Politecnico di Torino
Dipartimento di Elettronica e Telecomunicazioni

Master of Science in Communications and Computer Engineering

Satellite Navigation systems

Lab reports

Students:
Christian Lavezzari
Giuseppe Bruno

Academic year 2023/2024

Contents

1 Raw GNSS measurements in Android devices

Introduction on the Matlab script

The Matlab script is used to analyze the satellite information dataset, estimates the user's position using latitude and longitude. It displays various graphs that, when considered together, make the error associated with the estimation more clear.
Differents graphs are displayed but the main ones are:

- **Pseudoranges vs time**: shows the variation of the pseudoranges over time, from the first value (approximates to the real value) to the last one.
- **Pseudoranges changing from initial value**: shows the delta of the pseudoranges from the initial values (set to 0), to the values calculated at each iteration, for each of them.
- **Derivative of the pseudoranges over time**: shows the trend of the derivative of each pseudorange with respect to time
- **C/No in dB.Hz**: shows the power of the signals over the power generated by the noise, misured in dB-Hz
- **Weighted Least Squares solution**: shows all the solutions but in particular the median one, which represents the latitude and longitude of the device.
- **Horizontal Speed**: shows the trend over time of the different horizontal solutions obtained, considered in meters.
- **HDOP**: shows the trend of HDOP in relation to the quantity of visible satellites for that time instant. Its value tend to be inversely proportional to this second parameter.

Experiments

We considered two different scenarios, the first one in a **building area** (less visible satellites, more noise and multi path), the second regards an **open area** in which is supposed a better acquisition of the signals. For each scenario two different acquisitions are performed, the first regards a **stationary** user, while the second a **moving** one.

Measurements in a building area

The acquisitions are taken in **Via San Paolo on February 6, 2024, at 8:38 a.m.**, the real position of the device is **[45.062315 °, 7.649522 °]**, before getting the acquisition.

Static acquisition

It lasted **82** seconds and below are shown the results obtained using the Matlab script, considering **GPS**.

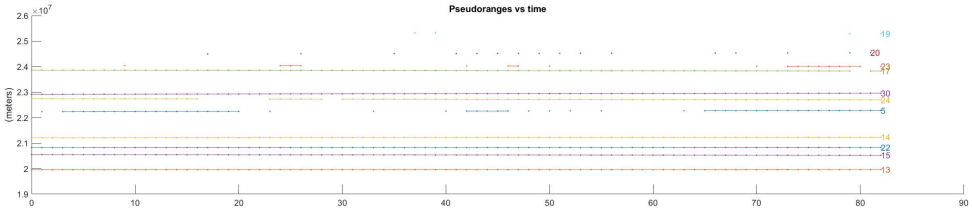


Figure 1: pseudoranges vs time

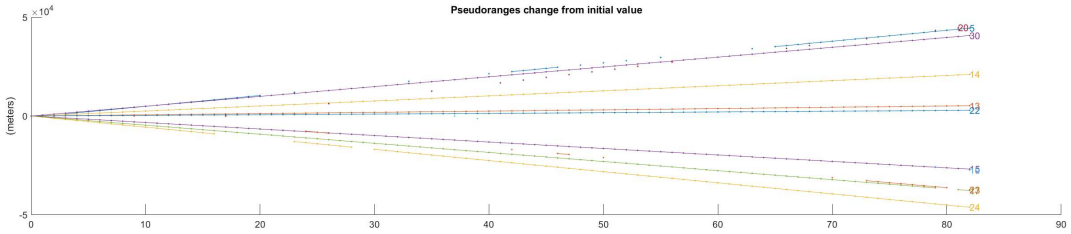


Figure 2: pseudoranges change from initial value

In figure 1 are showed **11** satellites which are ≈ **20.000 Km** apart from the device, so the pseudoranges are compatible with the GPS constellation. Each satellite is identified by its **SVID** (Space Vehicle Identifiers) displayed on the right side of the graphs. The worst signals comes from **19**, **20**, **23**, **5** but there are the **13**, **15**, **22**, **14**, **30**, **17** and **24** that are clearly visible, despite the location being surrounded by many buildings. It's important to notice that the satellites **less visible** are usually the ones moving **towards the horizon** (relative high positive slope in figure 2), moreover in case of building area the multi path of the signals and the noise are not negligible.
Stochastically speaking, there could be **destructive reflections/interference** or the **noise** might be such that it covers the entire signal.

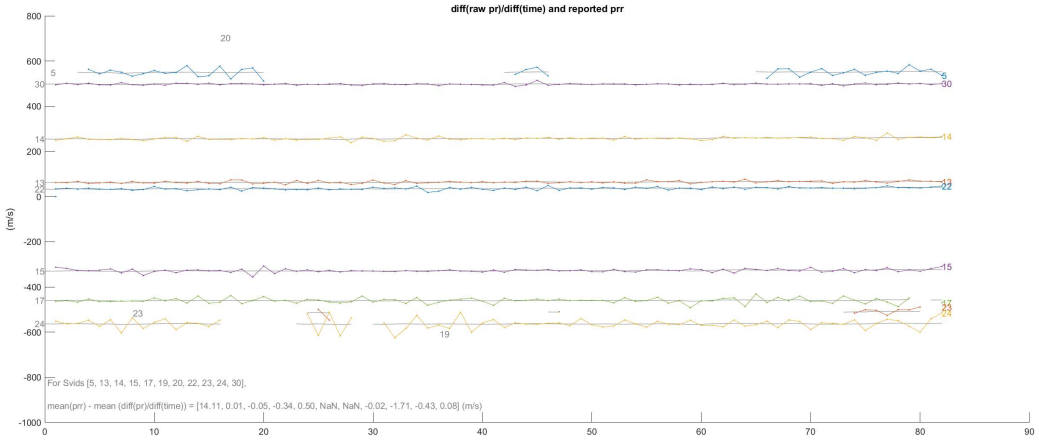


Figure 3: derivative of the pseudoranges

In figure 3 the derivative of the pseudoranges are relative stable, the satellites towards the **zenit** have a negative derivate unlike the others (a positive one). The values of the **24** and **5** are not stable, could be due to the presence of noise or other factors.
Observing the ratio between the power of the signal and the noise (figure 4), satellites **24** and **5** have an average $\frac{C}{N_0}$ that is under 30 dB.Hz (slight signal), while **20, 19, 23** even less. The lack of points does not means the absence of the signals: this is because the receiver is unable to capture it if the $\frac{C}{N_0}$ is lower than the minimum required by the smartphone. The satellites towards the zenit expect high values. In all the 82 seconds there are at least 4 visible satellites so a solution of the equation is always obtained. In presence of more than 4 satellites better solutions are obtained because a set of satellites with better **GDOP** is used.
The **HDOP** calculated (figure 5) shows that, despite the measurements are taken in an urban environment, the solution can be considered reliable. Infact the HDOP consistently remains at low values, and at times, it even drops below **1**.
At the end of the script (figure 6), the position is obtained as the median between all the solutions, with a range of **16.7** meters in the horizontal plane. It is possible to notice that the longitude and the latitude are close to the real ones. Moreover the distance between the two points is **13.2** meters. We can conclude that it is a good approximation considering a stationary user in an urban center.

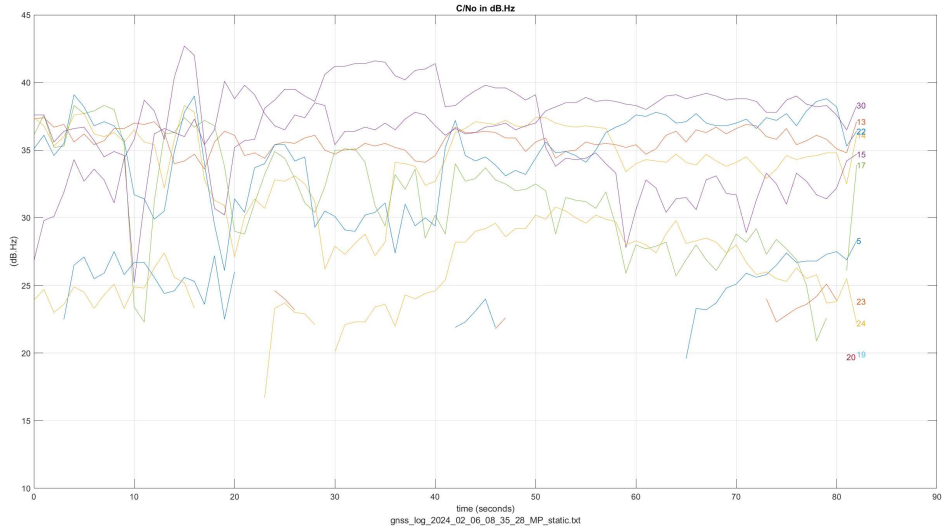


Figure 4: power over noise ($\frac{C}{N_0}$ for different time instants)

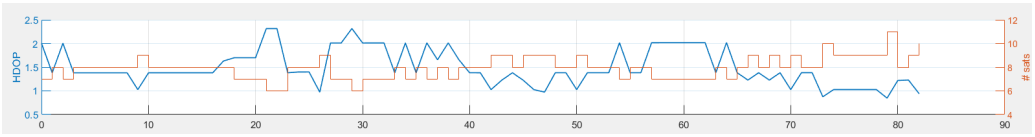


Figure 5: hdop

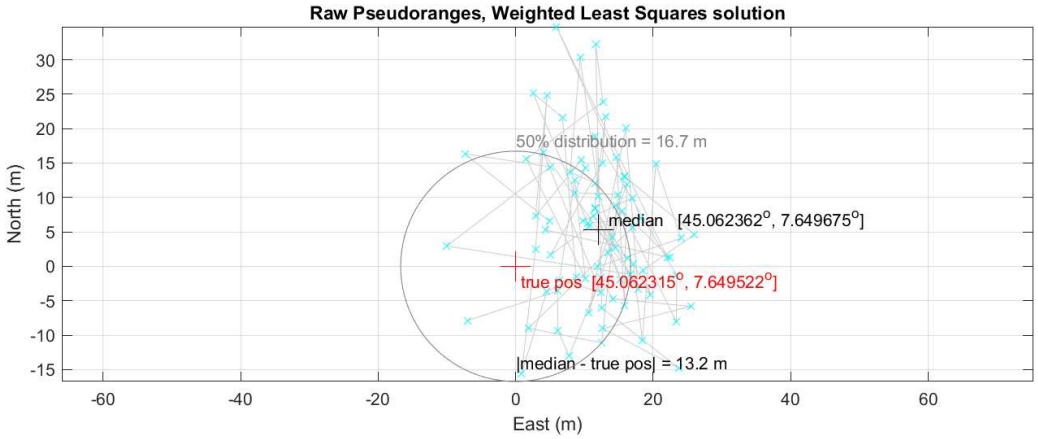


Figure 6: power over noise

Dynamic acquisition

The acquisition of the signals started after the end of the previously described one. It **started** at the same coordinates [**45.062315** °, **7.649522** °] and **ended** at [**45.065179**°, **7.654702**°], the test lasted **364** seconds. The satellites visible are the same **11** (represented by the same colors) but with some differences.

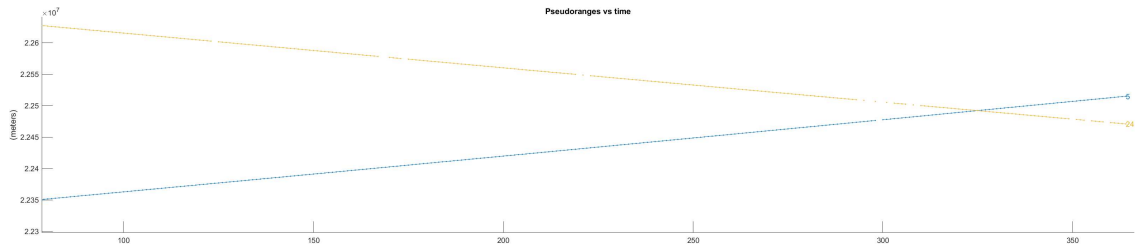


Figure 7: pseudoranges vs time in motion

There is a moment in which the satellites **24** and **5** have the same pseudorange (figure 7). During the motion, the signals captured are less stable due to the movement itself and also to the changing of the background (urban planning architecture), this behavior can be appreciated in the following graph.
The derivative of the pseudorange (figure 8), over the same time interval, exhibits a more general stochastic trend compared to the previous case, in addition satellites **5** and **23** are more visible.

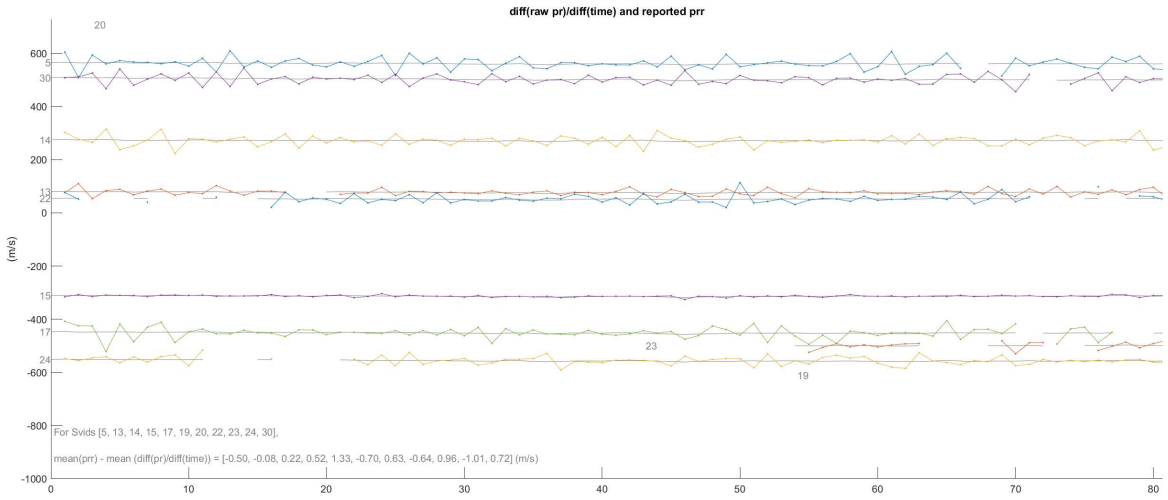


Figure 8: derivative of the pseudoranges in motion

In figure 9 is clearly visible the low power of the signal, only **30** remains around 40 dB.Hz while all the **others** have lower average values then the previous one (figure 4).

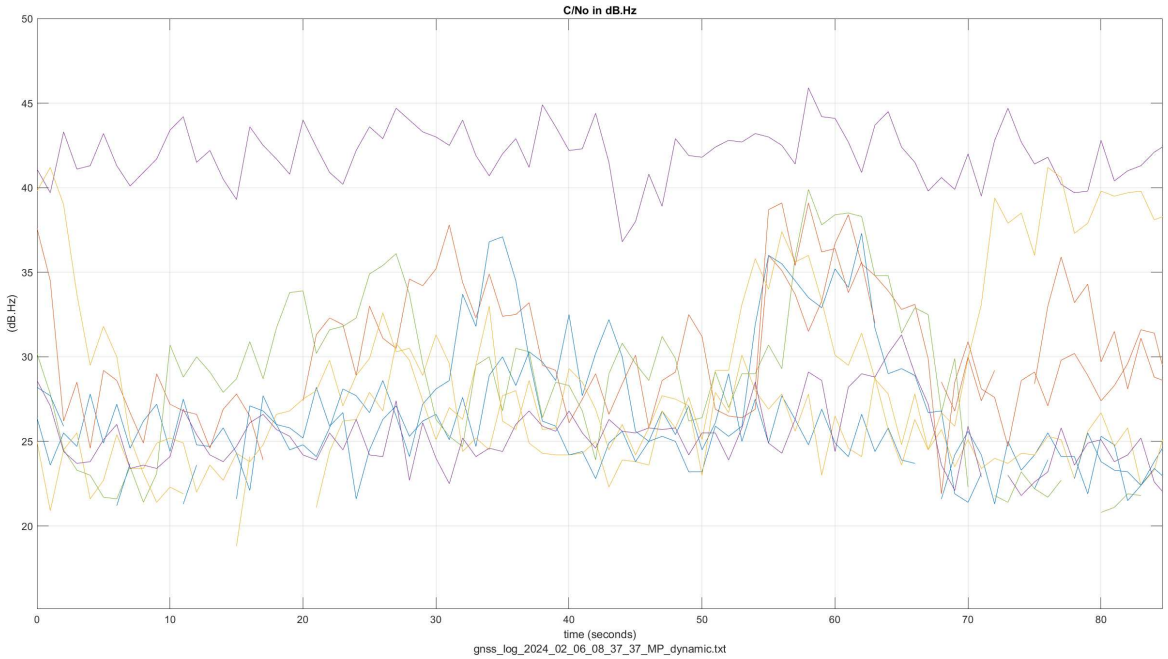


Figure 9: power over noise in motion

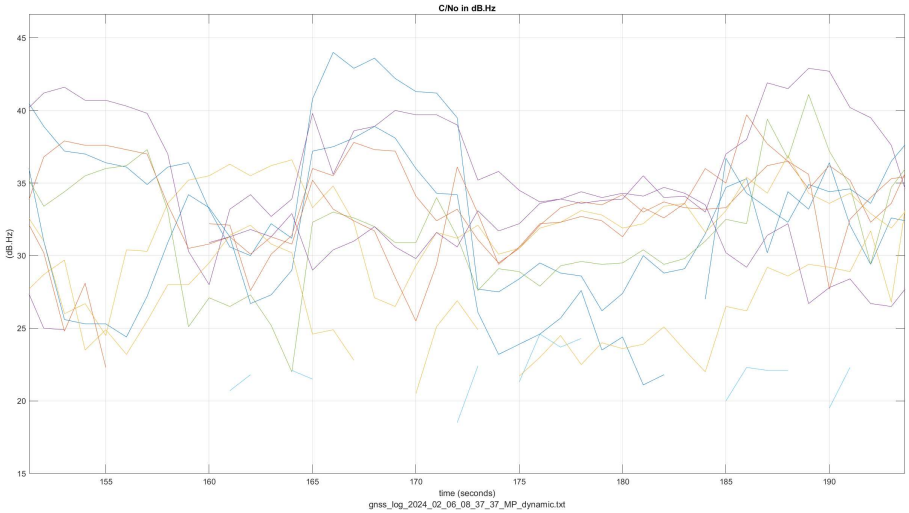


Figure 10: power over noise waiting for the semaphore

During the motion we had to wait for a traffic light to turn green, resulting in several moments of stationary state. This state improves the perfomance of the receiver infact the average of $\frac{C}{N_0}$ increases to $\approx$ **33** dB.Hz (figure 10), so we expect to reach more precision of the solution in that time interval.

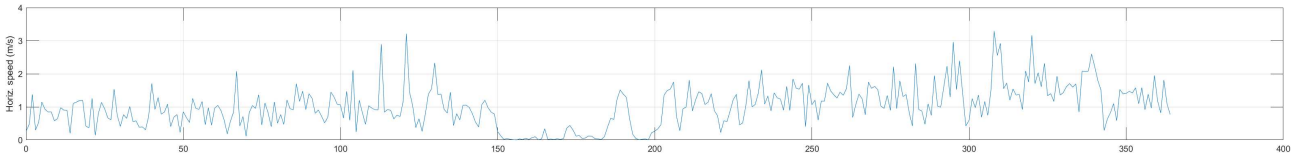


Figure 11: horizontal speed in motion

An index of low accuracy is the **horizontal speed of the solutions** (figure 11). The values have an **average** of **1** $\frac{m}{s}$ (which is compatible with an individual in motion) so the space of the solutions is broad.

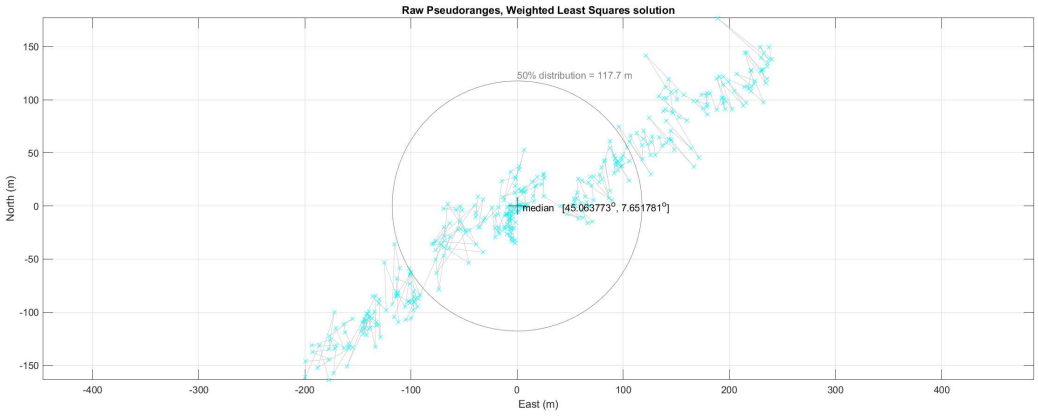


Figure 12: solution in motion

Figure 12 shows the space mentioned before and the position calculated over all this coordinates. As expected the solution tend to be in the middle of the space, oriented towards the region with the higher number of samples. Considering a urban center the solution are relative goods, the final position is just the result of the Matlab script (not intended for a dynamic environments).

Measurements in an open area

The measurements are taken at **Braccini Park on February 6, 2024, at 20:38**, the real position of the device is [**45.061072**°, **7.652208**°], before getting the acquisition. The **GPS** constellation is used, satellites have a **period of 12 hours**, so is expected to receive the signals from the **same ones**.

Static acquisition

The experiment lasted **116** seconds and the results are not completely as expected. Below the graphs obtained by Matlab.

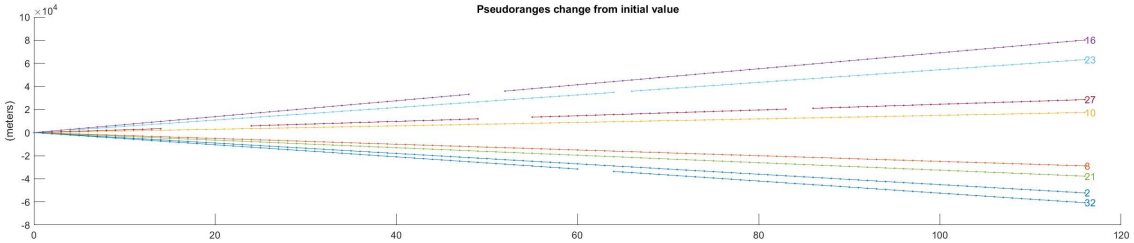


Figure 13: pseudoranges change from initial value in open area

In figure 13 are present not **11** satellites but **8**. In this case satellites **6**, **21**, **2** and **32** are towards the **zenit** while satellites **10**, **27**, **23** ad **16** are towards the **horizon**. All are more **visible** than the ones caputered in the morning activity but **different** SVID are registered (not as expected).

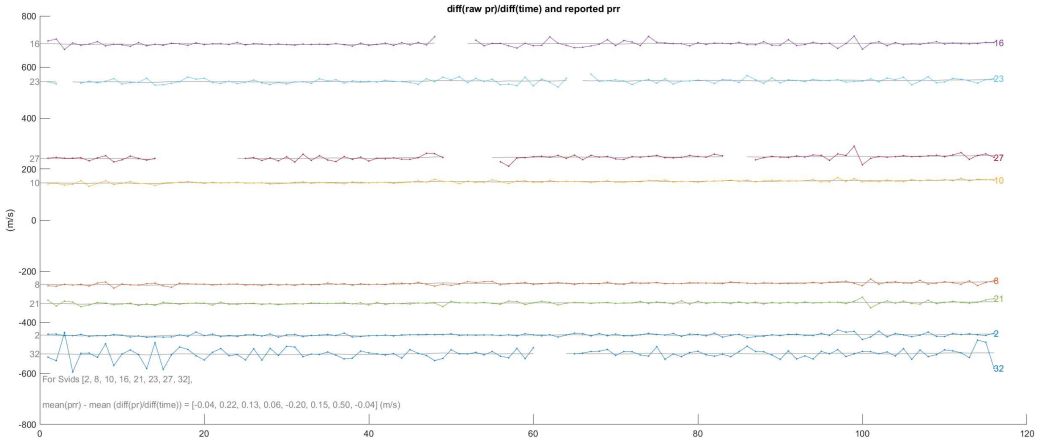


Figure 14: derivative of the pseudoranges in open area

The derivative of the pseudoranges are relative stable also in this scenario (figure 14). Despite the location, for some seconds the receiver lost the signals from satellites **16**, **23**, **27** and **32**. Particular attention need to be applyed to the last one, its pseudorange is probably affected to an error bigger than the others.
The **difference** between an open area and a closed one is clearly visible in graph 15. The overall **average** of $\frac{C}{N_0}$ is $\approx$ **37** dB.Hz so clearly a better performance compared to figure 4. Signals can be splitted in two different parts, the first one with values around 40 dB.Hz and the second one below that threshold. Seems not be a relevant connection between this ratio and the satellites towards the horizon. Satellite **32** has the lowest signal, concordant with the trend of its derivative.
As expected in an open area, more than 4 satellites are visible simultaneously for the most time so the **HDOP** has low values $\approx$ **1** (figure 16).

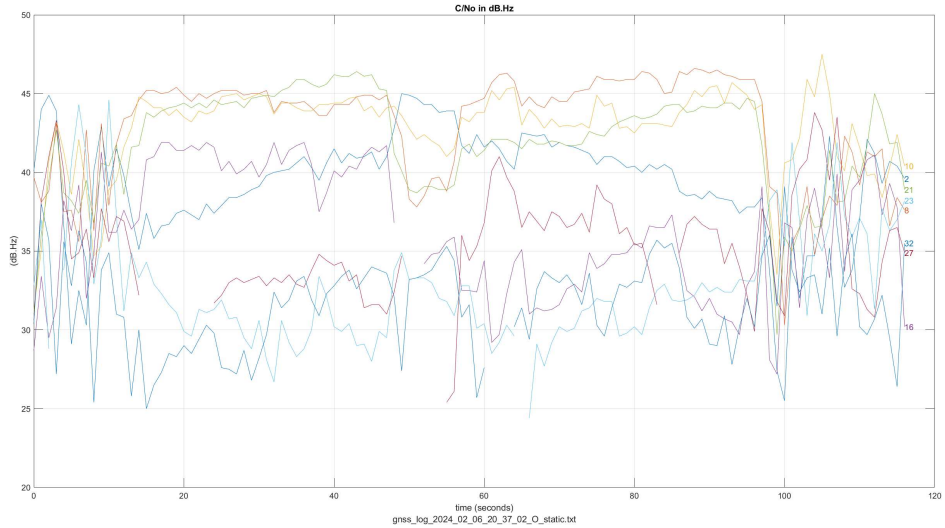


Figure 15: power over noise in open area

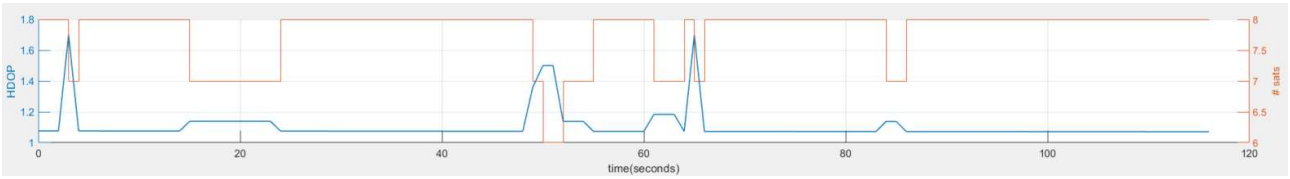


Figure 16: hdop open area

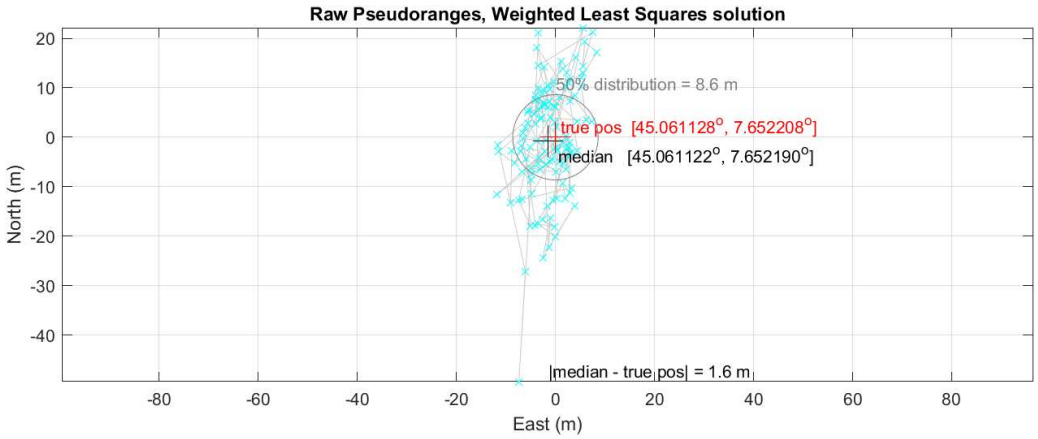


Figure 17: solution in open area

At the end of the script (figure 17), the position is obtained with a range of **8.6** meters in the horizontal plane. It is possible to notice that also in this case the longitude and the latitude are close to the real ones. The distance between the two points is **1.6** meters. We expect this better performance as can be seen comparing them with an urban environment, figure 6.

Dynamic acquisition

The test started after the end of the static one. A circular trajectory was followed, so the starting and ending coordinates are the same [**45.061072**°,**7.652208**°]. It lasted **195** seconds

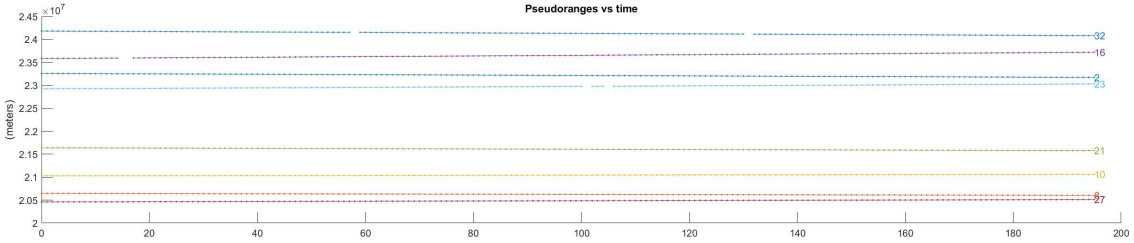


Figure 18: pseudoranges vs time in motion, open area

The **visible** satellites are the **same**, represented by the same colors.
In contrast to figure 14, the instances of signal loss are fewer. It is true that the park is an open space, however, there are trees and other structures present that can interfere with the signals (which are already weak in themselves). Statistically speaking, in different areas of the park the same signals can have different power.

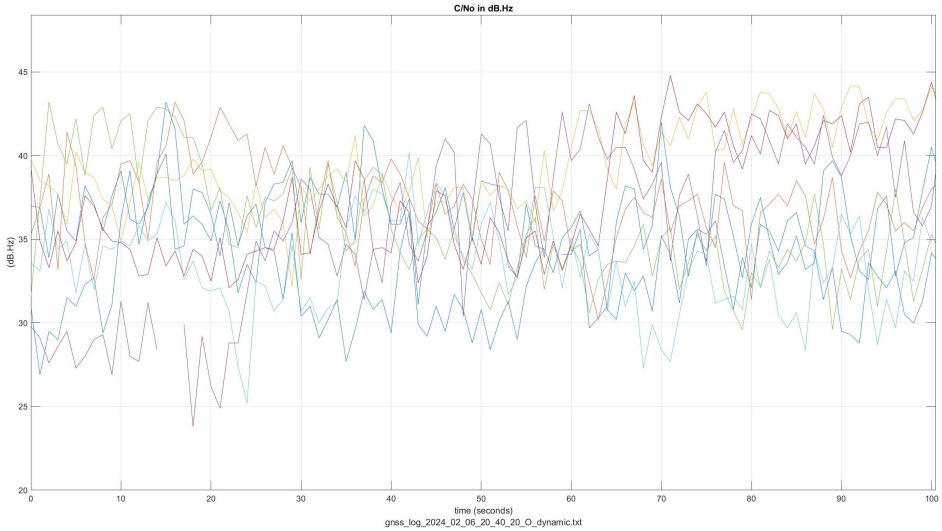


Figure 19: power over noise in motion, open area

Signals have a similar $\frac{C}{N_0}$ (figure 15, values are not too distant from themself. The behavior is different from the stationary case as can be seen in figure 19, they are not split in two part. Despite the receiver being in motion, here the average is $\approx$ **37** dB.Hz too. In addition, **32** (light blue) has a better ration than in stationary state, with a **peak** in **41** dB.Hz

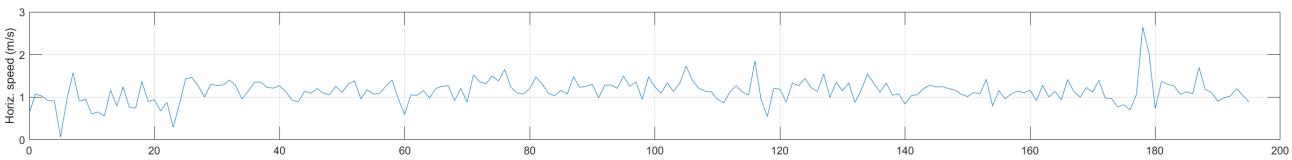


Figure 20: horizontal speed in motion, open area

The horizontal speed is always around **1** $\frac{m}{s}$, as is shown in figure 20. The number of peaks is lower then the case of a building area (figure 11): this is an index of cleaner signals. Following the trajectory we never stopped so there are **no** value around **0** $\frac{m}{s}$.

Figure 21 shows the latitude and longitude calculated. As expected the solution tend to be in the middle of the space. In this scenario the median of all the samples has no value but it's important to notice that the sky blue points apporoximate very well the followed trajectory. Observing the figures 20 and 15 is evident how performance improves in an open space compared to a closed space.

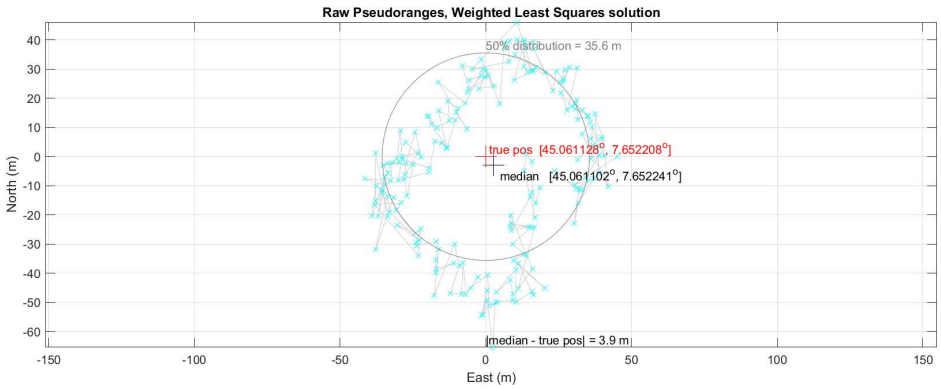


Figure 21: solution in motion, open area

2 State estimation

Introduction on state estimation

The aim of this second laboratory activity is state estimation: as we already know from the theory, the navigation provided by GNSS constellation is possible thanks to the set of informations (observables) that the final user extract from the broadcasted satellite navigation signals. As pointed out in the first experience, the observables contains, among the others, pseudorange and pseudorange rate, doppler shifts and so on. Since the observables depends on the actual state of the receiver, from them can be inferred the receiver state. The following sections present the same nomenclature as those of the laboratory slides to allow easier reading.

LMS - SPP - Single epoch

We know that, to obtain a PVT solution, we need to have at least 4 satellites in visibility. This is due to the fact that we need to solve a sistem of 4 equation in 4 unknown: the 3 user coordinates (x,y,z) and the user clock bias. This solution is called *trilateration*. Since the radius of the GPS, but also of the other GNSS constellations, is in the order of thousands of km, the pseudosphere equation can be linearized converting the complex trilateration process in the intersection of tangent planes to the pseudosphere in the user position.
This procedure is practically implemented with LMS algorithm and can be expressed as follow:

$$\Delta x^k = ((H^K)^T H^K)^{-1} (H^K)^T \Delta \rho^K$$

To solve this, and obtain the PVT solution, is to apply in a recursive manner the LMS. In this way obtain a progressive updating of the linearization point for each iteration, ending up in a more fine solution. The "K" parameter represents the iteration and, at each increment of K, the solution is expected to be more accurate with respect to the user position. It is worth to mention that, what presented before, is just for one epoch (n=1) indeed this part of the the laboratory rely on this. As requested we import in the matlab script the provided datasets and, for Galileo constellation, single epoch the obtained result is report below.

Figure 22: Estimated user position, single epoch

What we have obtained is validated thanks to the solutions provided in the laboratory slides. The latitude is -33.924819 and the longitude 18.424176, ending up in Cape Town, South Africa.
Finally, to have an accurate view of the obtained position we present a zommed version of the previous plot.

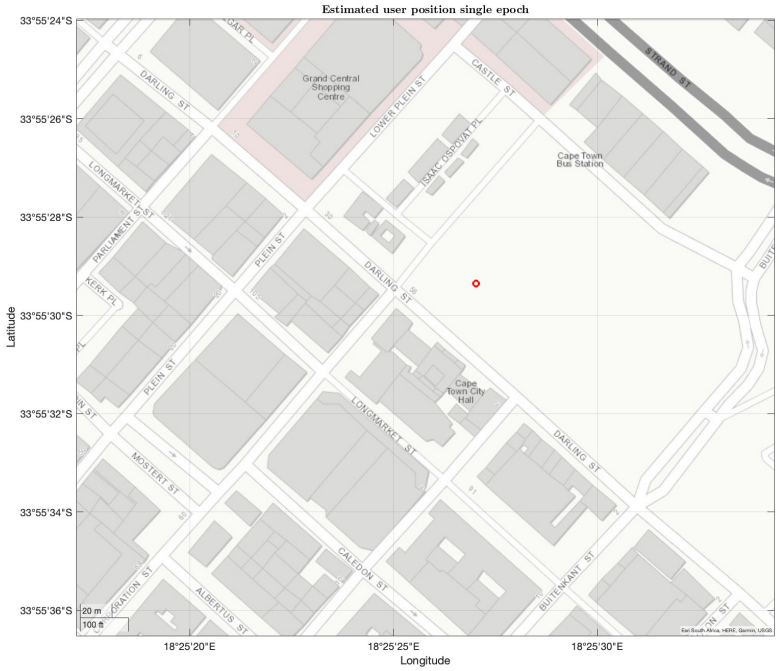


Figure 23: Zoom on the estimated user position, single epoch

Recursive LMS - Multi epoch

Passing onto the next section, we were asked to repeat the same operations but considering a multi epoch scenario. This means that now the time matters, so that the relation to be taken in to account is the following one:

$$\Delta x_n^k = ((H_n^K)^T H_n^K)^{-1} (H_n^K)^T \Delta \rho_n^K$$

In the implementation a key point is that for each new epoch n, new values are taken in to account, leading the iterative LMS to restart. It is worth to mention that, tipically, in the observables pseudorange rate measurements are inserted. Since this informations are not available we stick to compute the pseudoranges to obtain position and clock bias. Regarding our position results, we start considering the first dataset, from the "nominalUERE" folder, and GPS constellation. We found -33.924871 (lat), 18.424058(long) that is basically the same position of the previous section.

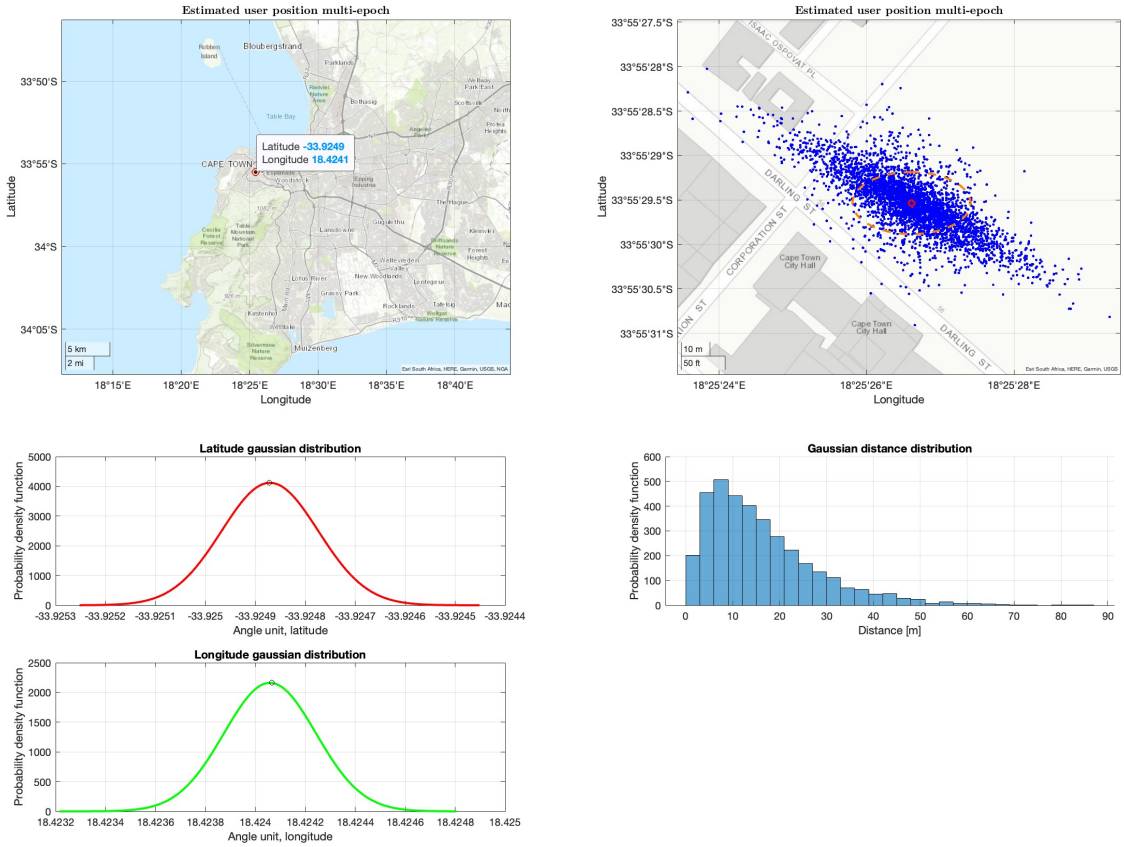


Figure 24: Multi epoch estimated user position, latitude, longitude and distance distributions.

In the rigth-upper part of the plot, all the computed position are showed as blue dots and the medium one, so our estimated user position, is highlighted in red. The orange ellipsoid describes the 1σ region of the guassian probability density function: that area includes the 68.3% of the computed points and indeed is the more populated one. The instogram presents the distance between each point and the computed user position taking in to account the heart geometry. In this we can conclude that the majority of the points were between 5 and 7,5 m from the user.
It is worth to mention that, during the theory classes, we have underlined that the positioning error is composed in the following way:

$$\delta x = [(H)^T H)^{-1} H^T)] \ \delta\rho$$

The second factor, that is to say $\delta\rho$, dependes on the estimation of the pseudorange. The elements that compose this vector are assumed to random variables

- Gaussian distributed with zero maen
- Indipendent and Identically distributed (i.i.d.)
- With variance σ^2_{UERE}

The variance is the same also on different satellites, so the probability density function will be always the same even changing dataset or constellation. This last statement will be clear observing the red PDF and the green one presented in the following sections.
Lastly we remaind the concepts of **accuracy** and **precision**: the first one is related to the distance between a point and the true position, that we do not know, while the second how close to each other are the estimated points. If we consider as true position the mean of the distribution, so our red point in the plots, we can say that good precision and accuracy were achieved. What it is important to avoid, is the bias because in that case the user would not be aware of the error (since do not know the true position).

Dataset 2 - GPS constellation

In task B.3 we were required to compare the quality of the obtained solutions for different datasets and constellations. To do so we consider the second dataset ending up the following final user position: 60.169855 (lat) and 24.938380 (long). This coordinates corresponds to a point inside the city of Helsinki, as can be appreciated in the following plots:

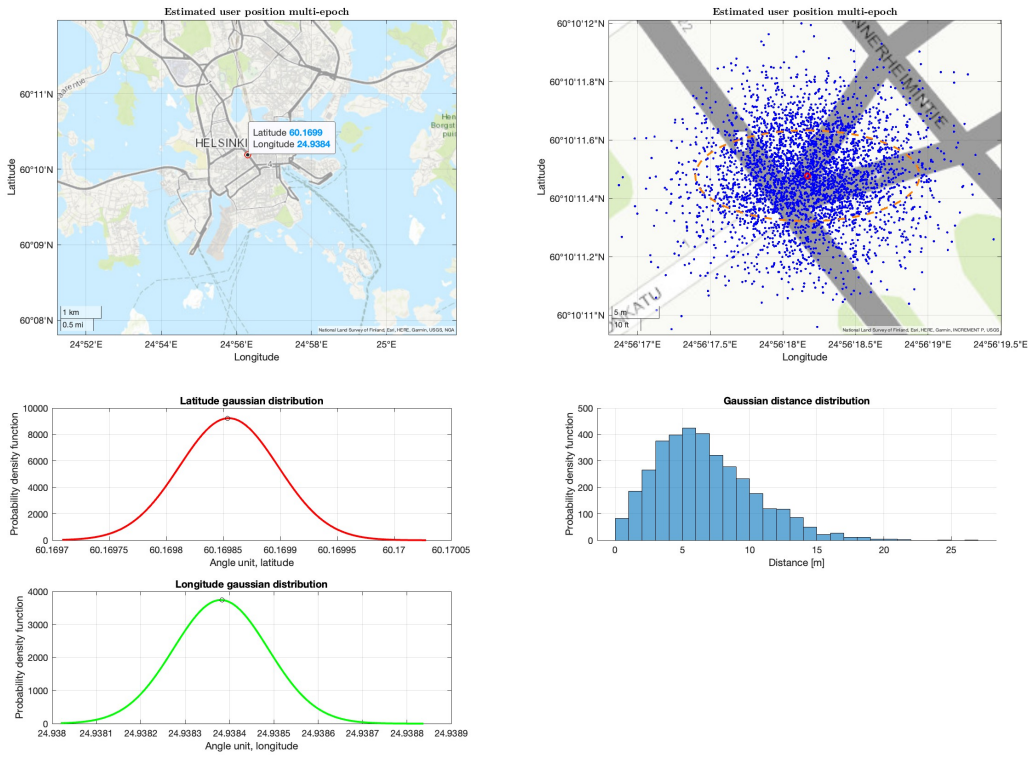


Figure 25: Multi epoch estimated user position, latitude, longitude and distance distributions.

It is clear that we have a more spread position dispersion as the points suggest. This can be noticed also analyzing the histogram of the distances: there is a wider interval, so a wider area under the curve, from 3 m to 7 m away with respect to the user position. One of the possible explanation is that, compared with the previous result, this is taken in a intersection of 4 roads surrounded by high buildings, so for sure this will affect the quality of the received signal at the user side.
Need to be noticed that this kind of consideration affects only the $\delta\rho$ factor of the previous presented formula 2, and can be reduced by using a better receiver. What can not be changed is satellites geometry affecting the rest of the mathematical relationship. Consequently in that position the user experience for sure a low number of visible satellites since the buidings were high and near.
Comparing this results with the previous one is clear how the environment affects the final position: in the first dataset the user ending up in a wide open area used as square and parking. Moreover, even if it is surrended building on two sides, these are not too high. The other two directions are more or less free. This means less signal reflections and multipath, but also a better visibility and an higher number of the satellites available. For clarity we attach here a comparison of the two user locations.

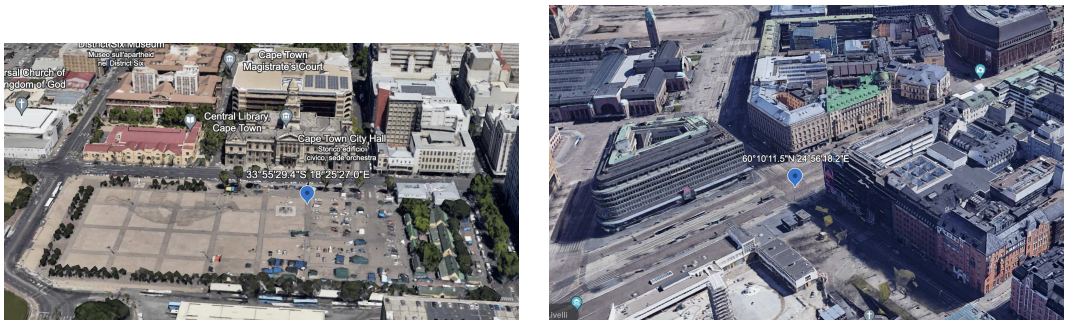
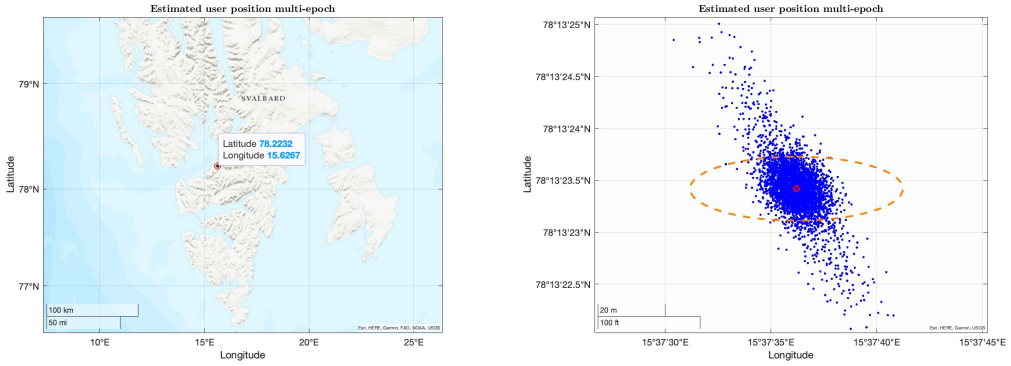


Figure 26: Rigth side: Capetown user position. Left side: Helsinki user position.

Dataset 3 - GPS constellation

For this part we want have a comparison with a better solution than the previous one to be properly commented. To do so we choose the third dataset in which the user position is at the Svalbard islands, precisely at 78.223172 (lat), 15.626723 (long). The obtained plots are reported here.



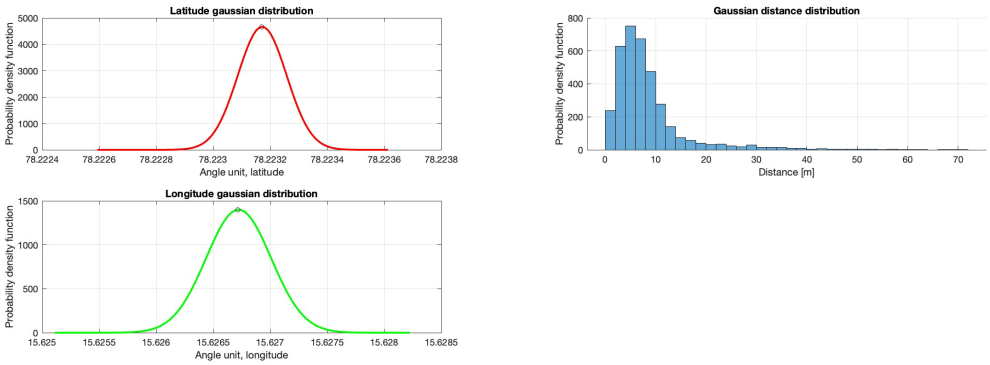


Figure 27: Multi epoch estimated user position, latitude, longitude and distance distributions.

Here it is clear that we have a better position dispersion: this can be noticed from both the right-up side graph, and from the bottom-right one. This specific locations do not sound new to us: we already know that at the Svalbard, precisely in the Spitsbergen island, there is the "SvalSat" station used by NASA, ESA and as sensor station to check the timing and accuracy of Galileo as reported here. This location was choosen because is one of the best, even not the best, under a lot of parameters like: availability, mean daily access, mean daily capacity, and SNR. We have practically achieved this results in a past year project but here just the final result is mentioned.
Coming back to more physical, or environmental considerations, we can say that this is the location with more open space conditions. In the area near the user position there are no high buildings and no urban environment, so we achieve good performances both on the satellite geometry and on the final signal quality (no or really low multipath and reflections). To allow the user visualize better the described situations we compare here below the Svalbard and Helsinki locations that are, respectively, the best and the worst of the analyzed datasets.

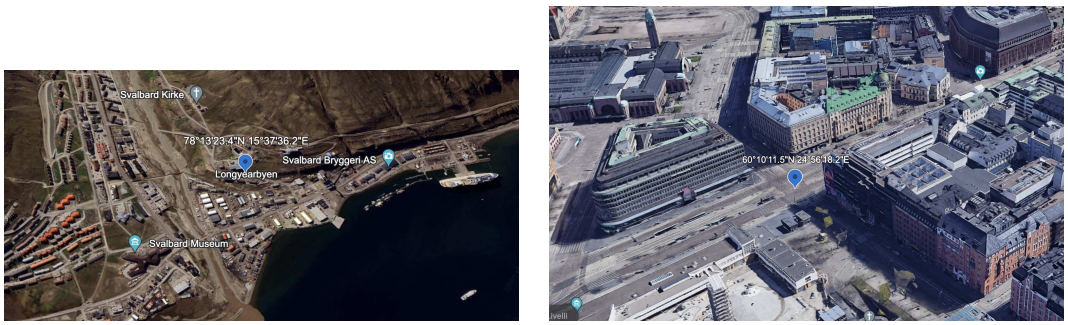


Figure 28: Rigth side: Capetown user position. Left side: Helsinki user position.

Comparison between GPS and Galileo for Svalbard location

Regarding the comparison for different constellations, we present what is obtained with GPS and Galileo for the Svalbard location.

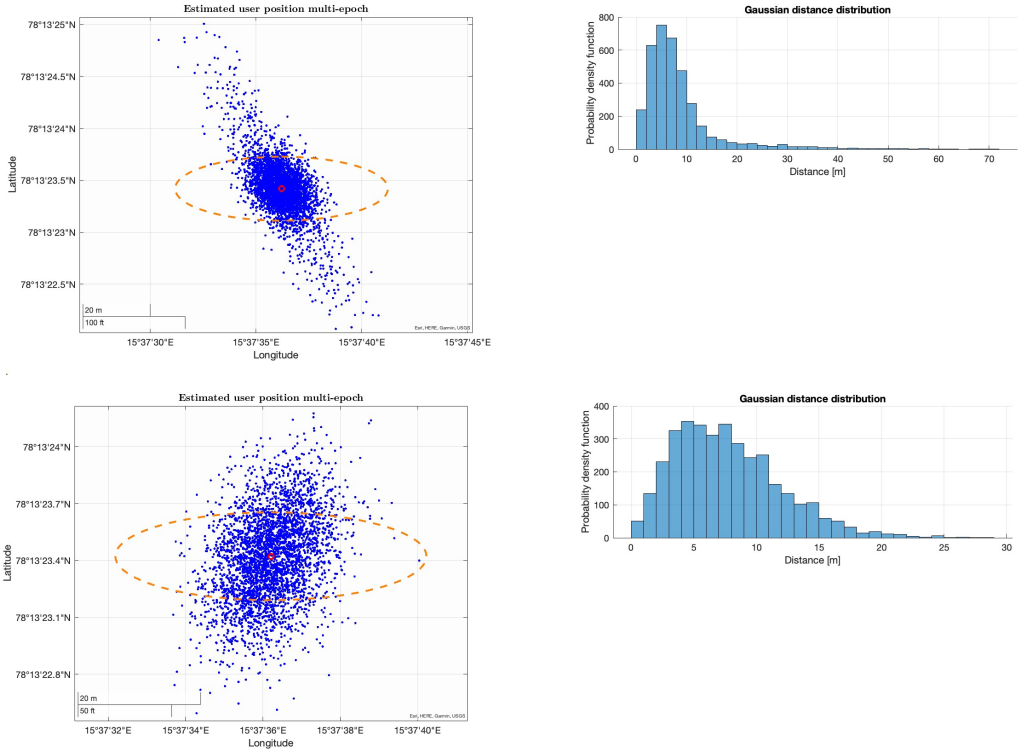


Figure 29: Up: Svalbard results with GPS. Down: Svalbard results with Galielo.

In this part we want highlight the difference in terms of position dispersion achieved by this two constellations. Using the GPS constellation we achieve better results as confirmed also by the histogram and by the point cloud. On the other hand, using Galielo we are not able to achieve that results: this is because Galileo constellation is not optimized for such extremely high latitude, even if some north country are served too.
Similar observations can be carried out using Galileo and GPS on other datasets, like the number 6. Doing so leads to a slightly better position dispersion using GPS, and this can be due to an higher number of visible satellites.

Weighted LMS - WLMS - Multi epoch

In this last section we were tasked to implement a Weighted LMS algorithm. The idea is to assign a smaller weight to the more noisy measurements. This is done assuming a non unitary covariance matrix R, that leads to a weighting matrix computed as W = R^{-1}.
From the data point of view this requires to load datasets form the realisticUERE folder. To compare the improvements in results with the one computed through the LMS, we consider always datasets number 1, 2 and 3. To optimize the space in the report we present directly the discussion regarding WLMS versus LMS: this is also because the user position and the σ_{UERE} are displayed as well.

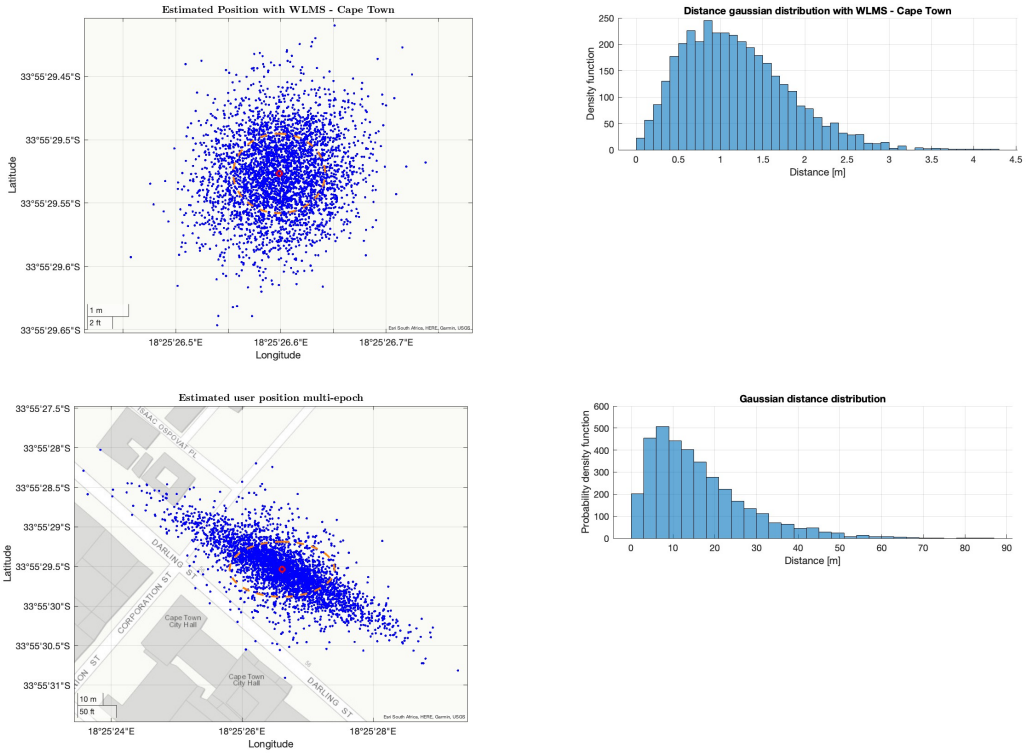


Figure 30: Up: performances using WLMS. Down: performances using LMS. (Cape Town)

As expected, the results for Cape Town, confirm the improved position dispersion and distance distribution with respect to the user position. Observing the σ_{UERE} distribution, is clear that the values passes from ≈ 8 m using LMS to less than 1 m using WLMS. This is because the weighted approach is capable of assign more importance to the less noisy measurements.
The following plots lead to the same conclusion but regards the second dataset, so Helsinki.

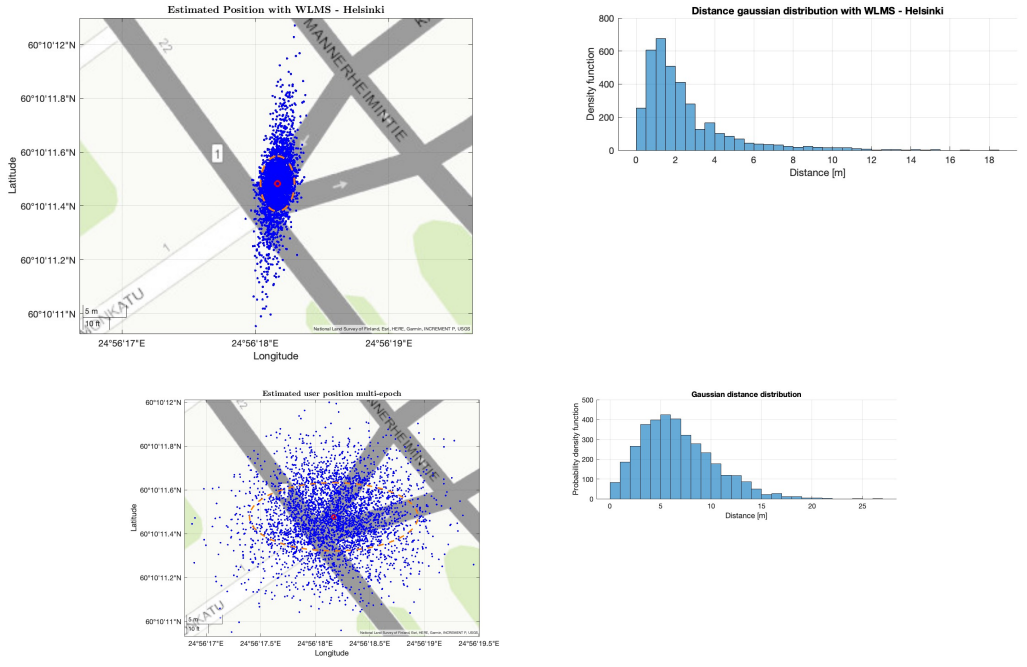


Figure 31: Up: performances using WLMS. Down: performances using LMS. (Helsinki)

Also in this case the improvements is clear: the position dispersion is significantly lower, with a more thick cloud of points around the user position. Also from the σ_{UERE} distribution, and mean value, we can retrieve the same conclusions: $\sigma_{UERE_{LMS}}$ is around 5 m, while $\sigma_{UERE_{WLMS}}$ is between 1 and 1.5 m.
Lastly, we focus or attention on the third dataset so the one regarding the Svalbard.

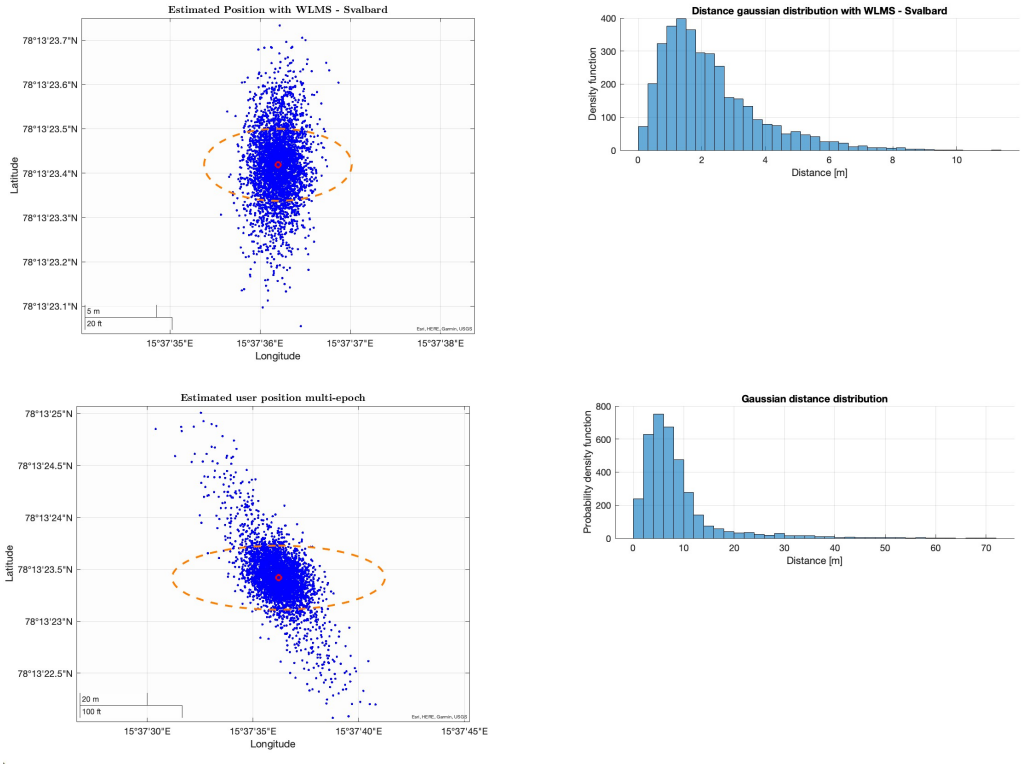


Figure 32: Up: performances using WLMS. Down: performances using LMS. (Svalbard)

Similar comments can be carried out also for this case observing a more accurate user position graph, in parallel to a lower σ_{UERE} that passes from $\approx$ 5 m to $\approx$ 1 m.

3 GNSS spreading codes

This third assignment is focus on GNSS spreading codes, in particular on GPS and Galileo ones. To occupy too much space in the report, we do not insert all the theoretical background related to this part, sticking to present just the most meaningful results we needed. Anyway it is wort to remember that to generate the GPS C/A code just 2 sequences are required. These sequences, call them G_1 and G_2, are maximum length sequences of order 10, so by keeping G_1 fixed and combine differents shifts of G_2 we are able to generate all the GPS C/A codes. Here need to be noticed that at each different shift of G_2 correspond a different satellite.
Considering, for example, G_1 and G_2, we can write the following:

$$\begin{cases} G_1 = 1 + x^3 + x^3 \\ G_2 = 1 + x^2 + x^3 + x^6 + x^8 + x^9 + x^{10} \end{cases}$$

Generation of GPS codes

Starting the practical implementation of what was required we were asked to write a function in matlab able to generate the chips of all the 32 Gold sequences that are used by GPS C/A

signals on L1. We know that the sequence order (m) need to be 10, indeed the lenght of the sequence is $2^m - 1$ so 1023.
Our results were consistent with the requirements: each sequence is long 1023 chips and contains $2^{m-1} - 1$ {+1} (511) and 2^{m-1} {-1} (512). This can be observed by open the 32x1023 obtained matrix (named gold) or using *"num2str()"* to display on the command window each rows and its content. The obtained results are reported below (just for space and graphic reason are reported side by side):

```
Riga 1: Numero di 1 = 511, Numero di -1 = 512
Riga 2: Numero di 1 = 511, Numero di -1 = 512
Riga 3: Numero di 1 = 511, Numero di -1 = 512
Riga 4: Numero di 1 = 511, Numero di -1 = 512
Riga 5: Numero di 1 = 511, Numero di -1 = 512
Riga 6: Numero di 1 = 511, Numero di -1 = 512
Riga 7: Numero di 1 = 511, Numero di -1 = 512
Riga 8: Numero di 1 = 511, Numero di -1 = 512
Riga 9: Numero di 1 = 511, Numero di -1 = 512
Riga 10: Numero di 1 = 511, Numero di -1 = 512
Riga 11: Numero di 1 = 511, Numero di -1 = 512
Riga 12: Numero di 1 = 511, Numero di -1 = 512
Riga 13: Numero di 1 = 511, Numero di -1 = 512
Riga 14: Numero di 1 = 511, Numero di -1 = 512
Riga 15: Numero di 1 = 511, Numero di -1 = 512
Riga 16: Numero di 1 = 511, Numero di -1 = 512
Riga 17: Numero di 1 = 511, Numero di -1 = 512
Riga 18: Numero di 1 = 511, Numero di -1 = 512
Riga 19: Numero di 1 = 511, Numero di -1 = 512
Riga 20: Numero di 1 = 511, Numero di -1 = 512
Riga 21: Numero di 1 = 511, Numero di -1 = 512
Riga 22: Numero di 1 = 511, Numero di -1 = 512
Riga 23: Numero di 1 = 511, Numero di -1 = 512
Riga 24: Numero di 1 = 511, Numero di -1 = 512
Riga 25: Numero di 1 = 511, Numero di -1 = 512
Riga 26: Numero di 1 = 511, Numero di -1 = 512
Riga 27: Numero di 1 = 511, Numero di -1 = 512
Riga 28: Numero di 1 = 511, Numero di -1 = 512
Riga 29: Numero di 1 = 511, Numero di -1 = 512
Riga 30: Numero di 1 = 511, Numero di -1 = 512
Riga 31: Numero di 1 = 511, Numero di -1 = 512
Riga 32: Numero di 1 = 511, Numero di -1 = 512
```

Figure 33: "num2str()" command line results.

To fully accomplish the third task we compare our generated codes with the ones from the official GPS webpage and they were correct.

GPS codes properties

The aim of this part is basically to confirm, through our laboratory results, the properties of the GPS codes. To do so we compute and plot the normalized linear and circular auto-correlation functions of the first code. We attach the obtained plots.

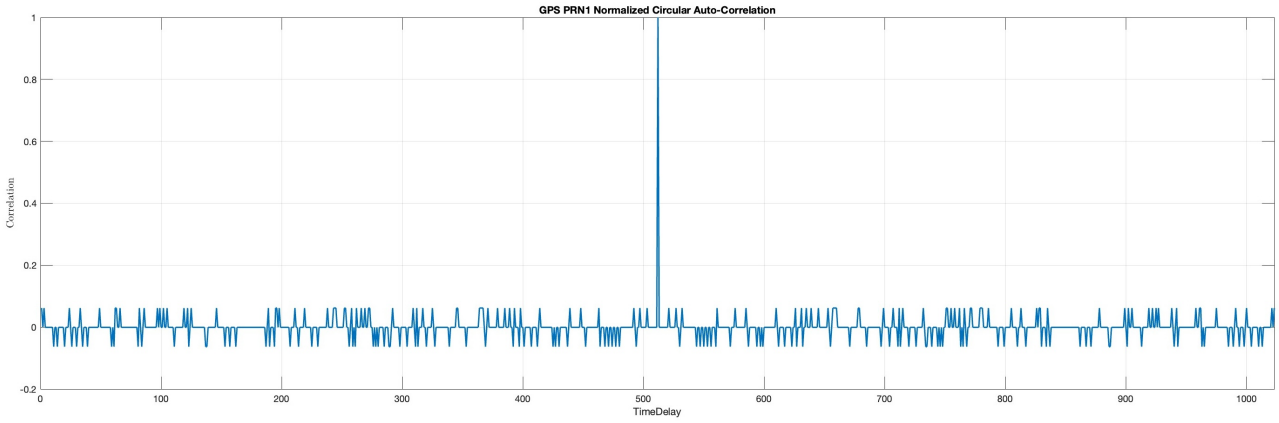


*Figure 34: GPS PRN1 normalized **circular** auto correlation*

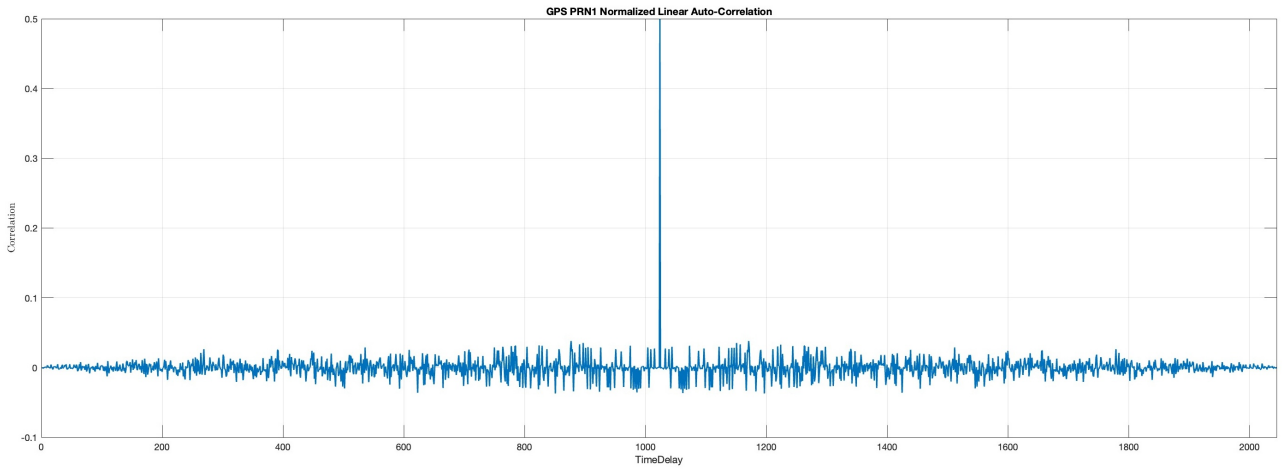


*Figure 35: GPS PRN1 normalized **linear** auto correlation*

In the previous two plots we need to impose a linewidth of 1.5 to not have a too thick graph that it would have been difficult to read.
Consequently we need to verify the values assumed by the circular auto correlation. These need to be consistent with:

$$\left\{1,\ -\frac{1}{p},\ \frac{-\beta(m)}{p},\ \frac{\beta(m)-2}{p}\right\} \text{ with } \beta(m) = 1 + 2^{\left\lfloor \frac{m+2}{2} \right\rfloor}$$

Substituting our values we obtain:

- $-\frac{1}{p} = -\frac{1}{1023} = 9.775171065\, e^{-4}$
- $\beta(m) = 1 + 2^{\left\lfloor \frac{m+2}{2} \right\rfloor} = \beta(10) = 1 + 2^{\left\lfloor \frac{10+2}{2} \right\rfloor} = 65$
- $\frac{-\beta(m)}{p} = \frac{-65}{1023} = 0.06353861$
- $\frac{\beta(m)-2}{p} = \frac{63}{1023} = 0.061583577$

To guarantee ad easier reading of this part, we have highlighted some points in the plots representing the values calculated above. Since these match perfectly the one computed analytically, we have prove the goodness of our results.
Analyzing the plots from a more mathematical point of view we can say that this functions ensure a very good separation, or difference, in dB between the main peak and the maximum side peaks. This is confirmed because the maximum values outside the peak are $20log_{10}\left|\frac{63}{1023}\right| = 24.211$ dB and $20log_{10}\left|\frac{65}{1023}\right| = 23.939$ dB.

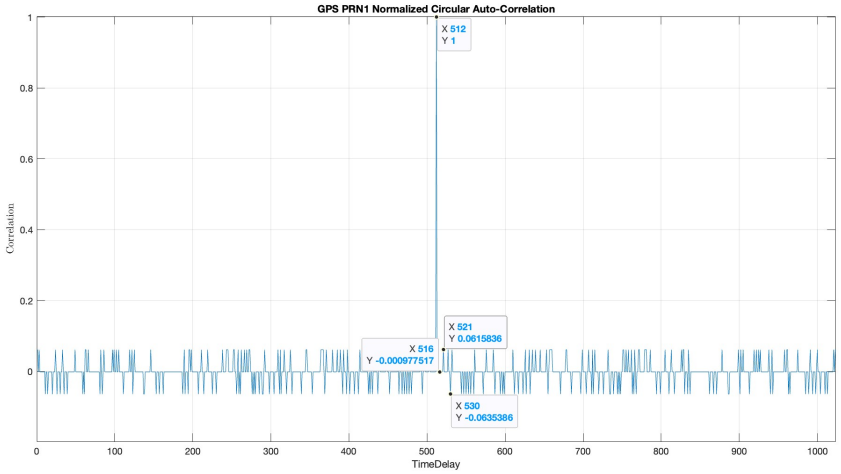


*Figure 36: GPS PRN1 normalized **circular** auto correlation with highlighted values.*

To overcome the next task we choose to compute the linear and circular cross correlation between GPS PRN1 and PRN2. The obtained plots are reported below:

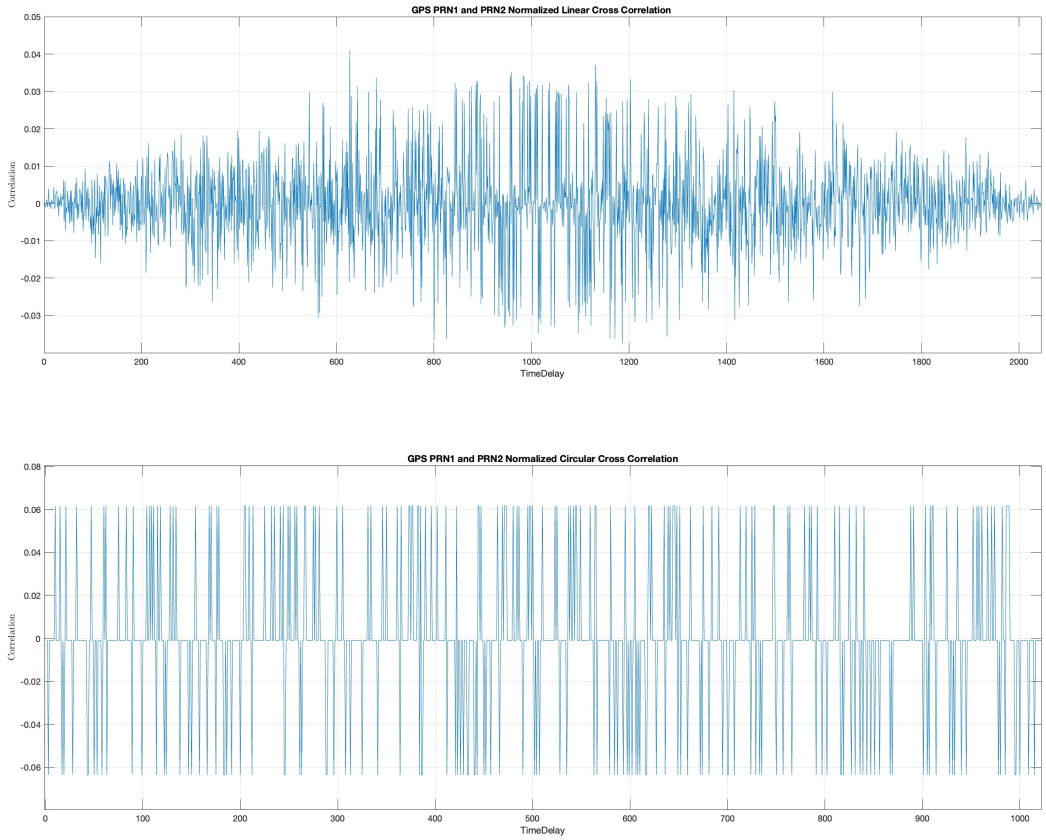


*Figure 37: Up: GPS PRN1, PRN2 normalized linear **cross** correlation. Down: GPS PRN1, PRN2 normalized circular **cross** correlation*

Passing onto the last task of step2 we need to verify that the circular cross correlation assumes the correct values, that is to say:

$$\left\{ -\frac{1}{p},\ \frac{-\beta(m)}{p},\ \frac{\beta(m)-2}{p} \right\} = \left\{ 9.775171065\, e^{-4},\ 0.06353861,\ 0.061583577 \right\}$$

We adopt the same approach: highlight the meaningful points in the graphs. Again the laboratory result match the analytical ones, proving the correcteness of our implementation.

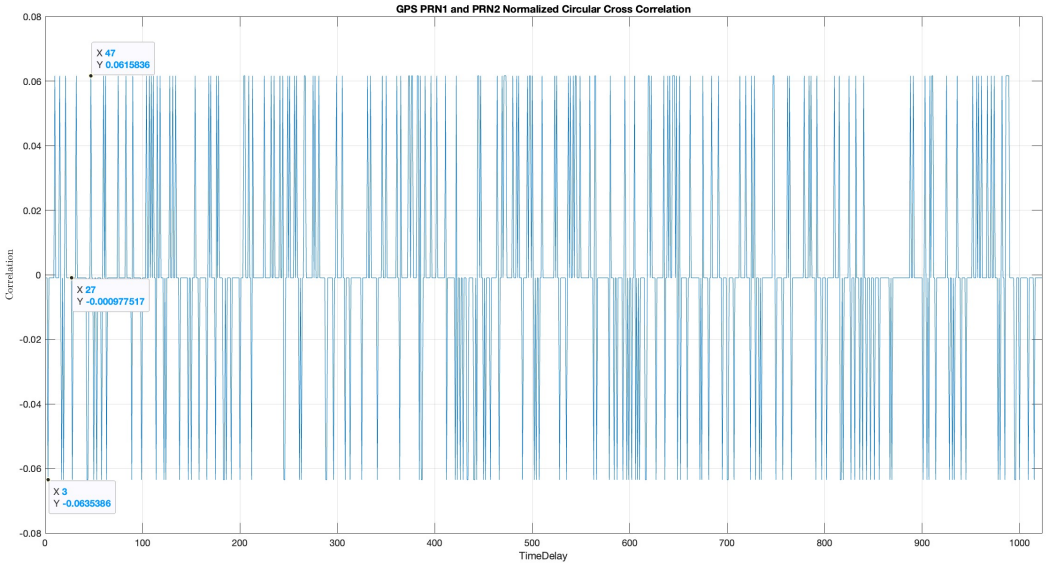


*Figure 38: GPS PRN1, PRN2 normalized circular **cross** correlation with highlighted values.*

Concluding this section can be sad that the shape of the normalized circular cross correlation is such becuase, missing the 1 in the expected values, we do not have the peak.

Generation of Galileo codes

Here we focus our attention on the Galielo constellation. Firstly we download the E1b memory codes and load them in to the matlab script by means of GalileoCodes.mat.
Our task is to repeat the same passages done for GPS but using the Galielo data, so compute and plot the linear and circular auto correlation and cross correlation functions.

Linear and circular auto correlation

We have split the presentation of the results in two subsection. This first rely on the auto correlation functions as can be appreciated from the next graphs.
The results match our expectations both from the point of view of the general function shape but also from the mathematical one. We know that Galileo has a code length of four times the GPS, so 4092. This information can be founded in the x-axes of both the graphs: in the linear one the peak is exaclty at 4092, while in the circular one is at 2046.

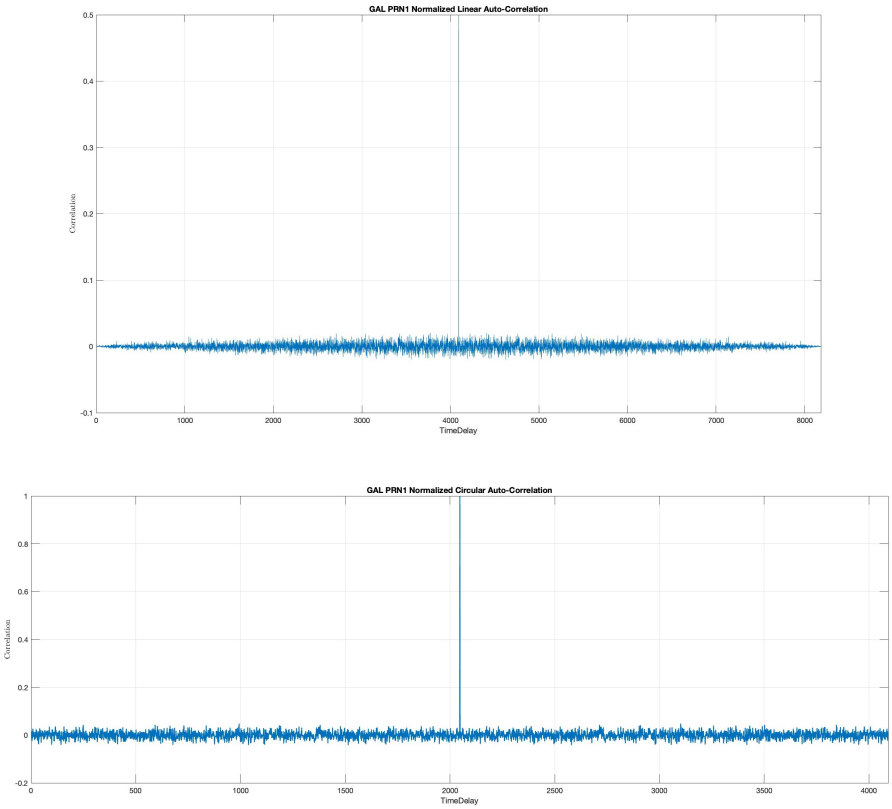


Figure 39: Up: Galielo PRN1 normalized linear auto-correlation. Down: Galielo PRN1 normalized circular auto-correlation

Linear and circular cross correlation

In this second sub section the results for the Galileo linear and circular cross correlation function are presented.

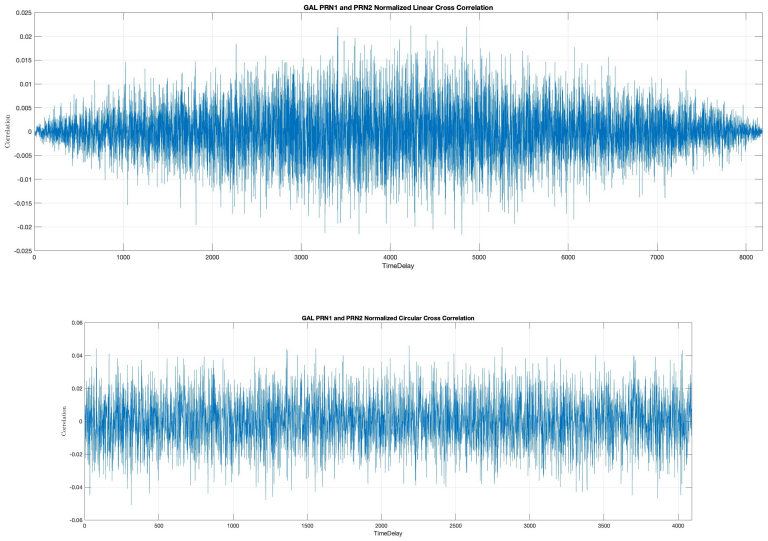


Figure 40: Up: Galielo PRN1-PRN2 normalized linear cross-correlation. Down: Galielo PRN1-PRN2 normalized circular cross-correlation

Passing onto the last request we need to compute the ratio in dB between the maximum amplitude of the circular auto correlation function and its highest side peak in absolute value. This need to be done for both GPS and Galileo codes. For GPS we found a value of 23.9392 dB (that is also aligned with already presented comment), while for Galielo the result was 26.2181 dB. This not sound surprising because Galielo use longer codes, four time the GPS ones, so it is better in "averaging out" the noise over a wider "integration interval".

Generation of the code local replica

In this step we were tasked to generate the local code replica using a sampling frenquency of 16.368 MHz and the chip rate reported below. In the following we report some key values that are the core of the GPS C/A codes:

- Code length equal to 1023 $(2^{10} - 1)$
- Code duration $= T_{code} = 1\ ms$
- Chip duration $\approx\ \mu s$
- Chip rate $= R_{chip} = 1023\ Mchips/s$

Codes in time and frequency domain

In this final part we were asked to plots the previously generated codes and comment what we obtained. The first two figures (41) regards the plots of the **GPS** in both time and frequency domain. Regarding the plot of the GPS PRN1 sequence in time domain, there is not much to say about: our result follow the expectations and show the pseudo random noise behaviour characteristic of this (gold) codes. Observing the GPS PRN1 code spectra, in frequency domain, it is worth to mention that what we observe is the envelope due to the rectangular shape of the code shaping. Mathematically what can be sad is that the shape depends on the following term and is indeed related to the bit duration.

$\left| \frac{sin\pi f T_c}{\pi f} \right|^2$ with T_c being the *chip time*, so the duration of one bit of the binary sequence.

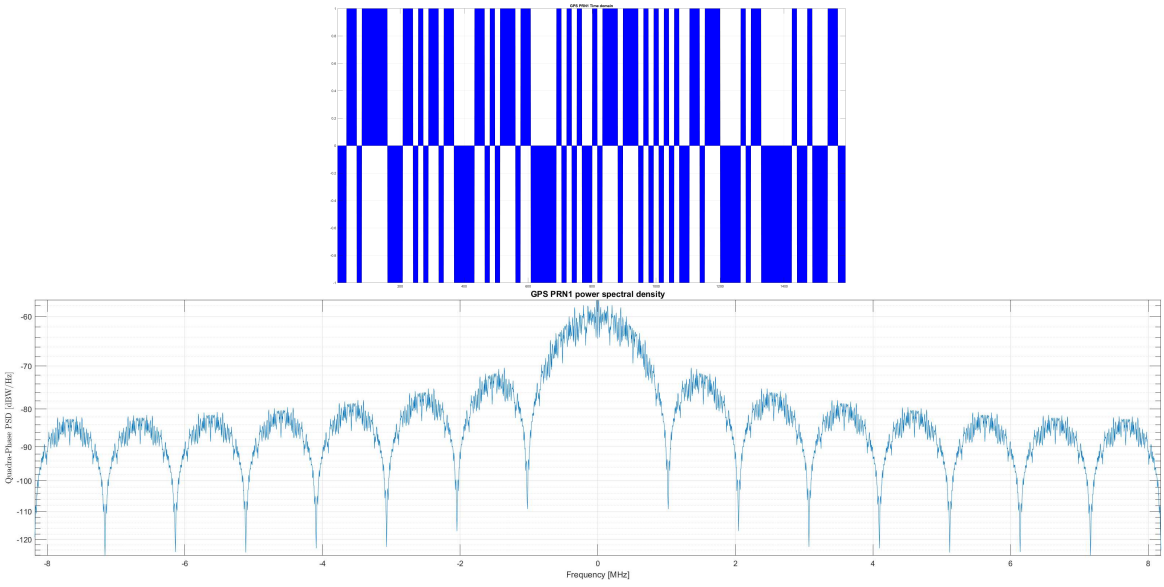


Figure 41: GPS codes in time (up) and frequency (down) domain

Passing onto the **Galileo** constellation, we need to introduce the BOC modulation. The acronym stands for "Binary Offset Carrier" modulation and it is obtained applying a square subcarrier to a BPSK signal with chips modulated by means of a rectangular shape. Such a operation is performed to **reduce** the mutual interference between signals from different GNSS constellations that are modulated over the same carrier. The general expression for the BOC modulation is BOC(n,m) with:

- n = sub carrier frequency (multiple of 1023 MHz, so of the GPS C/A code fundamental frequency)
- m = chip rate (multiple of 1.023 Mchip/s, so of the GPS C/A code chip rate)

In this context a key role is played by the subcarrier, which expression is:
$\left[sin\left(2\pi \frac{1}{T_{sc}} t\right) \right]$ with $T_{sc} = \frac{T_R}{n} \approx \frac{1\mu s}{n}$ being the *subcarrier period*.

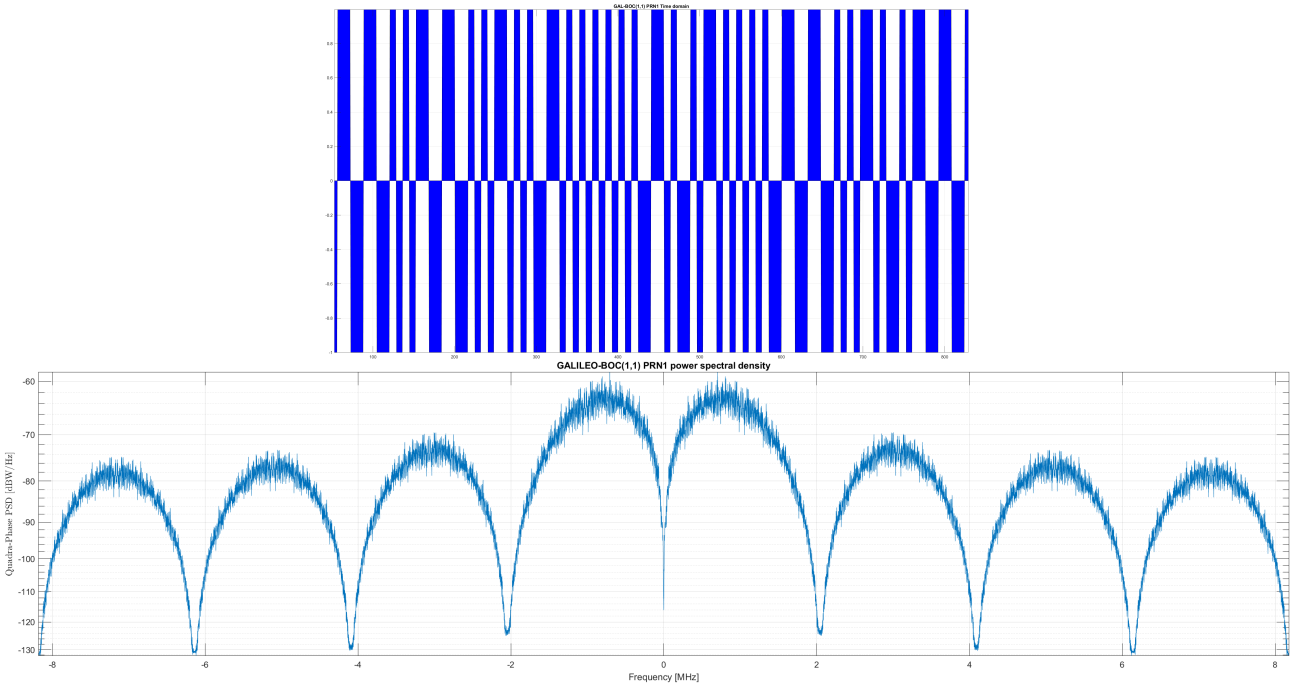


Figure 42: Galileo codes in time (up) and frequency (down) domain

It has a paramount importance that, since the GPS C/A code is taken as reference ($f_R = \frac{1}{T_R} = 1.023\ MHz$), there is **always** an entire number of sub carrier periods inside a chip duration.
In the laboratory we deal with BOC(1,1): this means that we have 1 period of the subcarrier in 1 code chip, $T_{sc} = \frac{T_R}{1} \approx 1\mu s$ and $T_r = \frac{T_R}{1} \approx 1\mu s$. Observing figure 42, it is clear also that the envelope of the singal is different from the GPS one. This is expected since applying the sub carrier, the shape of each chip is modified.
Moving towards the end of this quite long activity, we were asked to plot in the same figure the GPS and Galileo BOC(1,1) codes spectra 43: this, as usual when plots of spectrum are required, request to work with the welch periodogram and in the specific case to superimpose the results. What we obtain is reported here below:

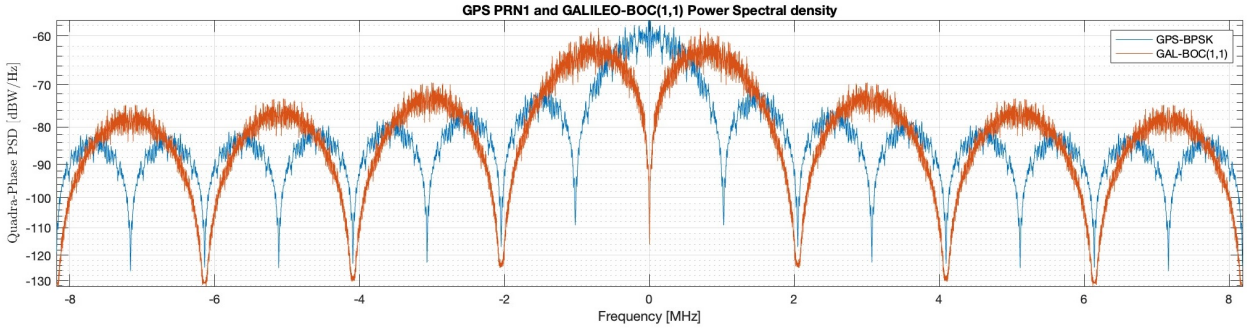


Figure 43: Galileo BOC(1,1) code spectra (orange) and GPS code spectra (blue)

This plots shows in a really clear way the differences between the GPS and Galileo BOC(1,1) spectrum: it is clear that, for Galileo, the maximum of the power spectra is shifted from the central frequency. This shift is related to n, while the width of the main lobe depends on m. Moving to the last task, we were asked to plot in the same graph the normalized **circular** auto correlation both for GPS and Galileo BOC(1,1) codes. Our result is reported in the following figure.

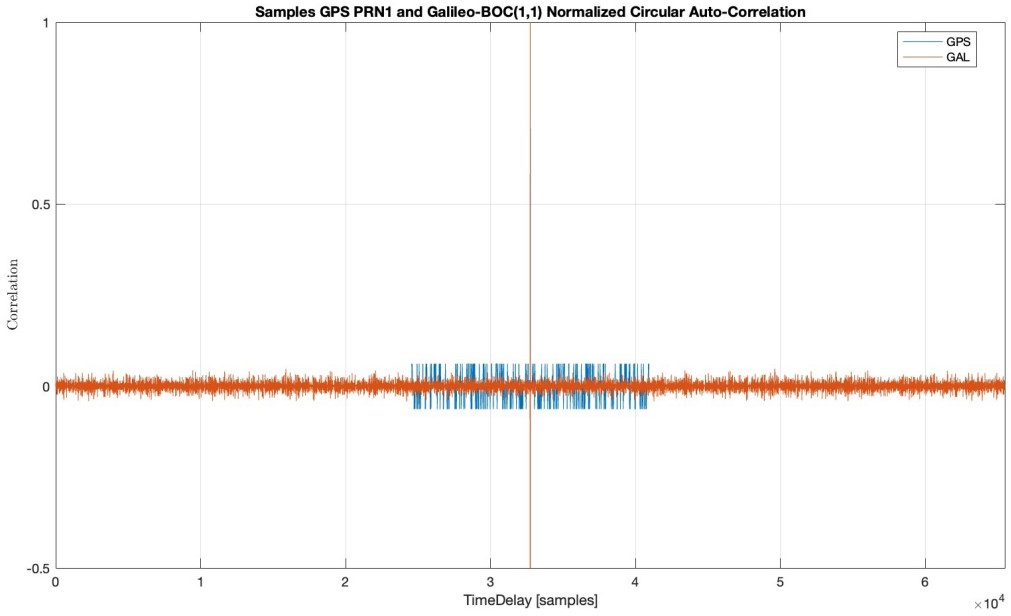


Figure 44: Galileo BOC(1,1) code spectra (orange) and GPS code spectra (blue)

4 IF Signals Correlations

The final goal of this fourth laboratory is to implement the **acquisition stage** of a GNSS receiver.
As basics for the next steps we need to download the material from the "portale della didattica" and to write the function *generateLocalIF*. This generates a complex carrier, that we named "carrierOut", that will be used to down convert a GNSS signal from IF to baseband. Just as tool to better analyze the reported results, we attach here a list of the key parameters:

- Sampling frequency = 16.368 MHz
- IF carrier frequency = 4.092 MHz
- Doppler = 0 Hz
- Amplitude = $\sqrt{2}$
- Signal length in time = 1 ms.

In the second step we were asked to plot the graphs of the **code** (previous laboratory), the real part of the **carrier** generated at the prior step, and finally the **product** between the code and the carrier. This need to be done and comment in both *time* and *frequency domain*, so in the following a total of six plots will be presented.

Code in time and frequency domain

Here we aim to present the GPS PRN1 code: this time, just to graphical reasons, we use "stem" so that the discrete time nature of the signal can be better appreciated. For the rest of the characteristics the plot is the same of the step 4 of Lab 3.

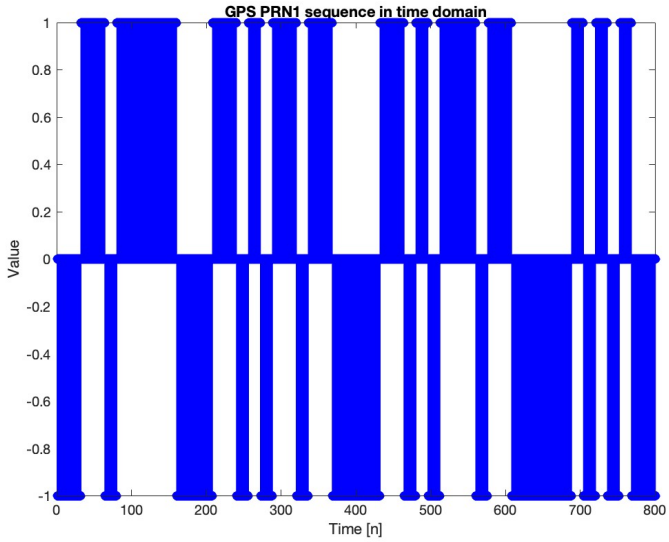


Figure 45: GPS PRN1 in time domain.

Moving to the frequency domain we need again to work with the pwelch and set properly all the parameters. What we obtained is shown below:

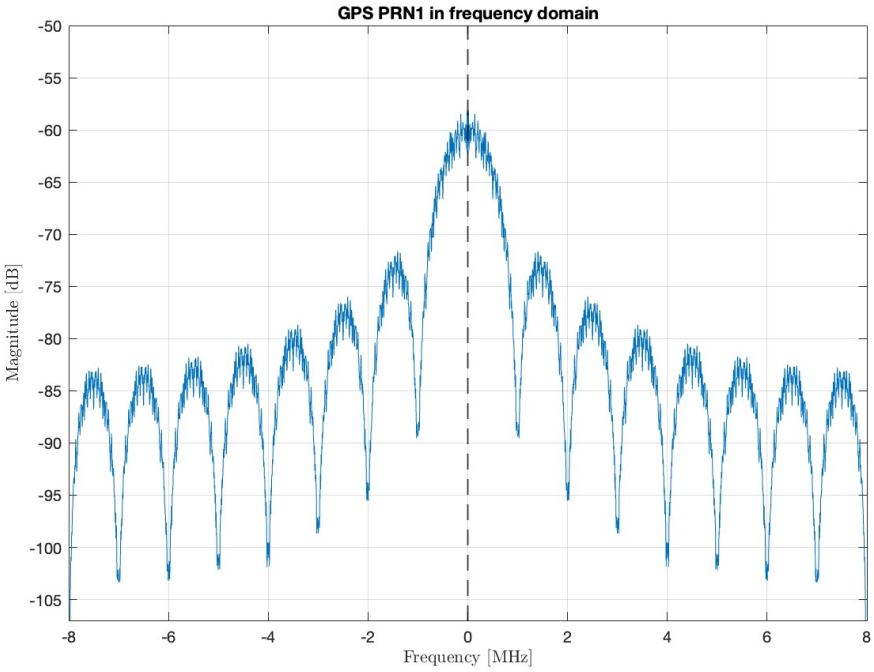


Figure 46: GPS PRN1 in frequency domain.

Here, the only difference from the laboratory 3 is that we display the shape of the signal: this is given by the minus and plus one of the code. For this reason often the envelope, that is equal for all the satellites, is preferred.

Real part of the carrier in time and frequency domain

The carrier that we generated in the first step is complex. We were asked to plot the real part and, in time domain, we reach the following result:

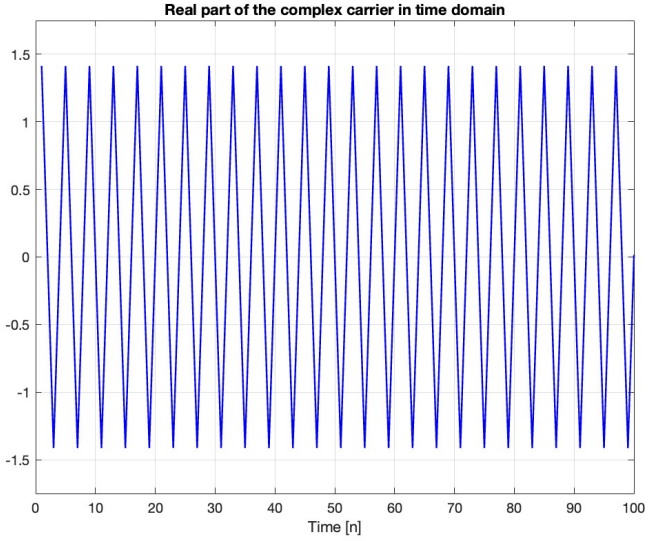


Figure 47: Real part of the generated carrier in time domain.

The amplitude factor can be appreciated on the vertical axes, while on the horizontal direction, we set a limit to better display the carrier behaviour.
Passing onto the plot in frequency domain we obtain the following:

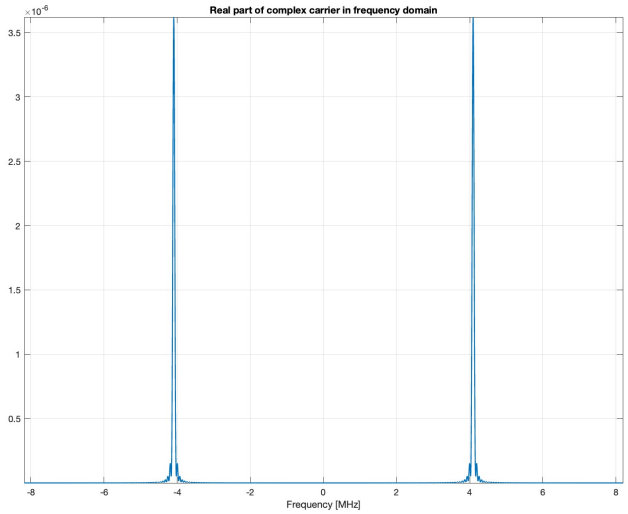


Figure 48: Real part of the generated carrier in frequency domain.

In this case the Intermediate Frequency (IF) can be surely noticed and will be crucial for the next shifting, or product, operations.

Product between code and carrier in time and frequency domain.

To conclude the third point of task 1, we need to multiply the code and the carrier and plot the results in the two domains.

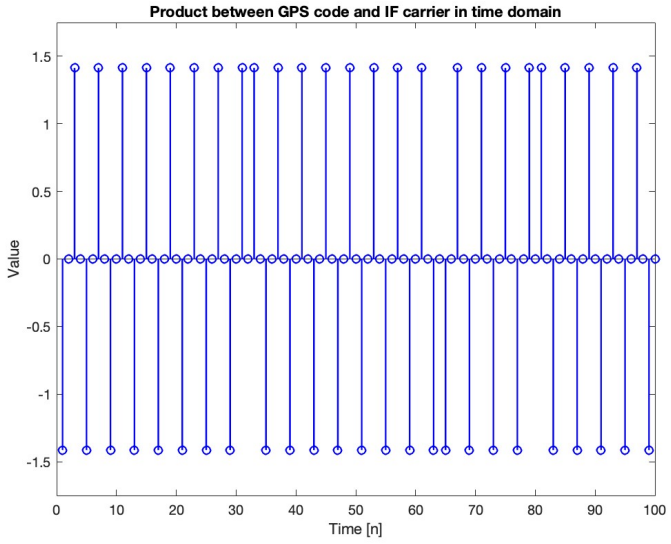


Figure 49: Product between code and the carrier in time domain.

Also in this case we have resorted to the use of "stem" to highlight the discrete nature of the plot and the amplitude factor is notable.
Passing onto the product in frequency domain is clear the effect of the IF carrier: the spectrum is centered at IF in both sides of the frequency axis.

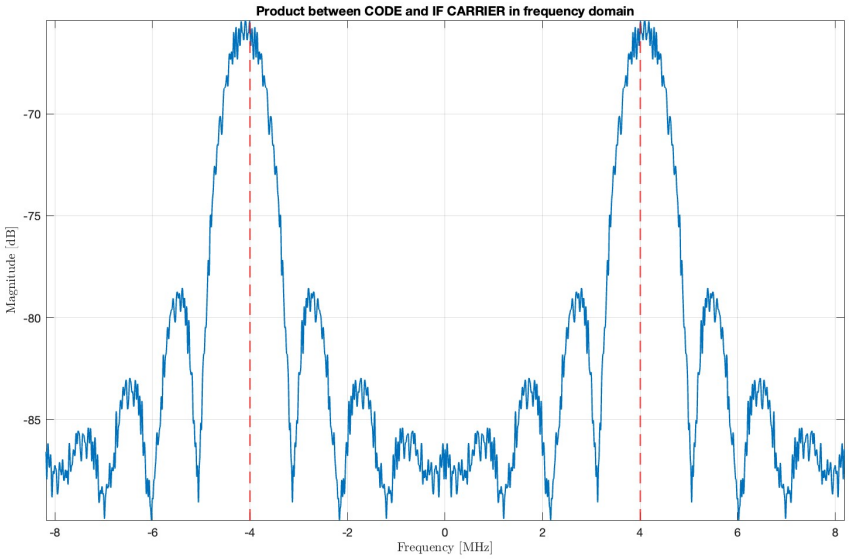


Figure 50: Product between code and the carrier in frequency domain.

Correlations

This part is completely devoted to correlations. The first requirement is to compute the **circular** and the **FFT-based** cross correlation between the local generated code and the received sequence. In this case the results shows that, from the end state point of view, there is no difference between these two approaches.

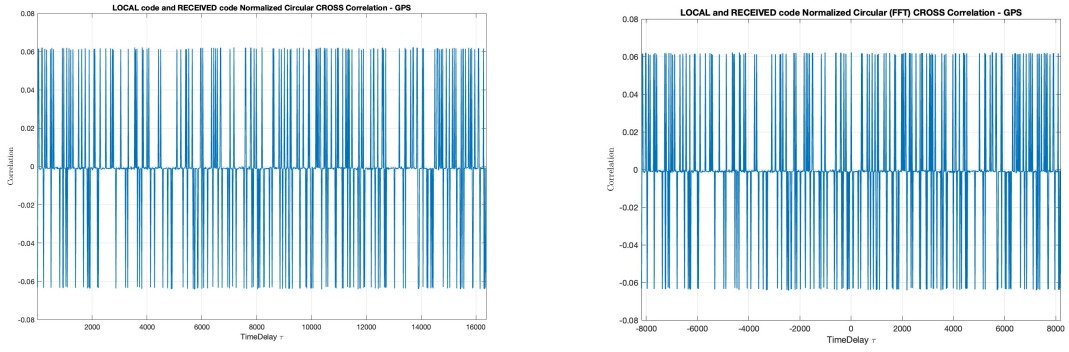


Figure 51: Left: circular based cross-corr. Right: FFT based cross-corr.

As general concept, can be noted that the FFT approach is faster than the traditional one, leading to a more efficient way to obtain the same result. This can be particularly useful when it is required to deal with long signals.

Moving to the next point we need to find which PRN code is inside the received sequence: after have tried all the possible comparison, we have found that the code is the PRN 3 as can be appreciated in the following plot:

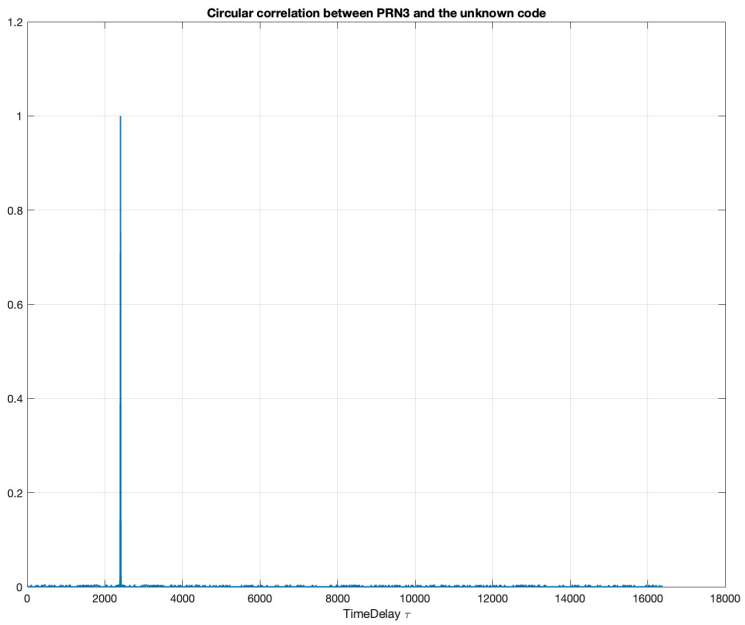


Figure 52: "Hidden" PRN in the received code: PRN 3

We have written a function (called *circular*) tailored specifically for the requirements of this sub task. Works by applying the circular correlation at the two signal vectors in order to find the peak, so the point in which the second one start. In our case the discovered delay is 2402.

Serial acquisition

We were required to implement a **serial** acquisition scheme to acquire an ideal GPS signal, i.e. no noise is considered. From the theory we know that this correspond to apply an exhaustive search to find the delay-doppler bin that maximize the CAF (Cross Ambiguity Function). This approach has the advantage that is "easy" to implement but the required time is proportional with the number of bins to explore.
In the following we present the used parameters:

- Signal length (time) = 50 ms
- Signal type = double
- Sampling frequency = 16.368 MHz
- IF frequency = 4.092 MHz
- GPS L1 C/A
- $T_{coh} = L * T_S = 1\, ms$
- $\Delta f = \dfrac{2}{3 * T_{coh}} = 666.67\, Hz$ (Rule of thumb)

Our results, are reported in the following three plots and are consistent with the expected ones.

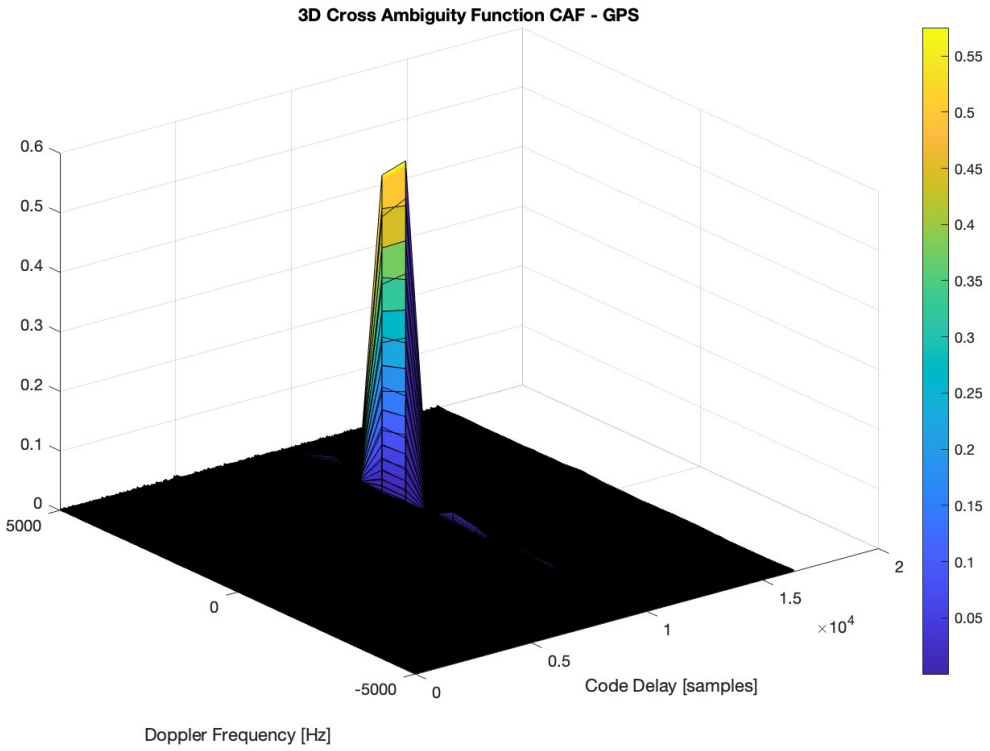


Figure 53: 3D GPS CAF

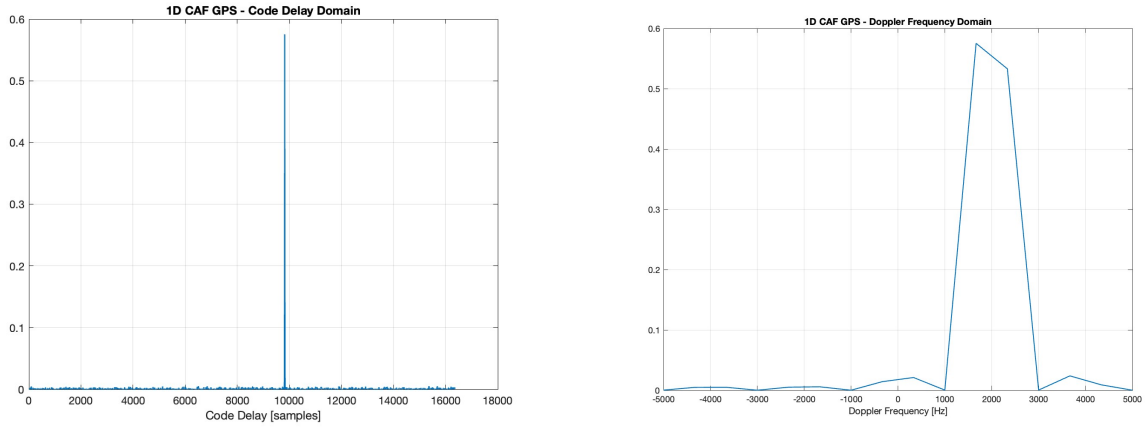


Figure 54: Left: 1D CAF in code delay domain. Right: 1D CAF in Doppler frequency domain.

5 GNSS Signal Acquisition

The aim of this fifth laboratory session is to implement an improved and more advanced GNSS acquisition stage: in particular we were asked to acquire real GNSS signals.

Parallel acquisition in time domain

As starting point we need to implement the code to acquire and *ideal* GPS signal by means of parallel acquisition in time domain. After have read the GNSS raw IF samples, generate both local carrier and code, we perform parallel acquisition. Finally we plot the obtained cross ambiguity function an we attach it below:

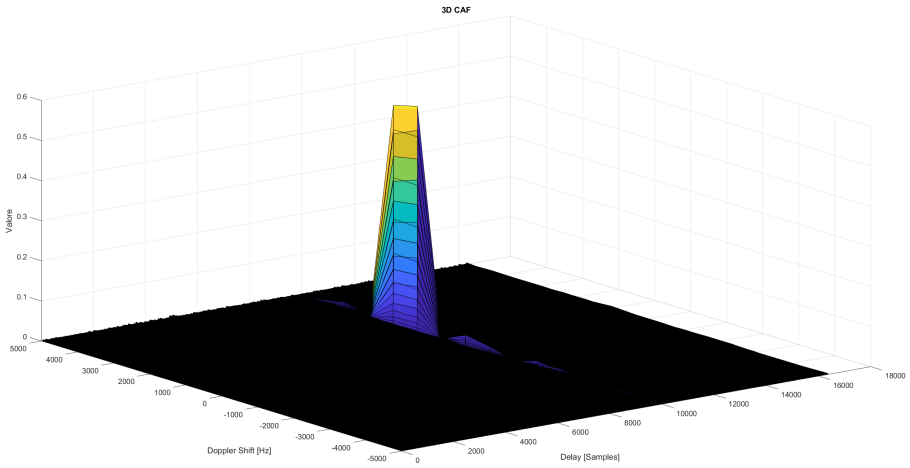


Figure 55: 3D CAF - Parallel Acquisition

The next task ask us to compare the performances of serial and parallel search in terms of processing time so basically computational complexity. We know that, using the serial approach, the computational complexity is proportional to $O(N^2)$ with N being the number of samples. This make sense because a single value of the correlation is obtained at each trial. Passing onto the parallel search, we now this is based on the FFT (Fast Fourier Transform). Its computational complexity is $O\left(Nlog(N)\right)$ so clearly less than the previous case.

Noise effect

In this part we process a new signal that is still ideal but contains noise. Three satellites are contained, 6-18-21. We were able to acquire only the satellite number 6 and 18, while for the satellite number 21 the noise was too high leading to a poor and not successful acquisition of the signal as can be appreciated from the following plots.

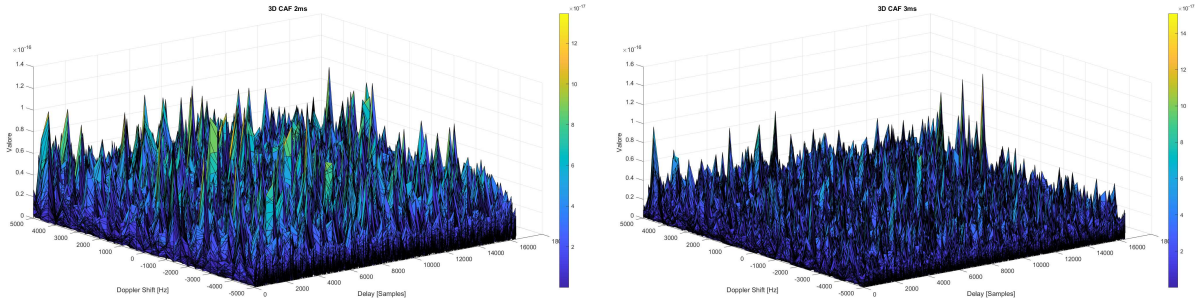


Figure 56: Right: 3D CAF 2ms for PRN 21. Left: 3D CAF 3ms for PRN 21.

Can be clearly noticed that, even increasing the coherent time to 3ms, we are not able to acquire this satellite because the noise floor level is too high and do not allow to correctly localize the peak. A possible solution to improve this situation is to use a non coherent time extension up to 10ms but this will be addressed in a following section.
For satellites number 6 and 18 the situation was better allowing to properly acquire the signals.

Coherent/Non-coherent integration

In this section we were asked to implement, in the parallel acquisition scheme, a non coherent integration time extension and a coherent one. First of all it is worth to mention that, by increasing the time of the acquisition, the noise is more averaged out so we appreciate a reduced noise floor level. We know that, focusing on the coherent accumulation, it is required to properly set the resolution in doppler domain accordingly to the coherent integration time. In particular we use $\Delta f = \frac{2}{3*T_{coh}}$ and we are able to appreciate another know effect of increasing T_{coh}. In particular this lead to a lower value of Δf, so to an higher resolution in the doppler domain, and to a peak shirinking also in the same domain.
Another aspect that it is notable in our results is that, this is a rule of thumb, that grant at worst a loss that is less than 3 dB.

PNR 6 - Coherent accumulation

We report here our results for satellite 6:

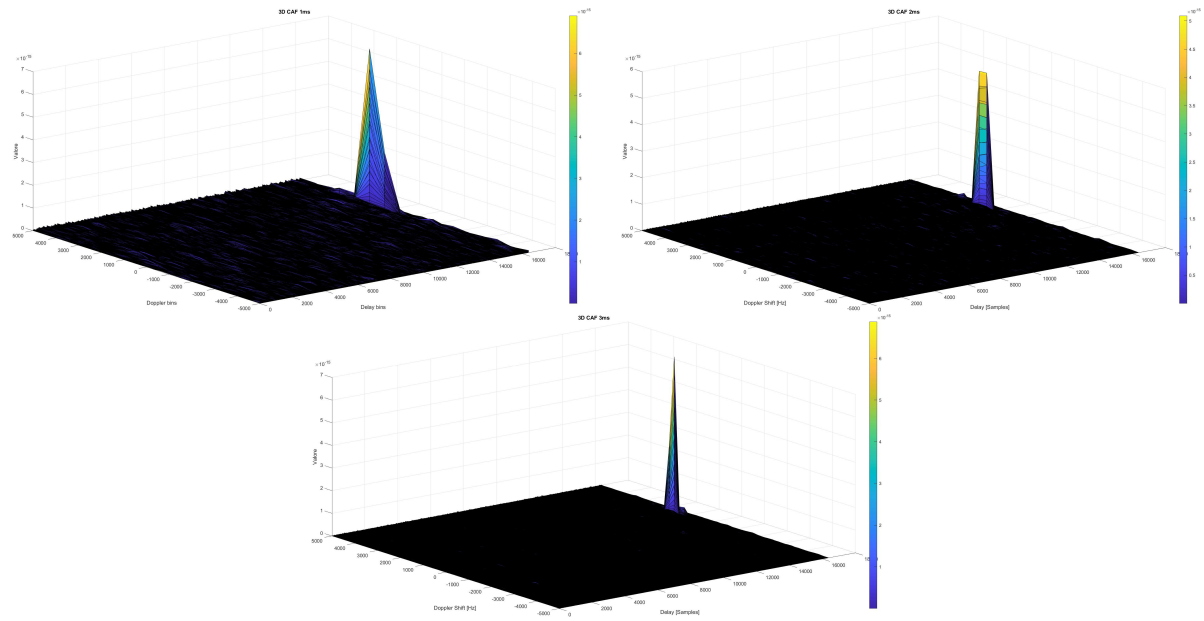


Figure 57: 3D CAF PNR 6 for 1, 2, 3 ms.

The peak shrinking is clearly notable, while the reduction of the noise floor is hard to notice (but it is present) due to the fact this is the better satellite in terms of noise so we start from a already "good" situation.

PNR 18 - Coherent accumulation

We report here our results for satellite 18:

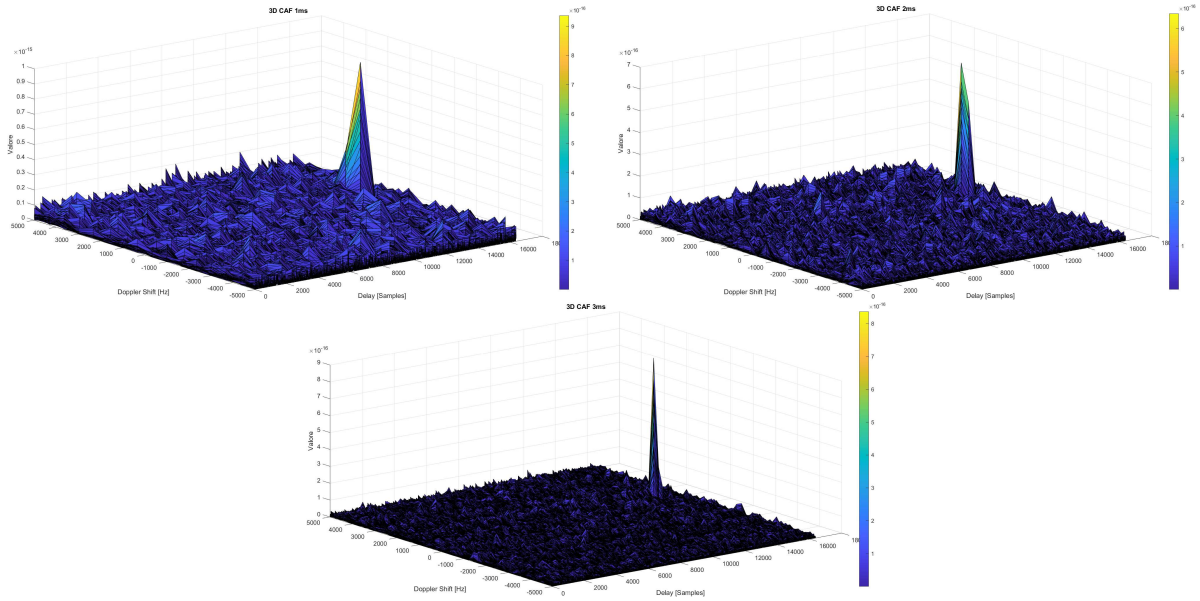


Figure 58: 3D CAF PNR 18 for 1, 2, 3 ms.

With respect to the previous PRN, in this case the noise was higher so the reduction of the noise floor is more graphically appreciable. The peak shrinking due to the increased doppler frequency resolution can be appreciated too.
Regarding the peak value we expect that this will be recuded due to the reduced noise floor level: this actually happened with just some minor displacement due to our use of the rule of thumb.

PNR 21 - Coherent accumulation

These results were already inserted in the 5 section to show that, for PRN 21 and using coherent accumulation, we were not able to acquire the signal.

PNR 6 - Non Coherent accumulation

We use this first section on the non coherent approach to highlight one concept: it suffer of squaring loss so it is less capable of separating the main peak from the noise floor with respect to the coherent one. One the ohter hand it is less heavy under the computation point of view, resulting in a trade off between performances and complexity.
In the following we report the Non Coherent accumulation function:

$$S^2_{y,r}(\bar{\tau}, \bar{f}_d) = \frac{1}{N_c} \sum_{i=1}^{N_{nc}} \left| R_{y,r}(\bar{\tau}, \bar{f}_d) \right|^2$$

Here N_{nc} is the number of coherently integrated and averaged block and it is the parameter that need to be tuned to achieve the required sensitivity.
What is important is also that N_{nc} can be used as a "tool" to improve the peak separation, when the T_{coh} can not be more increased anymore. For example, using GPS, T_{coh} can be at maximum 10 ms: if this is not enough, increasing N_{nc} can solve the situation.
In the following plots it si notable that, using $T_{coh} = 10$ ms and $N_{nc} = 5$, we achieve a better results with respect to the coherent case.

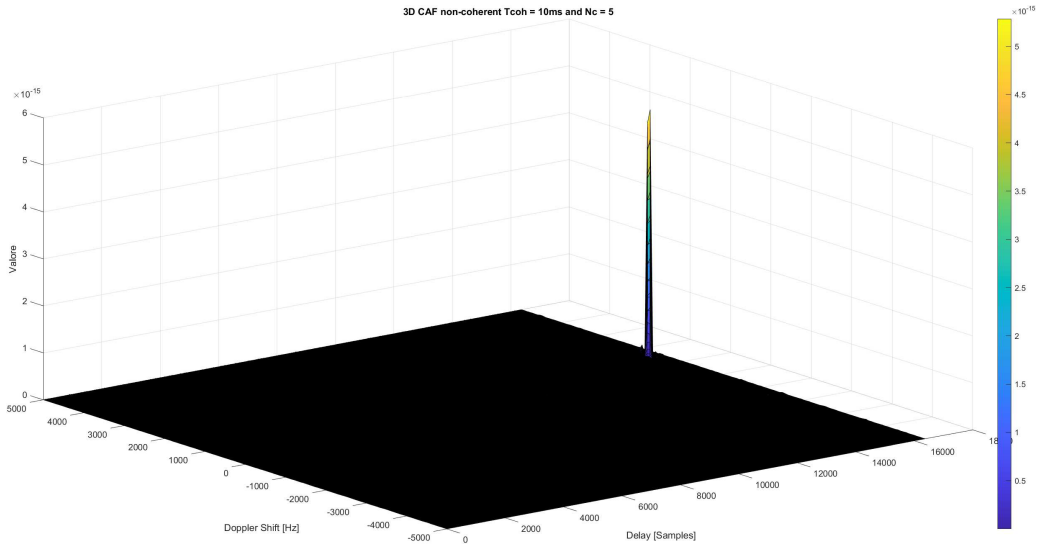


Figure 59: 3D CAF PNR 6 Non Coherent accumulation ($T_{coh} = 10$ ms and $N_{nc} = 5$)

PNR 18 - Non Coherent accumulation

Also in this case the presented concepts are applicable and the following plot is obtained.

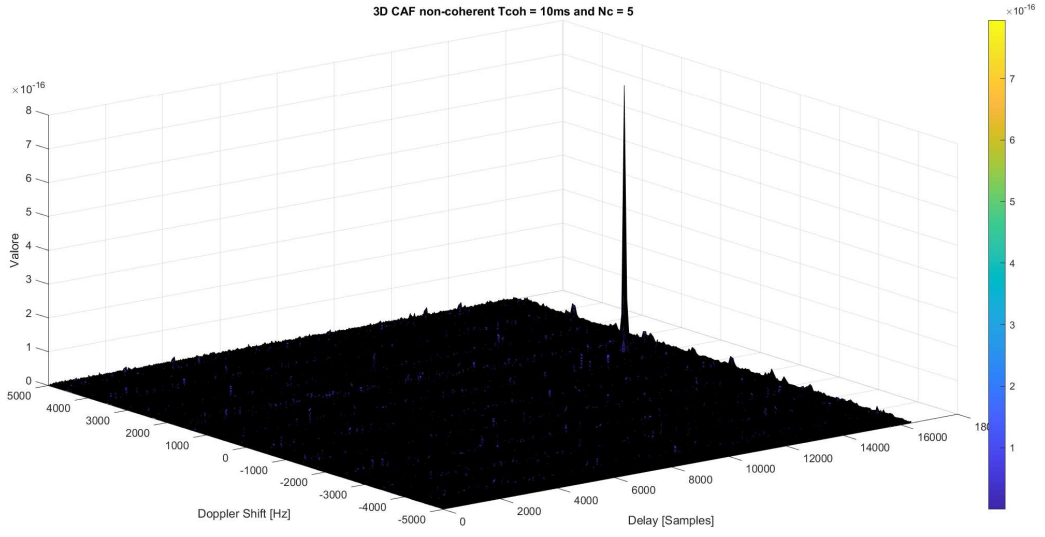


Figure 60: 3D CAF PNR 18 Non Coherent accumulation ($T_{coh} = 10$ ms and $N_{nc} = 5$)

PNR 21 - Non Coherent accumulation

This last case is very meaningful since can be compared with the coherent acquisition using $T_{coh} = 3$ ms as reported here 5.

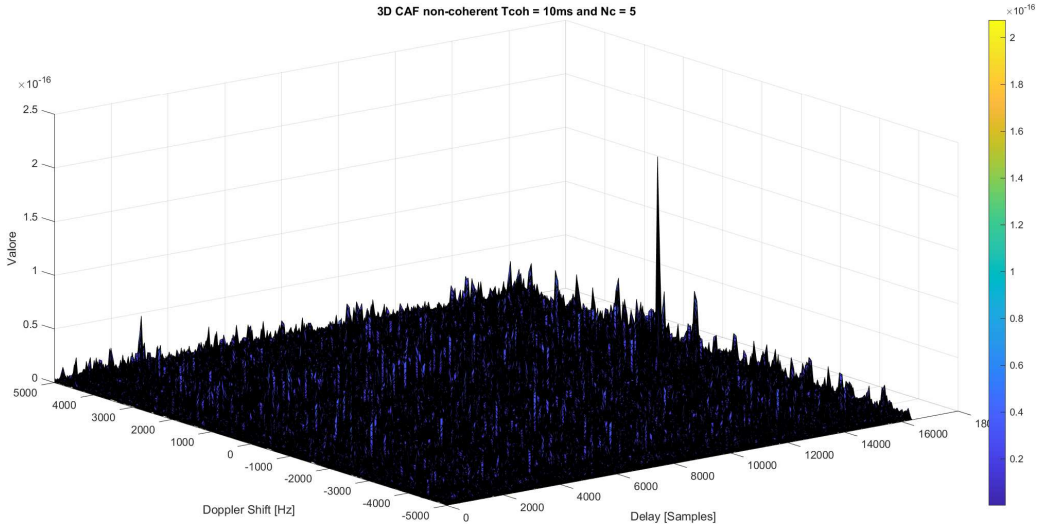


Figure 61: 3D CAF PNR 21 Non Coherent accumulation ($T_{coh} = 10$ ms and $N_{nc} = 5$)

In the coherent case, T_{coh} was too low, not permitting a satisfying noise averaging. In non coherent case, thanks to both T_{coh} and N_{nc} we were able to overcame this issue. It is worth to mention that, if we want a "clearer" plot (so a better $\frac{C}{N_0}$), we can increase only N_{nc} since 10 ms is already the maximum for GPS.

6 Real GNSS Signal

In this last part we deal with real GNSS signals acquired during the fifth training session in 01/12/2023 at 13:30. The collection was made using a patch antenna, a USB front end and a smartphone with a specific tailored application. Practically we used the *live_ data_ collection* collected signals, both a and b, focusing on the GPS constellation.
First of all it is important to mention the location and the atmospheric condition of the day at the collection time: we take the measuremnt out of room 5M in Polito building (45.06435472640277, 7.656608627593542) and the weather was cloudy. In the following two plots we show the location and the skyplot (cut off = 10°) at that specific time.

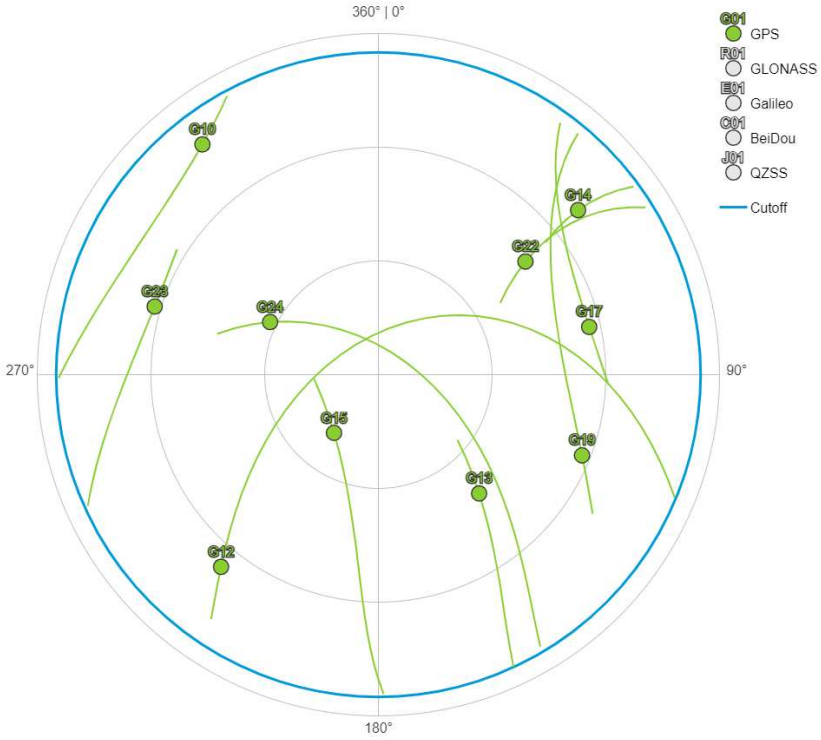


Figure 62: Sky plot (only GPS) at 13:30 of 01/12/2023 for room 5M location.

Figure 63: Room 5M location: 45.06435, 7.65660.

Collection a - Coherent

In this case we perform coherent acquisition with a coherent time of 10 ms to maximize our probability of success. We were able to acquire PRN number 13 and 17 as can be seen here:

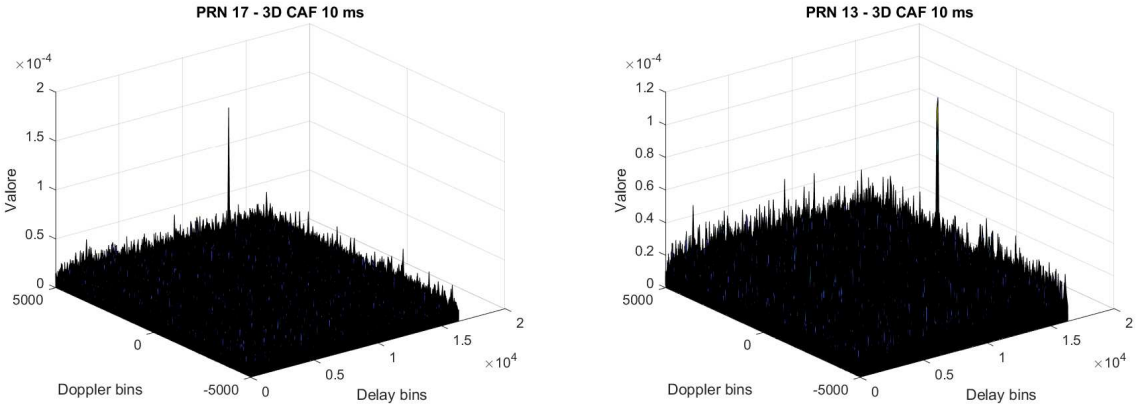


Figure 64: PRN_ 17 and PRN_ 13, coherent acquisition,T_{coh} = 10 ms.

Collection a - Non Coherent

In this case we perform non coherent acquisition with a integration time of 10 ms and N_{nc} = 5. We were able to acquire PRN number 5, 13, 17 and 22 as can be seen below. This improvement is due to the N_{nc} factor that can be tuned accordingly to the specific situation.

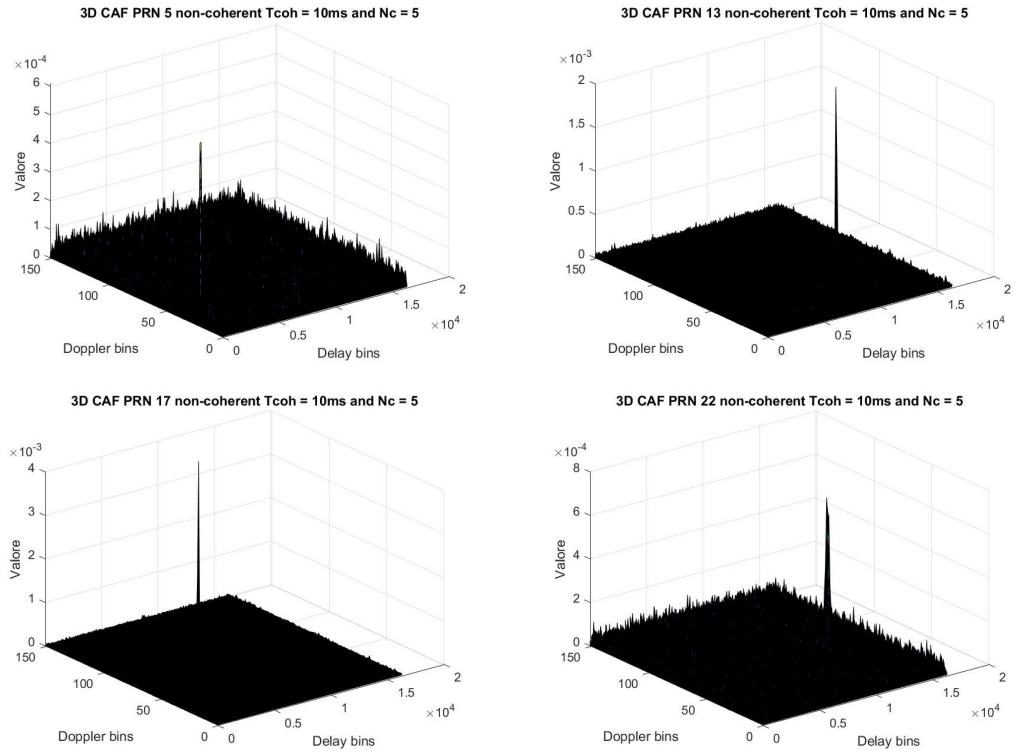


Figure 65: PRN_ 5,PRN_ 13,PRN_ 17,PRN_ 22 non coherent acquisition, T_{coh} = 10 ms and N_{nc} = 5.

Collection b - Coherent and Non Coherent

Here, to not make the report too long, we insert just the comments and the plots can be founded in the uploaded materials. In this case we were able to acquire PRN number 14 for the coherent case and PRN 5, 17, 15, 13 and 22 for the non coherent: this confirm what already stated before about the role of the integration time and N_{nc}. It is worth to remember that we have a quite coarse approximation: in reality measurements a and b were taken with two different groups of students one after the other so referring to a single sky plot for a specific time (13:30) is not so accurate. For sure if we have taken note of the correct time of each group , that in any case was between 13:00 and 14:30, we could have reported a more accurate analysis.