Work report

OBS de l'Ebre summer internship

Third year physics student

Universitat Autònoma de Barcelona

Author:

Josep Rubí Bort

Índex

1. Introduction

This first section is aimed to describe the process done week by week as a theoretical introduction to the topic.

1.1. Theoretical introduction

The earth magnetic field is mainly composed of the earth own field, which can variate every year, and the solar effect. This solar effect changes from day to day, as the activity of the Sun is not constant. The days when there is a low solar activity are called the Solar quiet days (Sq). The main objective of this summer practices is to analyse how does the sun affect the earth magnetic field during the Sq and from them extract a way to characterise the days that are not quiet using an index that will be created.

The first step that has been done is the selection of the quiet days. For doing this, the index of geomagnetic activity K_P will be used. *The K_P index is a 3-hour index of the level of worldwide geomagnetic activity* as stated in [1] . The K_P is divided in 10 numbers, and each number is divided in three different categories. The number ranges from 0 to 9, and each one has the categories -, 0 and +, being the '+' category the one with the most activity and '-' the one with the least activity.

To obtain the new index, the D component from the *Observatori de l'Ebre* will be analysed by extracting the contribution of the Earths magnetic field and a slight contribution that has an effect and it is due to the magnetosphere of the Earth.

It is expected that the contribution of the Sun during the Sq to the D component will have a sinusoidal pattern, as the effect of the Sun should be symmetrical. In the morning it hits by one side of the earth, and during the evening it is on the other side of the earth that is hit. Thus, it is expected that the D component goes through a maximum and a minimum and during noon it should be null. Furthermore, it should be expected that the amplitude of the sinusoidal functions has a maximum in the summer, as the sun is closer to the sun. In addition, one would suspect that the period of the same function would be of the order of the diurnal time. This to theoretical predictions will be later checked.

Moreover, at some point it will be classified the seasons of the year in three different groups. This ones are the winter months(D-months) that include: January, February, November and December, the summer months(J-months) that include: May, June, July, August and September, and the equinoxes months(E-months) that include: March, April and October[2].

This does not include the part of the Escape Room and the Atlas as they are projects that are not directly related to this project. However, it should be carried in mind that each week some time was dedicated to those two projects.

1.2. Week 1

During the first week, the research phase was conducted. A number of papers were consulted as well as some books. The main articles that were consulted were [3], [4], [2] and [5]. The books consulted were: [6] and [7] . This documents helped in the understanding of the Sq and the processes by which it deviates.

1.3. Week 2

During the beginning of this week, the research of information continued. The most important information found was about the indexes of magnetic activity K_P and K. The article found was [1]. Moreover, the program was started to be done. During the first phase of the code, it was intended to do a model of the Sq done month by month. Thus, the first objective was to look at which days were actually quiet by looking at the values of the index K_P. It was firstly stated that a quiet day included the values of K_P that went from 0- to 2+, as it was stated in the book [6]. Nevertheless, as some problems were introduced, later the values that range from 2- to 2+ were not taken into account. In addition, the mean values of D of each month were calculated, and a sinusoidal function was adjusted. The parameters that were to adjust are the amplitude, frequency, phase and additional constant. This was done with a library of python that has the capacity to adjust functions, if a first approximation of the values is given. To do this all the parameters where approximated. The amplitude was done by subtracting the smallest value of the biggest, the frequency was approximated by: subtracting the first hour of the day from the last one, and all of this was dividing 2π, the phase was approximated by knowing which values makes the sinus maximum or minimum, and lastly the additional constant was approximated by making the mean value of all the data points. In addition to this, it was stated that a correlation filter should be used, so as to filter the data that should not have any error, but for some reason it had some bad points.

1.4. Week 3

During this week the main focus was to generalise the code by using the Intermagnet web page. However, at the end of the week it was decided that a better option would be to do it only for the observatory of Ebre so as to make it for only this observatory first, and later generalised it more. So, at the end of the week when the code of the observatory of Ebre was continued there were a number of errors found and it was mainly due to the web page. The problem has arisen when some months had their linked changed from the standard one and it made a change of the code a necessity. Even if this seems a little problem, the fact that it was a big quantity made the problem seem like a real mess. On top of that, some articles about the SC where read as to make the Atlas. The requirement of a huge quantity of time due to the fact that it was necessary a big effort to understand them should be taken into account.

1.5. Week 4

During this week the focus was in making the code for having the D component of each day of the year. This means that with all the data of all years that we dispose of, we try to make a mean value of the D component for a daily model for the whole year. In addition, a sinusoidal function was fitted as explained earlier. Nevertheless, the additional constant was not considered as the D component was supposed to do, its sinusoidal movement around the 0. Moreover a Secular variation has been considered so as to allow the D to oscillate around the 0. By doing this a number of problems where found and solve. Despite that the whole week was centred in this, at the end of it the code was in the final steps of it.

One of the biggest problems that has been found is that, despite having 20 years of data, some days of the year where empty and had to be 'filled' with the nearest days around them. This was done by making a weighted average as it is explained in the code[1]. By doing this a problem was found as the days that were used to fill the empty ones may no have the same length. So as to solve it a simple condition was imposed that checked if they had the same length or not and solved the problem.

1.6. Week 5

An optimisation of the code is begun during this week, so as to reduce the time of compilation. Some problems are found as the function is not centred around the 0, as it should be, if all the magnetospheric component is not so important. An error in the function that fills the empty days is found, as it may coincide that the full days that are near the empty ones have a transition of diurnal time. Despite that this problem is easily solved by adjusting some code, it should be taken into account as to not repeat the same mistake. Moreover, it has been noted that the mean values do not correspond to the exact hours, but to the hours and a half. This is because the mean value is done with all the hourly data. Despite this, two corrections had to be done, one being the correction of the graphics. When correcting this, a little error in the code has been noticed, and, in spite of not being huge, it has prevented any other errors in the code. All of this has taken some time to fix, but now is corrected. So as to impose that the sun hours are not exact it also has been imposed that if the minutes are bigger or equal to 30 the hour should be counted as the next one[2]. When this was imposed an error was found as some years the minute of the sunset or sunrise change. By doing this an error appeared when the correlation was looked at. Even if ideally it should be done some research around this, as, at the end, the correlation was not checked[3] this research has not been done. Despite this and as to have results that are more coherent, a function was created to eliminate these points that did not have all the days.

1.7. Week 6

During this week some errors in the code kept appearing, but they were little the solution was fast enough. However, some problems were not so fast.

One of the main problems encountered were some days that appeared to be calmed by the index K_P but when the magnetograms where looked they actually were not. A possible explanation that was found was an error whenever the index for those days was selected. In order to solve this error, the days that caused problems where looked one by one, and the ones that are considered to be bad where discarded. In the Annex 4.1 the discarded days can be looked at. It should be noted that some days where causing doubts, but it was decided that those days should not be considered.

Another error occurred when the correlation filter was looked at. Nevertheless, the function to fit the sinusoidal function was drastically changed. Instead of fitting a function it was made so that the first and last data point had their fit at the value 0, and by doing this the frequency and phase were both fixed. Moreover, for the amplitude, the value was calculated by subtracting the lowest value from the biggest one. This was done as to fit a function that did not include the calm value from the earth, and the first condition was imposed as it is known that the D value from the Sq days has a 0 value before the Sun rises and

[1] As in the code it is explained in detail a further explanation will be omitted

[2] This is only for the sunrises and sunset hours

[3] As it will be commented later.

when the sun sets down. By doing this, the correlation filter had not a physical meaning anymore, so it is not used.

By doing all of this, the fits were considered good enough, so the differences between other days and the calmed one was calculated.

An addition of the data of 20 years has been done, so the total data ranges from 1980 to 2022. However, the number of quiet days has not increased in such a big way. The increase was about 20 %. This can be explained due to the decrease of Solar activity, as it can be seen on the official page of the **SPACE WEATHER PREDICTION CENTER**[8]. In the Fig. 1.1, it can be seen the general decrease of the activity of the Sun.

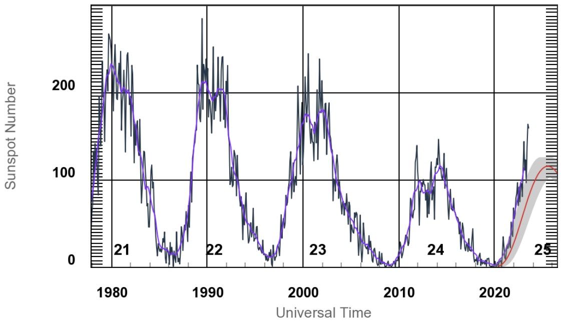


Fig. 1.1: Number of solar spots during the years. It should be remarked that the number of solar spots is closely related to the solar activity.

1.8. Week 7

An error was found, so the first day of this week spent solving this error. The error was seen when the code returned the days with big slope. These days where days that in theory should be accepted, but were not as the filter that selected big data was not placed correctly. When this error was found, the problem was solved immediately. On top of that, more days were to be removed, as in theory they should be calm, but when looking at their magnetograms it was clear that the days were not calmed, as it happened before, but this time it was done with days before the 2000. In addition, when revising all the days, it was noticed that the previous selection was not good enough, so other days from the 2000 above had to be removed.

Moreover, the mean value for all the ranges of the index was done and plotted, only for the years 1980 to 2000. It was seen a possible relation between all of the ranges, and the result was the Fig. 1.2. To do this a criteria to classify the days has been created. This criteria consisted in establishing a minimum number of K_P for each day that are in a certain interval. As it can be appreciated in Fig. 1.2 the intervals of K_P consisted in 4 intervals, and each of them has 6 possible values.

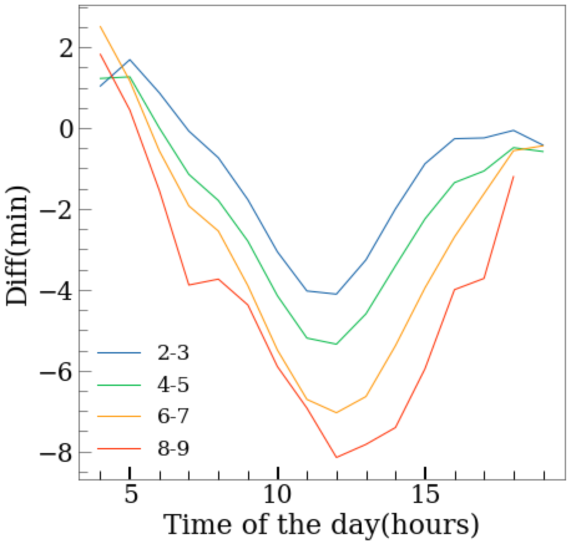


Fig. 1.2: Difference between the model and the non calm days. This is a mean value of all the days of all year that correspond to these ranges of the index.

1.9. Week 8

During the beginning of this week, it was plotted the mean value of the D for all the ranges of the index and for all the years available. It has been computed the difference between the model and the non calm days, in the same ways as Fig. 1.2 and a unexpected result has appeared. The result can be seen in Fig. 1.3 and the anomaly is the little lump of the curve of the values 4-5.

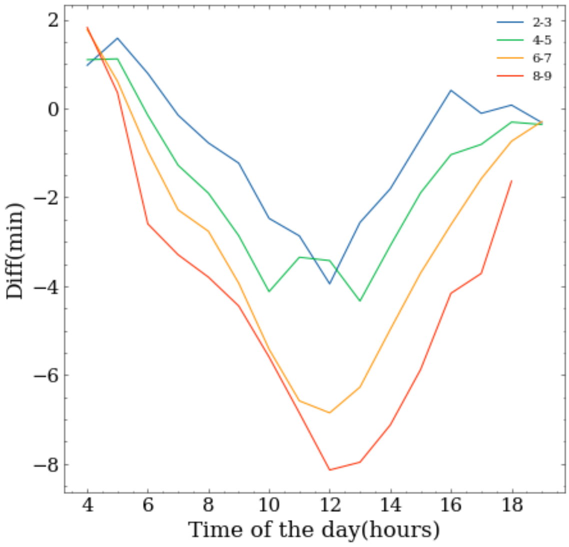


Fig. 1.3: Difference between the model and non calm days.

Moreover, at the end of the week an error in the code was found, but could not be solved in time.

1.10. Week 9

During the beginning of the week the error mentioned above was solved and found. By solving this error, the little lump of the Fig. 1.3 was gone. The final result can be seen in Fig. 1.4.

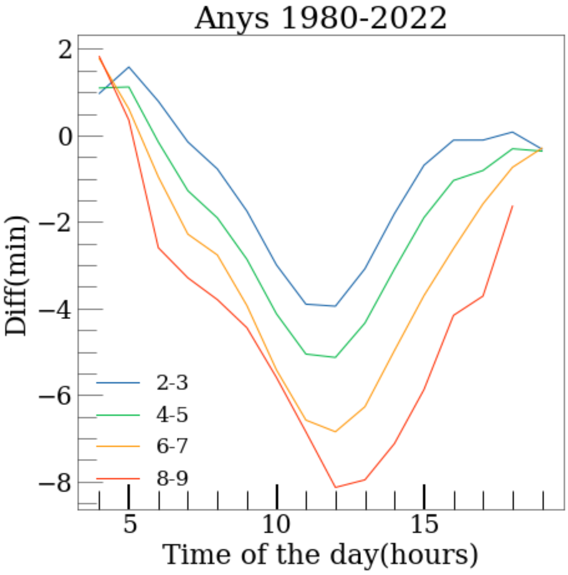


Fig. 1.4: Difference between the model and non calm days for the period 1980-2022.

Moreover, it was implemented the reading of the data from the computer, thus the code could compile faster.

2. Filtration of D data

2.1. Index selection

In first place, the accepted accepted values of the index that were considered to be that of a calmed day ranged from 0 to 2, including all their values, as it was stated in the book [6] that this values were considered that of a quiet day. However, and taking into account the problems found by using this, the range was lowered to 0 to 1, also including all their values. It should be taken into account that for the sole purpose to obtain a truly calmed day it has been considered a quiet day the one that has all of its value of K_P between 0 and 1.

The data of all the indexes was downloaded from the web page of the *ISGI*([9]) and read through a function of python, that is called `Classificacio_dies` in the code, and it returns the days that has an index between 0 and 1.

2.2. Paths followed

The first step done was making the average of the data of D month by month. The data to do this was acquired by the web page of the *l'Observatori de l'Ebre*. When this was done the next step followed was using the page *Intermagnet* as in that place there were more observatories that could be treated. Nevertheless, owing to the fact that the range of time that is available on the *Intermagnet* web page is very limited, and that it is not in our interest, at the moment, to use other observatories, this procedure was stopped and the code went back to the *l'Observatori de l'Ebre* web page.

2.3. Reading and treatment of the data

This code is made so that the data needed can be used directly from the web page, without downloading, or downloading it and reading it from the computer.

2.3.1. Online data

In first place only data from the year 2000 until the 2022 was being read. This data was found in the web page of the *l'Observatori de l'Ebre*. However there where some problems as not only the links kept changing, but also the organisation of the data. This caused the creation of some huge functions. The main functions to do this where: `open_link_with_condition`, which had the role to open the two links that kept changing, `read_file` has the function to read the file correctly, as the file from different years are organised differently, and `data_treatment`, that has the role of treating the data, as in some point the data that is displayed in the web page changed, and the conversion from one to another had to be done. This 3 functions, even if appear to be not important, were difficult to create, as a lot of time had to be spent to check if all the conditions were placed correctly and if there were no other exceptions happening.

To use this function one should go to the 2 and take off the comment on the function `read_file` that has commented the phrase `Aquesta funcio llegeix les dades online` and comment the other function named `read_file`.

2.3.2. Download data and offline data

This function is similar to the one before. First of all, the code 1 should be ran, so as to download all the data, and then the code 2 can be ran.

2.3.3. Data treatment

When all the data from the days where returned correctly it was seen that the data was not centred around the 0 as it should be. This was due to the secular variation and some contribution of the magnetosphere. To eliminate this two contributions two linear regressions were created. The linear regression of the secular variation was from one year to another, and as the secular variation was calculated with the average of the whole year, the final value corresponded to the middle of the year, so the regression went from the middle of one year to the next year. As for the linear regression of the magnetospheric contribution, the procedure was exactly the same, but this one used the values of the same day that we where looking, and to not used values affected by the Sun, the ones used where the one from the 00:00 H and the 23:00H. Thus, the regression was created using this two values, and the index was given directly knowing the hour of the day. To do all this some functions were created in the code. To solve the problem of the secular variation the functions `Coeff_secu` and `index_coef` were created. The role of `Coeff_secu` was to make the regressions for all the years of the available data and return a list with them. Thus, when the regression was needed the code it should only check which element of the list corresponded to that day. In order to do this the function `index_coef` was created. As for the regression of the magnetospheric contribution the function `regressio_magnetosfera` was created. This function returned the coefficients of the day, and as stated before, the index is the different hours of the days. However, as the regression was only useful to the first half of the year, and the 2023 is not over yet, to calculate its mean value it had to be done differently. To do it, a function that calculates the mean value of the 2023 with the data available on the web page of *Observatori de l'Ebre* had to be done. This function is `dada_extra_secular`.

Finally it has been noted that as the mean hourly values are made from the start of one hour to the end of it, the time that should be used it is not the initial part of the hour, but the middle one. For example, instead of using 7:00 it should be used 7:30.

2.4. Filters

Various filters had to be implemented as different problems arose.

One of the biggest problems is the fact that some data is missing, and instead of a number that could have a physical meaning the number 99999.00 was showed. In order to solve this the filter named `tercer_filtre` was imposed. However, as the data was given in different formats depending on the year, as stated 2.3. Thus the condition of the `file_type` was to be imposed.

Another filter that had to be imposed was the `segon_filtre` as there was some bad data, that simply was bad for an unknown reason, but it was not missing, so it did not show the number 99999.00. The filter works by comparing slopes from the day working with the previous day. The condition of the slope being as big as 5 times of the previous day was imposed. However, this condition is imposed as it solves the problem seen but without any physical criteria behind it. Moreover it must be taken into account that the first day has to be a quiet one with good data. Was this condition not to be followed and the code could give various problems.

Another filter was imposed using manual methods, as there where some days that did not have any wrong data, but it appeared to not be that of a quiet day. It was hypothesised that the reason behind this was a wrong classification of the index K_P. This problem came across when the plots of the fits[4] was seen, and the data was clearly not that of a quiet day. To solve this two functions where created. One is `selector_anys` which needs the input of the days that should be eliminated. With this input the function returns which years has that day as a quiet one. However this days should be manually checked as to not eliminate some useful data. The other function is `eliminador_dies` which requires an input of the days that should be eliminated and the day that we want to check. The first input is given by the first function, and the other one is given when the 'mother function' is working. The days that where eliminated are shown in the Annex 4.1.

Finally there is one last filter that it was initially implemented. Nevertheless, as the fit changed this filter had no meaning so it was deleted. Even if in the next section it will be explained in more detail, the reader should carry in mind that this filter worked with the correlation of the model and the days individually.

2.5. Curve Fit

To do the curve fit it was first thought to adjust a sinusoidal function with a library of python. By doing that it was intended to add a filter that searches which days deviate from the sinusoidal function as to remove them. All this process was done by calculating the correlation between the fit and the data of each day. However, as there is a general property that need to be taken into account and that way of adjusting did not do, it was removed. The property is the fact that at the sunrise and sunset the deviation should be null.

The new procedure to adjust the sinusoidal function was to make an approximation of the amplitude, frequency and the phase. This amplitude is adjusted by taking the smallest and biggest values and averaging them, while the frequency and phase are obtained by imposing that the initial and final value are null. By doing this the values of the frequency and phase were easily obtained.

2.6. Empty days

Another problem that occurred was the fact that, despite having a range of data of 40 years, there where some days of the year that where lacking data. To fill these empty days the nearest days to this empty ones where picked so as to do a weighted average. The weight depends on how far from the full day(f-day) the empty one(e-day) is and how many consecutive e-days there are. An example is given so as to make the explanation more comprehensible. Let's start from the point of having 4 e-days: the order would be: day 1, day 2, day 3, day 4, day 5 and day 6, where day 1 and 6 are the f-days. Now the day 2 would be filled by using 3/4 of the day 1 and 1/4 of day 6, days 3 and 4 would be by using day 1 and 6 times 1/2, and finally day 5 would be the opposite of day 1, thus, it would be 3/4 of day 6 and 1/4 of day 1. All of this was implemented in the function `empty_days`.

[4] These will be explained in the next section.

However it should be taken into account a little detail that can cause some problems, and that is the change of the hours of sunrise and sunset. The main problem is that between some e-days the sunrise or sunset hour can change, so this should be taken into account when the average is done. Should that not be taken into account and the average of the sunset and/or sunrise hours would be done incorrectly as there would be some hour that has a 0 value. This fact is also implemented in the function `empty_days`.

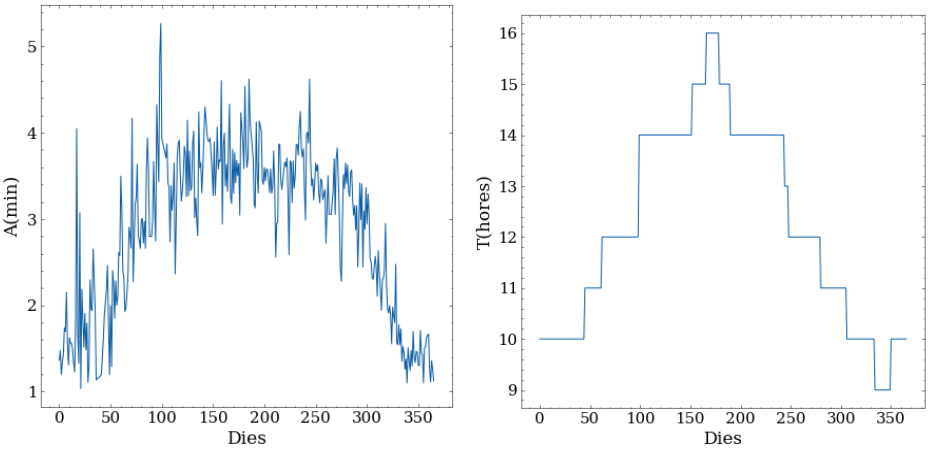
2.7. Sunrise and sunset problems

As said in the above section some problems arose by the fact that the sunrise and sunset hours changes. However a bigger problems arises when one encounters with the fact that within the same day of the year and for various years the minute of the sunrise and sunset shifts. For example, if one was to look to the sunset hour and minute of the 13^{th} of February of 2000, 2001 and 2002, one would encounter the following values: 7:29h, 7:30h and 7:29h. Were the reader to understand the whole code, it would now understand the direction that this is taking. In the function `filter_data` the sunset and sunrise hours are used to know which set of data is going to be used. In addition, in the function `notable_minutes` it is established that if the sunset and/or sunrise minute is 30 or bigger the hour changes to the next one. This can cause lots of problems as the same days of the year can change the set of data used when changing the year. To solve this 3 functions where created: `comprovacio_num_dies`, `arreglar_minuts` and `dies_min_dolent`. The function `comprovacio_num_dies` only checks if all the data of the same day of the year has the same length, and if it is not the day of the year is appended in a list to be latter checked. This check is done by the function `arreglar_minuts` which checks if the problem of not having the same lengths happens due to the sunset and/or sunrise hours. Moreover, this function also solves the error by eliminating some data that was going to be used to do the average values. Finally, the function `dies_min_dolent` only returns the days of the year that has to be checked in addition to their sunrises and sunsets hours, in order to solve problems in other functions.

It should be noted that this problem has been noted when the filter of correlation was implemented. In addition, this problem is not completely solved. Some extensive research should be conducted in order to determine exactly which days of the year changed their minute of sunrise and/or sunset in order to completely solve this problem. This research was not conducted due to the lack of time and the fact that the problem was mainly solved when the filter of correlation was eliminated. Despite all of this, the functions partially solve this problem, in a way that is more than enough for our purposes.

2.8. Results

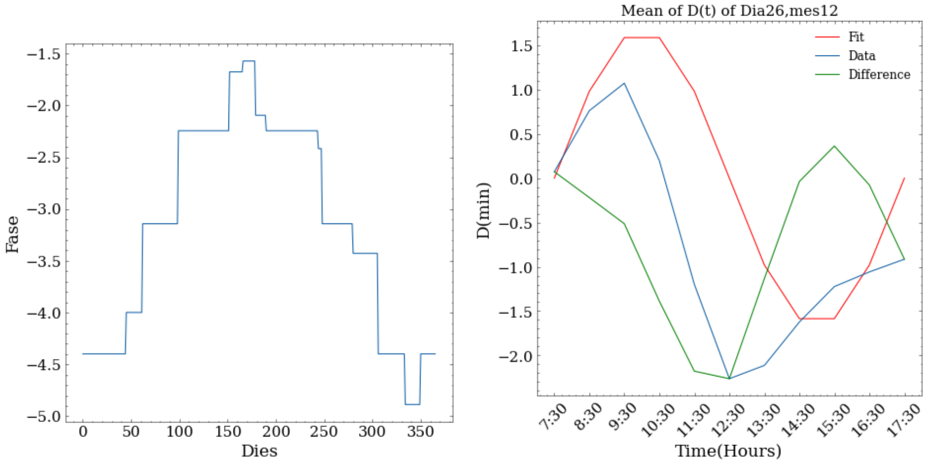
Of all the work mentioned above it has been obtained a prediction of the behaviour of the D component. The amplitude, phase and period are shown in Figs. 2.1a, 2.1b and 2.2a.It can be clearly seen the physical meaning of these results in the case of the amplitude and period. As mentioned in 1.1 the amplitude was expected to have a maximum on the summer time, while the period is of the order of the diurnal time. However no results were expected of the phase. Despite this, its behaviour is coherent with the way the code has been done, as its variation is not continuous but more stepped.



(a) Amplitude with respect to the day of the year.

(b) Period with respect to the day of the year.

Fig. 2.1: Plot of the amplitude and period with respect to the day of the year

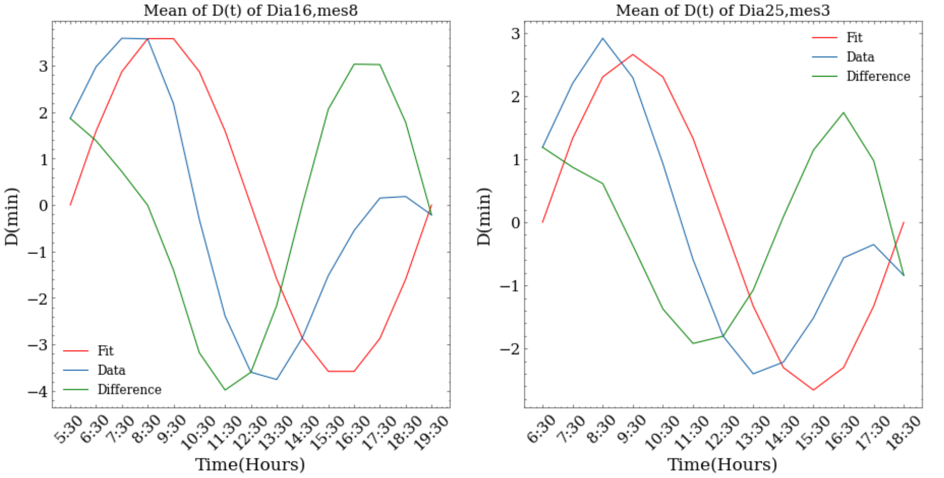


(a) Phase with respect to the day of the year.

(b) Example of a day of the D-month.

Fig. 2.2: Plot of the phase with respect to the day of the year and a example of the fit for a day of the D-months.

Additionally, Figs. 2.2b, 2.3a and 2.3b are shown as a more accurate example of the results. Each figure shows a day that corresponds to a different season of the year as to give a more appropriate range of time.



(a) Example of a day of the J-month. (b) Example of a day of the E-month.

Fig. 2.3: Examples of the fit of a day of the J-months and E-months.

3. Index creation

This section will explain the process of creation of an index so as to enable one to sort the disturbed days.

3.1. Index division

The first step to take is to make a classification of the disturbed days with the index K_P. Owing to the fact that the values of K$_P$ from 0- to 1+ are already classified as Sq, they should not be taken into account. Moreover, the classification has been slightly changed due to the fact that now the flexibility of the selection increases. Before, to classify the Sq it should be checked that all the day was not disturbed. However, the classification of the disturbed days is not so strict as one can think of different classifications to these days. First of all all, all of different values of K$_P$ should be separated into different groups. For a purpose of uniformity the following groups have been made: 2-3, 4-5, 6-7 and 8-9. Each group includes the subcategories of each value, so, in total there are 4 groups with 6 different values each one.

On one hand, very strict classification could be checking that every value of the day for the index K$_P$ is in the same range.

On the other hand, a very flexible classification could be by only checking if one value of the day is in the desired range. With this classification some problems could arise, as there are days that can be in more than one range. A possible solution could be to classify that day with the highest index.

However, the classification used is one that is in the middle of both extremes. It was decided that for the group of K$_P$ there should be at least 5 values each days that is in this gap. For the group 4-5 at least there should be 4 values, for the 6-7 3 values were required and for the group 8-9 only 2 values were needed. This was done as the probability of a day having all of the values with a high index K$_P$ is very low as these values indicates extreme magnetic storms. Moreover, the probability of these happening increases as K$_P$ decreases. Thus, this solution it was thought to be one that can accept days with extreme magnetic storms, and days with a lower magnetic activity. Despite all of this, the problem of a day being able to be in more than one group continues to be. The solution that has been given to this problem is the same one that has been mentioned before: to classify these days with the highest index possible. All this process is done with the function `Classificacio_dies_kps`.

3.2. Mean values

When all of the days were classified with the index a global mean value was calculated. The first step to take was to filter the data that was missing. This is done with the function `filtre_dies_seleccionats`. The next step to take was to obtain the mean values. However, it should be noted that the mean values are of the period 1980-2022, for each range of K$_P$. Thus, it was necessary to take into account that each day the sunrise and sunset can change. The function `Mitjes_anuals` takes that into account, and it returns a list of list, so that the mean values of each hour and each range of K$_P$ can be computed. In the 3.1 it can be see the graph mentioned.

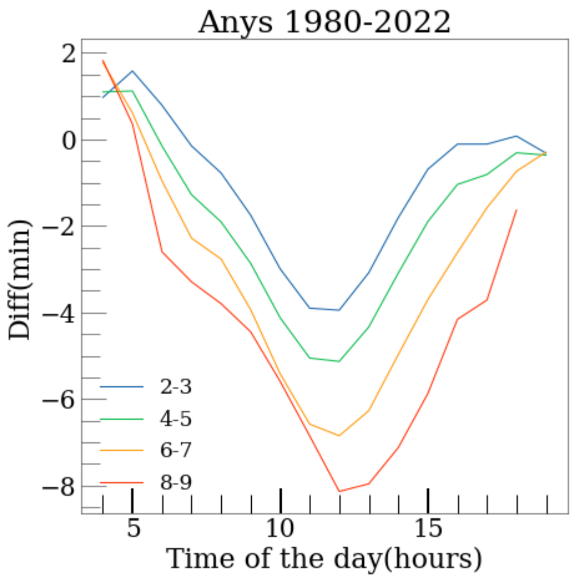


Fig. 3.1: Difference between the model and non calm days for the period 1980-2022.

3.3. Future work

Despite that time ran out and the index could not be created, this section is dedicated to explain one possible path to create the index.

Following the Fig. 3.1 it can be seen a possible route to compute create the new index based on the actual index K_P. This path is based on the fact that the difference between the model and the mean values has a variation that seems linear respect to the the K_P for each hour. Despite that the range of K_P 8-9 seems to be non linear at some hours, this is, probably, due to the fact that the available data is low. In fact, for this range there is a total of 55 days of data available.

4. Annex

4.1. Eliminated days

Table 4.1: Days that have been removed manually.

Day	Month	Year
7	1	1997
		2022
8	1	1984
		2007
		2021
9	1	1980
		1994
		2006
		2010
		2021
10	1	1980
		1994
		2006
		2013
		2021
12	1	2009
		2018
		2019
13	1	1982
		1986
		2007
		2009
		2019
		2021
		2022
15	1	1008
17	1	2009
		2020
		2021
18	1	2009
		2010
		2014
		2018
21	1	2021
23	1	2009
		2011
24	1	2007
		2009
28	1	2009
31	1	2006
		2013
		2014
		2021
2	2	2009
4	2	2001
7	2	1998
8	2	2009
9	2	2006
		2010
		2011
10	2	1981
		2009
		2010
		2021
12	2	1980
14	2	1981
		2006
		2017
		2018
15	2	2017
16	2	1992
		1998
18	2	2000
		2001
		2021
2	3	2006
		2009
12	3	2018
25	12	1997
		2007
		2012

Bibliography

[1] Gordon Rostoker. Geomagnetic indices. *Reviews of geophysics and space physics*, 10:935–850, 1972.

[2] J.M. Torta, J.J. Curto, and P. Bencze. Behavior of the quiet day ionospheric current system in the ropean region. *Journal of geophysical research*, 102:2483–2494, 1997.

[3] R.A. Langel, T.J. Sabaka, R.T. Baldwin, and J.A. Conrad. The near-earth magnetic field from magnetospheric and quiet-day ionospheric sources and how it is modeled. *Physics of the Earth and Planetary Interiors*, 98(3-4):235–267, 1996.

[4] J. Miquel Torta, Santiago Marsal, Juan J. Curto, and Luis R. Gaya-Piqué. Behaviour of the quiet-day geomagnetic variation at livingston island and variability of the sq focus position in the south american-antarctic peninsula region. *Earth Planets Space*, 62:297–307, 2010.

[5] N. M. Pedatella, J. M. Forbes, and A. D. Richmond. Seasonal and longitudinal variations of the solar quiet (sq) current system during solar minimum determined by champ satellite magnetic field observations. *Journal of geophysical research*, 116, 2011.

[6] Wallance H.Campbell. *Introduction to Geomagnetic Fields*. Cambridge University Press, 2003.

[7] Wallance Hall Campbell. *Earth Magnetism: A guided tour through magnetic fields*. A Harcourt Science and Technology Company, 2000.

[8] Space weather prediction center web page. https://www.swpc.noaa.gov/products/solar-cycle-progression. Accessed on 2023-08-17.

[9] Isgi data download. https://isgi.unistra.fr/data_download.php. Accessed on 2023-08-11.

```python
import os
import urllib.request
from urllib.error import HTTPError, URLError

def remove_empty_lines(data):
    lines = data.split('\n')
    non_empty_lines = [line.strip() for line in lines if line.strip()]
    return '\n'.join(non_empty_lines)

#Aquesta funcio m'obra els links, provant primer el mes avitual i despres el que alguns cops es fa sevrir
#Ja que alguns messos escampats va canviant( sense un patro aparent)
def open_link_with_condition(year, month, download_dir):
    month = int(month)
    year = int(year)
    month_list = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
                  'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
    month_name = month_list[int(month) - 1]

    urls = [f'http://www.obsebre.es/php/geomagnetisme/dhorta/{year}/{month_name}/ebr{year}{month:02d}dhor.hor',
            f'http://www.obsebre.es/php/geomagnetisme/dhorta/{year}/{month_name}/ebr20{year}dhor.hor'
           ]

    hdr = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36',
        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'
    }

    # Create the download directory if it doesn't exist
    if not os.path.exists(download_dir):
        os.makedirs(download_dir)

    for url in urls:
        try:
            req = urllib.request.Request(url, headers=hdr)
            response = urllib.request.urlopen(req)

            if response.status == 200:
                data = response.read().decode('utf-8')

                data = remove_empty_lines(data)


                # Construct the local file path using os.path.join
                filename = f'ebr{year}{month:02d}dhor.hor'
                full_path = os.path.join(download_dir, filename)

                # Save the data to the specified directory
                with open(full_path, 'w', encoding='utf-8') as local_file:
                    local_file.write(data)

                print(f"Data retrieved successfully and saved to {full_path}")
                return full_path
                break

        except HTTPError as http_err:
            if http_err.code == 404:
                print(f"The link was not found for URL: {url}")
            else:
                print(f"HTTP error occurred for URL: {url}, Error: {http_err}")
        except URLError as url_err:
            print(f"URL error occurred for URL: {url}, Error: {url_err}")

    else:
        print("Failed to retrieve data from all URLs.")

year_initial = int( input('Which initial year do you want to extract data from?'))
year_final = int( input('Which final year do you want to extract data from?'))

for year in range(year_initial, year_final+1):
    for j in range(0,12):
        month = str(j+1).zfill(2)
        file_path = f'C:/Users/pep/OneDrive - UAB/Escritorio/Variacio de D/Dades anys antics/{year}'
        open_link_with_condition(year, month, file_path)
```

Listing 1: Data download

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jul  7 12:59:48 2023

@author: pep

Si teniu dubtes sobre el codi escriviu a l'autor del codi en el correu peprubi@gmail.com
"""


import urllib.request
from datetime import datetime, date, timedelta
from astral.sun import sun
from astral import LocationInfo
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
from urllib.error import HTTPError
import scienceplots
plt.style.use(["science","no-latex"])


# # *****************************************************************************

#                                  FUNCTIONS
# # *****************************************************************************

```

```
#Aquesta funcio m’obra els links, provant primer el mes avitual i despres el que alguns cops es fa sevrir
#Ja que alguns messos escampats va canviant( sense un patro aparent)
def open_link_with_condition(year, month):
    month=int(month)
    year=int(year)
    # Define a list of URLs to try
    month_list = [’Jan’, ’Feb’, ’Mar’, ’Apr’, ’May’, ’Jun’,
                  ’Jul’, ’Aug’, ’Sep’, ’Oct’, ’Nov’, ’Dec’]
    month_name = month_list[int(month)-1]
    urls = [ f’http://www.obsebre.es/php/geomagnetisme/dhorta/{year}/{month_name}/ebr{year}{month:02d}dhor.hor’,
            f’http://www.obsebre.es/php/geomagnetisme/dhorta/{year}/’ +
    f’{month_name}/ebr’+’20’+f’{year}dhor.hor’
        # Add more URLs here if needed
    ]

    # hdr required to access the files
    hdr = {
        ’User-Agent’: ’Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36’,
        ’Accept’: ’text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8’
    }

    for url in urls:
        try:
            req = urllib.request.Request(url, headers=hdr)
            response = urllib.request.urlopen(req)

            # Check if the response status code is 200 (OK) before processing data
            if response.status == 200:
                data = response.read().decode(’utf-8’)
                # Process the data from the response here
                print("Data retrieved successfully:")
                file = urllib.request.urlopen(req)
                return file
                break  # Break the loop if a valid response is obtained

        except HTTPError as http_err:
            if http_err.code == 404:
                print(f"The link was not found for URL: {url}")
            else:
                print(f"HTTP error occurred for URL: {url}, Error: {http_err}")
        except urllib.error.URLError as url_err:
            print(f"URL error occurred for URL: {url}, Error: {url_err}")

    else:
        # The loop completed without a successful response from any URL
        print("Failed to retrieve data from all URLs.")

# *****************************************************************************

def read_file(d, m, y):
    #Aquesta funcio llegeix les dades offline
    print('>>>>>>>>>>>>>> El bucle de llegir el dia', d, m, y, 'ha comencat <<<<<<<<<<<<<<')
    day = d
    month = m
    year = y
    route = f'C:/Users/pep/OneDrive - UAB/Escritorio/Variacio de D/Dades anys antics/{year}/ebr{year}{month}dhor.hor'
    if float(year) > 2011:
        file_type = 'new'
        if (year == '2019' or year == '2018' or year == '2017' or year == '2016' or  year == '2014' or year == '2013'):
            data = pd.read_csv(route, skiprows=24, delimiter='\\s+')
        elif year=='2015' and (float(month)==6):
            data = pd.read_csv(route, skiprows=32, delimiter='\\s+')
        elif year == '2015' and (float(month)!=6):
            data = pd.read_csv(route, skiprows=24, delimiter='\\s+')
        elif year == '2012':
            data = pd.read_csv(route, skiprows=26, delimiter='\\s+')

        elif (year == '2020' or year == '2022'):
            data = pd.read_csv(route, skiprows=25, delimiter='\\s+')

        elif (year == '2021' and (float(month) >= 9 and float(month)<11)):
            data = pd.read_csv(route, skiprows=24, delimiter='\\s+')

        elif (year=='2021' and (float(month) <9 or float(month)>11)):
            data = pd.read_csv(route, skiprows=25, delimiter='\\s+')

        elif (year=='2021' and  float(month)==11):
            data = pd.read_csv(route, skiprows=25, delimiter='\\s+')
    elif float(year) > 2022:
        file_type = 'current'
    elif 1980 <= float(year) < 2000:
        file_type = 'old'
        if float(year) < 1995:
            data = pd.read_csv(route, skiprows = 24, delimiter='\\s+')
        elif float(year) >= 1995:
            data = pd.read_csv(route, skiprows = 26, delimiter='\\s+')
    else:
        file_type = 'old'
        if float(year) == 2000 and float(month) <= 4:
            data = pd.read_csv(route, skiprows=12, delimiter='\\s+', skip_blank_lines=True)
        elif ((float(year) == 2000 and float(month) >= 4) or float(year) > 2000) :
            data = pd.read_csv(route, skiprows=12, delimiter='\\s+', skip_blank_lines=True)
    #print(f'\n>>>>>>>>>>>>>> {day}-{month}-{year} <<<<<<<<<<<<<<')
    print('>>>>>>>>>>>>>> El bucle de llegir el dia', d, m, y, 'ha acabat <<<<<<<<<<<<<<')
    print('\n')
    print('\n')

    return data, file_type, day
"""
def read_file(d, m, y):
    #Aquesta funcio llegeix les dades online
    print('>>>>>>>>>>>>>> El bucle de llegir el dia', d, m, y, 'ha comencat <<<<<<<<<<<<<<')
    day = d
    month = m
    year = y
    if float(year) >= 2000:
        file = open_link_with_condition(year, month)

        if float(year) > 2011:
            file_type = ’new’
            if (year == ’2019’ or year == ’2018’ or year == ’2017’ or year == ’2016’ or  year == ’2014’ or year == ’2013’):
                data = pd.read_csv(file, skiprows=24, delimiter=’\\s+’)
            elif year==’2015’ and (float(month)==6):
                data = pd.read_csv(file, skiprows=32, delimiter=’\\s+’)
            elif year == ’2015’ and (float(month)!=6):
                data = pd.read_csv(file, skiprows=24, delimiter=’\\s+’)
            elif year == ’2012’:
                data = pd.read_csv(file, skiprows=26, delimiter=’\\s+’)

            elif (year == ’2020’ or year == ’2022’):
                data = pd.read_csv(file, skiprows=25, delimiter=’\\s+’)

            elif (year == ’2021’ and (float(month) >= 9 and float(month) <11)):
                data = pd.read_csv(file, skiprows=24, delimiter=’\\s+’)

            elif (year==’2021’ and (float(month) <9 or float(month) >= 11)):
                data = pd.read_csv(file, skiprows=25, delimiter=’\\s+’)

            elif (year==’2021’ and  float(month)==11):
                data = pd.read_csv(file, skiprows=25, delimiter=’\\s+’)
        elif float(year) > 2022:
            file_type = ’current’
        else:
            file_type = ’old’
            data = pd.read_csv(file, skiprows=12, delimiter=’\\s+’)
    elif 1980 <= float(year) < 2000:
        route = f’C:/Users/pep/OneDrive - UAB/Escritorio/Variacio de D/Dades anys antics/{year}/ebr{year}{month}dhor.hor’
        if float(year) < 1995:
            data = pd.read_csv(route, skiprows = 24, delimiter=’\\s+’)
        elif float(year) >= 1995:
            data = pd.read_csv(route, skiprows = 26, delimiter=’\\s+’)
        file_type = ’old’
    #print(f’\n>>>>>>>>>>>>>> {day}-{month}-{year} <<<<<<<<<<<<<<’)
    print(’>>>>>>>>>>>>>> El bucle de llegir el dia’, d, m, y, ’ha acabat <<<<<<<<<<<<<<’)
    print(’\n’)
    print(’\n’)
    return data, file_type, day
    """
# *****************************************************************************

#Aquesta funcio esta creada perque totes les llistes tinguin
#les mateixes dades
#Ja que segons quina epoca es llegeixi, les dades que dona
#l’observatori son unes o unes altres
def data_treatment(data, file_type):
    if file_type == ’new’ or file_type == ’current’:
```

```python
        data['EBRD'] = np.rad2deg(np.arctan(data['EBRY']/data['EBRX']))*60
        return data
    elif file_type == 'old':
        EBRx = []
        EBRy = []
        for i in range(len(data['EBRD'])):
            EBRx.append(np.cos(data['EBRD'][i]*2*math.pi/360)*data['EBRH'][i])
            EBRy.append(np.sin(data['EBRD'][i]*2*math.pi/360)*data['EBRH'][i])
        EBRx = pd.DataFrame(np.transpose(EBRx))
        EBRy = pd.DataFrame(np.transpose(EBRy))
        headers = ['DATE', 'TIME', 'EBRD', 'EBRX', 'EBRY']
        dades = pd.DataFrame({})
        dades = pd.concat([data['DATE'], data['TIME'], data['EBRD'], EBRx, EBRy], axis=1)
        dades.columns = headers
        return dades

# *****************************************************************************

#Aquesta funcio nomes em retorna les hores, i tot i que no  es
#necessaria la del migdia
#S'ha deixat ja que la tinc implementada en tot el codi i vull
#evitar els possibles errors que surgeixin per treure-la
def notable_times2(y, m, d):
    latitude, longitude = 40.820817, 0.495186
    city = LocationInfo("EBR", "Catalunya", "Europe", latitude, longitude)
    date = datetime.strptime(str(y) + '-'+ str(m) + '-' + str(d), '%Y-%m-%d')

    s = sun(city.observer, date=datetime.date(date))

    if s['sunrise'].minute >= 30:
        sunrise_index = (s['sunrise'].hour)+1
    elif s['sunrise'].minute < 30:
        sunrise_index = (s['sunrise'].hour)
    noon_index = (s['noon'].hour)
    if s['sunset'].minute >= 30:
        sunset_index = (s['sunset'].hour)+1
    elif s['sunset'].minute < 30:
        sunset_index = (s['sunset'].hour)


    t_indexes = [sunrise_index, noon_index, sunset_index]

    return t_indexes

# *****************************************************************************

#Aquesta funcio em retorna l'hora de la sortida i posta de
#sol aixi com els minuts
#Mes endavant ja es veura perque es necessaria
def notable_minutes(y, m, d ):
    latitude, longitude = 40.820817, 0.495186
    city = LocationInfo("EBR", "Catalunya", "Europe", latitude, longitude)
    date = datetime.strptime(str(y) + ’-’+ str(m) + ’-’ + str(d), ’%Y-%m-%d’)

    s = sun(city.observer, date=datetime.date(date))

    sunrise = s[’sunrise’].minute
    sunset = s[’sunset’].minute

    if s[’sunrise’].minute >= 30:
        sunrise_h = (s[’sunrise’].hour)+1
    elif s[’sunrise’].minute < 30:
        sunrise_h = (s[’sunrise’].hour)
    if s[’sunset’].minute >= 30:
        sunset_h = (s[’sunset’].hour)+1
    elif s[’sunset’].minute < 30:
        sunset_h = (s[’sunset’].hour)
    t_indexes = [sunrise, sunrise_h, sunset, sunset_h]

    return t_indexes

# *****************************************************************************

#Aquesta funcio serveix per filtrar les dades, primer segons el
#dia i despres segons la hora
#Tot i que es una mica ineficient es el que em semblava mes facil
# per no provocar possibles
#errors alhora de seleccionar els indexos
def filter_data(raw_data, file_type, day, a, b, secular):

    i_start, i_end = ((int(day)-1)*24), ((int(day))*24)#Primer torno els dies
    filtered_data = pd.DataFrame({})
    filtered_data = [raw_data[’DATE’][i_start:i_end+1],
                     raw_data[’TIME’][i_start:i_end+1],
                     raw_data[’EBRD’][i_start:i_end+1],
                     raw_data[’EBRX’][i_start:i_end+1],
                     raw_data[’EBRY’][i_start:i_end+1]]

    headers = [’DATE’, ’TIME’, ’EBRD’, ’EBRX’, ’EBRY’]
    data = pd.concat(filtered_data, axis=1, keys=headers)
    data = data.reset_index()
    data[’HOURS’] = data.index
    contador_random = tercer_filtre(data, file_type)
    if contador_random == 1:
        return data,i_start, contador_random

    dades_00 = data[’EBRD’][0] - secular
    dades_23 = data[’EBRD’][23] - secular

    coefficients = regressio_magnetosfera(dades_00, dades_23)
    model = np.poly1d(coefficients)
    for j in range(len(data[’EBRD’])):
        data[’EBRD’][j] -= (model(j)+secular)
    i_start_2, i_end_2 = a,b#Ara torno les hores
    filtered_data_2 = pd.DataFrame({})
    filtered_data_2 = [data[’DATE’][i_start_2:i_end_2+1],
                       data[’TIME’][i_start_2:i_end_2+1],
                       data[’EBRD’][i_start_2:i_end_2+1],
```

```python
                          data['EBRX'][i_start_2:i_end_2+1],
                          data['EBRY'][i_start_2:i_end_2+1]]

    headers = ['DATE', 'TIME', 'EBRD', 'EBRX', 'EBRY']
    data_2 = pd.concat(filtered_data_2, axis=1, keys=headers)
    data_2 = data_2.reset_index()
    data_2['HOURS'] = data_2.index
    return data_2, i_start, contador_random

# ******************************************************************************

#Aquesta funcio serveix per filtrar les dades nomes segons el dia
def filter_data_2(raw_data, file_type, day, secular):

    i_start, i_end = ((int(day)-1)*24), ((int(day))*24)
    filtered_data = pd.DataFrame({})
    filtered_data = [raw_data['DATE'][i_start:i_end+1],
                     raw_data['TIME'][i_start:i_end+1],
                     raw_data['EBRD'][i_start:i_end+1],
                     raw_data['EBRX'][i_start:i_end+1],
                     raw_data['EBRY'][i_start:i_end+1]]

    headers = ['DATE', 'TIME', 'EBRD', 'EBRX', 'EBRY']
    data = pd.concat(filtered_data, axis=1, keys=headers)
    data = data.reset_index()
    data['HOURS'] = data.index

    dades_00 = data['EBRD'][0] - secular
    dades_23 = data['EBRD'][23] - secular

    coefficients = regressio_magnetosfera(dades_00, dades_23)
    model = np.poly1d(coefficients)
    for j in range(len(data['EBRD'])):
        data['EBRD'][j] -= (model(j)+secular)

    return data

# ******************************************************************************

#Segon filtre serveix per si hi ha alguna dada malament, pero no surt un numero molt gran
#Per aixo es comprova la pendent i s'estableix un limit de 5 cops major a l'anterior
#Cal notar que el 5 es arbitrari pero s'ha fet amb la consciencia que solucionava algun problema
def segon_filtre(dades, dades_anterior):
    Slope_actual = []
    Slope_anterior = []
    contador_pendent = 0
    for j in range(len(dades['EBRD'])-1):
        #Faig aixo ja que se que estic agafant dades cada hora i per tant x_i-x_i+1=1
        slope = (dades['EBRD'][j]-dades['EBRD'][j+1])
        Slope_actual.append(slope)
    for j in range(len(dades_anterior['EBRD'])-1):
        slope_anterior = (dades_anterior['EBRD'][j]-dades_anterior['EBRD'][
    j+1])
        Slope_anterior.append(slope_anterior)

    #Imposo que el pendent del dia actual sigui similar al del dia
    #anterior bo amb un rang,amb un range de 5 cops l'anterior
    if max(Slope_actual) > 5*max(Slope_anterior):
        contador_pendent = 1
    return contador_pendent

#
    *****************************************************************************

#Aixo es basicament un filtre per comprovar que les dades siguin
#bones o no
#ja que pot haver dies on les dades siguin grans ja que hi ha hagut
#algun problema
#I aquell dia no s'han pogut agafar o algo
def tercer_filtre(data, file_type):
    contador_random=0
    if file_type=='current' or file_type=='new':
    #Diferencio si es nou o no ja que en l'OBS de l'Ebre
    #posen notacio diferent segons
    #L'any que estiguem mirant i per tant es important
    #fer aquesta diferenciacio
        for i in range(len(data['EBRX'])):
            if ((data['EBRX'][i]) >= 99999.00 and (data['EBRY'][i]) <=
    99999.00) :
            #Miro que sigui major a aquest valor ja que es el que hi ha
            #A la pagina web pels valors no acceptables
                data=data.drop([i], axis = 0)
                contador_random += 1
                break
            elif ((data['EBRX'][i]) <= 99999.00 and (data['EBRY'][i]) >=
    99999.00):
                contador_random += 1
                data=data.drop([i], axis = 0)
                break
            elif ((data['EBRX'][i]) >= 99999.00 and (data['EBRY'][i]) >=
    99999.00) and i%60!=0:
                contador_random += 1
                data=data.drop([i], axis = 0)
                break
    elif file_type=='old':
        for i in range(len(data['EBRD'])):
            if (data['EBRD'][i]) >= 90000.00:
                print('Dades massa grans')
                data=data.drop([i], axis = 0)
                contador_random += 1
                print(contador_random)
                break
    return contador_random

#
    *****************************************************************************

#Aquesta funcio simplement retorna el dia de l'any
#(doy, per 'day of the year')
#( en numeros natural, sent l'1 de gener l'1)
```

```python
#que em servira per despres
#Principalment la utilitat d'aquesta funcio sera alhora de fer
#llistes per tenir en compte que el dia X es el numero Y
#de la llista
def dia_any(d,m,y):
    #Comenco diferenciant si l'any es o no de traspas
    if ((int(y) % 4 == 0 and int(y) % 100 != 0) or (int(y) % 100 == 0 and int(y) % 400 == 0)):
        day_of_year = date(int(y), int(m), int(d)).timetuple().tm_yday
    else:
        if int(m)>2:#Si no ho es( de traspas) miro que el mes sigui major o menor a 2
            day_of_year = date(int(y), int(m), int(d)).timetuple().tm_yday+1
        else:#Quan hagi passat Febrer li haure d'afegir un 1 al doy per comptar que m'he saltat el 29/2
            day_of_year = date(int(y), int(m), int(d)).timetuple().tm_yday
    return day_of_year
# ******************************************************************************

def day_to_date(day_number):
    date_format = '%Y-%m-%d'
    # create a datetime object for January 1st of the given year
    start_date = datetime(2000 , 1, 1)
    #Poso l'any 2000 perque tinc les dades ordenades com si fosin
    #anys de traspas
    # add the number of days to the start date
    result_date = start_date + timedelta(days=day_number)
    # format the date string using the specified format
    return result_date.day, result_date.month

# ******************************************************************************

#Funcio per omplir els dies buits
def empty_days(Mitja_dies):
    Dies_buits = []
    for i in range(366):
        #En aquesta primera part simplement mirare quins dies estan buits i els ficare en una llista per comoditat
        Llista_dies = []
        Llista_dies_anteriors = []
        cond_1 = all(Mitja_dies[i][k] == 0 for k in range(24))#Imposo primer la condicio que el dia que estic mirant estigui buit
        if cond_1 == True:
            print('El dia buit es:', [day_to_date(i)] )
            Llista_dies.append([Mitja_dies[i],i])
            Dies_buits.append([day_to_date(i)])
            j = i+1#El +1 es important per no comptar dos cops el mateix dia
            if j >= 366:#Aquesta condicio s'imposa per si ens trobem amb l'ultim dia de l'any
                j = 0
            numero_dies_superior = 0
            while True:
                cond_1_2 = all(Mitja_dies[j][k] == 0 for k in range(24))#Imposo la condicio de que el seguent dia de l'any tambe estigui buit
                #I vaig guardant els dies de l'any buits
                if cond_1_2 == True:
                    numero_dies_superior += 1
                    Llista_dies.append([Mitja_dies[j],j])
                    Dies_buits.append([day_to_date(j)])
                    print('Un altre dia buit seguit:', [day_to_date(j)])
                    j += 1
                    if j >= 366:#S'imposa pel mateix motiu que s'ha imposat a dalt
                        j = 0
                else:#Quan el seguent dia esta ple l'afegeixo a la llista i surto del while
                    Llista_dies.append([Mitja_dies[j],j])
                    break
            l=i-1
            if l <= -1:#Aquesta condicio s'imposa per si es dona el cas del primer dia de l'any
                l = 365
            numero_dies_inferior = 0
            #Repeteixo el mateix procediment pero disminuint els dies. Cal notar que aquesta condicio no s'hauria d'activar si no es que s'esta treballant amb el primer dia de l'any ja que en principi ja agafo el primer dia buit
            while True:
                cond_1_3 = all(Mitja_dies[l][k] == 0 for k in range(24))
                if cond_1_3 == True:
                    numero_dies_superior += 1
                    Llista_dies_anteriors.append([Mitja_dies[l],l])
                    Dies_buits.append([day_to_date(l)])
                    l -= 1
                    if l <= -1:#S'imposa pel mateix motiu que s'ha imposat la de dalt
                        l = 365
                else:
                    Llista_dies_anteriors.append([Mitja_dies[l],l])
                    break
            Llista_dies_anteriors2 = []
            for m in range(len(Llista_dies_anteriors)):
                Llista_dies_anteriors2.append(Llista_dies_anteriors[len(Llista_dies_anteriors)-1-m])
            Llista_dies_total = Llista_dies_anteriors2 + Llista_dies
            numero_d_buit = len(Llista_dies_total)-2
            #La idea que he seguit aqui ha estat omplir els dies que hi havia buits amb els dos mes propers que hi havia plens
            #per aixo s'ha separat el proces en dues parts depenen de si el numero de dies buits era parell o no, i es sumara
            #per omplir el dia buit es tindran en compte diferents pesos dels dies ja plens mes propers segons quin es trobi mes proper

            dia_1 = day_to_date(Llista_dies_total[0][1])
            times_1 = notable_times2(2000, dia_1[1], dia_1[0])
            dia_2 = day_to_date(Llista_dies_total[-1][1])
            times_2 = notable_times2(2000, dia_2[1], dia_2[0])
            if times_1[0] >times_2[0]:
                hora_inici = times_2[0]
            elif times_1[0] <= times_2[0]:
                hora_inici = times_1[0]

            if times_1[2] > times_2[2]:
                hora_final = times_1[2]
```

```python
            elif times_1[2] <= times_2[2]:
                hora_final = times_2[2]

            if numero_d_buit%2 == 0:#Cas amb els dies buits parells
                for m in range(int(numero_d_buit/2)):
                    frac = (numero_d_buit-(1+m))/(numero_d_buit)#Aqui em declaro la fraccio que sera el meu pes alhora de fer els calculs
                    #La idea d'aquesta fraccio es dividir l'interval buit en fragments que estiguin pessats depenent de com de separats estan dels dies plens
                    #Anem a veure un exemple: si tenim els dies a, b, c, d, e, f on NOMES a i f estan plens, els dies s'ompliran de la seguent forma:
                    #c i d com estan en el centre tindran (a+f)/2 per altre banda b sera a*3/4+f*1/4 i e sera f*3/4+a*1/4
                    #La idea d'aquest fragment es fer aixo per qualsevol numero de dies buits
                    for n in range(len(Llista_dies_total[int(numero_d_buit/2)][0])):
                        Llista_dies_total[1+m][0][n] = frac*Llista_dies_total[0][0][n]+Llista_dies_total[numero_d_buit+1][0][n]*(1-frac)
                        Llista_dies_total[numero_d_buit-m][0][n] = Llista_dies_total[0][0][n]*(1-frac)+Llista_dies_total[numero_d_buit+1][0][n]*frac
                    #Estic imposant condicions per evitar problemes en els canvis d'hora, ja que es possible tenir entre mig d'un buit dies que tinguin hores diferents
                    if Llista_dies_total[0][0][hora_inici] != 0 and Llista_dies_total[-1][0][hora_inici] == 0:
                        Llista_dies_total[1+m][0][hora_inici] = Llista_dies_total[0][0][hora_inici]
                        Llista_dies_total[numero_d_buit-m][0][hora_inici] = Llista_dies_total[0][0][hora_inici]
                    elif Llista_dies_total[0][0][hora_inici] == 0 and Llista_dies_total[-1][0][hora_inici] != 0:
                        Llista_dies_total[1+m][0][hora_inici] = Llista_dies_total[-1][0][hora_inici]
                        Llista_dies_total[numero_d_buit-m][0][hora_inici] = Llista_dies_total[-1][0][hora_inici]
                    if Llista_dies_total[0][0][hora_final] != 0 and Llista_dies_total[-1][0][hora_final] == 0:
                        Llista_dies_total[1+m][0][hora_final] = Llista_dies_total[0][0][hora_final]
                        Llista_dies_total[numero_d_buit-m][0][hora_final] = Llista_dies_total[0][0][hora_final]
                    elif Llista_dies_total[0][0][hora_final] == 0 and Llista_dies_total[-1][0][hora_final] != 0:
                        Llista_dies_total[1+m][0][hora_final] = Llista_dies_total[-1][0][hora_final]
                        Llista_dies_total[numero_d_buit-m][0][hora_final] = Llista_dies_total[-1][0][hora_final]

            else:#Aqui es segueix el mateix procediment per un numero de dies buits senar
                for n in range(len(Llista_dies_total[int(np.ceil(numero_d_buit/2))][0])):
                    #Aqui s'omple el dia del mig(cal notar que ara nomes hi ha un mentres que abans hi havia dos)
                    Llista_dies_total[int(np.ceil(numero_d_buit/2))][0][n]
```

```
= (Llista_dies_total[0][0][n]+Llista_dies_total[numero_d_buit+1][0][n])/2
                for m in range(int(numero_d_buit/2)):
                    frac = (numero_d_buit-(1+m))/(numero_d_buit)
                    for n in range(len(Llista_dies_total[0][0])):
                        Llista_dies_total[1+m][0][n] = frac*Llista_dies_total[0][0][n]+Llista_dies_total[numero_d_buit+1][0][n]*(1-frac)
                        Llista_dies_total[numero_d_buit-m][0][n] = Llista_dies_total[0][0][n]*(1-frac)+Llista_dies_total[numero_d_buit+1][0][n]*frac

                    #Estic imposant condicions per evitar problemes en els canvis d'hora
                    if Llista_dies_total[0][0][hora_inici] != 0 and Llista_dies_total[-1][0][hora_inici] == 0:
                        Llista_dies_total[1+m][0][hora_inici] = Llista_dies_total[0][0][hora_inici]
                        Llista_dies_total[numero_d_buit-m][0][hora_inici] = Llista_dies_total[0][0][hora_inici]
                    elif Llista_dies_total[0][0][hora_inici] == 0 and Llista_dies_total[-1][0][hora_inici] != 0:
                        Llista_dies_total[1+m][0][hora_inici] = Llista_dies_total[-1][0][hora_inici]
                        Llista_dies_total[numero_d_buit-m][0][hora_inici] = Llista_dies_total[-1][0][hora_inici]
                    if Llista_dies_total[0][0][hora_final] != 0 and Llista_dies_total[-1][0][hora_final] == 0:
                        Llista_dies_total[1+m][0][hora_final] = Llista_dies_total[0][0][hora_final]
                        Llista_dies_total[numero_d_buit-m][0][hora_final] = Llista_dies_total[0][0][hora_final]
                    elif Llista_dies_total[0][0][hora_final] == 0 and Llista_dies_total[-1][0][hora_final] != 0:
                        Llista_dies_total[1+m][0][hora_final] = Llista_dies_total[-1][0][hora_final]
                        Llista_dies_total[numero_d_buit-m][0][hora_final] = Llista_dies_total[-1][0][hora_final]

            for h in range(int(np.ceil(numero_d_buit/2))):
                index = Llista_dies_total[h+1][1]
                for g in range(24):
                    Mitja_dies[index][g] = Llista_dies_total[h+1][0][g]
    return Mitja_dies, Dies_buits
#*****************************************************************************

def fun(x, A, w, phi):
    return A*np.sin(w*x+phi)

#Funcio per fer ajustar dies
def funct_curvefit_dies(Mitja_dies, Dies_buits, Dies_mal_min):
    def fun(x, A, w, phi):
        return A*np.sin(w*x+phi)
    Limits_mitja_dies = np.zeros((366,2))#Aquesta donal'interval on la mitja no es nula per tal de que es puguin plotejar be les grafiques, el primer elements es el maxim i el segon el minim
    Max_min_Md = np.zeros((366,2))#Aquesta llista dona els maxims i minims per tal de que es puguin plotejar be les grafiques, el primer elements
```

```
es el maxim i el segon el minim
    hores = []
    llegenda = ['Fit', 'Data', 'Difference']
    for j in range(24):
        hores.append(str(j)+':'+'30')
    As = []
    Freqs = []
    Phis = []
    Curve_fit = np.zeros((367,24))
    for i in range(24):
        Curve_fit[366][i] = i
    for i in range(366):
        llista_dies = []
        Md_no_nuls = []
        for j in range(len(Mitja_dies[0])):
            if Mitja_dies[i][j] != 0:
                llista_dies.append(j)
                Md_no_nuls.append(Mitja_dies[i][j])
        dia, mes = day_to_date(i)
        for j in Dies_mal_min:
            if int(j[0]) == dia and int(j[1]) == mes:
                sunrise = j[-2]
                sunset = j[-1]
                break
            else:
                times = notable_minutes(2000, mes, dia)
                sunrise = times[1]
                sunset = times [3]
                #No importa l'any que utilitzi ja que els casos que donen
problemes no els tinc en compte aqui
        #El primer maxim i minim es amb el qual plotejare
        max_1 = int(sunset)
        min_1 = int(sunrise)
        #Els segons maxims i minims son els que utilitzare per donar un bon
 guess de l'amplitud per fer el curve fit
        max_2 = max(Md_no_nuls)
        min_2 = min(Md_no_nuls)
        for j in range(len(Dies_buits)):
            if int(dia) ==int(Dies_buits[j][0][0]) and int(mes) == int(
Dies_buits[j][0][1]):
                dia_omplert = 'Y'
                break
            else:
                dia_omplert = 'N'
        print('<<<<<<<' + 'Dia' + str(dia) + '\t ,mes' + str(mes) + '
>>>>>>>\n')#Aqui he de posar el dia i el mes que estic fent
        Amplitud=(max_2-min_2)/2
        #Estic imposant les condicions que m'interesen, que son que en els
extrems trobi un valor de 0,
        #Per fer aixo he imposat que w*t_1+phi = 0 i w*t_2+phi = 2*pi, on
t_1 es la primera hora amb sol, i t_2 la ultima
        #Per com treballo es important que per calcular la fase i la freq
no tingui en compte que la hora es les 7:30( per exemple)
        #Ja que aixo fara que el grafic no quedi exactament en el 0
        freq = 2*math.pi/(max_1-min_1)
        fase = -2*math.pi*min_1/(max_1-min_1)
        #Fico el +1 en el max_1 per tal de que m'arribi on vull, ja que el
for es queda un per sota del maxim
        As.append(Amplitud)
        Freqs.append(freq)
        Phis.append(fase)
        Curve_fit[i][min_1:max_1+1]=fun(Mitja_dies[366][min_1:max_1+1], Amplitud, freq, fase)
        dif = []
        for j in range(len(Curve_fit[i])):
            dif.append(-Curve_fit[i][j]+Mitja_dies[i][j])
        plt.plot(hores[min_1:max_1+1],Curve_fit[i][min_1:max_1+1], color = 'r')
        plt.plot(hores[min_1:max_1+1], Mitja_dies[i][min_1:max_1+1])
        plt.plot(hores[min_1:max_1+1], dif[min_1:max_1+1], color = 'g')
        plt.yticks(fontsize = 15)
        plt.xticks(rotation=45, fontsize = 15)
        plt.xlabel('Time(Hours)', fontsize = 17)
        plt.ylabel('D(min)', fontsize = 17)
        if dia_omplert == 'Y':
            plt.title('Mean of D(t) of ' + 'Dia' + str(dia)+',mes' + str(mes) + 'Aquest dia ha estat creat', fontsize = 15 )
        else:
            plt.title('Mean of D(t) of ' + 'Dia' + str(dia)+ ',mes' + str(mes), fontsize = 15 )
        plt.legend(llegenda, fontsize = 12)
        plt.rcParams['xtick.major.size'] = 10
        plt.rcParams['ytick.major.size'] = 10
        plt.rcParams['xtick.minor.size'] = 7
        plt.rcParams['ytick.minor.size'] = 7
        plt.rcParams['xtick.top'] = False
        plt.rcParams['ytick.right'] = False
        plt.rcParams["figure.figsize"] = (7,7)
        plt.show()
    return Curve_fit, As, Freqs, Phis

# *****************************************************************************
# *****************************************************************************

def Classificacio_dies(Dades1):
    #Comenco llegint els dies que son bons depenent de l'index i mels guardo
    Dies_bons=[]
    n=0
    Dades_bones=0
    Dades_totals=0
    Dades_dolentes=0
    while n<len(Dades1[0]):
        Dades=Dades1[:,n:n+8]
        estat_dada='neutre'
        for i in Dades[3]:
            if i!='0+' and i!='0-' and i!='0o' and i!='1+' and i!='1-' and i!='1o' and i!=-1:
                Dades_dolentes+=1
                print('Dd', Dades_dolentes)
                estat_dada='Dada dolenta'
                Dades_totals+=1
                print('Dt', Dades_totals)
            elif i=='0+' or i=='0-' or i=='0o' or i=='1+' or i=='1-' or i=='1o'  or i==-1:
                if estat_dada=='neutre' or estat_dada=='bona':
                    estat_dada='bona'
        if estat_dada=='bona':
            Dies_bons.append(Dades[0][0])
            Dades_bones+=1
            Dades_totals+=1
            print('Dt', Dades_totals)
        n+=8#Vaig augmentat de 8 en 8 per avancar tot el dia, ja que cada dia nomes s'agafen 8 mesures
    Dies_Bons_2=[]#Canvio el format a un que em sigui mes util
    for i in Dies_bons:
        if i.day<10 and i.month<10:
            Dies_Bons_2.append([str(0)+str(i.day),str(0)+str(i.month), str(i.year)])
        elif i.day<10 and i.month>=10:
            Dies_Bons_2.append([str(0)+str(i.day),str(i.month), str(i.year)])
        elif i.day>=10 and i.month<10:
            Dies_Bons_2.append([str(i.day),str(0)+str(i.month), str(i.year)])
        elif i.day>=10 and i.month>=10:
            Dies_Bons_2.append([str(i.day),str(i.month), str(i.year)])
    return Dies_Bons_2
#*****************************************************************************
#Funcio que mira quins dies tenen dades dolentes i em torna els dies bons
#En aquesta funcio s'ajunten diversos filtres que he creat en altres funcions, ja que es una de les funcions 'mare' d'aquest codi
def Llistat_dades_dies(Dies_Bons_2, Dades2, Coeficients, Anys):

    dies_concrets = input('Do you want to look at some day?(y/n) ')
    if dies_concrets == 'y':
        LLista_dia_con = []
        day = input('Which day do you want to look at?').zfill(2)
        month = input('Of Which month?').zfill(2)
    shape = (366, 24)  # (dimension 0, dimension 1)

    # Generar la matriz tridimensional vacia
    Mitja_dies_l = [[[] for _ in range(shape[1])] for _ in range(shape[0])]
    Mitja_dies_l.append([])
    for i in range(24):
        Mitja_dies_l[366].append(i)

    Dies_bons_filtrats = []
    dies_dolents = 0
    i = 0
    Dies_pendent_gran = []
    Secular = []
    #Llegeixo les dades i vaig guardant les dades comprovant les diverses condicions que ja s'ha imposat
    #Com la del pendent o la que les dades siguin bones
    while True:
        if i == 0:

            in_coef, dif_dies = index_coef(int(Dies_Bons_2[i][2]), int(Dies_Bons_2[i][1]), int(Dies_Bons_2[i][0]), Anys)
            model = np.poly1d(Coeficients[in_coef])
            secular = model(dif_dies)
            Secular.append(secular)

            dades, file_type, day  = read_file(Dies_Bons_2[i][0], Dies_Bons_2[i][1], Dies_Bons_2[i][2])
            day = Dies_Bons_2[i][0]
            times = notable_times2(Dies_Bons_2[i][2], Dies_Bons_2[i][1], Dies_Bons_2[i][0])
            dades = data_treatment(dades,file_type)
            dades,start_time, contador_random = filter_data(dades, file_type,day, times[0], times[2], secular)
            if contador_random > 0 and i != (len(Dies_Bons_2)-1):
                dies_dolents += 1
                print(’El bucle ha acabat’)
                print(i, len(Dies_Bons_2))
                Dies_Bons_2.pop(i)
                continue
            elif contador_random>0 and i == (len(Dies_Bons_2)-1):
                print(’El bucle ha acabat’)
                print(i, len(Dies_Bons_2))
                Dies_Bons_2.pop(i)
                dies_dolents += 1
                break
            #Aquesta part pot ser una bona idea comentarla si no s’esta segur de quins dies son dolents ja que el que fa es mirar els dies que son dolents i no els te en compte
            #Pero cal aclarir que els dies dolents s’han vist de forma manual ja que es veien un grafics dolents pero en principi els dies eren calmats
            contador_dd = eliminador_dies(Dies_Bons_2[i][0], Dies_Bons_2[i][1], Dies_Bons_2[i][2])
            if contador_dd > 0 and i != (len(Dies_Bons_2)-1):
                print(i, len(Dies_Bons_2))
                Dies_Bons_2.pop(i)
                dies_dolents += 1
                print(’Dia dolent’)
                continue
            elif contador_dd > 0 and i == (len(Dies_Bons_2)-1):
                print(’El bucle ha acabat’)
                print(i, len(Dies_Bons_2))
                Dies_Bons_2.pop(i)
                dies_dolents += 1
                print(’pendent massa gran’)
                break

            dades_anterior=dades
            Dies_bons_filtrats.append([Dies_Bons_2[i][0], Dies_Bons_2[i][1], Dies_Bons_2[i][2]])

            for j in range(len(dades)):
                index_dia = dia_any(Dies_Bons_2[i][0], Dies_Bons_2[i][1], Dies_Bons_2[i][2])-1#Per tenir en compte que la llista comenca en el 0 li resto 1
                index_hora = dades[’index’][j]
                (Mitja_dies_1[index_dia][index_hora]).append(dades[’EBRD’][j])
                if i == (len(Dies_Bons_2)-1):
                    print(’El bucle ha acabat’)
                    break
            if dies_concrets == ’y’ and (day == Dies_Bons_2[i][0] and month == Dies_Bons_2[i][1]):
                LLista_dia_con.append([Dies_Bons_2[i][0], Dies_Bons_2[i][1], Dies_Bons_2[i][2]])
```

```python
        i += 1
    if i >= 1:

        in_coef, dif_dies = index_coef(int(Dies_Bons_2[i][2]), int(Dies_Bons_2[i][1]), int(Dies_Bons_2[i][0]), Anys)
        model = np.poly1d(Coeficients[in_coef])
        secular = model(dif_dies)

        dades, file_type, day = read_file(Dies_Bons_2[i][0], Dies_Bons_2[i][1], Dies_Bons_2[i][2])
        day=Dies_Bons_2[i][0]
        dades = data_treatment(dades,file_type)
        times = notable_times2(Dies_Bons_2[i][2], Dies_Bons_2[i][1], Dies_Bons_2[i][0])
        dades,start_time, contador_random = filter_data(dades, file_type, day, times[0], times[2], secular)
        if contador_random > 0  and i != (len(Dies_Bons_2)-1):
            dies_dolents += 1
            print(i,len(Dies_Bons_2))
            Dies_Bons_2.pop(i)
            continue
        elif contador_random > 0 and i == (len(Dies_Bons_2)-1):
            print('El bucle ha acabat')
            print(i, len(Dies_Bons_2))
            Dies_Bons_2.pop(i)
            dies_dolents+=1
            break

        #Segon filtre, serveix per si hi ha alguna dada malament, pero no surt un numero molt gran
        contador_pendent = segon_filtre(dades, dades_anterior)
        if contador_pendent > 0 and i != (len(Dies_Bons_2)-1):
            print(i, len(Dies_Bons_2))
            Dies_pendent_gran.append(Dies_Bons_2[i])
            Dies_Bons_2.pop(i)
            dies_dolents += 1
            print('pendent massa gran')
            continue
        elif contador_pendent > 0 and i == (len(Dies_Bons_2)-1):
            print('El bucle ha acabat')
            print(i, len(Dies_Bons_2))
            Dies_pendent_gran.append(Dies_Bons_2[i])
            Dies_Bons_2.pop(i)
            dies_dolents += 1
            print('pendent massa gran')
            break
        #Aquesta part pot ser una bona idea comentarla si no s'esta segur de quins dies son dolents ja que el que fa es mirar els dies que son dolents i no els te en compte
        #Pero cal aclarir que els dies dolents s'han vist de forma manual ja que es veien un grafics dolents pero en principi els dies eren calmats
        contador_dd = eliminador_dies(Dies_Bons_2[i][0], Dies_Bons_2[i][1], Dies_Bons_2[i][2])
        if contador_dd > 0 and i != (len(Dies_Bons_2)-1):
            print(i, len(Dies_Bons_2))
            Dies_Bons_2.pop(i)
            dies_dolents += 1
            print('Dia dolent')
            continue
            elif contador_dd > 0 and i == (len(Dies_Bons_2)-1):
                print('El bucle ha acabat')
                print(i, len(Dies_Bons_2))
                Dies_Bons_2.pop(i)
                dies_dolents += 1
                print('pendent massa gran')
                break

            dades_anterior = dades
            Dies_bons_filtrats.append([Dies_Bons_2[i][0], Dies_Bons_2[i][1], Dies_Bons_2[i][2]])
            for j in range(len(dades)):
                index_dia = dia_any(Dies_Bons_2[i][0], Dies_Bons_2[i][1], Dies_Bons_2[i][2])-1#Per tenir en compte que la llista comenca en el 0 li resto 1
                index_hora = dades['index'][j]
                (Mitja_dies_l[index_dia][index_hora]).append(dades['EBRD'][j])
            Secular.append(secular)
            if dies_concrets == 'y' :
                if day == Dies_Bons_2[i][0] and month == Dies_Bons_2[i][1]:
                    LLista_dia_con.append([Dies_Bons_2[i][0], Dies_Bons_2[i][1], Dies_Bons_2[i][2]])
            i += 1
        if dies_concrets == 'y':
            print(LLista_dia_con)
        print(i, len(Dies_Bons_2)-1)
        if i == (len(Dies_Bons_2)):#L'hi he tret el -1 ja que crec que em pot donar problemes per l'ultim dia
            print('El bucle ha acabat')
            break
    return Mitja_dies_l, Dies_bons_filtrats, Secular, Dies_pendent_gran
# *****************************************************************************
#Funcio que em comprova que tots els dies tinguin el mateix numero de dades
#En cas de que tots els dies no tinguin el mateix numero de dades em retorna els dies
#Que donen problemes
def comprovacio_num_dies(Mitja_dies_l, Dies_bons_filtrats):
    Dies_dolents = []
    numero_dies = np.zeros((367,24))
    for i in range(366):
        for j in range(24):
            numero_dies[i][j] = len(Mitja_dies_l[i][j])
        ind = [k for k in range(len(numero_dies[i])) if numero_dies[i][k] != 0]
        if len(ind) != 0:
            element_0 = numero_dies[i][ind[0]]
            cond_1 = all(numero_dies[i][k] == element_0 for k in ind)
            if cond_1 == False:
               #print('Algo falla en el dia', day_to_date(i), i)#El +1 es per tenir en compte com esta ordenada la llista
               Dies_dolents.append( day_to_date(i))
    Dies_comprovar=[]
    for j in range(len(Dies_dolents)):
        for i in Dies_bons_filtrats:
            y = int(i[2])
            if int(i[0]) == int(Dies_dolents[j][0]) and int(i[1]) == int(Dies_dolents[j][1]):
                #print('El dia', i ,'es dolent')
                Dies_comprovar.append(i)

    for i  in range(len(Dies_comprovar)):

        in_coef, dif_dies = index_coef(int(Dies_comprovar[i][2]), int(Dies_comprovar[i][1]), int(Dies_comprovar[i][0]), Anys)
        model = np.poly1d(Coeficients[in_coef])
        secular = model(dif_dies)

        dades, file_type, day  = read_file(Dies_comprovar[i][0], Dies_comprovar[i][1], Dies_comprovar[i][2])
        times = notable_times2(Dies_comprovar[i][2], Dies_comprovar[i][1], Dies_comprovar[i][0])

        dades=data_treatment(dades,file_type)
        dades,start_time, contador_random = filter_data(dades, file_type, day, times[0], times[2], secular)

        Dies_comprovar[i].append([len(dades), times[0], times[2]])

    """
    #Aixo es un plot opcional per visualtizar el problema que estan donant les dades, pero es podria eliminar perfectament
    for i in Dies_comprovar:

        in_coef, dif_dies = index_coef(int(i[2]), int(i[1]), int(i[0]), Anys)
        model = np.poly1d(Coeficients[in_coef])
        secular = model(dif_dies)

        dades, file_type, day = read_file(i[0], i[1], i[2])
        day=i[0]
        times = notable_times2(i[2], i[1], i[0])
        dades=data_treatment(dades,file_type)
        dades,start_time, contador_random = filter_data(dades, file_type, day, times[0], times[2], secular)
        plt.scatter(dades['TIME'], dades['EBRD'])
        plt.title(dades['DATE'][0])
        plt.show()
    """
    return Dies_comprovar, numero_dies

# ****************************************************************************
#Creo una funcio per intentar arreglar un problema que hi ha amb els minuts:
#La cosa es que alguns anys els dies el sol surt en el minut 29 i alguns altres en el 30
#Aixo provoca, que de la forma que estan filtrades les hores les llistes tinguin longituds diferents
#Per tant el que faig es eliminar les dades extres en cas que hi hagi
def arreglar_minuts(Dies_comprovar, Mitja_dies_l):
    i = 0
    while True:
        if len(Dies_comprovar) == 0:
            break
        D_malament = []
        sunrises = []
        sunrises_h = []
        sunsets = []
        sunsets_h = []
        D_malament.append(Dies_comprovar[i])
        doy = dia_any(int(Dies_comprovar[i][0]), int(Dies_comprovar[i][1]), 2000)-1#El -1 es perque la llista comenca a 0
        j = i+1
        times = notable_minutes(int(Dies_comprovar[i][2]), int(Dies_comprovar[i][1]), int( Dies_comprovar[i][0]))
        sunrises.append(times[0])
        sunrises_h.append(times[1])
        sunsets.append(times[2])
        sunsets_h.append(times[3])

        while True:
            if Dies_comprovar[i][0] == Dies_comprovar[j][0] and Dies_comprovar[i][1] == Dies_comprovar[j][1]:
                D_malament.append(Dies_comprovar[j])
                times = notable_minutes(int(Dies_comprovar[j][2]), int(Dies_comprovar[j][1]), int( Dies_comprovar[j][0]))
                sunrises.append(times[0])
                sunrises_h.append(times[1])
                sunsets.append(times[2])
                sunsets_h.append(times[3])
                Dies_comprovar.pop(j)
            else:
                break#Poso el break aqui perque se que estan ordenats per dies i mesos
            if  j >= len(Dies_comprovar)-1:
                  break
        if len(D_malament) > 1:
            Dies_comprovar.pop(i)
        else:
            i += 1
        cond_1 = all(sunrises[0] == sunrises[j]  and (sunrises[j] == 30 or sunrises[j] == 29 or sunrises[j] == 31) for j in range(len(sunrises)))
        cond_2 = all(sunsets[0] == sunsets[j] and (sunsets[j] == 30 or sunsets[j] == 29 or sunsets[j] == 31) for j in range(len(sunsets)))
        if cond_1 == False and cond_2 == False:
            Mitja_dies_l[doy][min(sunrises_h)] = []
            Mitja_dies_l[doy][max(sunsets_h)] = []
        elif cond_1 == True and cond_2 == False:
            Mitja_dies_l[doy][max(sunsets_h)] = []
        elif cond_1 == False and cond_2 == True:
            Mitja_dies_l[doy][min(sunrises_h)] = []
        if i >= len(Dies_comprovar)-1:
            break
    return Mitja_dies_l

# *****************************************************************************

#La idea darrera  d’aquesta funcio no es arreglar els dies que estiguin malament pel tema dels minuts( el mateix que en la funcio de dalt) perque per casualitat no hi son, sino que es tornar una llista que em digui en general quins son els dies que donen problemes
def dies_min_dolent():
    Dies_mal_min = []
    for i in range(366):
        dia, mes = day_to_date(i)
        times = notable_minutes(2000, mes, dia)
        sunrise_h = times[1]
        sunset_h = times[3]
        sunrise = times[0]
        sunset = times [2]
        cond_1 = (sunrise == 29 or sunrise == 30)
        cond_2 = (sunset == 29 or sunset == 30)
        if cond_1 == True and cond_2 == True:
            if sunrise == 29:
                Sunrise_def = sunrise_h+1
            elif sunrise == 30:
                Sunrise_def = sunrise_h
            if sunset == 29:
                Sunset_def = sunset_h
            elif sunset == 30:
                Sunset_def = sunset_h-1
            #La primera dada despres del mes es la hora que haure d'agafar el sunrise
            #La segona dada despres del mes es el sunset que haure d'agafar
            Dies_mal_min.append([str(dia).zfill(2), str(mes).zfill(2), Sunrise_def, Sunset_def])
        elif cond_1 == True and cond_2 == False:
            if sunrise == 29:
                Sunrise_def = sunrise_h+1
            elif sunrise == 30:
                Sunrise_def = sunrise_h
            Dies_mal_min.append([str(dia).zfill(2), str(mes).zfill(2), Sunrise_def, sunset_h])

        elif cond_1 == False and cond_2 == True:
            if sunset == 29:
                Sunset_def = sunset_h
            elif sunset == 30:
                Sunset_def = sunset_h-1
            Dies_mal_min.append([str(dia).zfill(2), str(mes).zfill(2), sunrise_h, Sunset_def])
    return Dies_mal_min
#****************************************************************************
#Aqui busco crearme una llista dels coeficients de la regressio
def Coeff_secu(y_1, m_1, d_1, y_2, m_2, d_2, Dades2, dia_mig_extra, year_extra):
    degree = 1  # Grado del polinomio a ajustar
    Coef_list = []
    Anys = []
    #Primer miro les condicions inicials sota les que he de crear la llista, depenent de si el primer i ultim dia es troben abans o despres de juny
    if(( m_1 < 6 or (m_1 == 6 and d_1 < 15)) and (m_2 > 6 or (m_2 == 6 and d_2 >= 15))):
        for i in range(y_1-1,y_2+1):
            Anys.append(i)
            index_any = i-math.floor(Dades2[0][0])#Es important agafar aquest any com el primer ja que es on comenca la llista de Dades2
            #En el cas que l'ultim dia es trobi despres de juny he d'anar a buscar les dades per internet ja que no
            #son definitives i no les tinc descarregades
            if index_any >= len(Dades2[0])-1:
                print('He de buscar les dades per internet i em canvia tot'
)
                final_day = dia_any(31, 12, year_extra-1)- dia_any(1, 6, year_extra-1) + dia_mig_extra#Estic agafant el numero total de dies fent: dies totals de l'any que estic mirant - dies que portem fins al 1 de juny+ dies que te per la mesura de l'any seguent
                #M'estableixo quin es el rang en les x que he d'agafar per fer el plot:
                #per fer aixo li resto a 365 el primer dia que comenco(l'1 de juny de l'any anterior) i li sumo els dies extres que poso
                X = [1,final_day]
                #Aqui agafo les dues dades que m'interesen
                Y = [Dades2[1][index_any]*60, Dades2[1][-1]*60]
            else:
                #Pels altres casos miro si es un any de traspas o no i poso les X en funcio d'aixo
                if (((int(i)-1) % 4 == 0 and (int(i)-1) % 100 != 0) or ((int(i)-1) % 100 == 0 and (int(i)-1) % 400 == 0)):
                    X = [1,366]
                else:
                    X = [1,365]
                #Agafo les Y dels dos anys que toca
                Y = [Dades2[1][index_any]*60,Dades2[1][index_any+1]*60]
            coefficients = np.polyfit(X, Y, degree)
            Coef_list.append(coefficients)
    elif (((m_1 == 6 and d_1 >= 15) or m_1 > 6) and ((m_2 == 6 and d_2 >= 15) or m_2 > 6)):#No es comenten els seguents elif ja que son identics al primer if
        for i in range(y_1,y_2+1):
            Anys.append(i)
            index_any = i-math.floor(Dades2[0][0])
            if index_any >=len(Dades2[0])-1:
                print('He de buscar les dades per internet i em canvia tot')
                final_day = dia_any(31, 12, year_extra-1)- dia_any(1, 6, year_extra-1) + dia_mig_extra#Estic agafant el numero total de dies fent: dies totals de l'any que estic mirant - dies que portem fins al 1 de juny+ dies que te per la mesura de l'any seguent
                X = [1,final_day]
                Y = [Dades2[1][index_any]*60, Dades2[1][-1]*60]
            else:
                if (((int(i)-1) % 4 == 0 and (int(i)-1) % 100 != 0) or ((int(i)-1) % 100 == 0 and (int(i)-1) % 400 == 0)):
                    X = [1,366]
                else:
                    X = [1,365]
                Y = [Dades2[1][index_any]*60,Dades2[1][index_any+1]*60]
            coefficients = np.polyfit(X, Y, degree)
            Coef_list.append(coefficients)
    elif (((m_1 == 6 and d_1 >= 15) or m_1 > 6) and (m_2 < 6 or (m_2 == 6 and d_2 < 15))):
        for i in range(y_1,y_2):
            Anys.append(i)
            index_any = i-math.floor(Dades2[0][0])
            if (((int(i)-1) % 4 == 0 and (int(i)-1) % 100 != 0) or ((int(i)-1) % 100 == 0 and (int(i)-1) % 400 == 0)):
                X = [1,366]
            else:
                X = [1,365]
            index_any = i-y_1
            Y = [Dades2[1][index_any]*60,Dades2[1][index_any+1]*60]
            coefficients = np.polyfit(X, Y, degree)
            Coef_list.append(coefficients)
    elif ((m_1 < 6 or (m_1 == 6 and d_1 < 15)) and (m_2 < 6 or (m_2 == 6 and d_2 < 15))):
        for i in range(y_1-1,y_2):
            Anys.append(i)
            index_any = i-math.floor(Dades2[0][0])
            if (((int(i)-1) % 4 == 0 and (int(i)-1) % 100 != 0) or ((int(i)-1) % 100 == 0 and (int(i)-1) % 400 == 0)):
                X = [1,366]
            else:
                X = [1,365]
            index_any = i-y_1
            Y = [Dades2[1][index_any]*60,Dades2[1][index_any+1]*60]
            coefficients = np.polyfit(X, Y, degree)
            Coef_list.append(coefficients)
    return Coef_list, Anys
# *****************************************************************************
#Funcio que em mira quins coeficients he d'agafar de la llista de la funcio anterior
def index_coef(y, m, d, Anys):#Aquesta funcio s'hauria de generalitzar una mica mes fent una dependencia de quins mesos comencen i tal
    for i in range(len(Anys)):
        if y == Anys[i]:
            if ((m == 6 and d >= 15) or (m > 6)):#Aqui imposo que em torni un index o un altre depenent de l'any que estic mirant i del mes
            #Si estic mirant l'any 2020 i estic en el mes 4 l'index sera el mateix que el del 2020 per com s'ha creat la llista de coeficient
            #En canvi si estic en el mes 6 ja es un index major al del 2020
                dia_inicial = date(Anys[i], 6, 1)
                dia_final = date(Anys[i], m, d)
                #Alhora li demano que em torni la diferencia de dies entre l'1 de juny de l'any que toqui
                #Per ferho servir a la regressio
                dif = (dia_final-dia_inicial).days
                index = i#He d'afegir una constant per tenir en compte si comenca abans o despres del mes 6
                #Aqui no va un +1 ja que la idea es que si tinc que el primer any comenca pel mes 7 l'element sigui el 0
            elif ((m < 6) or (m == 6 and d < 15)):
                dia_inicial = date(Anys[i]-1, 6, 1)
                dia_final = date(Anys[i], m, d)
                dif = (dia_final-dia_inicial).days
                index = i-1
            break
    return index, dif
# *****************************************************************************
#Aquesta funcio em torna una regressio entre les 00:30 i les 23:30 per eliminar
#les contribucions magnetosferiques que apareixen. NOTA IMPORTANT:
#Aquesta funcio s'ha creat ja que s'ha vist que les dades no queden centrades en el 0
#Pero si es veigues que a primeres( amb aixo em refereico sense aquesta funcio pel mig)
#queden centrades en el 0 aixo vol dir que la contribuacio d'aquesta funcio hauria de ser despreciable
```

```
def regressio_magnetosfera(dades_00, dades_23):
    X = [0, 23]
    Y = [dades_00, dades_23]
    degree = 1

    coefficients = np.polyfit(X, Y, degree)
    return coefficients

# ********************************************************************************

#Aquesta funcio simplement em dona una dada que necessito de la secular per fer una mitja que no tinc en la llista que va facilitar curto
def dada_extra_secular():
    url = ’http://www.obsebre.es/php/geomagnetisme/qdhorta/2023/ebr2023qmon.mon’


    # hdr required to access the files
    hdr = {
        ’User-Agent’: ’Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36’,
        ’Accept’: ’text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8’ }

    req = urllib.request.Request(url, headers=hdr)
    file = urllib.request.urlopen(req)
    data = pd.read_csv(file, skiprows=26, delimiter=’\\s+’)
    EBRD = []
    for j in range(len(data[’EBRX’])):
        EBRD.append(np.rad2deg(np.arctan(data[’EBRY’][j]/data[’EBRX’][j])))
    ebrd = np.mean(EBRD)

    #Condicio per tenir en compte si l’any es de traspas o no
    year = datetime.strptime(data[’DATE’][0], ’%Y-%m-%d’).year
    if (((int(year)-1) % 4 == 0 and (int(year)-1) % 100 != 0) or ((int(year)-1) % 100 == 0 and (int(year)-1) % 400 == 0)):
        c = 0
    else:
        c = 1
    dia_mig = np.ceil((data[’DOY’][len(data[’DOY’])-1] + c))/2
    return ebrd, dia_mig, year

# ********************************************************************************

#Aquesta funcio esta pensada per comprovar els dies que no acabao de veure bons, ja que mirare els magnetogrames i comprovare si hi ha algun any que es salvi o no, per despres cridar una altre funcio que eliminara els dies dolents
def selector_anys(Dies_Bons_2):
    #No li demano per pantalla els dies que vull mirar ja que son bastants i sera mes facil fer-ho a ma
    #Dies_mirar = [[’10’, ’01’], [’17’, ’01’], [’18’,’01’], [’21’ ,’01’], [’23’, ’01’], [’24’, ’01’], [’28’, ’01’], [’31’,’01’], [’02’, ’02’],[’08’, ’02’], [’09’, ’02’], [’10’, ’02’], [’14’, ’02’], [’15’, ’02’], [’18’, ’02’], [’02’, ’03’], [’12’, ’03’], [’25’, ’12’]]
    Dies_mirar = [[’07’, ’01’], [’08’, ’01’], [’09’, ’01’], [’10’, ’01’], [’12’, ’01’], [’13’,’01’], [’15’ ,’01’],  [’04’, ’02’],[’07’, ’02’], [’
    08’, ’02’], [’09’, ’02’], [’10’, ’02’], [’12’, ’02’], [’14’, ’02’], [’
    15’, ’02’], [’16’, ’02’], [’18’, ’02’], [’25’, ’12’]]

    #He creat una llista de llistes en la que cada subllista consta del dia
     i el mes(en aquest ordre)
    Anys_comprovar = []
    for j in range(len(Dies_mirar)):
        Anys_comprovar.append([Dies_mirar[j][0], Dies_mirar[j][1]])
        for k in range(len(Dies_Bons_2)):
            if Dies_mirar[j][0] == Dies_Bons_2[k][0] and Dies_mirar[j][1]
    == Dies_Bons_2[k][1]:
                Anys_comprovar[j].append(Dies_Bons_2[k][2])
    return Anys_comprovar

#
    ****************************************************************************

#Aquesta funcio em retorna un contador que val 1 si el dia que miro
    coincideix amb un de la llista i amb aquest contador passare al dia
    seguent en la funcio ’mare’
#I cal aclarir que els dies dolents s’han vist i imposat a pic i pala, ja
    que els grafics es veien com si no fossin calmats tot i que ho eren
def eliminador_dies(d , m, y):
    Llista_dies_dolents = [[’07’, ’01’, ’1991’],  [’07’, ’01’, ’2022’], [’
    08’, ’01’, ’1984’], [’08’, ’01’, ’2007’], [’08’, ’01’, ’2021’],  [’09’,
     ’01’, ’1980’], [’09’, ’01’, ’1994’], [’09’, ’01’, ’2006’], [’09’, ’01’
    , ’2010’], [’09’, ’01’, ’2021’], [’10’, ’01’, ’1980’], [’10’, ’01’, ’
    1994’], [’10’, ’01’, ’2006’], [’10’, ’01’, ’2013’], [’10’, ’01’, ’2021’
    ], [’12’, ’01’, ’2009’],  [’12’, ’01’, ’2018’],  [’12’, ’01’, ’2019’],
    [’13’, ’01’, ’2009’],  [’13’, ’01’, ’1982’], [’13’, ’01’, ’1986’],  [’
    13’, ’01’, ’2007’], [’13’, ’01’, ’2019’], [’13’, ’01’, ’2021’], [’13’,
    ’01’, ’2022’], [’15’, ’01’, ’1998’],  [’17’, ’01’, ’2009’], [’17’, ’01’
    , ’2010’], [’17’, ’01’, ’2016’], [’17’, ’01’, ’2018’], [’17’, ’01’, ’
    2020’], [’17’, ’01’, ’2021’], [’18’,’01’, ’2009’], [’18’,’01’, ’2010’],
     [’18’,’01’, ’2014’], [’18’,’01’, ’2018’], [’21’ ,’01’, ’2021’], [’23’,
     ’01’, ’2009’],[’23’, ’01’, ’2011’], [’24’, ’01’, ’2007’], [’24’, ’01’,
     ’2009’], [’28’, ’01’, ’2009’], [’31’,’01’, ’2006’],  [’31’,’01’, ’2013
    ’], [’31’,’01’, ’2014’], [’31’,’01’, ’2021’], [’02’, ’02’, ’2009’],  [’
    04’, ’02’, ’2001’],  [’04’, ’02’, ’2007’],  [’07’, ’02’, ’1998’], [’08’
    , ’02’, ’2009’], [’09’, ’02’, ’2006’], [’09’, ’02’, ’2010’], [’09’, ’02
    ’, ’2011’], [’10’, ’02’, ’1981’], [’10’, ’02’, ’2009’], [’10’, ’02’, ’
    2010’], [’10’, ’02’, ’2021’], [’12’, ’02’, ’1980’], [’14’, ’02’, ’1981’
    ], [’14’, ’02’, ’2006’], [’14’, ’02’, ’2017’],[’14’, ’02’, ’2018’],  [’
    15’, ’02’, ’2017’], [’16’, ’02’, ’1992’], [’16’, ’02’, ’1998’], [’18’,
    ’02’, ’2000’],  [’18’, ’02’, ’2001’], [’18’, ’02’, ’2021’], [’02’, ’03’
    , ’2006’],  [’02’, ’03’, ’2009’], [’12’, ’03’, ’2018’], [’25’, ’12’, ’
    1997’], [’25’, ’12’, ’2007’], [’25’, ’12’, ’2012’]]
    for j in range(len(Llista_dies_dolents)):
        if Llista_dies_dolents[j][0] == d and Llista_dies_dolents[j][1] ==
    m and Llista_dies_dolents[j][2] == y:
            contador = 1
            break
        else:
            contador = 0
    return contador

#
    ****************************************************************************

#Com el nom indica, aquesta funcio nomes es per plotejar els dies extres
```

```python
def plot_dies_extres(As, Freqs, Phis, Dies_seleccionats, Coefficients, Anys, Dies_mal_min):
    def fun(x, A, w, phi):
        return A*np.sin(w*x+phi)
    j = 0
    while True:
        EBRD = []
        doy = dia_any(Dies_seleccionats[j][0], Dies_seleccionats[j][1], Dies_seleccionats[j][2])
        in_coef, dif_dies = index_coef(int(Dies_seleccionats[j][2]), int(Dies_seleccionats[j][1]), int(Dies_seleccionats[j][0]), Anys)
        model = np.poly1d(Coeficients[in_coef])
        secular = model(dif_dies)

        dades, file_type, day  = read_file(Dies_seleccionats[j][0], Dies_seleccionats[j][1], Dies_seleccionats[j][2])
        for k in Dies_mal_min:
            #Aqui comprovo si he de vigilar amb les hores de sortida i posta de Sol
            if int(Dies_seleccionats[j][0]) == int(k[0]) and int(Dies_seleccionats[j][1]) == int(k[1]):
                    sunrise = k[-2]
                    sunset = k[-1]
                    contador_srss = 1#Em poso aquest contador per fer un if fora del bucle
                    break
            else:
                contador_srss = 0
        if contador_srss == 1:
            X = []
            Hores = []
            for l in range(sunrise, sunset+1):
                Hores.append(str(l)+':'+str(30))
                X.append(l)
            dades = data_treatment(dades,file_type)
            dades,start_time, contador_random = filter_data(dades, file_type, day, sunrise , sunset, secular)
        else:
            times = notable_times2(Dies_seleccionats[j][2], Dies_seleccionats[j][1], Dies_seleccionats[j][0])
            dades = data_treatment(dades,file_type)
            dades,start_time, contador_random = filter_data(dades, file_type, day, times[0] , times[2], secular)

            X = []
            Hores = []
            for l in range(times[0], times[2]+1):
                Hores.append(str(l)+':'+str(30))
                X.append(l)
        for p in range(len(dades['EBRD'])):
            EBRD.append(dades['EBRD'][p])
        Fit = []
        for n in range(len(X)):
            Fit.append(fun(X[n], As[doy], Freqs[doy], Phis[doy]))
        dif = []
        for m in range(len(Fit)):
            dif.append(EBRD[m]-Fit[m])
        plt.plot(Hores, Fit, color = 'r')
        plt.plot(Hores, EBRD)
        plt.plot(Hores, dif, color = 'g')
        plt.xticks(rotation=45)
        plt.xlabel(’Time(Hours)’)
        plt.ylabel(’D(min)’)
        plt.title(’Dia’+ Dies_seleccionats[j][0]+’Mes’+Dies_seleccionats[j][1]+ ’Any’+Dies_seleccionats[j][2])
        plt.show()
    return Dies_seleccionats


#****************************************************************************
#Aquesta funcio el que fara sera demanar quins indexos k vull mirar i em seleccionara els dies que tinguin aquests indexos
def Classificacio_dies_kps(Dades1):
    #Demano per pantalla quins son els indexos que es vol mirar
    #k_inf =int( input(’What will be the inferior limit of K_P?(number between 0 to 9)’))
    #k_sup = int(input(’What will be the inferior limit of K_P?(number between 0 to 9 and bigger than the previous  one)’))
    Dies_bons = [[], [], [], []]
    ks = [[2,3],[4,5], [6,7], [8,9]]#Creo una llista amb tots els indexos
    kps = [[], [], [], []]
    for j in range(len(ks)):#Aqui creo els diferents kps que tinc amb la notacio que fa servir l’excel
        for k in range(len(ks[j])):
            kps[j].append(str(ks[j][k])+’+’)
            kps[j].append(str(ks[j][k])+’o’)
            kps[j].append(str(ks[j][k])+’-’)
            kps[j].append(-1*ks[j][k])
    n = 0
    #En aquesta part simplement vaig comprovant els criteris que hem donat per
    #dir quin dia pertany a quin index. La idea es que perque pertanyi al rang
    #8-9 de kp necessita 2 valors entre 8 i 9, pel rang 6-7 en necessita 3
    #valors dins d’aquest rang, pel 4-5 en necessita 4 i pel 2-3 en necessita 5
    while n<len(Dades1[0]):
        Dades=Dades1[:,n:n+8]
        contador_2 = 2
        for j in range(len(kps)):
            contador_1 = 0
            for l in range(len(kps[len(kps)-1-j])):
                for i in Dades[3]:
                    if i == kps[len(kps)-1-j][l]:
                        contador_1 += 1
            if contador_1 >= contador_2:
                contador_3 = 1
                break#Aquets break es important ja que la funcio ha de parar un
                #cop ha triat un index, i se li dona mes importancia a l’index
                #que sigui mes gran
            else:
                contador_3 = 0
            contador_2 += 1
        if contador_3 == 1:
            Dies_bons[len(Dies_bons)-1-contador_2+2].append(Dades[0][0])
        n += 8#Vaig augmentat de 8 en 8 per avancar tot el dia, ja que cada
```

```
 dia nomes s'agafen 8 mesures
    Dies_Bons_2 = [[], [], [], []]#Canvio el format a un que em sigui mes util
    for i in range(len(Dies_bons)):
        for j in range(len(Dies_bons[i])):
            dia = pd.Timestamp(Dies_bons[i][j])
            Dies_Bons_2[i].append([str(dia.day).zfill(2), str(dia.month).zfill(2), str(dia.year).zfill(2)])
    return Dies_Bons_2

# *****************************************************************************
#Aqui simplement filtro les dades per saber si son bones o no com ja s'ha fet
#anteriorment
def filtre_dies_seleccionats(Dies_seleccionats, Coeficients, Anys, Dies_mal_min):
    def fun(x, A, w, phi):
        return A*np.sin(w*x+phi)
    Dies_seleccionats_filtrats = []
    for i in range(len(Dies_seleccionats)):
        Dies_seleccionats2 = Dies_seleccionats[i]
        j = 0
        while True:
            in_coef, dif_dies = index_coef(int(Dies_seleccionats2[j][2]), int(Dies_seleccionats2[j][1]), int(Dies_seleccionats2[j][0]), Anys)
            model = np.poly1d(Coeficients[in_coef])
            secular = model(dif_dies)

            dades, file_type, day  = read_file(Dies_seleccionats2[j][0], Dies_seleccionats2[j][1], Dies_seleccionats2[j][2])
            for k in Dies_mal_min:
                if int(Dies_seleccionats2[j][0]) == int(k[0]) and int(Dies_seleccionats2[j][1]) == int(k[1]):
                        sunrise = k[-2]
                        sunset = k[-1]
                        contador_srss = 1#Em poso aquest contador per fer un if fora del bucle
                        break
                else:
                    contador_srss = 0
            if contador_srss == 1:
                dades = data_treatment(dades,file_type)
                dades,start_time, contador_random = filter_data(dades, file_type, day, sunrise , sunset, secular)
            else:
                times = notable_times2(Dies_seleccionats2[j][2], Dies_seleccionats2[j][1], Dies_seleccionats2[j][0])
                dades = data_treatment(dades,file_type)
                dades,start_time, contador_random = filter_data(dades, file_type, day, times[0] , times[2], secular)

            if j == 0:
                if contador_random > 0 and j != (len(Dies_seleccionats2)-1):
                    print(j, len(Dies_seleccionats2))
                    Dies_seleccionats2.pop(j)
                    continue
                elif contador_random > 0 and j == (len(Dies_seleccionats2)
    -1):
                    print('El bucle ha acabat')
                    print(j, len(Dies_seleccionats2))
                    Dies_seleccionats2.pop(j)
                    break
                j += 1
            elif j >= 1:
                if contador_random > 0 and j != (len(Dies_seleccionats2)-1)
    :
                    print(j, len(Dies_seleccionats2))
                    Dies_seleccionats2.pop(j)
                    continue
                elif contador_random > 0 and j == (len(Dies_seleccionats2)
    -1):
                    print('El bucle ha acabat')
                    print(j, len(Dies_seleccionats2))
                    Dies_seleccionats2.pop(j)
                    break
                j += 1
            if j == (len(Dies_seleccionats2)):#L'hi he tret el -1 ja que
    crec que em pot donar problemes per l'ultim dia
                print('El bucle ha acabat')
                break
        Dies_seleccionats_filtrats.append(Dies_seleccionats2)
    return Dies_seleccionats_filtrats

#
    ****************************************************************************

#Aqui afegeixo els valors dels dies ja filtarts a una llista
# per poder calcular les mitjes
def Mitjes_anuals( Dies_seleccionats, Coefficients, Anys, Dies_mal_min, As,
     Freqs, Phis):
    shape = (4, 24)  # (dimension 0, dimension 1)

    # Generar la matriz tridimensional vacia
    Mitja = [[[] for _ in range(shape[1])] for _ in range(shape[0])]
    Mitja.append([])
    for j in range(24):
        Mitja[-1].append(j)
    for i in range(len(Dies_seleccionats)):
        Dies_seleccionats2 = Dies_seleccionats[i]
        j = 0
        while True:
            in_coef, dif_dies = index_coef(int(Dies_seleccionats2[j][2]),
    int(Dies_seleccionats2[j][1]), int(Dies_seleccionats2[j][0]), Anys)
            model = np.poly1d(Coeficients[in_coef])
            secular = model(dif_dies)
            doy = dia_any(Dies_seleccionats2[j][0], Dies_seleccionats2[j
    ][1], Dies_seleccionats2[j][2])
            dades, file_type, day  = read_file(Dies_seleccionats2[j][0],
    Dies_seleccionats2[j][1], Dies_seleccionats2[j][2])
            for k in Dies_mal_min:
                if int(Dies_seleccionats2[j][0]) == int(k[0]) and int(
    Dies_seleccionats2[j][1]) == int(k[1]):
                        sunrise = k[-2]
                        sunset = k[-1]
                        contador_srss = 1#Em poso aquest contador per fer
    un if fora del bucle
                        break
                else:
                    contador_srss = 0
            if contador_srss == 1:
                dades = data_treatment(dades,file_type)
                dades = filter_data_2(dades, file_type, day, secular)
                for l in range(sunrise, sunset+1):
                    Mitja[i][l].append(dades['EBRD'][l]-fun(l, As[doy-1], Freqs[doy-1], Phis[doy-1]))

            else:
                times = notable_times2(Dies_seleccionats2[j][2], Dies_seleccionats2[j][1], Dies_seleccionats2[j][0])
                dades = data_treatment(dades,file_type)
                dades = filter_data_2(dades, file_type, day, secular)
                for l in range(times[0], times[2]+1):
                    Mitja[i][l].append(dades['EBRD'][l]-fun(l, As[doy-1], Freqs[doy-1], Phis[doy-1]))

            j += 1
            if j == (len(Dies_seleccionats2)):#L'hi he tret el -1 ja que crec que em pot donar problemes per l'ultim dia
                print('El bucle ha acabat')
                break

    return Mitja

# *****************************************************************************

Dades1 = pd.read_excel("Dades dels indexos 40 anys.xlsx")
Dades1 = np.transpose(np.array(Dades1))

Dades3 = pd.read_excel("Dades dels indexos 20 anys antics.xlsx")
Dades3 = np.transpose(np.array(Dades3))

Dades2 = pd.read_excel("VSEC_AN.xlsx")
Dades2 = np.transpose(np.array(Dades2))
Dades2 = Dades2.tolist()
ebrd, dia_mig_extra, year_extra = dada_extra_secular()

for j in range(len(Dades2)):
    if j == 0:
        Dades2[0].append(int(year_extra))
    elif j == 1:
        Dades2[1].append(ebrd)
    else:
        Dades2[j].append(0)

Dies_Bons_2 = Classificacio_dies(Dades1)

Anys_comprovar = selector_anys(Dies_Bons_2)

y_1 = int(Dies_Bons_2[0][2])
m_1 = int(Dies_Bons_2[0][1])
d_1 = int(Dies_Bons_2[0][0])
y_2 = int(Dies_Bons_2[-1][2])
m_2 = int(Dies_Bons_2[-1][1])
d_2 = int(Dies_Bons_2[-1][0])

```

```
Coeficients, Anys = Coeff_secu(y_1, m_1, d_1, y_2, m_2, d_2, Dades2, dia_mig_extra, year_extra)

Mitja_dies_l, Dies_bons_filtrats, Secular, Dies_pendent_gran = Llistat_dades_dies(Dies_Bons_2, Dades2, Coeficients, Anys)

Dies_comprovar, numero_dies = comprovacio_num_dies(Mitja_dies_l, Dies_bons_filtrats)
Mitja_dies_l = arreglar_minuts(Dies_comprovar, Mitja_dies_l)
Dies_mal_min = dies_min_dolent()
#Aqui simplement faig la mitja
Mitja_dies = np.zeros((367,24))
for i in range(24):
    Mitja_dies[366][i] = i

for k in range(366):
    for l in range(len(Mitja_dies_l[0])):
        if len(Mitja_dies_l[k][l]) == 0:
            Mitja_dies[k][l] = 0
        else:
            Mitja_dies[k][l] = np.mean(Mitja_dies_l[k][l])

Mitja_dies, Dies_buits = empty_days(Mitja_dies)

#Aqui estic agafant les regressions i nomes les dades que no son 0
Primers_ajustos, As, Freqs, Phis = funct_curvefit_dies(Mitja_dies, Dies_buits, Dies_mal_min)

Dies_graficar = [['01', '01', '2001'], ['03', '02', '2001'], ['15', '03', '2001'], ['27', '04', '2001'], ['21', '05', '2001'], ['23', '06', '2001'], ['28', '07', '2001'], ['16', '08', '2001'], ['10', '09', '2001'], ['24', '10', '2001'], ['03', '11', '2001'], ['18', '12', '2002']]

fig = plt.figure(figsize=(30,30))
X = []
Xs = []
Ys = []
a = 1
for j in range(24):
    X.append(str(j)+':30')
for j in range(len(Dies_graficar)):
    in_coef, dif_dies = index_coef(int(Dies_graficar[j][2]), int(Dies_graficar[j][1]), int(Dies_graficar[j][0]), Anys)
    model = np.poly1d(Coeficients[in_coef])
    secular = model(dif_dies)

    doy = dia_any(Dies_graficar[j][0], Dies_graficar[j][1], Dies_graficar[j][2])

    dades, file_type, day  = read_file(Dies_graficar[j][0], Dies_graficar[j][1], Dies_graficar[j][2])
    dades = data_treatment(dades,file_type)
    dades = filter_data_2(dades, file_type, day, secular)
    for l in Dies_mal_min:
        if l[0] == '01' and l[1] == Dies_graficar[j][1]:
            sunrise = l[2]
            sunset = l[3]
            break
        else:
            sunrise, noon, sunset = notable_times2(Dies_graficar[j][2],
    Dies_graficar[j][1], Dies_graficar[j][0])
    plot = []
    plot_1 = []
    plot_2 = []
    for l in range(sunrise, sunset+1):
        plot.append(-fun(l, As[doy],Freqs[doy], Phis[doy])+dades['EBRD'][l])
    for l in range(sunrise+1):
        plot_1.append(-fun(l, As[doy],Freqs[doy], Phis[doy])+dades['EBRD'][l])
    for l in range(sunset, 24):
        plot_2.append(-fun(l, As[doy],Freqs[doy], Phis[doy])+dades['EBRD'][l])
    ax = fig.add_subplot(12,1,j+1)
    if j < 11:
        ax.xaxis.set_ticks([])
    ax.plot(X[0:sunrise+1],plot_1, color = 'b')
    ax.plot(X[sunrise:sunset+1], plot, color = 'r')
    ax.plot(X[sunset: 24], plot_2, color = 'b')
    ax.axvline(sunrise, c = 'r', label = 'sunrise')
    ax.axvline(sunset, c = 'b', label = 'sunset')
    ax.axvline(11.5, c = 'k', label = 'noon')
    Ys.append(plot)
    Xs.append(X[sunrise: sunset+1])
plt.rcParams["figure.figsize"] = (7,7)
plt.rc('xtick', labelsize=22)
plt.rc('ytick', labelsize=22)
plt.rcParams['xtick.top'] = False
plt.rcParams['ytick.right'] = False
plt.rcParams['xtick.major.size'] = 20
plt.rcParams['ytick.major.size'] = 20
plt.rcParams['xtick.minor.size'] = 15
plt.rcParams['ytick.minor.size'] = 15
plt.savefig('Grafiques')
plt.show()


Primers_ajustos_net=[]
for j in range(366):
    llista_brut=[]
    dia, mes = day_to_date(j)

    for i in range(len(Dies_mal_min)):
        if dia == int(Dies_mal_min[i][0]) and mes == int(Dies_mal_min[i][1]):
            times = [Dies_mal_min[i][-2], 0, Dies_mal_min[i][-1]]
            break
        else:
            times = notable_times2(2000, mes, dia)
    for k in range(24):
        if times[2] >= k >= times[0]:
            llista_brut.append(Primers_ajustos[j][k])
    Primers_ajustos_net.append(llista_brut)

Dies_seleccionats = Classificacio_dies_kps(Dades1)
Dies_seleccionats = filtre_dies_seleccionats( Dies_seleccionats, Coeficients, Anys, Dies_mal_min)
Mitjes_seleccio_ll = Mitjes_anuals( Dies_seleccionats, Coeficients, Anys, Dies_mal_min, As, Freqs, Phis)

```

```python
Mitjes_seleccio = [[], [], [], []]
Mitjes_seleccio.append(Mitjes_seleccio_ll[-1])
Hores_sol = []
llegenda = ['2-3', '4-5', '6-7', '8-9']
for j in range(len(Mitjes_seleccio_ll)-1):
    for i in range(len(Mitjes_seleccio_ll[j])):
        if len(Mitjes_seleccio_ll[j][i]) != 0:
            Mitjes_seleccio[j].append(np.mean(Mitjes_seleccio_ll[j][i]))
        else:
            Mitjes_seleccio[j].append(0)
    ind = [k for k in range(len(Mitjes_seleccio[j])) if Mitjes_seleccio[j][k] != 0]
    Hores_sol.append([min(ind), max(ind)])
for j in range(len(Mitjes_seleccio)-1):
    plt.plot(Mitjes_seleccio[-1][Hores_sol[j][0]:Hores_sol[j][1]], Mitjes_seleccio[j][Hores_sol[j][0]:Hores_sol[j][1]])
plt.legend(llegenda, fontsize = 17)
plt.title('Anys 1980-2022', fontsize = 25)
plt.ylabel('Diff(min)', fontsize = 22)
plt.xlabel('Time of the day(hours)', fontsize = 22)
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
plt.rcParams["figure.figsize"] = (7,7)

plt.savefig('Resultats_finals_1980_2022.png')
plt.show()

Dies_grafic = []
Periode = []
for j in range(366):
    Dies_grafic.append(j)
    Periode.append(2*math.pi/Freqs[j])
plt.plot(Dies_grafic, As)
plt.rcParams["figure.figsize"] = (7,7)
plt.yticks(fontsize = 15)
plt.xticks(fontsize = 15)
plt.ylabel('A(min)', fontsize = 17)
plt.xlabel('Dies', fontsize = 17)
plt.savefig('Amplitude')
plt.show()

plt.plot(Dies_grafic, Periode)
plt.rcParams["figure.figsize"] = (7,7)
plt.ylabel('T(hores)', fontsize = 17)
plt.xlabel('Dies', fontsize = 17)
plt.yticks(fontsize = 15)
plt.xticks(fontsize = 15)
plt.savefig('Periode')
plt.show()

plt.plot(Dies_grafic, Phis)
plt.rcParams["figure.figsize"] = (7,7)
plt.ylabel('Fase',fontsize = 17 )
plt.xlabel('Dies', fontsize = 17)
plt.yticks(fontsize = 15)
plt.xticks(fontsize = 15)
plt.savefig('Fase')
plt.show()
```

Listing 2: Main code