COURSE ENV-542

ADVANCED SATELLITE POSITIONING

LAB 4: TRACKING OF GNSS SIGNALS

DOCUMENT REFERENCE: STUDENT VERSION
AUTHOR(S): VINCENZO CAPUANO, MIGUEL ANGEL RIBOT, CYRIL BOTTERON
ORGANIZATION: EPFL-IMT-ESPLAB
DATE OF PREPARATION: 02/04/2020
REVISION: 1.0

DATE OF LAB: 03.04.2020, 10.04.2020

STUDENT 'S NAME : ALBERIC DE LAJARTE & STÉPHANIE LEBRUN

References

[Borre 2007] Borre, K., Akos, D.M., Bertelsen, N., Rinder, P., Jensen, S. H. (2007). A Software-Defined GPS And Galileo Receiver: A Single-Frequency Approach. Springer.

Acronyms

GNSS	Global Navigation Satellite System
PRN	Pseudo-random Noise
PLL	Phase Lock Loop
DLL	Delay Lock Loop

Goal of the lab

The first goal of this lab is to show how the tracking of GNSS signals is performed, and to give an idea of its implementation issues. The second goal is to show how some tracking (and acquisition) parameters have to be set for specific applications and/or receiver architectures.

The lab is divided in two parts. The first part (Part I) introduces the student to the GPS C/A carrier tracking and code tracking, i.e., the student will implement a typical PLL and a DLL. In the second part (Part II), the student will need to change the value of one or more parameters in order to make the software receiver able to acquire and track a signal with particular characteristics.

Information

For each exercise, there is a folder with the templates for the carrier tracking and code tracking loops to implement and the files of the complementary Matlab functions and sub-functions that you need to perform the full tracking of different signal cases.

Important: Please write your answers in this electronic document and document well your Matlab code as you will also need to provide it in a .zip file named as "lab4_ LastName_FirstName.zip". The deadline to hand out the .zip file and this document without penalty is April 17th before lunch.

PART I: Introduction to GPS C/A carrier and code tracking

In the folder "Part I", you will find all the needed Matlab files to acquire and track the same signal recording `ENV542_GPS_CA_data_capture.bin` that you already used previously in the acquisition lab. This time, you are also going to calculate the corresponding navigation solution. The only incomplete Matlab script is `tracking.m`, where you will have to implement the missing carrier and code loop discriminators.

The SoftGNSS "reduced" receiver for ENV-542

As in the previous acquisition lab, you will also use the SoftGNSS "reduced" receiver. The SoftGNSS receiver is a GPS software receiver implemented in MATLAB based on the open source SoftGNSS v3.0 code developed by Darius Plausinaitis and Dennis M. Akos.

First step: SoftGNSS "reduced" configuration: `initSettings.m`

The function `initSettings.m` creates a data structure with all the settings that the receiver will need to work. To help you, all the fields that you might need to modify are already defined in this file, so you will only need to set the right values.

Please, carefully read the code comments for a description of each field.

Remember to run `initSettings.m` function every time you change some of its parameters.

Example of use: `settings = initSettings;`

GNSS Tracking

The acquisition provides only rough estimates of the frequency and code phase parameters. The main purpose of tracking is to refine these values, track them, and demodulate the navigation data from the tracked satellites.

The figure below represents the simplified scheme (the in-phase and quadrature signals are not represented) used to demodulate the input signal and obtain the navigation message. First, the input signal is multiplied with a carrier replica in order to wipe off the carrier wave from the signal. Then, the signal is multiplied with a code replica, giving as output of this multiplication the navigation message (with a 180 phase shift ambiguity). So the tracking module has to generate two replicas, one for the carrier and one for the code, to perfectly track and demodulate the signal of one satellite [Borre 2007]. Note that the order of the two operations (i.e., carrier wipe-off and code wipe-off) can also be reversed.

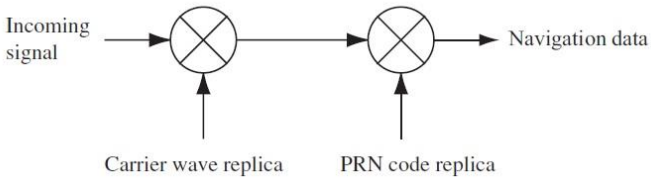


Figure 1: Basic demodulation scheme used to demodulate the navigation message [Borre 2007].

1. Exercise 1: Implement Carrier and Code Loop Discriminators

Figure 2 shows a block diagram of a GNSS receiver **carrier tracking loop**. The first multiplication wipes off the PRN code, and the next multiplications wipe-off the carrier of the input signal. The carrier loop discriminator block is then used to find the phase error (PLL) or the frequency error (FLL) on the local carrier wave replica. The output of the discriminator, which is the phase error or the frequency error (or a function of one of them), is then filtered and used as a feedback to the numerically controlled oscillator (NCO), which adjusts the frequency of the local carrier wave. In this way, the local carrier wave can be a precise replica of the input signal carrier wave [Borre 2007].

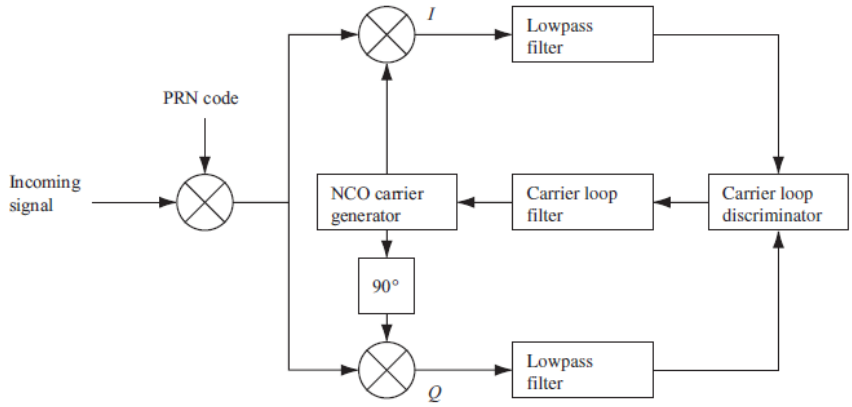


Figure 2: Basic GPS receiver carrier tracking loop [Borre 2007].

After converting the C/A code to baseband, by multiplying the incoming signal with a perfectly aligned local replica of the carrier wave, the goal for a **code tracking loop** is to keep track of the code phase of a specific PRN code in the signal. The simplest code tracking loop in a GPS receiver is a delay lock loop (DLL) called an early minus late (EML) tracking loop. As shown in figure 3, the basic principle of the DLL is to correlate the input signal (output of the multiplication of the incoming signal with a perfectly aligned local replica of the carrier wave) with three replicas of the code, nominally generated with a spacing of ±1/2 chip. After these multiplications, the three outputs are integrated and dumped. The output of these integrations (a numerical value indicating how much the specific code replica correlates with the code in the incoming signal) are inputted to the code discriminator function. The output of the code discriminator function is fed back to the PRN code to adjust its code phase. Figure 4 provides the block diagram of a complete tracking channel on a GPS receiver.

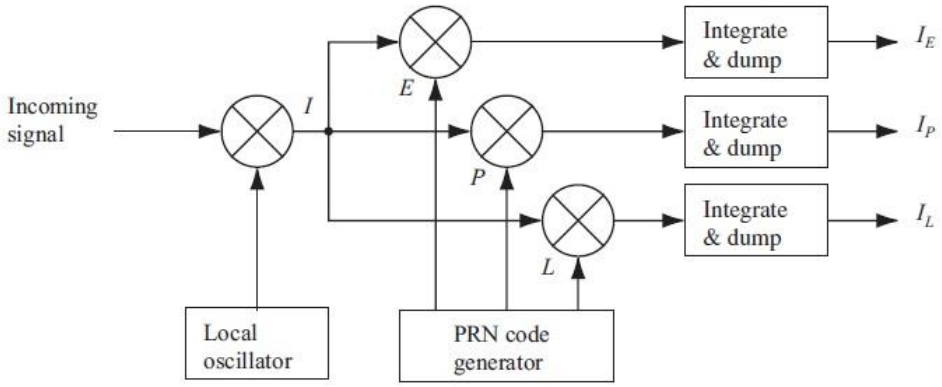


Figure 3: Basic code tracking loop block diagram (the code loop discriminator is not represented) [Borre 2007].

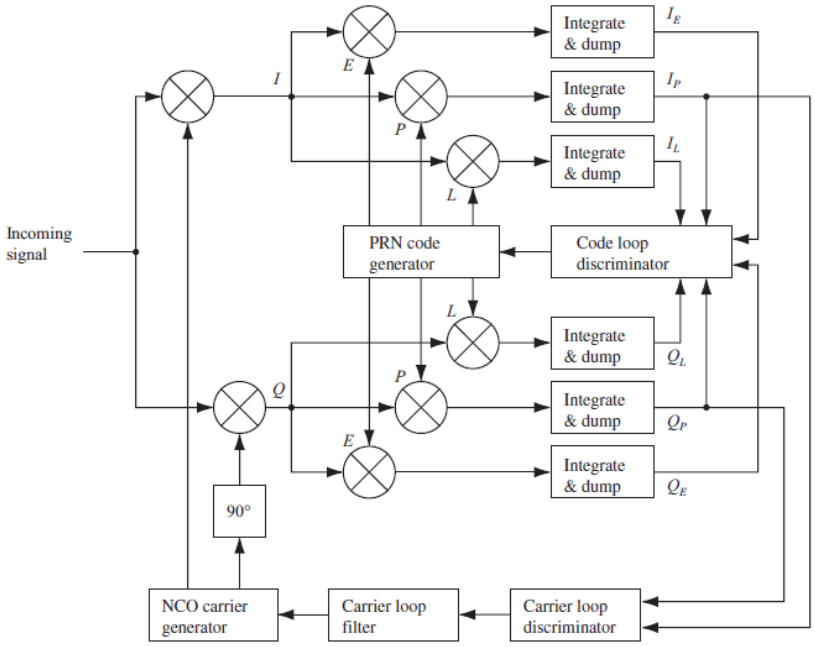


Figure 4: block diagram of a complete code and carrier tracking channel on a GPS receiver [Borre 2007].

1.1 Task 1

In this first task you will implement only one part of the carrier tracking loop: the carrier loop discriminator. In particular, in the file `tracking.m` (in Part I\Kai Borre GNSS SDR reduced_part1 folder) the ATAN discriminator has to be implemented in such a way to provide the frequency error, indicated in the template as `carrError`.

Hints:

- The missing carrier discriminator function should be placed in `tracking.m` where specified in the comments, within section "%% Find PLL error and update carrier NCO".
- The output of the ATAN discriminator is the phase error. Modify it in such a way it gives the frequency error.
- The carrier loop period is equal to $\Delta t = 1$.
- Use the correlators output (e. g. I_E, Q_E, I_P, Q_P, etc.).

1.2 Task 2

Implement the non coherent early minus late envelope (EMLE) discriminator. In particular, in the file "`tracking.m`" the EMLE discriminator has to be implemented in such a way to provide the code error, indicated in the template as "`codeError`" using:

$$E - L$$

where $E = \sqrt{{I_E}^2 + {Q_E}^2}, L = \sqrt{{I_L}^2 + {Q_L}^2}$.

Then normalize $E - L$ by $E + L$ to remove the amplitude sensitivity.

Hints:

- The missing carrier discriminator function should be placed in `tracking.m` where specified in the comments, within section "%% Find DLL error and update code NCO".
- Use the correlators output (e. g. I_E, Q_E, I_P, Q_P, etc.).

1.1 Task 3

Run the SoftGNSS "reduced" receiver by performing the following steps:

- select the "Part I" folder as your working Matlab folder
- add to Path the folder "geoFunctions" and "include"
- define the general settings for the SoftGNSS "reduced" receiver (i.e. run >>`settings = initSettings;` in your Matlab Command Window)
- run `postProcessing.m`. If Task 1 and Task 2 have been completed correctly, you will acquire and track the signal recording `ENV542_GPS_CA_data_capture.bin` and plot:
  - the acquisition results (that you already got in the previous lab)
  - the tracking results (defined in `plotTracking.m`), that contain:
    - ✓ Discrete-Time Scatter Plot, which plots the in phase and quadrature prompt correlator output I_p and Q_p

- ✓ Bits of the navigation message, obtained by plotting I_p over time
- ✓ Raw PLL discriminator, which corresponds to the carrier error (in frequency) at the output of the PLL discriminator.
- ✓ Raw DLL discriminator, which corresponds to the code delay error (in chips) at the output of the DLL discriminator.
- ✓ Correlation results.

Q.1 By observing the tracking plots, what are the tracked satellites?

The acquired satellite numbers are 03, 05, 06, 11, 12, 16, 18, 19,21, 22, 30, 31.
All of them are then successfully tracked, even if it took a little bit more time for the synchronization to start

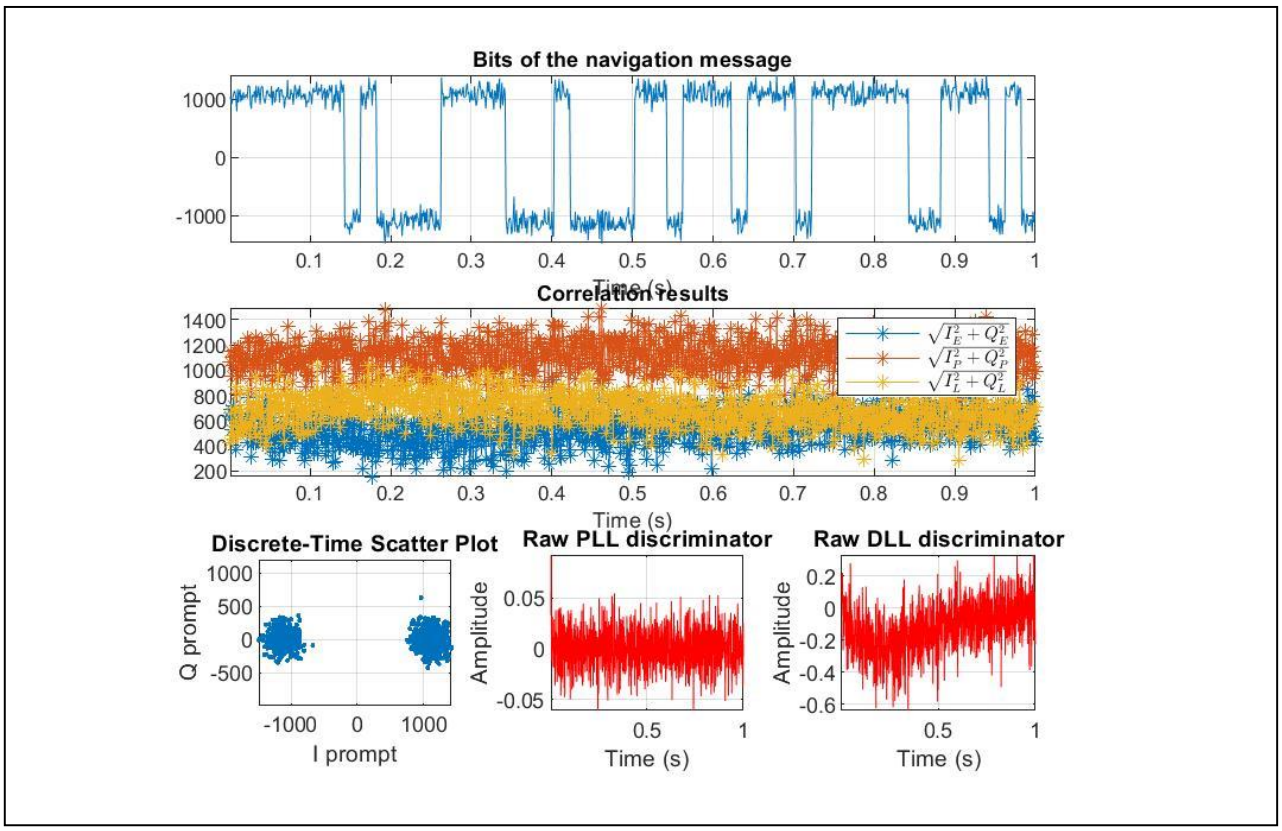
Q.2 What is the impact of correctly tracking a channel on the above described plots (i.e., by considering each tracking plot for a given satellite, tell how you can see that it is tracked with success?). Paste here the tracking plots for one satellite and comment them.
We can see that tracking was successful for satellite 30 for instance, thanks to following plot characteristics:

- The correlation between the satellite signal and the prompt code is the highest, while the early and late codes have similar correlations on average. This means the incoming signal is synchronized with the prompt code and note the early or late. The more the correlation with the prompt code is above in the graph, the more precise the synchronization.

- On phasor diagram: The values for the quadrature arm are clustered around 0, and the values in the in-phase arm are maximum. So the phase error is minimum, the incoming signal is synchronized with the in-phase component. We have data points on one side or the other because of the 180° phase ambiguity of the arctan discriminator.

- On PLL diagram: the discriminator's output corresponds on the error that should then be filtered and corrected by the oscillator. Here we can see that its value converges to 0, so it means the error is null, the receiver is locked on the satellite's phase.

- On DLL diagram: as in previous diagram, the discriminator value converges to 0, the receiver is locked on the satellite's delay.



Please, validate your results with the instructor before moving to **Exercise 2**.

2. Exercise 2: Impact of the early minus late correlator spacing

2.1 Task 1

Change the (E-P) DLL correlator spacing in `initSettings.m` from 0.5 chip to 0.3 and to 0.7. Quantify your observations by filling the following table.

	Correlator spacing 0.5 (chips)	Correlator spacing 0.3 (chips)	Correlator spacing 0.7 (chips)
Standard deviation of DLL discriminator for channel 1 (PRN 31)*	0.1423	0.0724	0.2965

*Use only the last 0.1 s of tracking output data for computing the std.

Q.3 What do you observe? Explain the outcomes of the results. What is the effect of making smaller or bigger the correlator spacing?

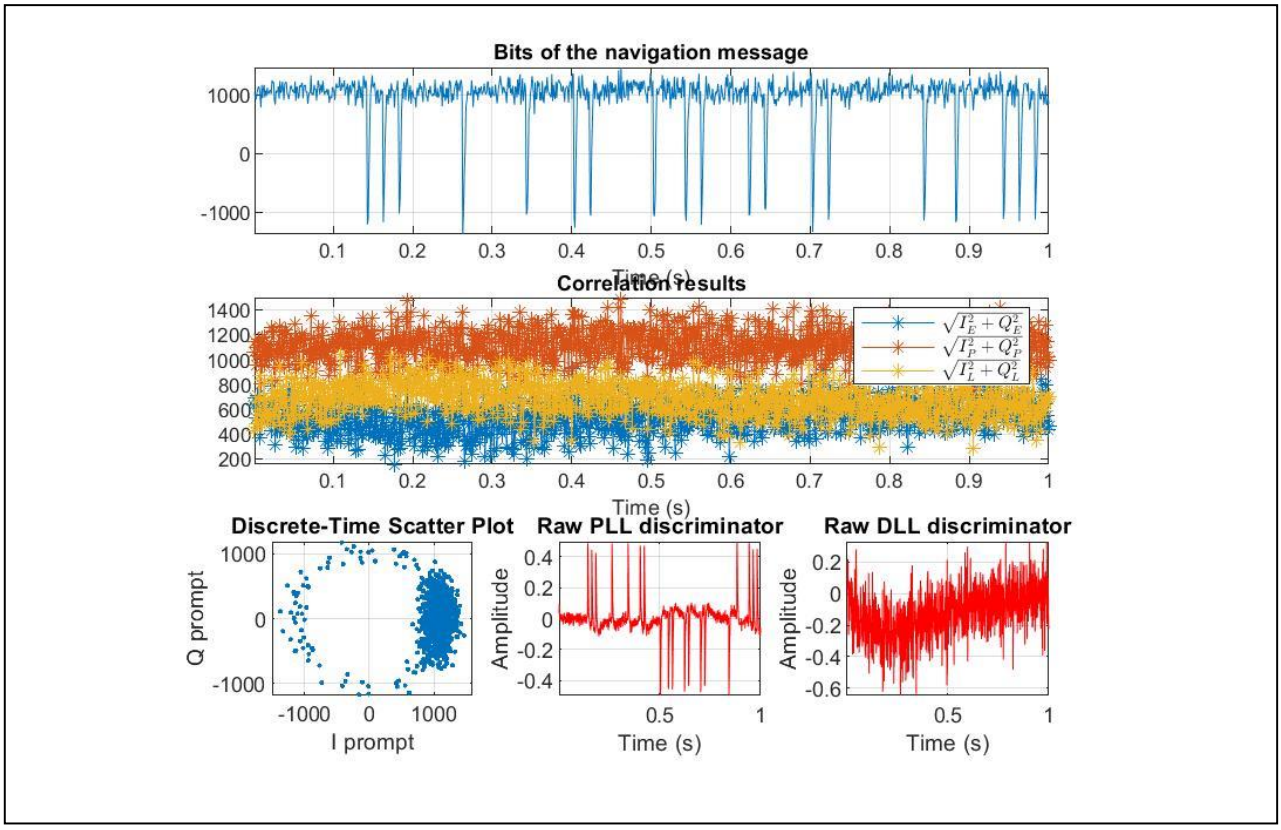
We can observe that the standard deviation grows with the spacing. A small correlator spacing gives higher accuracy and is better for multipath mitigation. On the other hand, a larger correlator spacing handles better dynamics and noise.

3. Exercise 3: Change the Carrier Loop Discriminator

The ATAN2 discriminator is the only one that remains linear over the full input error range of $\pm 180°$, that means it tracks the full four quadrant range of the input signal.

1.3 Task 1

In Exercise 1 you implemented the ATAN discriminator, that is two quadrant arctangent; replace it with the ATAN2 in `tracking.m` and (setting the DLL correlator space to 0.5 as at the beginning) run `postProcessing.m`. Paste here the tracking plots for the same satellite you considered in **Q.2**.
(We considered channel 3, PRN30)



Q.4 Is there any difference with the tracking plots obtained by using the ATAN carrier discriminator? If yes, why?

Yes, 2 plots in particular are different:

- Plot of the phasors, the data points take values over the 4 quadrants now. They are more clustered on the positive side of I prompt than with the other discriminator, but now we also have data points with a Q value different than 0.
- Plot of PLL discriminator: we see that the discriminator takes multiple times a non-zero value over the time window. This is due to transition bits occurring. A bit transition involves a 180° phase shift occurs, and the system will try to correct this.
  The Costa arctan discriminator is insensitive to transition bit on the other hand, so we didn't observe this behavior previously.

4. Exercise 4: Calculate the navigation solution from the tracking results to validate that software receiver is functioning correctly

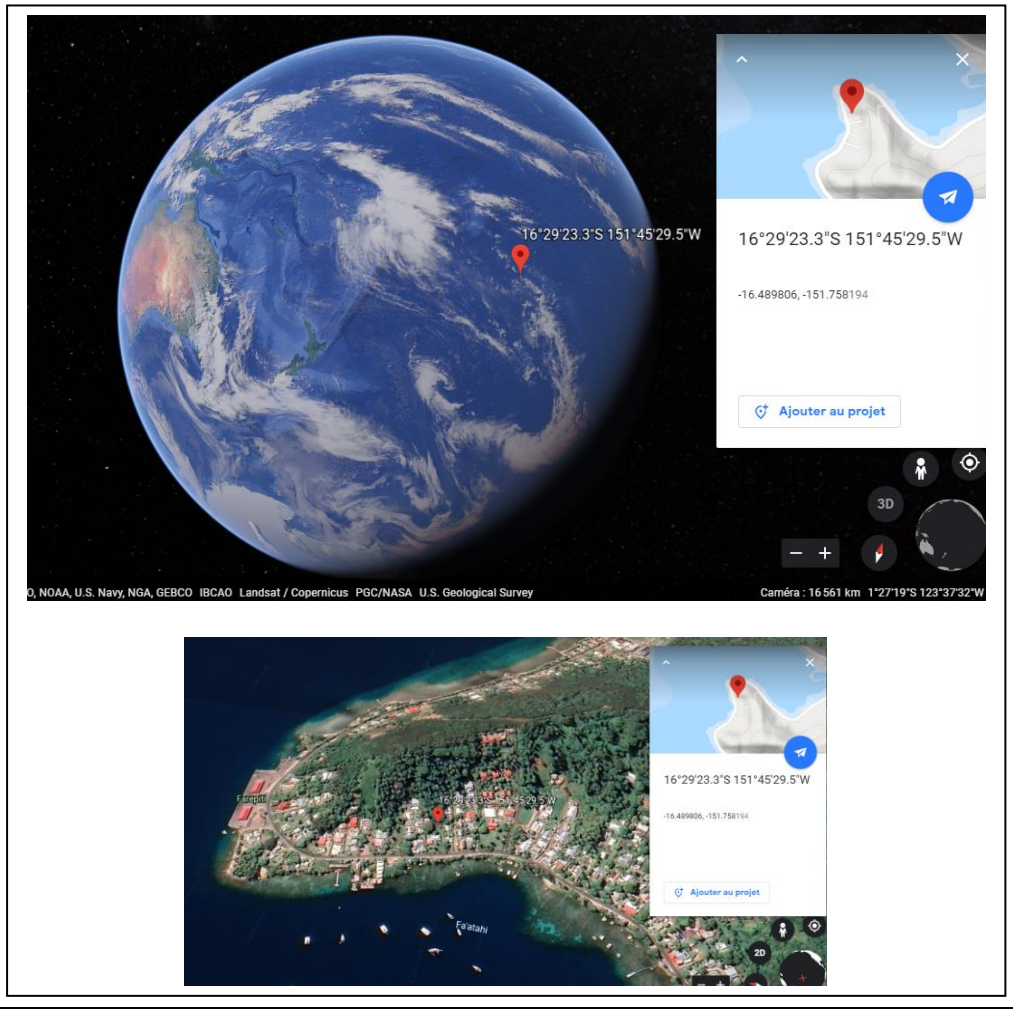
1.4 Task 1

Follow these steps:

- Following the results of Exercise 2, set the proper carrier discriminator (*ATAN* or *ATAN2*) in `tracking.m`.
- Uncomment the section "%% Calculate navigation solutions" in `postProcessing.m` and run it.
- Check the resultant navSolution variable in your Matlab workspace.
- Fill the following table, by considering the mean values over time.
- Paste here a Google Earth snapshot of the location corresponding to the obtained latitude, longitude and altitude.

Note: Consider the main values over time and a position error within ± 50 m, due to range errors and GDOP.

Latitude	Longitude	Altitude
-16.4898	-151.7582	-25.7791



PART II: GPS C/A Acquisition and Tracking for specific signal case

In this exercise, you will need to acquire and track a different recording of GPS L1 C/A signals embedded in `ENV542_GPS_CA_data_capture2.bin`, and to calculate the corresponding navigation solution. In the folder "Part II" you will find this new recording as well as all the necessary Matlab files that you will need. However, in order to acquire and track with success this new signal recording, one or more default acquisition or/and tracking parameters defined in `initSettings.m` has/have to be changed as the recorded user scenario is not the same as before.

1. Exercise 1: Set properly one or more unknown parameters

4.1 Task 1

Using the standard initSettings parameters in `initSettings.m`, run the SoftGNSS "reduced" receiver by following these steps:

- select the "Part II" folder as working folder in your Matlab
- add to Path the folder "geoFunctions" and "include"
- define the general settings for the SoftGNSS "reduced" receiver (i.e. run `settings = initSettings` in your Matlab Workspace)
- run `postProcessing.m`.

Q.1 How many satellites are acquired? How many satellites are tracked? Is it possible to calculate the navigation solution?

Only two satellites (PRN 18 and 19) were acquired and tracked. It is thus not possible to calculate the navigation solution, as we need at least four satellites to do so.

4.2 Task 2

The value of some parameter(s) in `initSettings.m` has/have to be changed in such a way the receiver will be able to acquire and track 7 satellites. Identify the parameter(s) that has/have to be changed, set their value(s) properly and explain why.

Hints:

- Try increasing the coherent acquisition time from 1 ms (default time) to 10 ms and evaluate the acquisition plots to infer what parameter/parameters you need to change in the initial setting.

By increasing the coherent acquisition time from 1 ms to 10 ms, we are able to detect another satellite signal (PRN 15), without being able to track it. By looking at its ambiguity function we see that the received signal from satellite 15 has a doppler shift of -6900 Hz, which is at the limit of the ± 7 kHz range that we search. By increasing this range, we are able to track the satellite 15 completely.
To detect more satellite, we can continue increasing this range, as we seem to be in a situation where high doppler shift can be expected. With a total range of 48 kHz, we are able to detect 4 more satellites (PRN number 6, 9, 16, 22), however their large doppler shift doesn't allow the

tracking to work properly. Looking at the acquisition result, we see that the signals are all received clearly, so increasing the integration time won't help much.
It is thus on the tracking parameters that we should work on. From the PLL graph, we see that the carrier tracking loop seems to converge, so we should adjust the code tracking loop parameters, which is confirmed by the correlator output graph which shows that the incoming signal is not synchronized with the prompt signal. Thus, by increasing the bandwidth of the DLL from 2 Hz to 20 Hz, we can correct the DLL to take into account the increased integration time of 1 ms to 10 ms.
This final change allows to track correctly the 7 satellites.

4.3 Task 3

For the satellite with the lowest PRN you tracked, the raw DLL settling time is approximately 0.4 s to remain within an amplitude error of 0.4. Modify one or more parameters in order to reduce the settling time to 0.3 s remaining within approximately the same amplitude error, without preventing the tracking lock of any satellites. Report here your changes and the resultant raw DLL plot.

The satellite 6 is the one with the lowest PRN number. To reduce the DLL settling time, we can either increase its bandwidth, or increase its damping ratio[1]. However, we can see on figure 5 that the DLL discriminator is already converging in less than 0.4 ms, with an amplitude error of around 0.2.
Figure 6 shows the result with a damping ratio of 0.99 and a bandwidth of 40 Hz, which slightly improve the convergence speed of the DLL

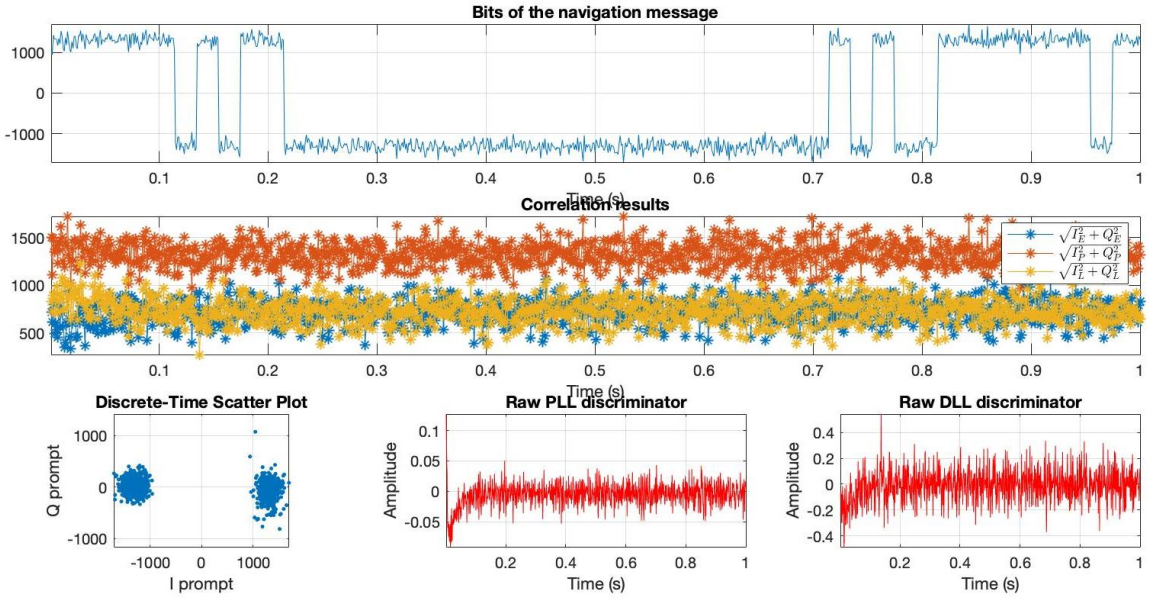


Figure 5: Tracker plots of satellite number 6, with a DLL bandwidth of 20Hz, a damping ratio of 0.7 and a total integration time of 10 ms

[1] Tomé G, Post-processed acquisition & tracking of GPS C/A L1 signals, Universidade de Lisboa

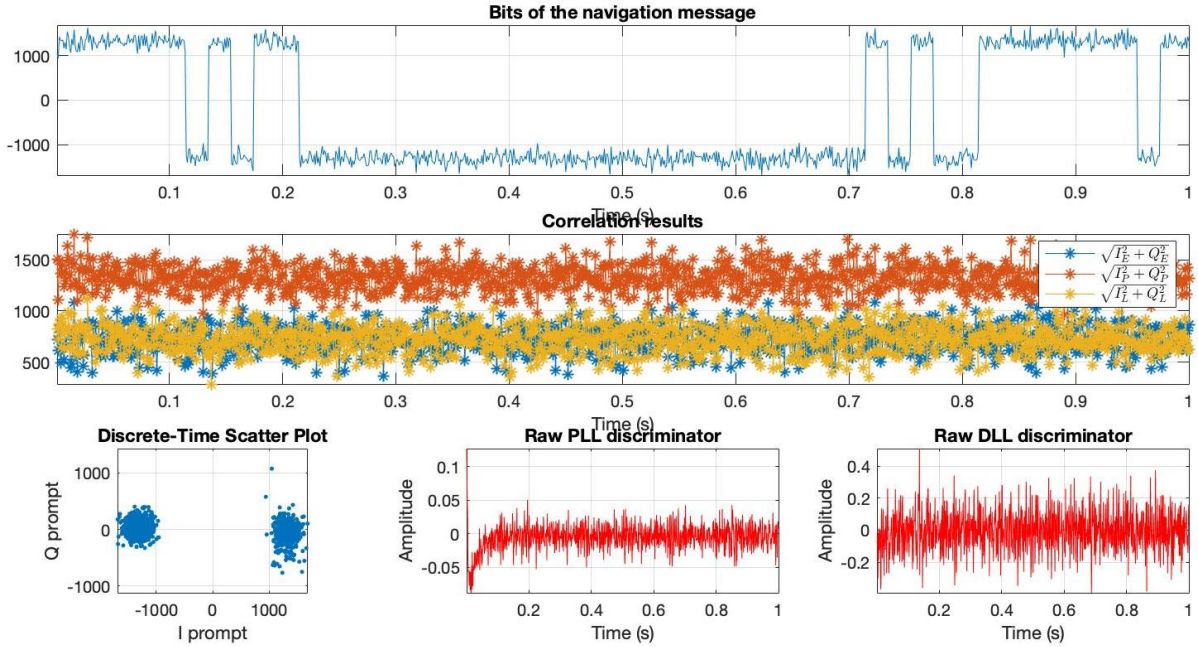


Figure 6: Tracker plots of satellite number 6, with a DLL bandwidth of 40Hz, a damping ratio of 0.99 and a total integration time of 10 ms

5. Exercise 2 Calculate the navigation solution from the tracking results to validate the software receiver functioning

1.5 Task 1

Follow these steps:

- Uncomment the section "%% Calculate navigation solutions" in `postProcessing.m` and run it.
- Check the resultant navSolution variable in your Matlab Workspace
- Fill the following table, by considering the mean values over time.
- Paste here a Google Earth snapshot of the location corresponding to obtained latitude, longitude and altitude.

Note: Consider the final estimated values.

Latitude	**Longitude**	**Altitude**
0.6125°	-136.8215°	592.89 km

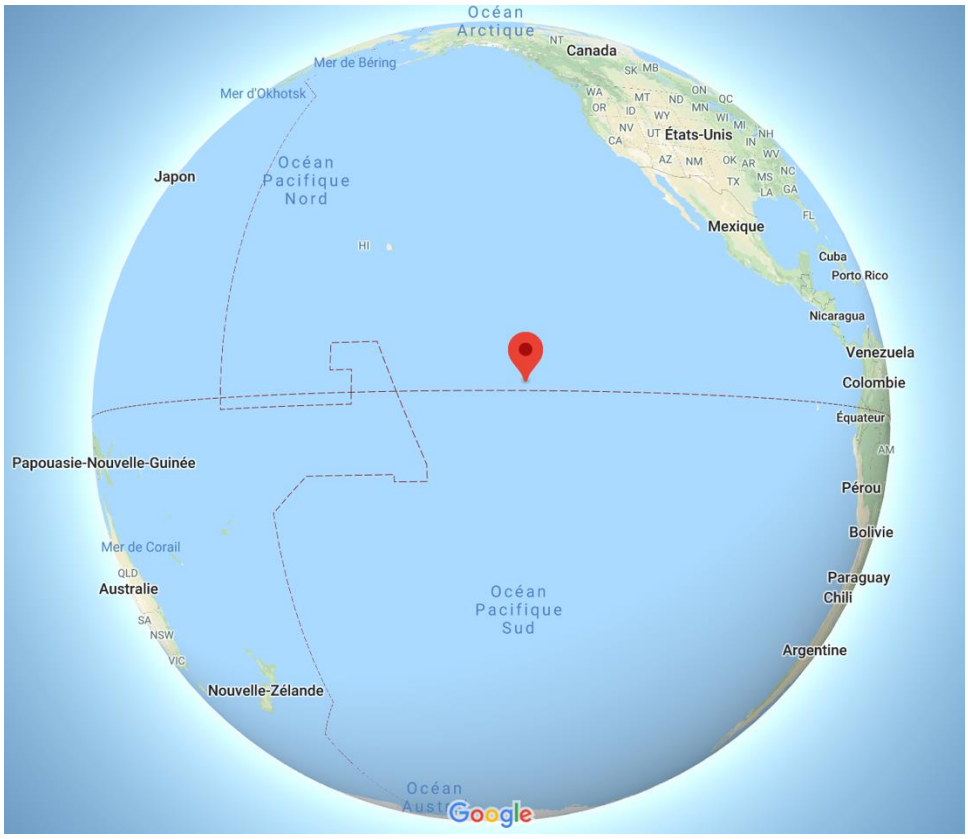


Figure 7: Snapshot of the GNSS receiver

Q.2 By considering the mean position of the receiver, can you explain why you needed to modify one or more parameters setting?

From these coordinates, we can see that the receiver is actually a satellite in Low Earth Orbit around the equator. This means that the receiver is moving at a velocity of around 7.5 km/s with respect to the ground, so at a very high velocity with respect to some GNSS satellite orbiting in the other direction. This result in a significantly higher doppler effect than expected.