Course ENV-542

Advanced satellite positioning

Lab 3: Acquisition of GNSS signals

Document reference: student version

Author(s): Miguel Angel Ribot, Jérôme leclère, Cyril Botteron

Organization: EPFL-IMT-ESPLAB

Date of preparation: 19/03/2020

Revision: 3.1

Date of lab: 20.03.2020 / 03.04.2020

Student ’s name : Stéphanie Lebrun, Albéric de lajarte

1. Acronyms

CAF Cross Ambiguity Function

FFT Fast Fourier Transform

GNSS Global Navigation Satellite System

PRN Pseudo-random Noise

IF Intermediate Frequency

1. Goal of the lab

This lab exercise has two main goals. The first goal is to help you understand how the acquisition of GNSS signals is done, and to give you an idea of the processing required. The second goal is to highlight some of the difficulties and challenges encountered during the acquisition of the GNSS signals.

The lab exercise is divided in two parts: the first part introduces you (the student) to the GPS C/A codes correlation properties, while in the second part you will implement a simple acquisition scheme.

1. Information

For each exercise, we provide you a folder with the files to generate the GNSS codes and the templates for the functions to implement.

The Gold codes are 1023-chip long, i.e. 1 ms with a chipping rate of 1.023 MHz.

For a static user, the maximum carrier Doppler for the GPS L1 C/A signal is usually considered to be ± 5 kHz, which means a maximum code Doppler of ± 3.2 Hz.

**Important: The files for each exercise have been stored in individual folders. When working on an exercise, remember to set its folder as your Matlab current working folder before running any of the script files. If you don’t to do so you are likely to face naming conflict problems.**

Here are some useful MATLAB functions:

* load : to load variables from a .mat file,
* plot : to display a 2D graph,
* mesh : to display a 3D graph,
* length : to determine the length of a vector,
* size : to determine the size of a vector or a matrix,
* find : to perform a search inside a vector or a matrix,
* max : to find the value and the position of a maximum inside a vector,
* fft : to perform the fast Fourier transform (FFT).

**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 “lab3\_LastName\_FirstName.zip”. The deadline to hand out the .zip file and this document without penalty is one week after the end of this lab, i.e., on April 3th before lunch.**

PART I: Introduction to GPS C/A codes correlation

1. Exercise 1: Circular cross correlation

The goal of this exercise is to help you understand the properties of the C/A Gold codes used in GPS signal processing. The correlation properties of the C/A codes allow to measure the delay difference (or code phase) between a locally generated replica and the received signal. In addition, they are fundamental to detect the presence of a signal, or, in other words, “to acquire” the signal.

* 1. Task 1

In this first task you will create a function that computes the circular cross-correlation between two signals of the same length, in this case, two generated PRN codes.

To start, use the function generateCAcode() to generate one period of the C/A Gold codes for satellites 4 and 12. Then, create a function computeCorrelation() that will compute the circular cross correlation between the two PRN codes.

The circular cross-correlation can be computed as:

|  |  |
| --- | --- |
|  | Figure 1 : Illustration of the circular correlation between two PRN codes (the values are in chips) |

The template can be found in the file computeCorrelation.m, with the following prototype:

function corr = computeCorrelation(signal\_1, signal\_2)

Hint:

* Use the function circshift(vector, shift]) to implement the circular shifting.

Finally, test your computeCorrelation.m by:

* Computing the auto-correlation for PRN code 12  
  (i.e. computeCorrelation(PRN\_12,PRN\_12)) and plotting the results obtained.

Une image contenant capture d’écran

Description générée automatiquement

* Computing the cross-correlation for PRN codes 4 and 12  
  (i.e. computeCorrelation(PRN\_4,PRN\_12)) and again, plotting the results obtained. Use the same scale as in previous plot (e.g. Y-axis [-100, 1100]).

Une image contenant capture d’écran

Description générée automatiquement

**Q.1** Observing your plots, what are the possible values for the correlation of the C/A Gold codes?

When the two signals correlate, we expect a value of 1023, which correspond to the length of the signal.

When the two signals don’t correlate, we should expect a zero mean noise, within the interval [-65; 63]

**Q.2** The Gold codes are selected as spreading sequences for the signal because of their “interesting” characteristics. The most important characteristics of the C/A codes are their correlation properties. From your plots and the course materials, can you state the two important correlation properties?

The first interesting characteristic is the autocorrelation of the signal with itself. This means that when the signal is aligned with itself, the autocorrelation will produce a peak of amplitude 1023 at zero shift. This allow to calculate the relative shift between the received signal and the receiver code for the computation of the pseudo-range.

The second characteristic is the orthogonality of the different C/A code between themselves, which allow the receiver to distinguish the different satellites, as the cross-correlation of two different C/A code will produce zero mean pseudo random noise with no visible peak.

* 1. Task 2

The file Ex1T2\_mistery\_signal.mat contains a signal composed of the sum of two GPS L1 C/A Gold codes (PRN between 1 and 10) buried in a Gaussian noise, as shown by Eq 1. The signal contains 1 ms of data. The PRN number of these two codes is unknown, as well as their delay (or code-phase) (i.e. and both in this case).

|  |  |  |
| --- | --- | --- |
|  |  |  |

The signal has been sampled with a sampling frequency = 6.5 MHz.

The goal of this task is to determine the PRN codes present and their delays.

In the file Solution\_ex1.m, call the function computeCorrelation() to determine visually the PRN codes present in the file Ex1T2\_mistery\_signal.mat and their phase. Fill the following table and add a plot simultaneously showing the correlation results for all the PRNs (1 to 10).

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

**Hints:**

* Use the function generateGoldCodeSampled() to generate the code sampled.
* Use the function tic and toc to measure the time.

|  |  |  |
| --- | --- | --- |
|  | First code | Second code |
| PRN number | 1 | 8 |
| PRN Delay  (samples) | 4086 | 3368 |

**Q.3** What is the processing time to obtain the correlation for all 10 codes?

The processing time was measured to be 1.131902 seconds. This time also includes the time to find the PRN number of the two codes

**Q.4** From your previous results, determine the relative time difference (code delay in s) between the local generated codes and the PRN codes in Ex1T2\_mistery\_signal.mat*.*

**Q.5** What is the width of the main peak (in samples)? How do you explain this?

The full width at half maximum of the peak is 6 samples for PNR number 1 and 8. This is because our sampling frequency fs = 6.5 Mhz is approximately 6.4 times higher than the chip frequency, which means that a shift by ±3 samples still result in a significant correlation as the chip shift is less than one.

1. Exercise 2: Fast Correlation using FFT

The time required to compute the cross-correlation between PRN codes can be relative long. Minimizing this computation time is very important because, during the acquisition stage, the receiver needs to perform correlations with multiple PRN codes with several different carrier Doppler frequencies. In this exercise, we will make a function to perform the correlation faster, and find the codes present in the file Ex2T2\_mistery\_signal.mat, which is a signal composed of the sum of several GPS L1 C/A Gold codes (only PRNs between 1 and 10) buried in white Gaussian noise. The sampling frequency is still 6.5 MHz, and the total duration of the signal is one C/A code period, i.e. 1 ms.

The parallel code phase[[1]](#footnote-1) search acquisition method is used to parallelize the code delay search. Instead of multiplying the input signal with a PRN code with each different code delays to obtain as done before, the circular correlation can be computed more efficiently in the frequency domain as:

where is the correlation vector.

Remark: represents the complex conjugate of , that can be computed using the conj() MATLAB function.

Create a function that computes the circular correlation between the input signal and a locally generated PRN code using the FFT. Use the template FFTCorrelation.m, with the following prototype:

function corr = FFTCorrelation(signal\_1, signal\_2)

**Q.6** In the file Solution\_ex2.m, use the function FFTCorrelation() to determine the PRN codes present in the file Ex1T2\_mistery\_signal.mat and their code phase. Do you find the same result as in Exercise 1?

No. The PRN codes we find are the ones from satellites 4 and 9. The code delays are respectively

0.000736 s and 0.00049477 s.

**Q.7** Is the computation faster than in Exercise 1? By which factor?

Yes, about 6.5 times faster.

PART II: GPS C/A Acquistion

The SoftGNSS “reduced” receiver for the ENV-542’s course

In this second part of the lab, you will be introduced to the SoftGNSS “reduced” receiver. 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.

In a non-assisted GPS receiver’s cold-start, we need to wait at least 37 seconds until the ephemeris information can be successfully retrieved. Only after that the receiver is able to compute the navigation solution. In order to store 37 seconds of data recorded at a sampling frequency 6.5 MHz, we would need hundreds of megabytes. In addition, depending on the computer, SoftGNSS 3.0 might need several minutes to track 4 channels during 37 seconds. This is not very practical for a course lab exercise.

To overcome those limitations, we decided to use a special stripped down version of the original SoftGNSS software to be used in this course. In this version of the software, we are able to reduce the required processing time significantly by using some pre-loaded information (i.e. satellite ephemeris, subframe start position and TOW) specific for the recorded data.

First step: SoftGNSS “reduced” configuration: initSettings.m

The function initSettings() creates a data structure with all the settings that the receiver will need to work. To help you, all the fields specifying the required settings have been already defined in this file, so you will only need to modify their values when needed.

Please, read carefully the code comments for each field’s description.

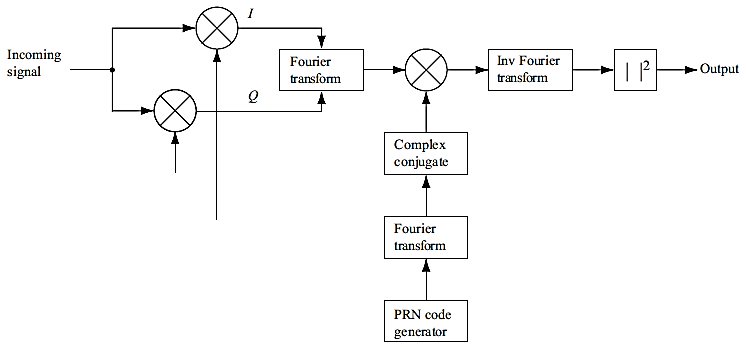
Remember to run initSettings() function every time you change some of its parameters.

Example of use:settings = initSettings;

1. Exercise 3: The acquisition function

The goal of this exercise is to implement acquisition\_1ms(), a simple signal acquisition function to be used with recorded GPS data, and as part of our SoftGNSS receiver.

We want to implement the parallel code phase acquisition scheme introduced in Exercise 2 following the scheme depicted in Figure 2.



**Signal\_1**

**fft()**

**abs()^2**

**ifft()**

**conj()**

**fft()**

**generateGoldCodeSampled()**

Figure 2: Block diagram of the parallel code phase search algorithm to implement in acquisition\_1ms().   
Figure Source: Kai Borre et. al. “A Software-Defined GPS and Galileo Receiver”.

You must complete the acquisition\_1ms() template function provided:

function acqResults = acquisition\_1ms(settings, longSignal),

where “settings” is the settings structure generated by initSettings() and longSignal vector contains data input signal.

Do not run acquisition\_1ms()directly, but instead use the provided script Main\_Ex3.m to call your function. Remember to set “Exercise 3 – Acquisition” as your MATLAB’s working folder before running Main\_Ex3.m

**Follow these steps:**

1. You should first create a smaller vector signal\_1 from longSignal, containing the first 1 ms of data.
2. Generate a 1 ms local replica of the PRN code sampled at 6.5 MHz using generateGoldCodeSampled() function.
3. As we have seen during the course, the carrier frequency of the received GNSS signals is shifted due to the Doppler Effect, as a consequence of the relative motion between the receiver and the GNSS satellites. The Doppler frequency shift is a-priori unknown by the receiver. It is part of the acquisition process to find it out by assuming different frequency shift values and trying to detect the signal.

You must generate two carrier signals with a phase difference of 90º (sin and cos) of 1 ms length. The carrier must have a frequency corresponding to the intermediate frequency (IF) the frequency step (settings.acqFreqstep) according to the considered search band (settings.acqSearchBand).

**Hint**: you can generate 1 ms carrier signals in MATLAB, use sin(2\*pi\*fi\*Ts\*n); where fi is the receiver’s IF plus the current frequency offset, Ts is the sampling period , and n is an array with numbers from 0 to 6500. Same for cos().

1. Multiply signal\_1 with these local replicas of the carrier frequency to obtain the **I** (in-phase) and **Q** (quadrature) signals.
2. Just like in Exercise 2, perform:
3. Finally, take .

What you have now is just the code autocorrelation function (ACF) for a certain Doppler frequency shift. You must repeat steps 3 to 5 for all the frequency shifts that you are considering to obtain the Cross-Ambiguity Function (CAF), i.e.:

for frqBinIndex = 1:numberOfFrqBins

**Steps 3 to 5**

End

Figure 3 shows an example of the CAF where the signal is present. As you can see, there is a peak corresponding to the signal delay and frequency shift.

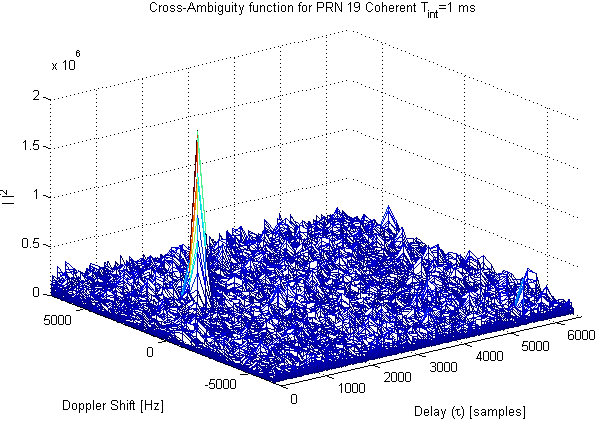


Figure 3: Example of CAF.

1. The last step of acquisition is to identify the position of this peak and determine if it actually corresponds to a signal and not to just noise. To do so you should:
2. Obtain the CAF value and the position of the highest peak.
3. Use the provided function secondPeak() to obtain the value of the second highest peak on the CAF.
4. Compare the ratio between the max peak and the second highest peak to   
   settings.acqThreshold. If the ratio is above the threshold, we will consider that the signal is present.
5. If the signal is present, fill acqResults as:
   1. acqResults.carrFreq = settings.IF + Doppler shift corresponding to the max peak.
   2. acqResults.codePhase = code phase of the max peak.
   3. acqResults.peakMetric = ratio between max and second peak.

Once again, run the script Main\_Ex3.m to perform the acquisition and obtain the results.

Plot the bar graph that is generated with the acquisition results.

Une image contenant stationnaire

Description générée automatiquement

**Q.8** How many satellite signals were acquired?

10 satellite were acquired: satellites number 3, 6, 11, 16, 18, 19, 21, 22, 30 and 31.

**Q.9** Which is the satellite with the highest detection metric?

From the bar graph, we observe that satellite number 31 has the highest detection metric, with around 7.789.

Plot the CAF function for satellite 22. Use MATLAB’s mesh() function and don’t forget to name figure’s axis (use Figure 3 as example).

Une image contenant carte

Description générée automatiquement

**Q.10** What range and step size do you use for the frequency search space? What is the corresponding maximum frequency mismatch loss (check course materials for the definition of mismatch loss)?

In the frequency space, we use a range of 14 kHz and a step size of 500 Hz. The maximum frequency mismatch loss is thus:

Now plot the CAF function for satellite 6.

Une image contenant carte

Description générée automatiquement

**Q.11** What do you observe compared with the CAF for satellite 22? How do you explain it ?

Here we observe two peaks in the CAF. This may be due to a data bit transition, so the two peaks correspond to two different bits.

Please, validate your results with the instructor before moving to **Exercise 4**.

1. Exercise 4: High Sensitivity Acquisition

The goal of this exercise is to increase the acquisition sensitivity by increasing the coherent integration time.

To do so, you have two possible options (pick one):

1. Modify the function acquisition\_1ms(), that you have previously completed for Exercise 3, to implement a 10 ms integration time. In this case please rename it to  
    acquisition\_Tcoh()function.
2. Modify directly Main\_Ex4.m to include the necessary code to perform the 10 ms acquisition.

If you these two options are not clear for you, please ask the instructor for more details.

In any implementation, the FFTs should still be performed on 1 ms data, and then, the ten results for each individual ms must be summed.

We recommend you to use the settings.cohInt (by default = 10) property, in the initSettings script, to define the integration time in your code.

**Hint:** Use generateGoldCodeSampled() function to generate 10 ms of sampled code. Then extract the subvectors of 1 ms (from 1 to 10) to compute the correlation with the received signal.

To perform the acquisition and obtain the results, run the script Main\_Ex4.m. Remember to set “Exercise 4 – High Sensitivity Acquisition” as your working folder before running Main\_Ex4.m

**Q.12** For satellite 22 found in previous Exercise 3, is the peak more or less identifiable than before?

**Q.13** Were you able to detect all the satellites found in Exercise 3? What happened to the peak metric?

No, we detect less satellites than previously. This is because of bit transition effect: for some satellites, the peaks will balance each others so the resulting gain is lower. On the other hands, some other satellites will benefit from the longer integration time, their peaks will be higher.

Are there, any new satellites detected?

If the answer is yes, indicate its/their corresponding code and Doppler frequency.

|  |  |  |  |
| --- | --- | --- | --- |
| Satellite PRN  number |  |  |  |
| Code delay  (samples) |  |  |  |
| Frequency Doppler (Hz) |  |  |  |

**Q.14** We have seen how increasing the integration time can increase the sensibility of the receiver for some of satellite signals, while others seem to be not present now. How do you explain this?

Are there any limitations that prevent us to further increase the integration time to acquire even weaker signals? (Consider the GPS C/A case)

1. Exercise 5: High Sensitivity Acquisition, Bonus Question

Based on your findings in Q.14, modify your code from Exercise 4 function to circumvent the observed problem.

Once again, remember to set “Exercise 5 - High Sensitivity Acquisition - Non-Coherent T\_int 10 ms” as your working folder before running Main\_Ex5.m

**Hint:** think about “non-coherent” combinations of intermediate results!

**Q.15** Plot the bar graph that is generated with the acquisition results and compare it with the plots that you obtained in Exercise 4 (both with short and long coherent integration times). What do you observe and why?

1. The term “code phase” is equivalent to “code delay”. [↑](#footnote-ref-1)