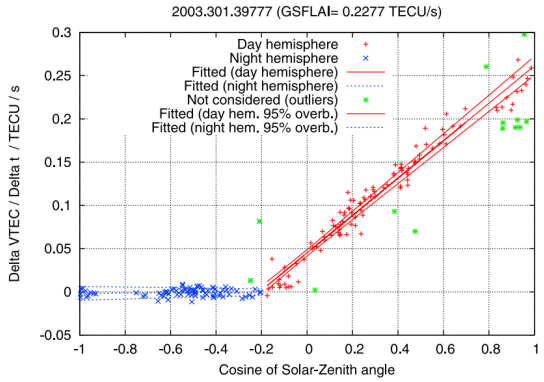


Figure 4.2: VTEC distribution throughout the day for all IPPS (a) and for IPPs that have Vill as the receiver (b)

As we can see the plot of the flare called X17.2 took place exactly at 2003.301.39777 (year.day.seconds of GPS time). In hours, 39777 seconds of a day is $39777s * 1h/3600s = 11.049...h$, around 11AM.

The ti files provided contained data from 10.5h to 11.5h (with a sampling rate of 30 seconds), so that we could see the VTEC distribution throughout the day.

With this data we can obtain the two parameters that will yield the plot shown in figure 4.2: the **VTEC value** and the **cosine of the solar-zenith angle**.

4.2 Vertical Total Electron Content (VTEC)

Fisrt we wanted to obtain VTEC distribution throughout the day, to visually see if any spikes appeared confirming that the moment we were going to study based on the paper [8] was correct.

For each epoch in our data set (from 10.5 to 11.5 with a sampling rate of 30 seconds) we needed to compute an estimation of the VTEC value.

4.2.1 Computing the VTEC

As we have mentioned before, one of the main paramenters relevant to the computation is the **double derivate of LI**, the d2li field in the ti file. The Li is the "ionospheric combination of carrier phases" [8], a direct measurement of TEC. Because this is a derivative it is the increment in VTEC, this will be observed in figure 4.5.

The VTEC increment can be estimated using the following approach:

$$\frac{d^2V}{dt^2} = \frac{d^2Li}{M} \tag{4.1}$$

Where $M = \frac{1}{\cos Z}$ is the "ionospheric mapping function", the inverse of the cosine of the satellite-zenith angle that we have for each IPP. [8]. This is the **xmappingion** field in the ti file.

Therefore, we can estimate the VTEC increment of an IPP by dividing the two given parameters:

$$\Delta V \approx \frac{d^2Li}{M} \tag{4.2}$$

Although the data that will be used throughout the project is going to be ΔV, the VTEC increment, it will be referenced as simply VTEC from now on.

Implementation

```
1 double precision function estimateVTEC (mapIon, d2Li)
2   implicit none
3   double precision, intent(in) :: mapIon, d2Li
4   double precision :: vtec
5
6   vtec = d2Li/mapIon
7   return
8 end function estimateVTEC
```

Listing 4.2: Simple Fortran function to compute the VTEC value

4.2.2 Distribution throughout the day

Because the only operation that had to be performed was the previous division, a simple AWK script was used to filter out the two necessary fields from the data file and print the resulting value as a function of time.

```
{
  /a/
  d2li = $21;
  mappingFunc = $43;
  vtec = d2li/mappingFunc;
  print $3 " " vtec
}
```

Listing 4.3: AWK script to estimate the VTEC

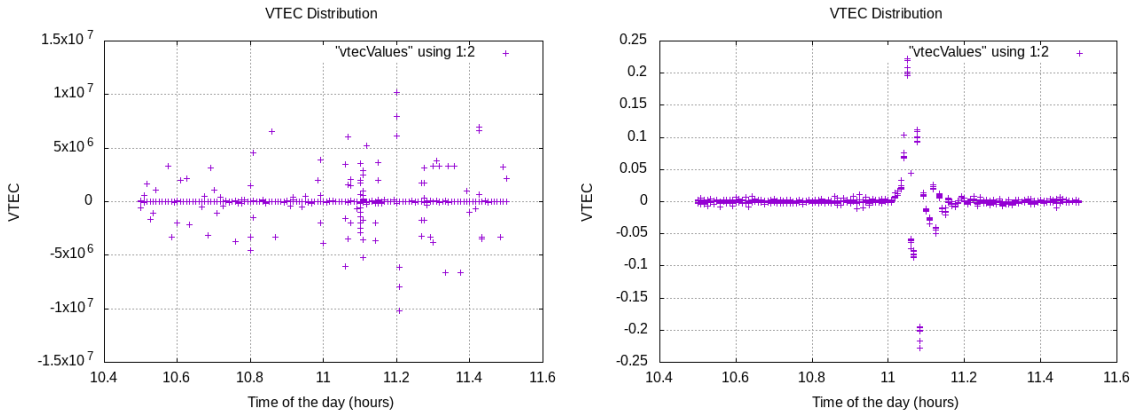
```
#!/bin/bash
tiDataFile="../data/ti.2003.301.10h30m-11h30m.gz"

zcat "$tiDataFile" | gawk -f previewVTECDistribution.awk >
    vtecValues
gnuplot -e "set terminal png; set output 'vtecDistribution.png';
    set title 'VTEC Distribution'; set xlabel 'Time of the day (
    hours)'; set ylabel 'VTEC'; set grid; plot \"vtecValues\" using
    1:2 with point"
rm vtecValues
```

Listing 4.4: Bash script to execute the procedures

The bash script executes the AWK process with the data as the input and outputs n rows with two columns: the time of the day and the calculated VTEC, plotting the results using Gnuplot. This results can be seen in figure 4.5(a), where we can see how the VTEC value evolves throught the day. Visually, a spike can be seen between 11 and 11.2 hours.

As mentioned before this value is estimated using the derivative of li, because this is the increment in VTEC, we can see that the value becomes negative after the spike, due to the VTEC value decreasing (negative gain).



(a) All IPPs (b) Villafranca station

Figure 4.3: VTEC distribution throughout the day for all IPPS (a) and for IPPs that have Vill as the receiver (b)

To see the event more clearly, though, we can focus on one specific receiver (which will still yield multiple IPPs, as the receiver works with different satellites). For the particular case of the Villafranca, Spain station (identified as Vill), we obtain the plot from figure 4.5(b). At this time of the day around 11:00h the Sun would have a greater effect on the IPPs of this station, so the spike can be seen more clearly.

As mentioned before, the flare took place at 11.05, so we could proceed using the studied data range and this epoch in particular.

4.3 Solar-zenith angle

The solar-zenith angle (denoted χ from now onward) plays a major role when studying this event: it is the angle formed by the Sun and the Earth's zenith and indicates the effect the flare is having on a particular IPP. It is expected that this variable presents a correlation with the increase in VTEC, which is what we aim to observe in this chapter.

Figure 7.1, at the end of the chapter, provides a visual representation of this variable that along with the results depicts how it can affect the VTEC value.

Obtaining the angle between two celestial objects has been shown in the previous section by means of equations 4.3, 4.4 and 4.5, when calculating the angle between the Sun and a detected GRB.

TODO: JUSTIFICAR ESTAS DOS EQUACIONES!!!

$$unitVectorObjectA = \begin{bmatrix} \cos\delta_g * \cos\alpha_g \\ \cos\delta_g * \sin\alpha_g \\ \sin\delta_g \end{bmatrix} \tag{4.3}$$

$$unitVectorObjectB = \begin{bmatrix} \cos\delta_s * \cos\alpha_s \\ \cos\delta_s * \sin\alpha_s \\ \sin\delta_s \end{bmatrix} \tag{4.4}$$

$$\cos\beta = unitVectorObjectA \cdot unitVectorObjectB \tag{4.5}$$

For this case, though, the angle is computed using the IPP's Right Ascension and Latitude (equivalent to declination), yielding the cosine of χ, the **solar-zenith angle**. The previous dot product can be simplified to:

$$\cos\chi = \sin\delta_{IPP} * \sin\delta_{Sun} + \cos\delta_{IPP} * \cos\delta_{Sun} * \cos(\alpha_{IPP} - \alpha_{Sun}) \tag{4.6}$$

The following Fortran code is the function that implements equation 4.6 and returns $\cos\chi$:

```
1 double precision function computeAngle (raIPP, decIPP, raSun,
    decSun)
2   implicit none
3   double precision, intent(in) :: raIPP, decIPP, raSun, decSun
4   double precision :: solarZenithAngle
5
6   solarZenithAngle = sin(decIPP)*sin(decSun) + cos(decIPP)*cos(
    decSun)*cos(raIPP - raSun)
7   return
8 end function computeAngle
```

Listing 4.5: Computation of the solar-zenith a angle's cosine

4.4 Results

Taking 212.338 and -13.060 as the Sun's right ascension and declination, respectively, and the measurements of all IPPs at 11.05 hours, figure 4.4 shows the plot of the output of our program.

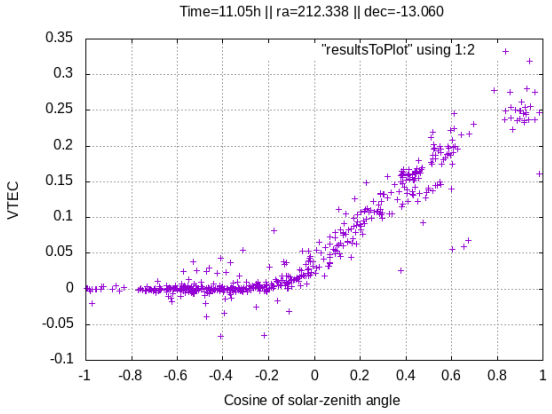
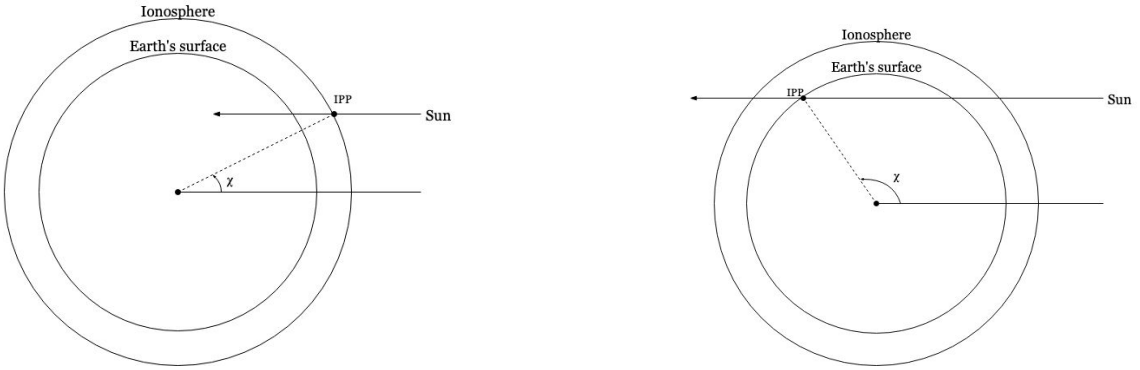


Figure 4.4: VTEC value as a function of the solar-zenith angle cosine

As we can observe, the resulting plot, similar to the one from figure 4.2, shows a strong corelation between the cosine of the solar-zenith angle and the VTEC content, which increases from $\cos\chi = 0$ ($90^{\underline{o}}$) to $\cos\chi = 1$ ($0^{\underline{o}}$) and it doesn't seem to be affected from $\cos\chi = -1$ ($180^{\underline{o}}$) to $\cos\chi = 1$ ($0^{\underline{o}}$).

Visually, this can be seen as follows:
TODO: SOLO FALTA ESTO, DIBUJO Y EXPLICACION DE PORQUE SUCEDE ESTO



(a) All IPPs (b) Villafranca station

Figure 4.5: Effect of the solar-zenith angle on the VTEC

When the Sun is vertically on the.....TODO

In conclusion, we can see that there appears to be correlation between the two variables. This correlation will be studied in more detail in the following section, where a first approach of the algorithm will be presented to detect the flare without knowing the location of its source.

Chapter 5

Brute Force Approach

In the previous chapter the correlation between the solar-zenith angle's cosine and the estimated VTEC value was studied. Here, we aim to provide a first, brute force approach, of the *Blind GNSS Search of Extraterrestrial EUV Sources (BGSEES)* algorithm to estimate the location of a EUV source. This approach is done as a first approximation to be problem to see more clearly how the algorithm will work, regardless of the performance.

For this first approach the Sun is used as the source (to check the corectness of the solution). It considers possible Sun locations (with a certain degree of precision) and checks the fitness of each to consider which could be the real location.

5.1 Key elements

5.1.1 Mean VTEC as a reliable indicator of the moment of the flare

The first part of the algorithm was, without going into the actual computations regarding the position of the IPPs, the possible Sun locations, etc, finding out when to perform the study, that is, detecting a spike in the VTEC content throughout the provided data range. In the previous chapter we already knew the specific moment of the flare: 11.05h, and could work based on this information, but the first step of the algorithm has to determine which moment is going to be studied.

For each epoch[1] we computed the mean VTEC of all IPPs and returned the epoch which had the highest VTEC mean.

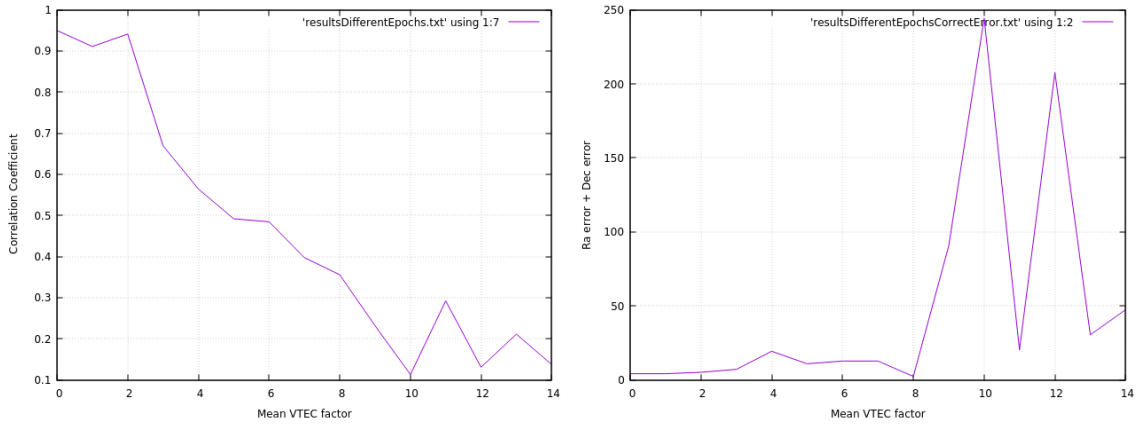
??????An alternative which performs an insertion sort (by inserting the epoch candidates into a priority queue) was also considered and implemented, but for this chapter we only used the epoch with the largest mean.

[1]In our data set the epochs ranged from 10.5 to 11.5, that is, 10:30AM to 11:30AM with a sampling rate of 1/120 hours or 30 seconds

To see if the mean VTEC could be used as a reliable indicator, the algorithm was tested with all available epochs of the data set, in order to study the effect of this indicator in the resulting estimation of the source's location.

To do this we ordered the different epochs available in our data set by their mean VTEC, inserting them into a priority queue.

Figure 5.1 shows the evolution of the correlation coefficient of the best estimated location (a) and its total error (right ascension error + declination error) (b) as the mean VTEC of the epoch decreases.



(a) Correlation coefficient (b) Error of the estimated source

Figure 5.1: Correlation and error of the solution as the mean VTEC decreases

As we can see, the correlation rapidly decreases from 1 (almost linear) to a negative coefficient. The error, on the other hand, doesn't experience much change for the first epochs but finally increases considerably.

The fact that the error remains near 0 as the correlation rapidly decreases for the first epochs can be due to the fact that different location of the source reduces its effect on the the estimated location with the highest correlation coefficient is still similar to that of the source. As the position of the source changes (the Sun in this case) TODO: este parrafo

5.1.2 Correlation

As seen in the previous chapter, there exists a correlation between the VTEC value and the cosine of the solar-zenith angle.

Once the moment of the flare is found, the aim of the algorithm is to study the correlation for each of the possible Suns and yield a fitness indicator for them.

The main idea for the algorithm is that the higher the correlation, the more accurate the estimated location should be, compared to the real one of the Sun.

The results will be discussed in the last section of this chapter to see if the previous expectations are true.

Computation

The correlation between two independent variables is defined as follows:

$$r_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2}\sqrt{\sum(y_i - \bar{y})^2}} \quad \text{for} \quad i \in (0, n) \tag{5.1}$$

Although optimization is not the aim of this section, the previous formula would require passing the data twice: first to compute the mean of the cosine and VTEC and second to compute the coefficient itself. The formula can also be expressed as follows:

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}} \quad \text{for} \quad i \in (0, n) \tag{5.2}$$

With can be implemented with a single pass algorithm, as opposed to the former. Because of this we decided to initially start with this one.

To ensure that the implementation of the previous method in Fortran was correct, we used the R language and its already implemented correlation function to test our results, in the previous section the Fortran implementation is shown.

5.2 Algorithm

The algorithm works as follows: the spike of VTEC value throughout the day is found by finding the epoch with the maximum mean VTEC of all IPPs for that epoch[2]. The data is then filtered by that epoch (only the info of IPPs for that epoch will be used for the computations) and the algorithm starts considering possible Suns.

There are $360 * 180 = 64800$ possible Sun's. Thus, in order to test the algorithm, the factor *STEP* is used which defines the step between the angles of possible Suns. The smaller the step, the more Suns will be considered.

For each of these possible Suns, the VTEC value and the cosine of the solar-zenith angle ($\cos\chi$) are computed for every IPP in that epoch, but the $\cos\chi$ is computed using the location of the IPP and the one of the possible Sun. Each of this Suns yields a data set with the aforementioned variables that can be plotted to obtain images such as the one studied at the end of the previous section. This two variables are used for computing the correlation coefficient for every considered Sun.

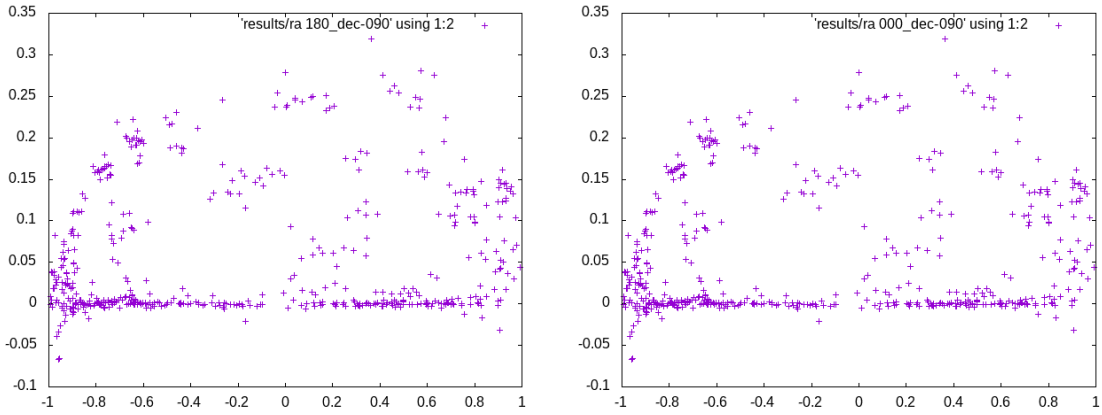
The following is the pseudocode for the brute force approach of the algorithm. Which returns the Sun that has yielded the highest correlation coefficient.

[2]The use of this method for finding the best epoch is discussed in chapter ****

Algorithm 1 Brute Force Approach

```
1: procedure MAIN
2:     epoch ← findSpikeInData()
3:     filterDataByEpoch(epoch)
4:     bestSun ← nil
5:     for ra = 0; ra <= 360; ra+ = STEP do
6:         for dec = −90; dec <= 90; dec+ = STEP do
7:             currentSun ← computeCorrelationPossibleSun(ra, dec)
8:             if currentSun.correlation > bestSun.correlation then
9:                 bestSun ← currentSun
10: return bestSun
```

One thing that has to be taken into consideration is that all possible locations that have a declination of 90^{o} or -90^{o} yield the same results (figure 5.2).



(a) ra=180^{o}, dec=-90^{o} (b) ra=0^{o}, dec=-90^{o}

Figure 5.2: Two examples of possible Sun locations

Using a *diff* tool we can observe that although the data files used to plot the results [3] have different values, the resulting plots are exactly the same image.

This is due to the fact that declinations 90^{o} and -90^{o} correspond to the poles, when the Sun is directly on top or below the Earth, and therefore all possible right ascensions are exactly the same position.

[3]The data is $cos\chi$ as the X axis and VTEC as the Y axis

5.2.1 Implementation

Finding the epoch

```
1 double epochIn, vtecIn, raIPPIn, latIPPIn;
2 int n = 0;
3
4 data >> epochIn >> vtecIn >> raIPPIn >> latIPPIn;
5 double totalEpochVTEC = vtecIn;
6 double previousEpoch = epochIn;
7 while (data >> epochIn >> vtecIn >> raIPPIn >> latIPPIn) {
8   totalEpochVTEC += vtecIn;
9   n++;
10  if (previousEpoch != epochIn) {
11    insertCandidate(previousEpoch, totalEpochVTEC/n);
12    previousEpoch = epochIn;
13    totalEpochVTEC = 0;
14    n = 0;
15  }
16 }
```

Listing 5.1: Finding a VTEC spike

The loop traverses the data and computes the mean VTEC of each epoch, inserting it in a priority queue.

Iterating over the possible Suns

```
for (int dec = -90; dec <= 90; dec += step) {
  if (dec != -90 and dec != 90) {
    for (int ra = 0; ra <= 360; ra += step) {
      pearsonCoefficient = computeCorrelation(&ra, &dec);
      if (pearsonCoefficient > maxCoefficient) {
        maxCoefficient = pearsonCoefficient;
        location = "[" + to_string(ra) + ", " + to_string(dec) + "]
    ";
      }
    }
  }
  else {
    int ra = 0;
    pearsonCoefficient = computeCorrelation(&ra, &dec);
    if (pearsonCoefficient > maxCoefficient) {
      maxCoefficient = pearsonCoefficient;
      location = "[" + to_string(ra) + ", " + to_string(dec) + "]";
    }
  }
}
```

Listing 5.2: Main loops

Computing the correlation

This is the main code of the Fortran function *computeCorrelation(ra, dec)* called for every possible *(ra, dec)* pair (a "virtual" Sun) considered in the previous loop. *computeCorrelation* has more auxiliary functions such as *openFile()* that are not included for readability.

It reads every line of the file that contains the information of the IPPs filtered by the found epoch and computes both the solar-zenith angle and the VTEC as seen in the previous chapter.

Given these results, the necessary values for computing the correlation are updated each iteration:

- $\sum x_i$ and $\sum y_i$
- $\sum x_i y_i$
- $\sum x_i^2$ and $\sum y_i^2$

Once all the IPPs have been processed, the previous values are used to compute the Pearson Correlation Coefficient using equation 5.2.

```
1  double precision function traverseFile (raSunIn, decSunIn)
2    implicit none
3    integer, intent(in) :: raSunIn, decSunIn
4    double precision :: raIPP, decIPP, raSun, decSun, mapIon, d2Li,
      cosX, vtec
5    double precision :: sumx = 0, sumy = 0, sumxy = 0, sumx2 = 0,
      sumy2 = 0
6    double precision :: rxyPearson
7    integer :: i = 0
8
9    sumx = 0
10   sumy = 0
11   sumxy = 0
12   sumx2 = 0
13   sumy2 = 0
14   i = 0
15
16   raSun = raSunIn
17   decSun = decSunIn
18   raSun = toRadian(raSun)
19   decSun = toRadian(decSun)
20   call openFile()
21   do while (1 == 1)
22     read (1, *, end = 240) raIPP, decIPP, mapIon, d2Li
23     raIPP = toRadian(raIPP)
24     decIPP = toRadian(decIPP)
25     vtec = estimateVTEC(mapIon, d2Li)
26     cosx = computeSolarZenithAngle (raIPP, decIPP, raSun, decSun)
27     if (cosx > CORRELATION_THRESHOLD) then
28       write(34,*) cosx, vtec
29       call updateCorrelationParameters (cosx, vtec, sumx, sumy,
       sumxy, sumx2, sumy2)
30       i = i + 1
31     end if
32   end do
33   240 continue
34   close(1)
35   rxyPearson = computePearsonCoefficient(i, sumx, sumy, sumxy,
      sumx2, sumy2)
36   return
37 end function traverseFile
```

Listing 5.3: Correlation computation

As can be seen in the code, the line (IPP) is only considered if $\cos\chi$ is higher than a *"correlation threshold"* (-0.1$^{\text{o}}$ in this case). This is done because we want to study only the "part" of the ionosphere where the Sun is having an effect. This can be seen in figure 7.1, where we observed that the VTEC value remained the same from $\cos\chi = -1$ (180^{o}) to $\cos\chi = 1$ (0^{o}).

Compiling

The C++ and Fortran compilers allow us to compile both languages and their libraries together. With this, the main part of the algorithm can be implemented using C++ which can then call Fortran for the parts that require heavy numerical computation.

This can be done by compiling the object of the Fortran code using the $-c$ flag, and then linking it with the C++ code using the $-lgfortran$ flag so that the standard Fortran libraries are included:

gfortran fortranFunctions.f90 -c -o functions.o
g++ functions.o bruteForce.cc -o bruteForce.x -lgfortran

5.2.2 Results

Executing the algorithm with a STEP of 10^{o}, this is the output of the execution:

```
1 [C++: Finding a spike in the VTEC distribution]
2   -> Spike found: 11.05
3 [AWK: Filtering all data by best epoch: 11.05]
4 [C++ -> Fortran: Finding the Person coefficients for possible Suns]
5   -> Input degree step: 10
6 [631 possible Suns considered]
7 [C++: Results]
8   -> Largest correlation coefficient: 0.926959
9   -> Estimated Sun's location: [ra=210, dec=-10]
```

Listing 5.4: Brute force approach algorithm output

As we can see, the possible Sun with the highest correlation coefficient (0.9269) has a location with a right ascension of 210^{o} and a declination of -10^{o}. Considering that for the epoch we are working with the Sun position was 212.338 and -13.060 as the right ascension and declination, respectively, we can see that the estimated Sun's location returned by the algorithm is close to the real one, using a step of 10^{o}.

It can be interesting to see how the computation time grows as more precision is demanded from the algorithm, and if the precision of the results does as well. The following table shows this relation for some input cases:

As we can see this approach provides the expected results, but has a large computational complexity that increases with the precision we demand.

Furthermore, we can see that a problem appears: in the last two cases there is an increase in the correlation coefficient as expected, but the estimated Sun location does not improve, it is actually less accurate than the previous one.

In the next chapter, the *Blind GNSS Search of Extraterrestrial EUV Sources* (BGSEES) algorithm is presented to perform this computations with several optimizations to reduce its complexity and certain special cases are studied, like the aforementioned one.

Step	Considered Suns	Correlation coefficient	Estimated location	Execution time
100	5	0.695358	[ra=200, dec=10]	867ms
50	25	0.695358	[ra=200, dec=10]	303ms
25	106	0.866293	[ra=225, dec=-15]	820ms
12	436	0.92287	[ra=216, dec=-6]	956ms
6	1771	0.934663	[ra=216, dec=-12]	1s 385ms
3	7141	0.937349	[ra=213, dec=-12]	7s 169ms
1	64621	0.939114	[ra=214, dec=-11]	1m 9s 564ms

Table 5.1: Results

Chapter 6

Decrease search range method

In the previous chapter we saw that the BGSEES algorithm for detecting the location of an EUV source (in this case, the Sun) is possible with a first, brute force approach. Here, the algorithm is studied in more detail considering a different method aiming to increase its precision and reduce its computational complexity.

6.1 Decreasing the range of the search

As we saw in the previous chapter, to increase the precision of the algorithm, the *step* with which we iterate over the possible angles (right ascension and declination) is reduced. This causes more possible Suns to be considered. For example, with a step of one degree, we consider all possible right ascensions [0,360] and declinations [-90,90] which is $360 * 180 = 64800$ Suns, minus the $2 * 360 - 2 = 718$ right ascensions we don't consider because as we have seen right ascensions for declinations of -90 and 90 are the same location.

We want to have the highest precision possible without having to consider all $64800 - 718 = 64082$ possibilities by progressively reducing the search range.

6.2 Pseudocode

This first optimization works as follows: the entire possible range is considered with a large starting step (e.g 60). Once the best Sun is found within this range, the precision is increased (the step is decreased) and the search range is reduced. This way the precision is increased but the number of considered possibilities remains similar each iteration of the algorithm. The following is the pseudocode[1] for the algorithm using this optimization:

[1]The code for the special cases of -90 and 90 degree declinations is not included for more readability. It is included in the implantation, in the following section.

Algorithm 2 Search range decrease

```
1: procedure MAIN
2:     epoch ← findSpikeInData()
3:     filterDataByEpoch(epoch)
4:     bestSun ← nil
5:     r ← defaultRange()
6:     for step = initStep; step >= min; step /= 2 do
7:         for ra = r.lowerRa; ra <= r.upperRa; ra+ = step do
8:             for dec = r.lowerDec; dec <= r.upperDec; dec+ = step do
9:                 currentSun ← computeCorrelationPossibleSun(ra, dec)
10:                if currentSun.correlation > bestSun.correlation then
11:                    bestSun ← currentSun
12:                    r ← newRange(bestSun, step)
13: return bestSun
```

A very important part of the algorithm is computing the correlation. In the previous chapter we didn't consider discarding outliers from our data set because it appeared that considering all samples yielded good results, but testing with a different flare resulted in problems because of that. That is why in the previous section a method for discarding outliers was introduced.

The main problem of discarding outliers is that while before we the correlation could be computed with a single pass, multiple are needed to be able to discard outliers. This is explained in more detail in the next chapter, along with the time comparison and error of both possibilities (discarding outliers or not).

6.3 Implementation

This first piece of code is the main loop of the method, which starts with a default range of ra=[0, 360], dec=[-90, 90] and reduces it every iteration based on the current best Sun's estimated location.

Furthermore, because an increase in the precision of the tested location doesn't necessarily imply an improvement in the solution, the new candidate is inserted into a *priorityQueue* ordered by the coefficient to assure that the method returns the best one found throughout the entire loop.

```
1 void TraverseGlobe::decreasingSTEP() {
2   int rangeSize = 3;
3   int initStep = 60;
4   possibleSunInfo currentSun;
5   searchRange range = setRange(currentSun, true, initStep,
     rangeSize);
6   for (double step = initialStep; step >= 0.5; step /= 2) {
7     currentSun = considerPossibleSuns(step, range, plotData);
8     bestSuns.push(currentSun);
9     range = setRange(currentSun, false, step, rangeSize);
10  }
11 }
```

Listing 6.1: Decreasing the range and increasing the precision

The *setRange* function sets a new range based on the estimated location that depends on the precision used for the values and assure that valid value is returned.

```
1 searchRange setRange(possibleSunInfo sun, bool defaultR, double
    step, int rangeSize) {
2   searchRange range;
3   if (defaultR) {
4     range.lowerRa = 0;
5     range.upperRa = 360;
6     range.lowerDec = -90;
7     range.upperDec = 90;
8   }
9   else {
10    double raRange = step*rangeSize;
11    double decRange = step*rangeSize;
12    range.lowerRa = sun.ra - raRange >= 0 ? sun.ra - raRange : 0;
13    range.upperRa = sun.ra + raRange <= 360 ? sun.ra + raRange :
     360;
14    range.lowerDec = sun.dec - decRange >= -90 ? sun.dec - decRange
     : -90;
15    range.upperDec = sun.dec + decRange <= 90 ? sun.dec + decRange
     : 90;
16  }
17  return range;
18 }
```

Listing 6.2: Setting the new range based on the estimated source location

Finally, the *considerPossibleSuns* function has the same functionality than the one from the previous chapter, it iterates over the possible locations, this time, however, it does so over the given range, rather than just the default one.

The names of the lower and upper bound variables have been changed (*uRa* instead of *upperRa*, for example) for readability.

```
1 possibleSunInfo considerPossibleSuns(double step, searchRange range
    ) {
2   FortranController fc;
3   double pearsonCoefficient;
4   int i = 0;
5   possibleSunInfo bestSun;
6   bestSun.coefficient = -23;
7   bestSun.location = "salu2";
8
9   for (double dec = range.lDec; dec <= range.uDec; dec += step) {
10    if (dec != -90 and dec != 90) {
11      for (double ra = range.lRa; ra <= range.uRa; ra += step) {
12        pearsonCoefficient = fc.computeCorrelation(&ra, &dec);
13        if (pearsonCoefficient > bestSun.coefficient) {
14          bestSun.coefficient = pearsonCoefficient;
15          bestSun.ra = ra;
16          bestSun.dec = dec;
17        }
18      }
19    }
20    else {
21      //Do only once
22      double ra = 0;
23      pearsonCoefficient = fc.computeCorrelation(&ra, &dec);
24      if (pearsonCoefficient > bestSun.coefficient) {
25        bestSun.coefficient = pearsonCoefficient;
26        bestSun.ra = ra;
27        bestSun.dec = dec;
28      }
29    }
30  }
31  return bestSun;
32 }
```

Listing 6.3: Iterating over possible locations within the given range

Finally, the *computeCorrelation* calls the Fortran code (the same used in the Brute Force chapter) to compute the correlation

Another change that could be done to improve this method's performance would be to adopt a sort of *"Dynamic Programming"* strategy in order to avoid repeated calculations (the new range will always be inside the previous range). However, the problem is that because we are dealing with a different precision for the right ascension and declination values every loop, the same values are never used. TODO: justificar mejor esto?

6.4 Linear fitting: discarding outliers

In the previous chapter we used a simple method to discard outliers for the VTEC value (using a cutoff value between -0.3 and 0.3). While this worked with that case in particular, using it with other data sets can lead to problems.

Because of this, another approach is presented in order to discard outliers: linear fitting. The aim is to find the line that fits best the relation between both variables (cosine and VTEC) by means of linear regression to discard samples that do not fit in it.

We can see that this procedure has been used before in figure 4.2 from the paper *"GNSS measurement of EUV photons flux rate during strong and mid solar flares"*[8].

Manuel Herández-Pajares, the author, provided the Fortran program that performs this computation. After integrating it with the rest of the code, figure 6.1 shows the result of testing it with the flare used in the previous chapter:

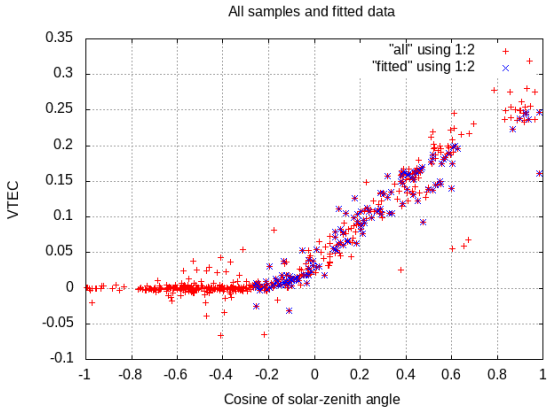


Figure 6.1: All data (red) and fitted samples (blue)

The main problem of this method is that before the correlation could be calculated in a single pass of the data. Now, on the other hand, the algorithm needs multiple iterations: first, the solar-zenith angle cosine for each IPP is calculated; then, the outliers are discarded, and finally the filtered data is traversed one last time to compute the correlation.

Here is the new code in order to compute the correlation for each possible location:

```
1 double computeCorrelation(double* ra, double* dec) {
2   FileManager fileManager;
3   int sigma = 1;
4   int iterations = 6;
5   computecosinesofcurrentsourcefortran_(ra, dec);
6   fileManager.discardOutliersLinearFitFortran(sigma, iterations);
7   return computecorrelationfortran_(ra, dec);
8 }
```

Listing 6.4: Discarding outliers and computing the correlation

As we can see, the data is traversed **at least** 3 times now (more depending on the number of iterations).

6.5 Results

Comparacion sencilla entre outliers/no-outliers, decir que en results mas detalle con otros ejemplos

Seeing the increase in computational complexity of this method because of the fact that outliers have to be discarded using the linear fitting method, we decided to instead focus on a different method that would rely only on the data itself, instead of considering the many possible locations of the source.

Chapter 7

Least Squares method

As we have seen in previous chapters the overall process to determine the location of the source is studying the correlation between the VTEC value and the solar-zenith angle (or source-zenith angle, speaking in general terms) for a possible location.

That is, for an IPP with a location and an associated VTEC value, given the location of a possible source, compute the cosine between them, and see that the closer the cosine to 1 (or $180^{\underline{o}}$), the higher the VTEC value.

The idea that we wanted to test with this method was finding the location of the source by performing the inverse operation: having the VTEC, location of the IPP and correlation (1, assuming a near-linear correlation), obtaining the source's right ascension and declination.

7.1 The equation system

The source-zenith cosine between the source and the IPP was computed using the following equations:

$$unitVectorIPP = \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\delta_g * \cos\alpha_g \\ \cos\delta_g * \sin\alpha_g \\ \sin\delta_g \end{bmatrix} \tag{7.1}$$

$$unitVectorSource = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \cos\delta_s * \cos\alpha_s \\ \cos\delta_s * \sin\alpha_s \\ \sin\delta_s \end{bmatrix} \tag{7.2}$$

$$\cos\chi = unitVectorIPP \cdot unitVectorSource \tag{7.3}$$

Having the VTEC and source-zenith cosine, we could find the correlation of the two variables, expecting that the real source would yield a near-linear correlation.

Visually, we can see the relation between VTEC and the computed cosine in figure 7.1, obtained in chapter 3 when studying the a specific case for the Sun.

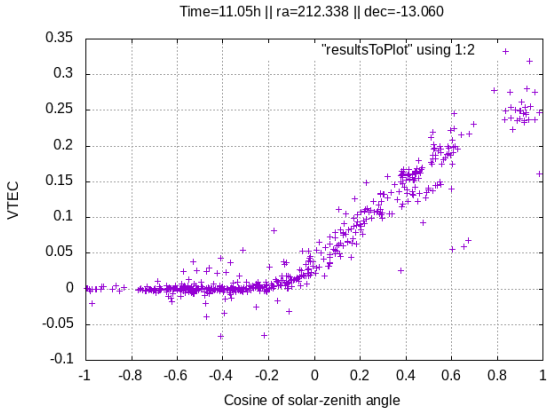


Figure 7.1: Visual representation of the solar-zenith angle

As we have previously seen for this case, there appears to be a linear relation starting around $\cos\chi = -0.1$ between the two studied parameters. Therefore, we could define this linear relation as a **straight line** ($y = mx + b$) by expressing the estimated VTEC value (ΔV) as a function of the source-zenith cosine (7.4).

$$\Delta V = a\cos\chi + b \tag{7.4}$$

Because the cosine is computed using the previous equations (7.1, 7.2, 7.3). It can be expressed as follows:

$$\cos\chi = XX' + YY' + ZZ' \tag{7.5}$$

Where X', Y', Z' are the components of the IPP's unit vector obtained from equation 7.1, and X, Y, Z are the unknowns of our equation: the components of the source's unit vector, which could be used to easily find the right ascension and declination of the source by using trigonometric operations.

However, finding the value of this unknowns is the difficult part of this method. Taking the cosine as 7.5 we can express the linear function as:

$$\Delta V = aXX' + aYY' + aZZ' + b \tag{7.6}$$

Because a and b are unknowns as well as X, Y and Z, we can group them as follows:

$$\Delta V = \alpha X' + \beta Y' + \gamma Z' + b \tag{7.7}$$

Our aim now would be to solve the previous equation, but then we would need to obtain only the values of X, Y and Z. We can see that:

$$\sqrt{\alpha^2+\beta^2+\gamma^2} = \sqrt{a^2(X^2+Y^2+Z^2)} = \sqrt{a^2} = a \tag{7.8}$$

Because X, Y and Z are the components of a unit vector[1]. The previous allows us to, once we know the values of α, β and γ, obtain X, Y and Z by doing:

$$\frac{\alpha}{\sqrt{\alpha^2+\beta^2+\gamma^2}} = \frac{\alpha}{a} = \frac{aX}{a} = X \tag{7.9}$$

In our data we can find, for each IPP: ΔV, X', Y', Z' (because we have the right ascension and declination of the point).

For each of these IPPs, we have an equation of the form $\Delta V = \alpha X' + \beta Y' + \gamma Z' + b$ and, therefore, we have an overdetermined system of equations, with more equations (unknown, depends on the input data) than variables (four: α, β, γ and b)

Knowing how to obtain X, Y and Z from α, β and γ, we can now focus on solving the system of equations to obtain the latter unknowns .

Because we have an overdetermined system of equations, the solution can be approximated using the Least Squares approach. The system can be represented in matrix form $y = AX$ as follows:

$$\begin{bmatrix} \Delta V_0 \\ \Delta V_1 \\ . \\ . \\ . \\ \Delta V_n \end{bmatrix} = \begin{bmatrix} X'_0 & Y'_0 & Z'_0 & 1 \\ X'_1 & Y'_1 & Z'_1 & 1 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ X'_n & Y'_n & Z'_n & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ b \end{bmatrix} \tag{7.10}$$

This way, X, the least square estimate, can be computed by:

$$X = (A^T A)^{-1} A^T y \tag{7.11}$$

7.2 Pseudocode

7.3 Implementation

When

One of the main advantages of this method over the others is that we don't need to compute the correlation, that it is, discard outliers. (o si????)

[1] $\sqrt{(X^2+Y^2+Z^2)} = 1$

Chapter 8

Other methods

Here other possible optimizations are considered that could be used to, in the future, extend the algorithm and perhaps improve its performance and accuracy.

8.1 Hill Climbing

explicar que no es hill climbing exactamente, algo mas simple rollo greedy Using the previous optimization we can see a plot of all the possibilities the algorithm considers:

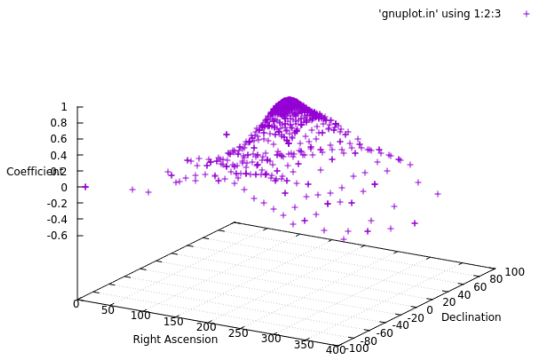


Figure 8.1: All visited candidates of the solution space

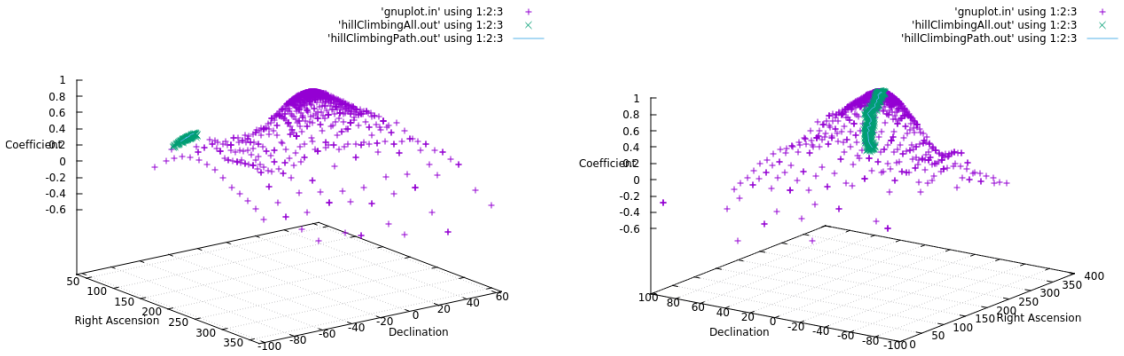
As we can see in the previous figure, there appears to be a "hill" (our solution) so an attempt to solve the problem using a *Hill Climbing* approach was also considered.

While not exactly Hill Climbing, a simple greedy algorithm was implemented as a first attempt to test if this method could yield good results:

```
1   // Starting position
2   sourceInfo current;
3   current.ra = 160;
4   current.dec = -20;
5   int i = 0;
6   // Loop with limit or until no progress can't be made
7   while (++i < 100) {
8   vector<sourceInfo> candidates = getNeighbourList(current);
9   sourceInfo newCandidate = getBestCandidate(candidates);
10  if (newCandidate < current) {
11  break;
12  }
13  current = newCandidate;
14  }
15
```

Listing 8.1: Hill Climbing

The following figure shows the results of the execution for two different starting states. For both of them, the algorithm ran until it couldn't progress any further (wasn't interrupted by the iteration limit). The plots contain both the possibilities considered by the decrease range method (in purple, the same plot as 8.1) and the path taken by the Hill Climbing method (in green).



(a) Start: ra=100º, dec=-60º (b) Start: ra=160º, dec=-20º

Figure 8.2: Paths taken by the Hill Climbing algorithm

Visually, we can see that the number of considered possibilities is inferior and the top is reached for case (b), but a problem appears: **a local maxima**.

If we take a starting right ascension of 160º and a declination of -20º, the algorithm takes a path that gets to the top of our solution, yielding similar results to the previous, decrease range method.

However, with a starting right ascension of 100º and a declination of -60º, the algorithm finds a local maxima on its way and can't progress to the real best solution.

As a result, we can't rely on the Hill Climbing approach for all cases, considering that local maxima may exist for our type of problem. Because of this, another possibility would be to use the Simulated Annealing algorithm so that we can explore other parts of the solution space and find other paths that might lead to our desired solution, instead of only.

8.2 Simulated Annealing

8.3 OpenMP

Explicar OpenMP

8.4 Discarding the Sun hemisphere

So far the algorithm has been studied for the case of the Sun, as it is a source that should be detected more easily than far-away stars.

The main idea is that the algorithm should detect extraterrestrial EUV sources (if possible) other than the Sun. it follows that it should, before any other computations, discard the Sun hemisphere based on the current Sun location. The greater impact of the Sun's radiation due to its proximity would blind the algorithm from detecting sources that may be having an effect on that hemisphere because of the noise.

bla bla bla

This will be used in the next chapter, in which the presented methods will tested with flares from the Sun and far-away stars.

Chapter 9

Results

Here discuss results, error. And compare different data sets First sun, then far-away

9.1 Sun

the two data sets for the best epoch
the first one, study all epochs and see what happens even when there's no flare

9.1.1 Decrease range method

Instead of focusing on only the best epoch[1], the following table presents the 15 best epochs from best to worst (with 11.05 at the top, the one studied in previous chapters) and their results:

- The right ascension and declination of the estimated source location (the Sun in this case)
- The error of the previous results compared to the real Sun's location
- The Pearson correlation coefficient of the estimated Sun

9.2 Far-away stars

[1]The best epoch is the one with the highest mean VTEC of all its IPPs

Epoch	RA	Dec	RA Error	Dec Error	Correlation Coefficient
11.05	213.75	-10.3125	1.412	2.7475	0.949659
11.075	215.625	-12.1875	3.287	0.8725	0.910935
11.0417	210.938	-9.375	1.4005	3.685	0.941128
11.1167	211.875	-19.6875	0.463	6.6275	0.669265
11.0333	225.938	-18.75	13.5995	5.69	0.564307
11.025	219.375	-16.875	7.037	3.815	0.49188
11.0917	216.562	-21.5625	4.2245	8.5025	0.48441
11.1333	214.688	-23.4375	2.3495	10.3775	0.397166
11.1917	211.875	-11.25	0.463	1.81	0.355636
11.3667	185.625	-76.875	26.713	63.815	0.23138
10.8583	20.625	-65.625	191.713	52.565	0.112926
11.1583	211.875	-32.8125	0.463	19.7525	0.29224
10.9917	62.8125	-71.25	149.525	58.19	0.130575
11.0167	203.438	8.4375	8.9005	21.4975	0.211087
11.4583	235.312	-37.5	22.9745	24.44	0.137234

Table 9.1: Results for different epochs

Chapter 10

Annexes

Poner aqui filemanager? debugger makefile cosas asi?

Bibliography

[1] (Archived website) The Starlink Project. https://web.archive.org/web/20120123062045/http://starlink.jach.hawaii.edu/starlink/WelcomePage. [Online; accessed 10-March-2019].

[2] International GNSS Service Website. http://www.igs.org/about. [Online; accessed 8-March-2019].

[3] J. M. Dow, R. E. Neilan, and C. Rizos. The international GNSS service in a changing landscape of global navigation satellite systems. *Journal of geodesy*, 83(3-4):191–198, 2009.

[4] L.-L. Fu and A. Cazenave. *Satellite altimetry and earth sciences: a handbook of techniques and applications*, volume 69. Elsevier, 2000.

[5] N. Gehrels, G. Chincarini, P. Giommi, K. Mason, J. Nousek, A. Wells, N. White, S. Barthelmy, D. Burrows, L. Cominsky, et al. The swift gamma-ray burst mission. *The Astrophysical Journal*, 611(2):1005, 2004.

[6] Goddard Space Flight Center (NASA). Swift GRBs Archive. https://swift.gsfc.nasa.gov/archive/grb_table.html/. [Online; accessed 1-March-2019].

[7] C. J. Hegarty and E. Chatre. Evolution of the Global Navigation Ssatellite System (GNSS). *Proceedings of the IEEE*, 96(12):1902–1917, 2008.

[8] M. Hernández-Pajares, A. García-Rigo, J. M. Juan, J. Sanz, E. Monte, and A. Aragón-Àngel. GNSS measurement of EUV photons flux rate during strong and mid solar flares. *Space Weather*, 10(12):1–16, 2012.

[9] M. Hernández-Pajares, J. Juan, J. Sanz, R. Orus, A. Garcia-Rigo, J. Feltens, A. Komjathy, S. Schaer, and A. Krankowski. The igs vtec maps: a reliable source of ionospheric information since 1998. *Journal of Geodesy*, 83(3-4):263–275, 2009.

[10] M. Hernández-Pajares, J. M. Juan, J. Sanz, À. Aragón-Àngel, A. García-Rigo, D. Salazar, and M. Escudero. The ionosphere: effects, gps modeling and the

benefits for space geodetic techniques. *Journal of Geodesy*, 85(12):887–907, 2011.

[11] B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle. *GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more*. Springer Science & Business Media, 2007.

[12] Institute of Space Sciences (CSIC-IEEC. Magnetar Outburst Online Catalog. `http://magnetars.ice.csic.es/#/parameters`. [Online; accessed 3-March-2019].

[13] Jochen Greiner, Max-Planck-Institute for extraterrestrial Physics. GRB collection. `http://www.mpe.mpg.de/~jcg/grbgen.html`. [Online; accessed 3-March-2019].

[14] D. Martínez Cid. First study on the feasibility of stellar flares detection with gps. B.S. thesis, Universitat Politècnica de Catalunya, 2016.

[15] S. W. P. C. N. Oceanic and A. Administration). Ionosphere. `https://www.swpc.noaa.gov/phenomena/ionosphere`. [Online; accessed 25-March-2019].

[16] P. W. Roming, T. E. Kennedy, K. O. Mason, J. A. Nousek, L. Ahr, R. E. Bingham, P. S. Broos, M. J. Carter, B. K. Hancock, H. E. Huckle, et al. The swift ultra-violet/optical telescope. *Space Science Reviews*, 120(3-4):95–142, 2005.

[17] T. Singh, M. Hernandez-Pajares, E. Monte, A. Garcia-Rigo, and G. Olivares-Pulido. GPS as a solar observational instrument: Real-time estimation of EUV photons flux rate during strong, medium, and weak solar flares. *Journal of Geophysical Research: Space Physics*, 120(12):10–840, 2015.

[18] S. Solar Center. The Earth's Ionosphere. `http://solar-center.stanford.edu/SID/activities/ionosphere.html`. [Online; accessed 13-March-2019].

[19] Solar System Exploration, NASA Science. Reference Systems. `https://solarsystem.nasa.gov/basics/chapter2-2/`. [Online; accessed 5-March-2019].