Chapter 5

Tests and Results

In Chapter 4 different algorithms typical of a snapshot GNSS mass-market receiver have been presented. In order to validate their performance and to compare them, several tests and simulations have been performed, the results of which are reported hereafter.

First the description of a fully software receiver implementing these algorithms is provided. Then, results concerning the performance of the multi-correlator unit, the accuracy of the code delay and of the Doppler frequency estimators are reported and discussed in Sections 5.2, 5.3 and 5.4. Afterwards, some results concerning search space windowing and on demand duty cycle processing are reported. Section 5.8 describes the result obtained processing real Galileo IOV signals with the software receiver developed and with the techniques proposed. Finally the results of some tests with a real commercial receiver and with the real Galileo IOV signals are given.

5.1 A fully software receiver

A GNSS SDR receiver has been designed from scratch as a proof of concept, so as to test and validate all the techniques proposed in Chapter 4. It exploits the flexibility offered by SDR implementation and represents a fundamental research working tool. In particular, it accompanied all the phases of the work:

- the test of state-of-the-art algorithms;
- their adaptation to a software platform;
- the design of their extensions to new signals and constellations;
- the first tests with real Galileo signals;
- the development of innovative algorithms;
- the final performance assessment in terms of estimates accuracy and PVT solution.

5.1.1 The concept of SDR

SDR refers to an ensemble of hardware and software technologies enabling a reconfigurable communication architecture [26]. Despite the fact that software implementations are generally less efficient than hardware dedicated receivers, SDR has entered the field of GNSS receivers as soon as satellite navigation enlarged its popularity. The availability of digital signals opened new opportunities and lead to the design, directly in the digital domain, of new algorithms, which were too complex to be implemented in an analog way. For example, the possibility to access intermediate processing stages and observables, which is not available in commercial hardware receivers, paved the way towards innovative interference [66] or spoofing detection techniques [67]. SDR GNSS receivers are nowadays tools of paramount importance for Research and Development (R&D) and still offer new interesting and un-explored perspectives, like the case of a multi-GNSS scenario. Among the wide variety of solutions that employ different combination of processing platforms, three categories of architectures can be identified [26]:

classic SDR architectures, based on an FPGA, and performing positioning algorithms on a DSP;

hybrid architectures, based on an FPGA with hardware accelerator, and performing the positioning software algorithm on Personal Computer (PC);

fully software architectures, entirely hosted on a General Purpose Processor (GPP), such as a PC or an embedded system.

The receiver developed belongs to the category of **fully software receivers**: the full receiver, including the baseband functions and the positioning algorithms, is implemented using MATALB programming language and runs on a PC. It accepts as input different kind of raw IF GNSS data, coming from different sources. It is able to process both GPS L1 C/A signals and Galileo OS E1B and E1C signals It is based on a multi-correlator open-loop snapshot processing approach, as described in Section 4.1 and can provide as output the usual receiver's observables, such as code delay, Doppler frequency, C/N_0, phase and navigation message. The receiver is very flexible and fully configurable, and represents an important tool to assess performance and accuracy of the selected techniques in different circumstances.

5.1.2 Initialization of the receiver

Mass-market receivers usually benefit of assistance data, according to the 3GPP AGNSS standard [68]. The receiver software implemented follows the same principle and, despite the fact it could potentially work as a standalone receiver, the presence of some aiding greatly improves performance. As depicted in Figure 5.1, mass-market receivers exchange information with assistance servers as follows.

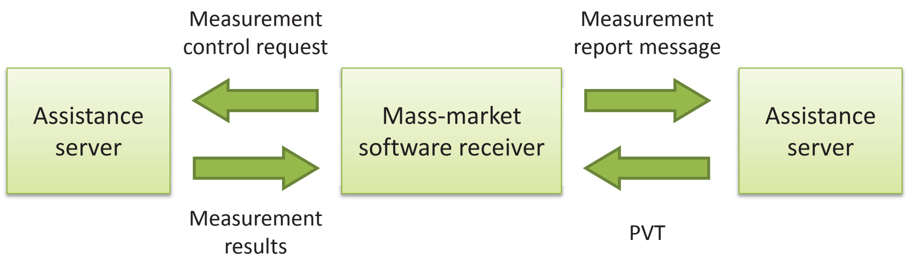

Measurement control request
Measurement report message
Assistance server
Mass-market software receiver
Assistance server
Measurement results
PVT

Figure 5.1: Scheme of message exchange between assistance server and mass-market receiver.

- **Measurement control request:** the receiver asks for certain measurements, such as timing information, list of PRNs in view, rough Doppler frequencies and rough code delay.
- **Measurement results:** the server replies with the proper information
- **Measurement report message:** the receiver sends to the server the observables, i.e. the results of the processing of raw data.

- **PVT:** the server computes the PVT and sends it back to the receiver.

Notwithstanding, the message exchange is only *emulated* by the software receiver: the information to initialize the multi-correlator open-loop engine is supplied manually or exploiting a standard acquisition scheme, and the PVT is computed with a standalone software routine.

5.1.3 Block scheme of the final version

A block scheme of the final version of the fully software receiver for Galileo E1BC processing implements is reported in Figure 5.2.

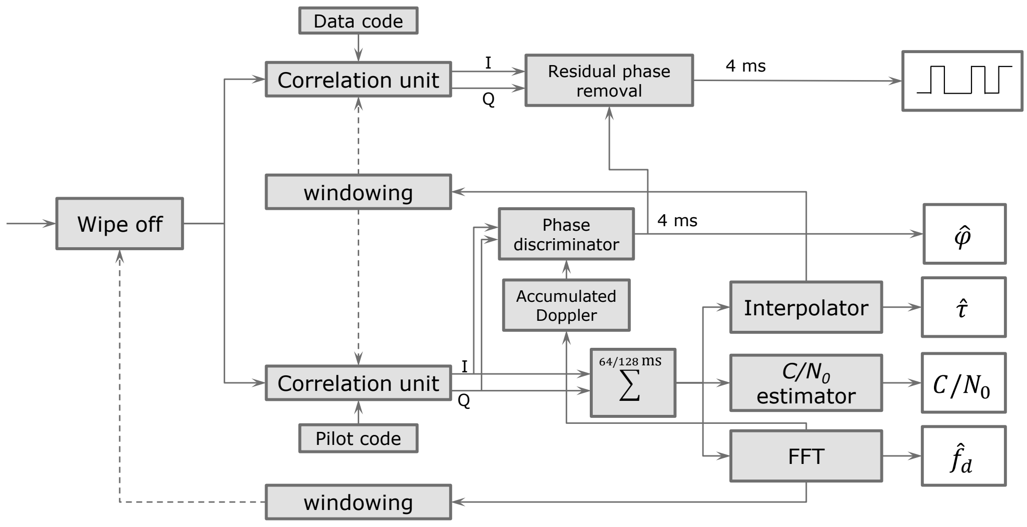


Figure 5.2: Fully software receiver implementation block scheme.

5.1.4 Signals

Three types of GNSS signals are considered in all the work:

- **Software simulated signals**, obtained with a software signal generator (N-FUELS), described in Appendix A.1. They give the best flexibility in terms of frequency plan, modulation, navigation message, power level, frequency and code delay definition, at the expenses of a longer simulation time and not realistic signal conditions. Some signal impairment, such as interference, multipath can be added to the signal. However they can be used only by software receivers.

- **Hardware simulated signals**, obtained with the Spirent signal generator available at ESTEC premises and described in Appendix A.3.1. This tool is flexible, powerful, but its configuration is not trivial and its availability limited. The LMS model described in Sections 3.4.2 and 3.4.3 can be included in the Spirent configuration. The RF signal is conditioned by a front-end (Section 3.1.1) grabbed with a bit-grabber, such as the ones described in Section A.2, and then stored to a memory for post-processing.

- **Real GPS and Galileo signals**, captured by an antenna, conditioned by a RF front-end and stored in a memory through a bit-grabber. They are less flexible, completely unknown in principles, but obviously extremely realistic.

5.2 Multi-correlation results

A software routine performing the multi-correlation snapshot algorithm described Section 4.1 has been implemented, representing the core of the fully software receiver. In the first stage, in order to test the features of the technique, an N-FUELS input signal is considered. Its characteristics are reported in Table 5.1; most of them, such as the sampling frequency, are compatibles with the ones of real mass-market hardware devices.

Table 5.1: N-FUELS signal parameters for snapshot processing test.

Signal length	0.5 s
Frequency band	L1 (1575.42 MHz)
Sampling frequency	16.3676 MHz
IF carrier frequency	4.1304 MHz
number of satellites	1
Doppler	500 Hz, fixed
Code delay	0.1 ms
Modulation	GPS L1 C/A or Galileo BOC(1,1)
Navigation message	Absent
Front-end filter	Butterworth 4th order, 4.092 MHz

The signal is processed by the software routine and the non-normalized correlation function is shown as result of the signal correlation, both for the I and for the Q channels. The main differences between this simulation and a real scenario have to be pointed out:

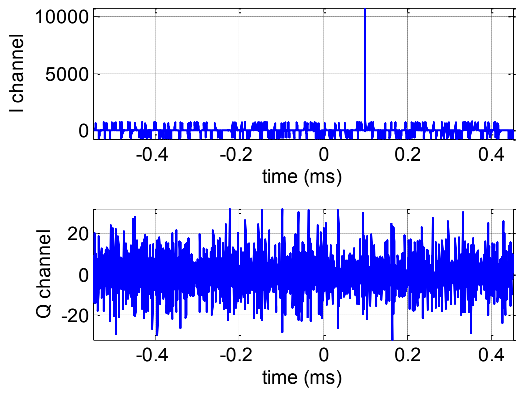
- noise is absent, at least for the first configuration;
- multipath is absent;
- ionosphere and troposphere propagation is not simulated;
- the phase of the signal is perfectly known;
- the Doppler frequency of the signal is perfectly known (perfect carrier wipe-off);
- only 1 signal component is considered.

In Sections 5.8 and 5.2.6, the same algorithm is applied to real GNSS signals. Some results are shown and discussed.

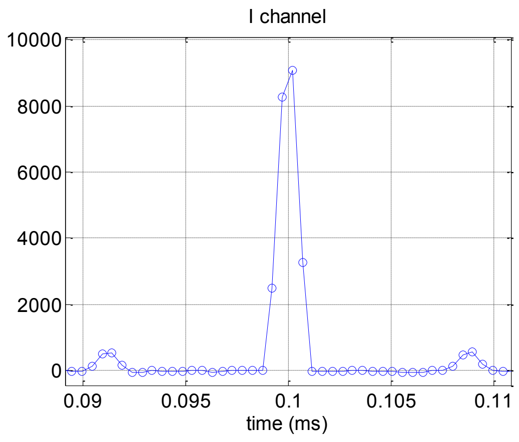
5.2.1 Example of a search space

Figure 5.3a shows an example of correlation function obtained exploiting the technique described. The input signal is a GPS L1 C/A signal, PRN 1, with a very high C/N_0. The phase and frequency of the input signal are perfectly known so a perfect wipe-off is performed. The domain of the correlation function covers a snapshot of 1 ms, which is the nominal length of a code period. The simulation mode is the standard mode, with 2 samples per chip ($P = 2$). The shape of the theoretical auto-correlation function of the BPSK modulation is clearly recognisable in the I plot. A peak appears around 0.1 ms, which exactly matches the true code delay of the signal, as configured in the simulation. In addition, Figure 5.3b shows a zoom around the peak. The shape of the triangle is clear, and it can be assumed, from a “visual interpolation”, that the upper vertex is exactly located at 0.1 ms. At the same time, in the Q channel only noise is present.

More in details, Figure 5.3 shows the magnitude of the correlation in the vicinity of a peak. Points on the x axis are spaced at an interval equal to the C/A code chip length divided by P; in



(a) I and Q channels



(b) I channel (zoom)

Figure 5.3: Example of the correlation function for a GPS L1 C/A signal obtained with the multi-correlation snapshot processing technique.

this case the sampling interval corresponds to 150 m in the range domain). Therefore the width of the triangle base is exactly 2 C/A code chips, or 4 points of the Auto Correlation Function (ACF). The height of the base of the triangle is the magnitude of the noise. The peak of the triangle corresponds to the signal magnitude.

The obtained plot is useful to compare the performance, in terms of signal detection capability, sensitivity, time to perform correlation, under different scenarios:

- different input C/N_0;
- different coherent and non-coherent integration times;
- different operation mode (standard, high resolution, super-high resolution); different signals (Galileo and GPS);

In particular the C/N_0 is computed according to the following formulation:

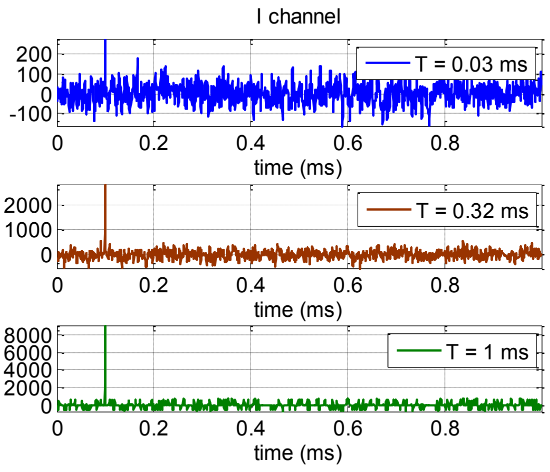
$$C/N_0 = \frac{1}{T_\mathrm{c}} \frac{A^2}{2\sigma^2} \, , \tag{5.1}$$

where T_c is the coherent integration time, A is the magnitude of the peak, and σ^2 is the noise variance, which can be computed as

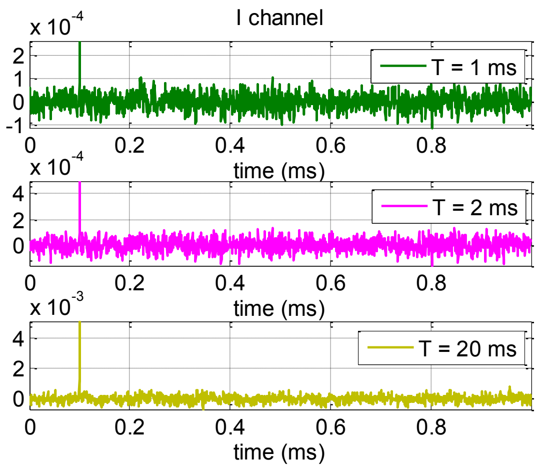
$$2\sigma^2 = \mathbf{E}\left[I^2 + Q^2\right] \, . \tag{5.2}$$

5.2.2 Effect of different integration times

The coherent integration time is configurable. The minimum value corresponds to the length of a the shift register, e.g. 66 samples for the standard mode, corresponding to 33 chips, or 0.032 ms. Other values are possible, for example a value corresponding to the length of the shift register times L, or as in the general case the full code length, 1023 chips, corresponding to 1 ms (equal to the register length times M), or even more, as long as no data transition occurs. It is obvious



(a) $T_c < 1$ ms



(b) $T_c > 1$ ms

Figure 5.4: Effect of different integration times in the snapshot processing correlation function.

that, by increasing the coherent integration time, the peak emerges from the noise floor; however the dynamics of the input signal must be accounted for, and bit synchronizations is required.

Figure 5.4a shows the effects of different coherent integration times on the I channel correlation function. The signal is a GPS L1 C/A signal, without noise, in standard resolution mode. In the first case the integration time is set equal to the length of the shift register. The peak is visible, although the noise level due to cross-correlation is quite high. In the second case the integration time is equal to 10 times the size of the shift register, in the third case equal to 31 times the length of the shift register, i.e. 1 ms. Table 5.2 reports some numerical values, among which the approximate time required to perform the correlation in the Matlab simulation and the SNR between the peak and the noise floor. It is noted that the time information assumes interest only as a comparison, and not as absolute value.

Table 5.2: Effect of different integration times lower than 1 ms in snapshot processing.

Integration time	SNR	Time elapsed
66 samples - 0.03 ms	17 dB	0.30 s
660 samples - 0.32 ms	26.4 dB	0.43 s
2046 samples - 1 ms	31.6 dB	0.78 s

It is interesting to notice how the SNR and the time elapsed increase when increasing the integration time. In particular, when increasing the integration time by a factor of 10 (from 0.03 ms to 0.32 ms), the SNR increases by a factor of 10 dB, while when increasing the integration time by a factor of 3 (from 0.32 ms to 1 ms), the SNR increases by a factor of about 5 dB, as expected from theory.

Clearly, integration times higher than 1 ms are possible. Results for 2 ms and 20 ms coherent integrations are shown in Figure 5.4b. In this case the signal is affected by noise, the C/N_0 is equal to 50 dBHz. Similarly Table 5.3 reports the C/N_0 after the correlation and the time elapsed in the same three cases. As expected, by increasing the integration time up to 20 ms sensitivity is higher, and the receiver is able to acquire also signals that would be under the noise floor for lower integration times.

Table 5.3: Effect of different integration times longer than 1 ms in snapshot processing.

Integration time	SNR	Time elapsed
1 ms	19.8 dB	0.78 s
2 ms	21.5 dB	1.29 s
20 ms	28.8 dB	8.6 s

Figure 5.5 reports the results for 1 ms and 20 ms integration times for a signal with a C/N_0 equal to 35 dBHz. It is evident that the code delay peak is visible only when integrating for a longer time, confirming the benefits of this technique in the case of low power signals (high sensitivity).

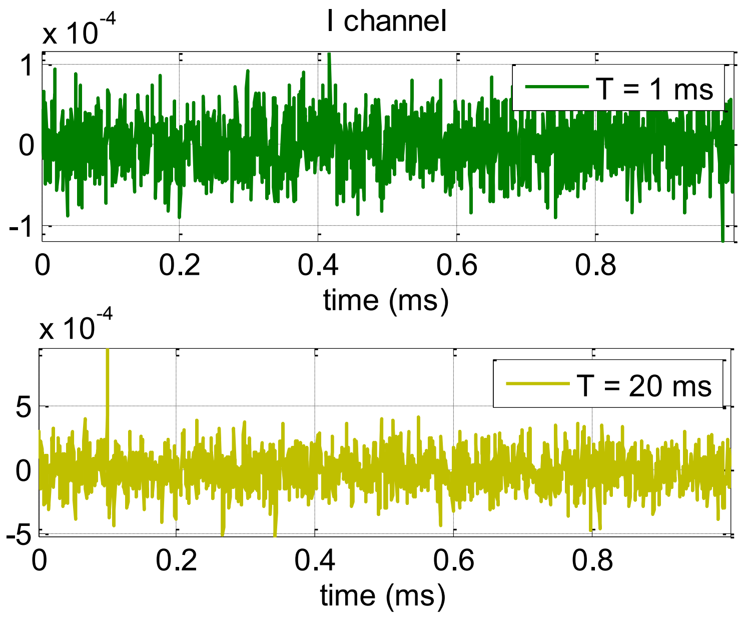


Figure 5.5: Effect of 20 ms integration times in the snapshot processing correlation function for low power signals.

5.2.3 Effect of different input C/N_0

It is expected that the technique proposed behaves differently according to different input C/N_0. In this section GPS C/A signals characterized by different C/N_0 are tested. Figure 5.6 shows the correlation function for four different signals with a C/N_0 respectively equal to 35, 40, 45 and 50 dBHz, coherently integrated for 1 ms. While in the first case the signal is not acquired, in the remaining cases the correlation peak is clearly visible, and the noise floor is lower and lower as the C/N_0 increases.

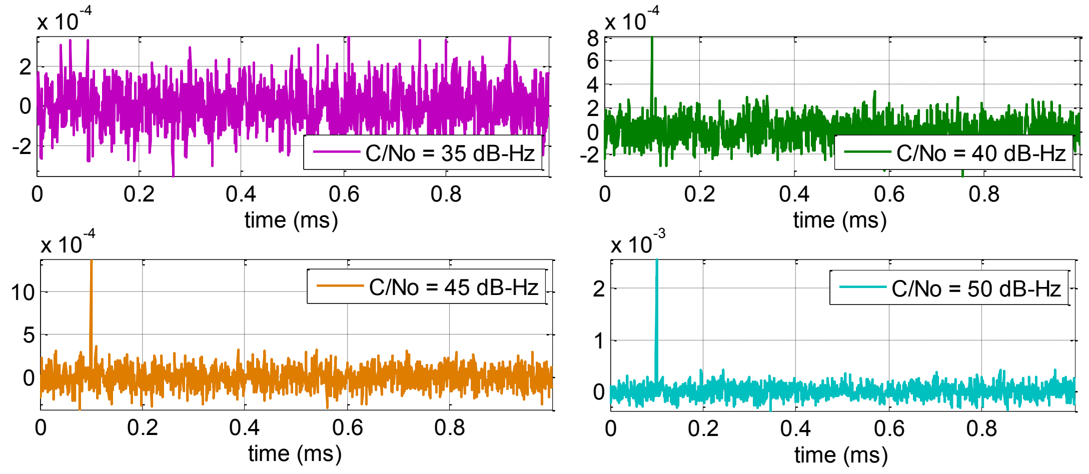


Figure 5.6: Effect of different signal strength in the snapshot processing correlation function with 1 ms coherent integration.

Table 5.4 reports the SNR and the C/N_0 after correlation in the different cases. As expected it increases with the signal strength. It is noted that also low power signals can be acquired by increasing the integration time.

Table 5.4: Effect of different signal strength in the snapshot processing correlation function with 1 ms coherent integration.

Signal C/N_0	SNR	Estimated C/N_0
35 dBHz	not acquired	not acquired
40 dBHz	9.3 dB	39.3 dBHz
45 dBHz	15.7 dB	45.7 dBHz
50 dBHz	19.8 dB	49.8 dBHz

In order to acquire the signal for signal strengths as low as 35 dBHz it is necessary to increase the integration time. As detailed in Section 3.3.1, this is possible only if navigation data are absent or wiped-off. For this purpose, N-FUELS signals with no navigation data are exploited. Table 5.5 summarizes the input parameters chosen and reports the SNR and the C/N_0 for the different cases.

Figure 5.7 shows that in the first four cases it is possible to acquire the signal, since the peak is clearly visible and the resulting SNR is higher than 15 dB. Instead in the last two, the peak values are similar to the noise realization and the probability of acquisition is then lower.

Table 5.5: Effect of different signal strength C/N_0 in the snapshot processing correlation function with different coherent integration times.

Signal C/N_0	T_c	SNR	Estimated C/N_0
35 dBHz	20 ms	17.2 dB	34.2 dBHz
29 dBHz	120 ms	18.1 dB	27.3 dBHz
20 dBHz	640 ms	17.6 dB	19.5 dBHz
15 dBHz	1200 ms	14.7 dB	13.9 dBHz
10 dBHz	2560 ms	11.5 dB	7.5 dBHz
5 dBHz	2560 ms	9.4 dB	5.3 dBHz

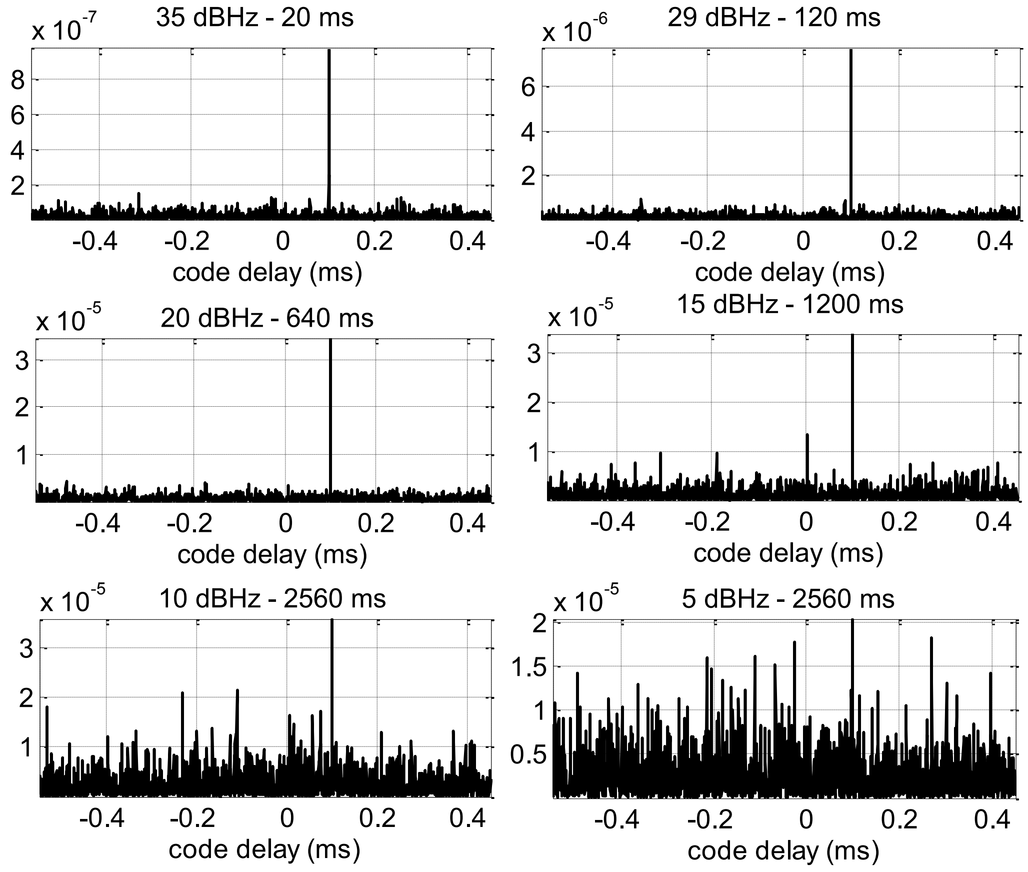


Figure 5.7: Effect of different signal strengths and integration times on the correlation peak function.

5.2.4 Effect of different operation modes

The most interesting feature of snapshot multi-correlation algorithms is represented by the possibility to dynamically switch between different operating modes. The parameters for the standard, the high-resolution and the super-high resolution modes are reported in Table 4.1. Briefly summing up, the standard mode is able to look for the code delay in the entire code epoch, but has a reduced accuracy (2 samples per chip). On the contrary high and super-high resolution modes can account for a larger accuracy (5 and 10 samples per chip respectively), despite the fact that they consider a reduced search space, and therefore need a priori rough information on the code delay. The standard case has been extensively analysed in Sections 5.2.1, 5.2.2 and 5.2.3. The high resolution mode considers a subset of the code equivalent to 1/8 of the code epoch. A rough estimate of the code delay should then be sufficiently accurate to fall within ±63 code chips, otherwise the signal has no chance to be acquired, as described in Section 5.6.3 concerning signal windowing.

Figure 5.8 shows the correlation function in the I and Q channels when processing the signal in high definition mode. The signal is a GPS L1 C/A signal, without noise, the integration time is equal to 1 ms. The peak is clearly emerging, and it is interesting to notice that the correlation spans a time interval of only 0.125 ms = 1 ms)/8.

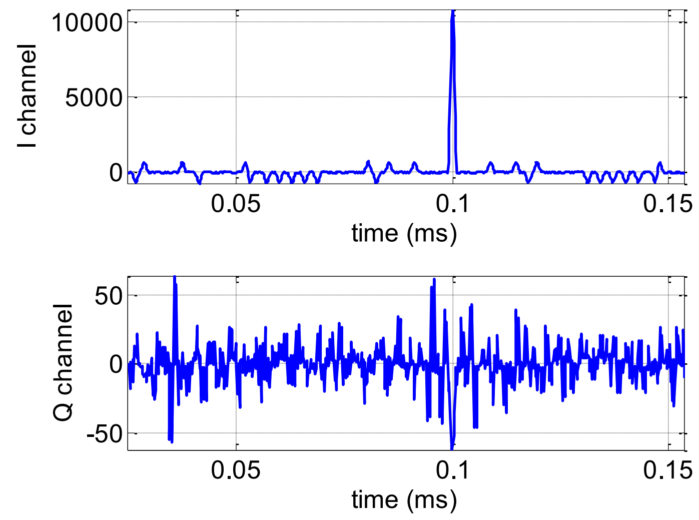


Figure 5.8: Example of the correlation function for a GPS L1 C/A signal in high resolution mode.

In the super-high resolution case the search space is further reduced, and in the delay domain it corresponds to 1/16 of the code epoch. Figure 5.9 depicts the zoom on the correlation peak for the high resolution and the super-high resolution cases. In the left plot the resolution is equal to 5 samples per chip, indeed the base of the triangle corresponds to 10 samples. In the right case the resolution is doubled, 10 samples per chip, allowing a theoretical higher accuracy in the delay estimation. These results can be compared with the standard resolution peak zoom, reported in Figure 5.3b.

Finally, Figure 5.10 depicts the case in which the true signal code delay falls outsides the actual reduces search space; therefore the peak is not detected and an invalid measure is obtained. This problem is considered in the windowing of the search space, in Section 5.6.

It is also possible to further reduce the number of correlations, by decreasing the L parameter. For example, Figure 5.11 shows the correlation peak in the standard resolution mode for $L = 31$, corresponding to the case of Figure 5.3a, for $L = 15$, so considering about half of the code-epoch

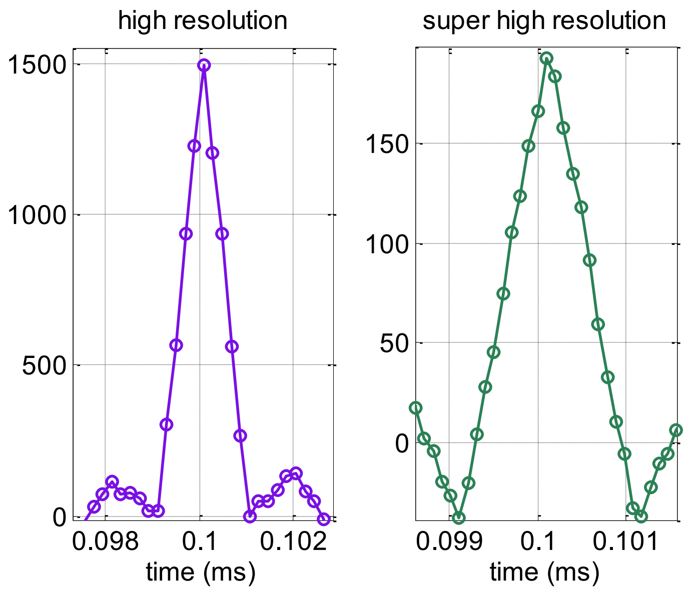


Figure 5.9: Example of the correlation function for a GPS L1 C/A signal in different resolution modes.

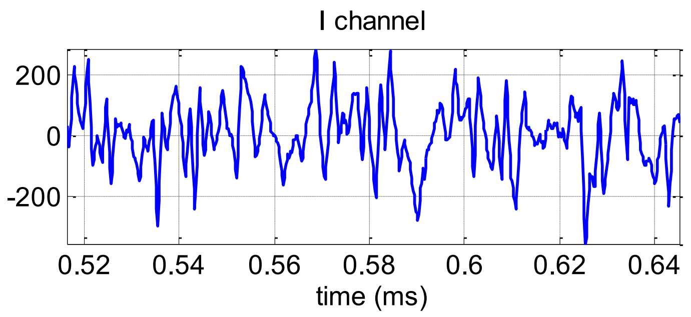


Figure 5.10: Acquisition failure in the case of false aiding information in high resolution mode.

length, and for $L = 5$, equivalent to approximately 1/6 of the code length. It is clear from the picture that the accuracy and the SNR of the result is equivalent, while the computational complexity is reduced by a factor of 2 and of 6 approximately.

5.2.5 Comparison between GPS and Galileo signals

The technique can be extended to Galileo signals as described in Section 4.1.2; the only difference consist of a new design of the parameters K, M and L, according to the structure of the BOC(1,1) modulation. In particular, since the Galileo OS code-epoch is 4 times larger than in GPS, either a larger shift register or a higher frequency are required. First an example of correlation function is reported in Figure 5.12, with $P = 2$, $K = 33$, and $L = 124$. In this case 8184 parallel correlators are required. The dimension of the shift register is the same of the GPS case, 66 samples, but the clock frequency is four times larger. The signal is a Galileo E1B BOC(1,1) signal without noise and the integration time is equal to a full code epoch, i.e. 4 ms.

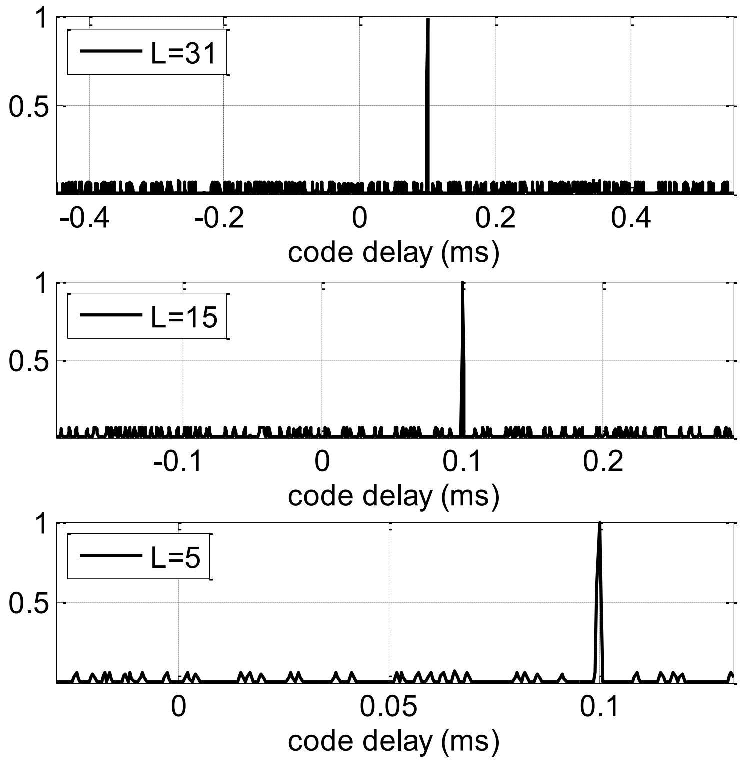


Figure 5.11: Correlation peak for standard resolution and for different values of L.

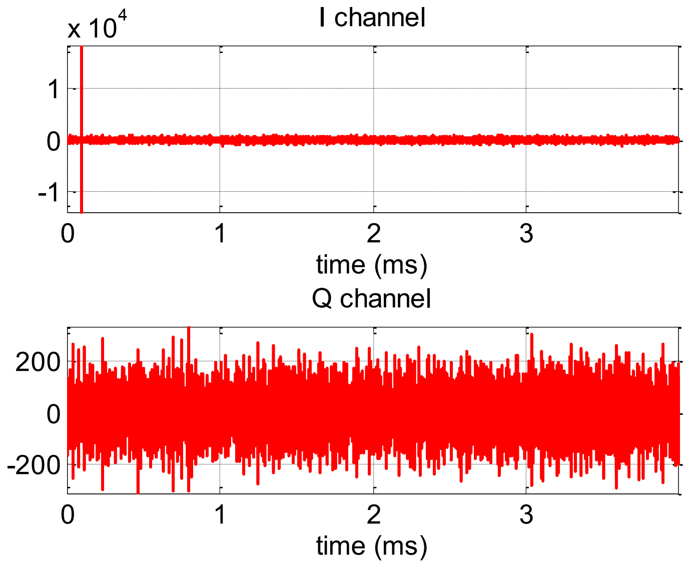


Figure 5.12: Example of the correlation function for a Galileo E1B BOC(1,1) signal obtained through snapshot algorithms.

The peak around 0.1 ms is clear and the noise level is sufficiently low to allow acquisition of the signal. By enlarging the figure around the peak the theoretical sharper correlation function of the BOC(1,1) modulation can be recognized. As in the case of GPS, depicted in Figure 5.3b, the resolution is equal to 2 samples per chip.

The main issue concerning the processing of Galileo signal is the increased complexity. With respect to GPS C/A, the code is 4 times longer and the modulation is more elaborated. The standard mode operation has been performed by considering the third combination of parameters, as described in Table 4.2, i.e. with 2 samples per chip ($P = 2$), $K = 132$, a shift register of 264 bins, $L = 31$, giving a clock frequency of 127 MHz and a total of 8184 correlators. So, with respect to GPS, the clock frequency is the same and the shift register is 4 times larger. Table 5.6 shows a summary of parameters and the result in terms of execution time, which is approximately 4 times larger for Galileo. This result is acceptable, since the code epoch, and so the integration time is 4 times larger.

Table 5.6: Comparison between GPS an Galileo, standard resolution.

	GPS L1 C/A	Galileo E1B BOC(1,1)
f_{clk}	127 MHz	127 MHz
Shift register (PK)	66	264
Integration time	1 ms	1 ms
L	31	31
Time elapsed	0.56 s	2.13 s

By reducing the Galileo integration time to 1 ms, corresponding to 1/4 of a code epoch (sub correlation), it is possible to compare the results in a fairer way. Table 5.7 shows the results; the elapsed time is reduced by approximately a factor of 4, giving a performance comparable with the GPS case.

Table 5.7: Comparison between GPS an Galileo, standard resolution, same integration time.

	GPS L1 C/A	Galileo E1B BOC(1,1)
f_{clk}	127 MHz	127 MHz
Shift register (PK)	66	264
Integration time	1 ms	4 ms
L	31	62
Time elapsed	0.56 s	0.63 s

By considering another approach, for example a double shift register dimension and a double clock frequency, the results are different. As reported in Table 5.8, the elapsed time increases from 0.6 s to almost 1 s.

Table 5.8: Comparison between GPS an Galileo, standard resolution, double shift register.

	GPS L1 C/A	Galileo E1B BOC(1,1)
f_{clk}	127 MHz	254 MHz
Shift register (PK)	66	132
Integration time	1 ms	1 ms
L	31	31
Time elapsed	0.56 s	0.95 s

However it has to be pointed out that an accurate comparison between GPS and Galileo execution times should be carried out by counting the number of floating and or fixed point operations rather than measuring the elapsed time. In addition, the software implementation cannot consider the presence of more correlators working in parallel.

Better results can be achieved passing to high resolution and super-high resolution operating modes. As for the GPS case, the resolution of the peak in samples per chip can be increased. In addition, by exploiting for each correlation a subset of the reference code instead of the full code, the computational burden decreases. The value of the P, K and L parameters for Galileo E1B are reported in Table 4.3.

Figure 5.13 reports a plot of the correlation peak in the case of super-high resolution. Further techniques, such as least square interpolation, can assure even better results.

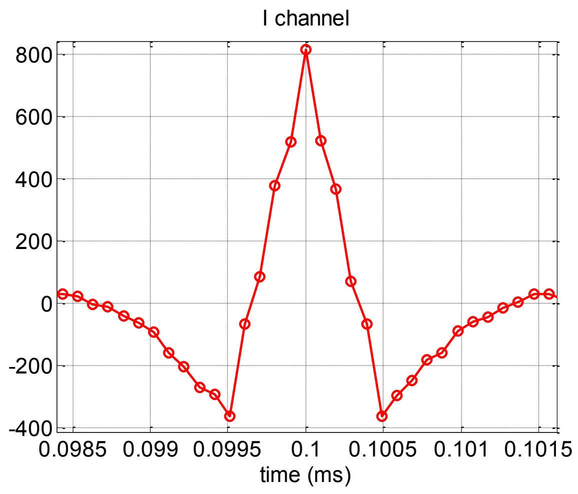


Figure 5.13: Super-high resolution in the case of Galileo signal.

Finally a comparison in terms of execution time is given in Table 5.9. GPS and Galileo signals are compared, with a comparable integration time (1 ms), with no noise and with the parameters described above. The execution times for GPS and Galileo are comparable. However is has to be pointed out that the execution time depends strictly on the code implemented, its optimization and the computational capacity of the machine on which it is run.

Table 5.9: Total execution time for different operation modes and signals.

	GPS L1 C/A	Galileo E1B BOC(1,1)
Standard resolution	0.56 s	0.63 s
High resolution	0.21 s	0.22 s
Super-high resolution	0.16 s	0.18 s

5.2.6 Real data snapshot processing

Similar results can be obtained by processing real GNSS data, captured at ESTEC premises. It is interesting to point out that the algorithms is able to process not only simulated signal but also real signals, characterized by a real propagation scenario and then affected by several errors from different sources.

120 seconds of raw GNSS L1 data have been collected with the front-end/bit-grabber (described in Section A.2), with a sampling frequency equal to 16.368 MHz and an IF equal to 4.092 MHz, and stored in a file for post-processing, on December 18, 2013, at 12.30 UTC+1 , at ESTEC Navigation laboratory (52.218° N, 4.419° E). Figure 5.14 shows the GPS and Galileo satellites visible at that time in that location.

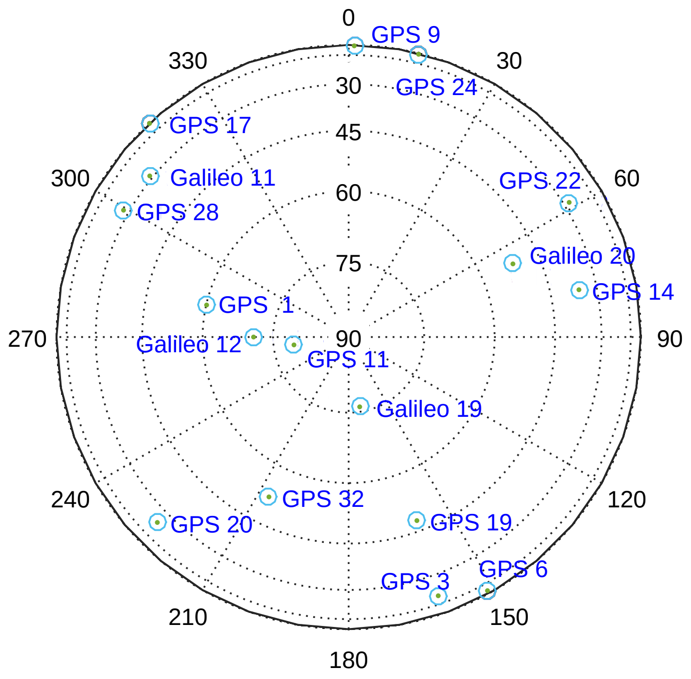


Figure 5.14: Skyplot at ESTEC, December 18, 2012, 12.30.)

The data have been processed with the fully software snapshot receiver described in Section 5.1, conveniently modified in order to fit front-end frequency plan. The main differences with the simulated signal considered above are:

- the presence of noise;
- the possible presence of multipath (however the roof top position of the antenna assures a low multipath impact);
- errors induced by ionospheric and tropospheric propagation;
- the phase of the signal is totally unknown;
- the Doppler frequency of the signal is roughly known (it accounts also for the grabber oscillator clock drift); in particular, the code is initialized with some aiding information. For this reason both the I and the Q branches of the processor are considered and combined: $S = I^2 + Q^2$;
- several GPS and Galileo signals are present;
- a real front-end characterized by a certain bandwidth is present;
- the signal is quantized over a certain number of bits.

For example, Figure 5.15 shows the multi-correlation function over an entire code epoch, obtained correlating a snapshot of input signal with local code of GPS PRN 11. The standard resolution mode is exploited, with $P = 2$; the samples are coherently accumulated for $T_c = 1$ ms, and then 5 accumulations are performed, for a total period equal to 5 ms. The code delay, corresponding to the correlation peak is located around -0.27 ms.

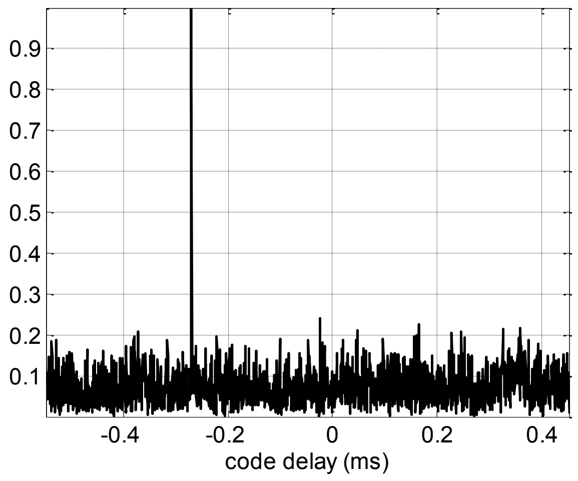


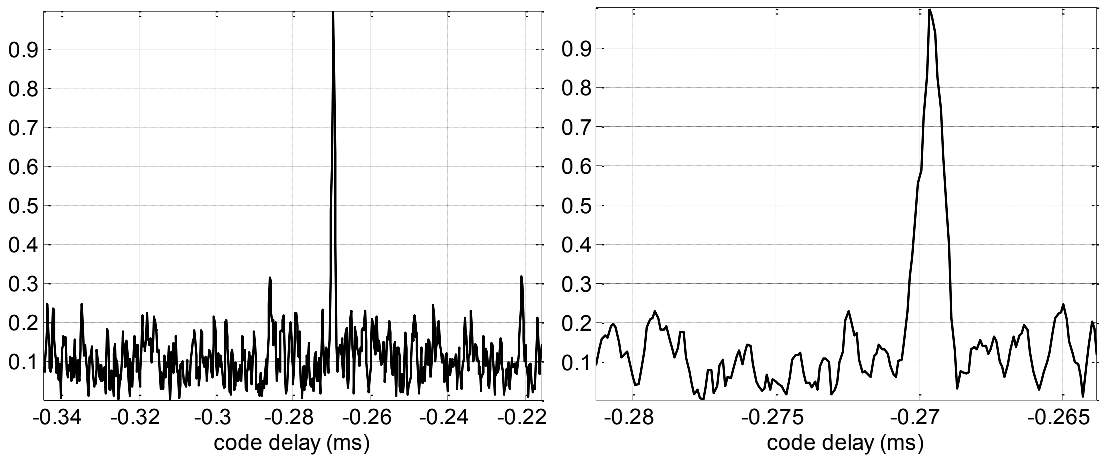
Figure 5.15: Code correlation function for GPS PRN 11 with standard resolution and 5 ms integration time.

Better results can be achieved performing interpolation on the samples of the correlation triangle or changing the resolution mode to high and super-high resolution. A priori information on the code delay is required, for example the output of the standard correlation processing. In these cases, reported in Figure 5.16, 5 and 10 samples per chip are considered respectively. In both cases the peak is evident.

It is interesting to look at the evolution of the correlation function. In fact, the algorithm is able to correctly follow the correlation function peak over time. In the example below, the GPS signal with PRN 11 is processed over a total integration time of 2 s. In particular, 100 snapshot correlations, each one corresponding to a 20 ms integration, are performed. Each 20 ms, an interpolation on the non-coherent accumulation (I^2+Q^2) is computed, and the result is reported in Figure 5.17. The navigation data bit synchronization is achieved manually by starting the processing at a given time.

With respect to simulated data, real data are affected by multipath; in particular low elevation satellites are more conditioned by the multipath effect, for geometrical reasons. Figure 5.18 shows the correlation peak, computed with super high resolution and with a non-coherent integration time equal to 10 ms. The signal corresponds to the PRN 22, which at that moment had an elevation of about 28°. It is possible to see in the plot the typical distortion of the peak induced by the presence of reflected rays, reaching the receiver with a larger time delay. The effect of a front-end filter is not present or negligible.

The same results can be obtained processing Galileo signals. An example, considering Galileo signal with code 22, coherent integration time equal to 4 ms, i.e. a BOC(1,1) code-epoch) and non-coherent integration time equal to 20 ms, is reported. In particular Figure 5.19 shows the correlation peak for the standard and the high resolution modes.



(a) High Resolution ($P = 5$) **(b)** Super-high Resolution ($P = 10$)

Figure 5.16: Code correlation function for GPS PRN 11 with different resolutions and 5 ms integration time.

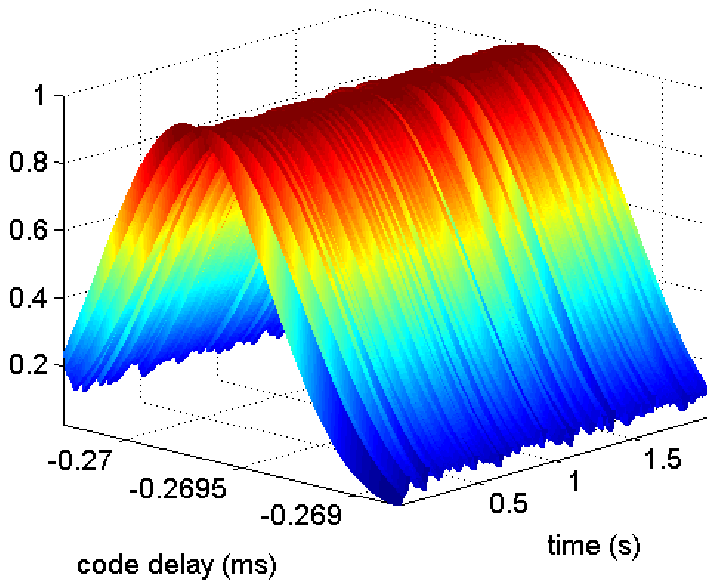


Figure 5.17: Correlation peak (interpolation) of a GPS signal over a time interval of 2 s.

5.3 Code delay estimation

In the simulations performed in this section the code delay estimation accuracy is evaluated. It has been proved that the code delay accuracy may depend on several factors and parameters used in the code delay estimation:

- the presence of a front-end filter;
- the true code delay, i.e. the delay between the received signal and the local replica;

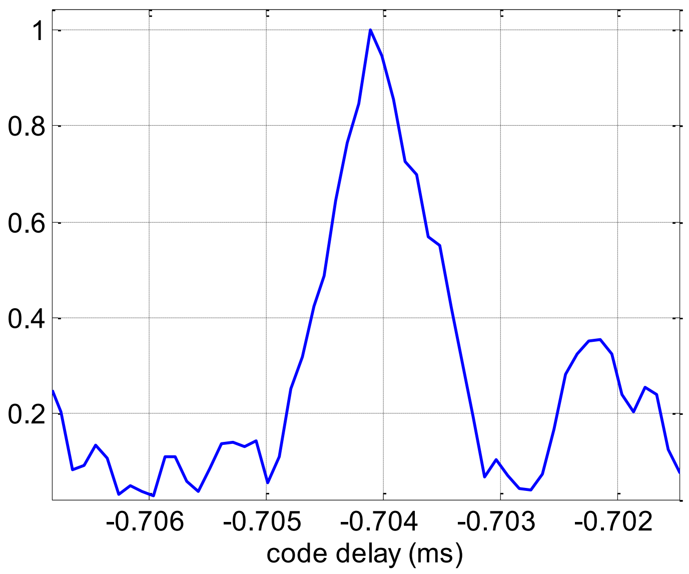


Figure 5.18: Code correlation function for GPS PRN 22 with super-high resolution; effect of multipath.

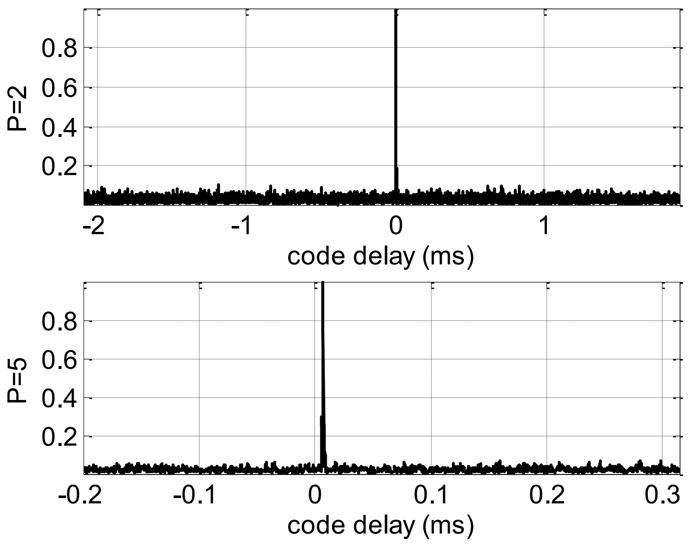


Figure 5.19: Correlation peak for standard and high resolution for Galileo code number 22.

- the C/N_0;
- the coherent integration time;
- the chip spacing (resolution of the correlation function, resolution mode in the software receiver);
- the resolution of the interpolation curve;
- the number of points of the correlation function used to interpolate.

First a Matlab routine implementing semi-analytical simulations for code delay estimation has been implemented, in order to compute the accuracy of the code delay estimate compared to different parameters. In all the following semi-analytical simulation, unless specified, the following parameters are assumed:

- BPSK signal, correspondent to the GPS L1 C/A signal;
- $T_c = 1$ ms;
- chip spacing equal to 0.1 chip; 21 samples of the correlation are considered, thus spanning an interval of 2 chips (the entire correlation peak). This is equivalent to the super high resolution;
- resolution of the interpolation equal to 100, i.e. 100 samples every sample of the correlation function
- all the points of the correlation, namely 21;
- a number of trials of the Monte Carlo simulation sufficient to smooth the effect of noise.

Then some results have been validated and confirmed exploiting N-FUELS and the fully software receiver and performing Monte Carlo simulations.

5.3.1 Effect of the front-end filter

It has to be reminded that the signal entering the receiver is far different from the ideal GNSS signal, mainly because of the effect of the front-end filtering. For example, a 4 MHz bandwidth front-end, typically adopted in mass market receivers, can save most of the power of a GPS L1 C/A signal, while cuts some useful signal from a Galileo E1 signal. Table 5.10 reports the typical error standard deviation on the pseudo-range for a GPS signal at 45 dBHz for different values of the front-end filter bandwidth, i.e. considering a different number of lobes of the input signal, for $T_c = 1$ ms [69].

Table 5.10: Standard deviation of code delay estimates at 45 dBHz for 1 ms coherent integration time.

Number of lobes	Bandwidth (MHz)	Code delay standard deviation (ns)	Pseudo-range error (m)
0.5	1.023	31.7	31
1	2.046	24.8	22
2	4.092	17.4	18
3	6.138	14.2	16
5	10.23	7.4	15

A front-end filter has been added to the semi-analytical model, in order to simulate the low-pass filter of the front-end. A Butterworth filter, of the 4th order and with a bandwidth of 4.092 MHz has been implemented in Matlab. The theoretical effect of the filter is to cut the high frequencies of the signal; therefore, the resulting correlation peak is smoothed, leading to a deterioration in the estimate accuracy. Figure 5.20 shows this effect: the theoretical and real (filtered) correlation functions for a BPSK signal are depicted. At the same time, especially in the presence of low C/N_0, the results can improve because of the rejection of out of band noise.

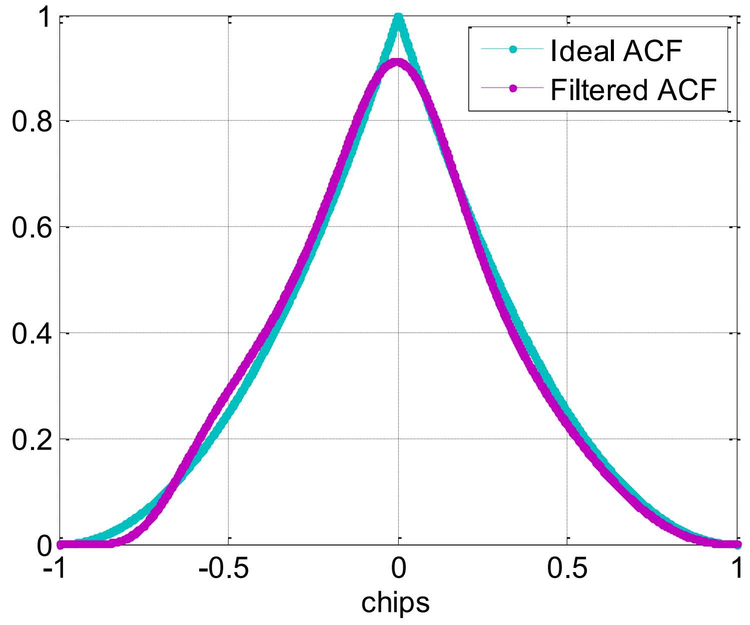
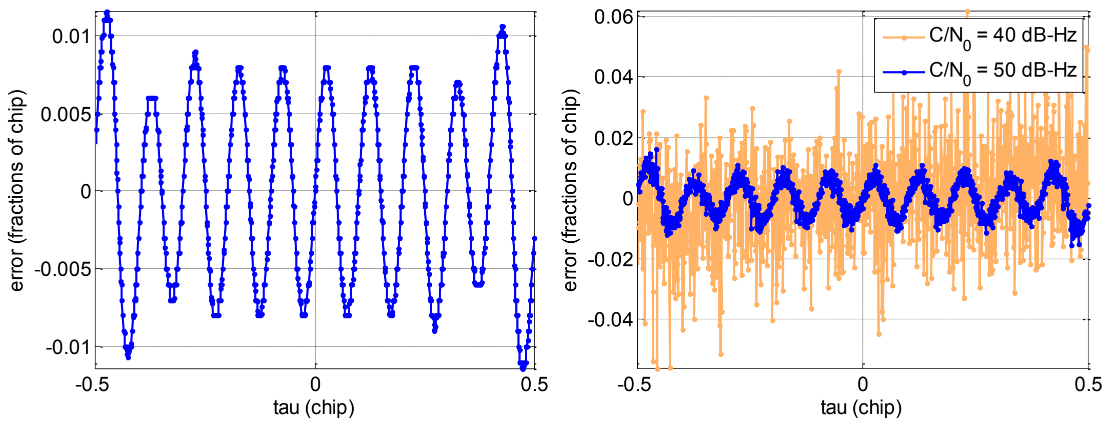


Figure 5.20: Effect of the front-end filter.

5.3.2 Effect of the initial code delay estimate

Let's assume that the noise is absent ($C/N_0 = 100$ dBHz) and that there is no front-end filter, it is possible to analyse the code delay accuracy.



(a) Absence of noise **(b)** Presence of noise

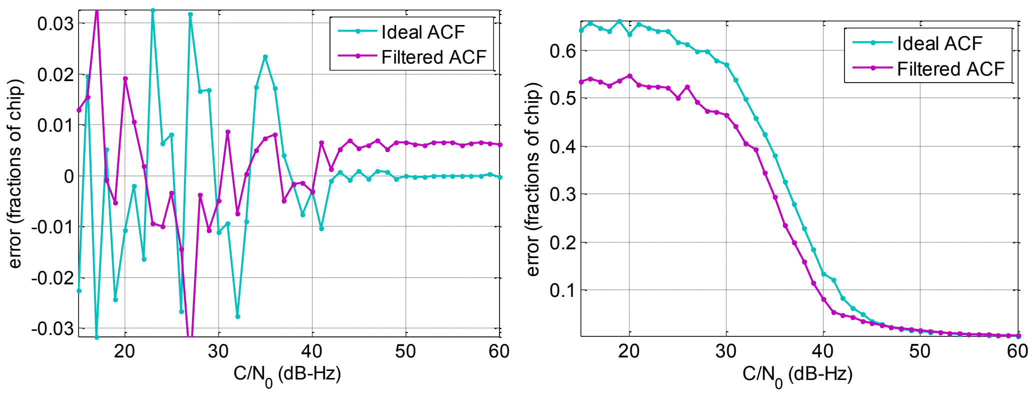
Figure 5.21: Code delay accuracy versus initial delay for different C/N_0.

Figure 5.21a shows the code delay accuracy versus the initial delay estimate used to initialize the algorithm. It is interesting to notice that the delay estimate has a bias, dependent on the initial code delay. The variance of the estimate is almost zero, since the noise is almost absent Also for small delays, for example 0.025 chips, the error in the estimate is equal to 0.008 chips, that are equivalent to 2.3 m in the pseudo-range. This is probably due to the fact that a limited number of samples of the correlation is considered. When considering 200 samples, i.e. much more than 2 chips, the result improves. However, this solution is not computationally affordable. By increasing the interpolation resolution no improvements are experienced By increasing noise to respectively 50 dBHz and 40 dBHz, the accuracy decreases, as shown in Figure 5.21b. In other words, the effect of noise overcomes the bias for low C/N_0.

In addition, it has to be noted that the trend is periodic, with period equal to 0.1 chips, which is the resolution of the correlation function. By changing this value the plot changes accordingly. The nulls correspond exactly to 0, 0.1, 0.2, chips, confirming the fact that the error is a limitation of the interpolation technique. When the delay is a multiple of the chip spacing, the error is minimum. In other cases, the interpolation solution is less accurate. The error could be computed analytically summing up all the terms of the summation cut off; in fact, in theory, the summations should go from $-\infty$ to $+\infty$. In principle this should be negligible, but it could affect the final result. It will be shown in the following that by increasing the number of points used to compute the interpolation, i.e. by spanning an interval larger than 2 chips, results improve.

5.3.3 Effect of the C/N_0

It is evident that by decreasing the C/N_0 the accuracy decreases. Figure 5.22 shows the mean and the standard deviation over 1 000 Monte Carlo simulation runs, in the presence and in the absence of the front-end filter and with code delay equal to zero. The error in fraction of chip is lower than 0.01 chips, corresponding to 3 m, for $C/N_0 > 52$ dBHz. The filtered version presents a bias of about 0.007 chips, but exhibits a lower standard deviation.



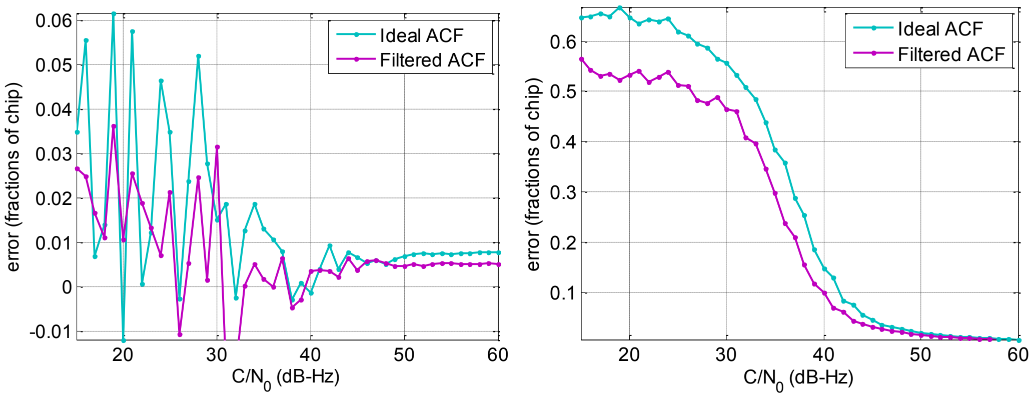
(a) Mean value **(b)** Standard deviations

Figure 5.22: Code delay accuracy versus C/N_0.

The same test is repeated and reported in Figure 5.23 for an initial delay equal to 0.025 chips. The result in terms of standard deviation is slightly worse, as expected. Moreover it is interesting to notice the same bias of 0.008 chips for the ideal case.

DLL comparison

The same effect is analysed performing Monte Carlo simulations with the snapshot algorithms described above and using N-FUELS generated signals at different C/N_0. In order to evaluate the goodness of the results, the curve obtained with Monte Carlo simulations is compared to the



(a) Mean value **(b)** Standard deviations

Figure 5.23: Code delay accuracy versus C/N_0 and delay equal to 0.025 chip.

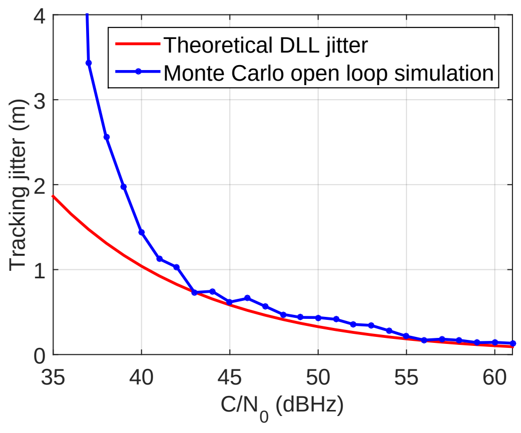
theoretical DLL tracking jitter [17]. The tracking jitter of a closed-loop DLL is a measure of the amount of noise transferred from the input signal to the output of the PLL on the final phase estimate. It is the first measure of a loop performance and allows one to quantify the impact of the thermal noise on the PLL. The tracking jitter is a normalized version of the noise standard deviation, taking into account the bandwidth of the complete loop. It can be expressed (in meters) as:

$$\sigma_{\mathbf{DLL}} = \sqrt{\frac{B_n}{2C/N_0}\frac{R_c}{2B_{\mathbf{fe}}}\left(1+\frac{1}{T_c C/N_0}\right)} \cdot \frac{c}{R_c}, \tag{5.3}$$

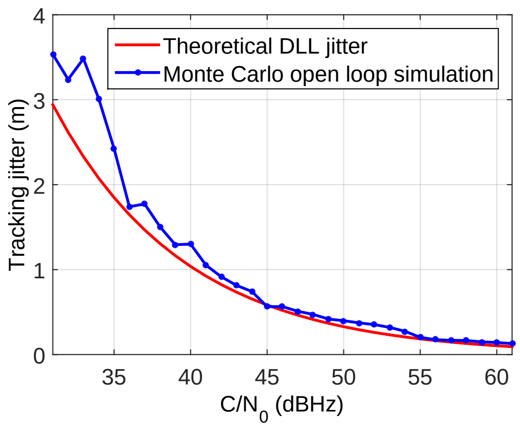
where B_n is the code loop noise bandwidth, R_c is the chipping rate, $B_{\mathbf{fe}}$ is the single sided front-end bandwidth, T_c is the coherent integration time and c is the speed of light. Nevertheless, it is not trivial to compare results obtained with two different techniques, because of the different quantities and variables involved. In particular, In order to liken the results, a E-L spacing equal to $D = 0.2$ chip is chosen, to emulate the 10 samples per chip correlation spacing of the open/loop multi-correlator snapshot architecture. In addition, it has to be said that a DLL has a loop filter, the main task of which is to improve the delay estimate, reducing the noise present at the output of the discriminator. A filter bandwidth is defined when designing a DLL, typical values for the DLL are around 1 Hz. This means that the code delay estimate is the result of average over about 1 s of samples, approximating the loop filter with a Finite Impulse Response (FIR)) filter. On the contrary, open-loop estimation gives as output non averaged results, resulting in an obvious degraded accuracy. To overcome the issue, the code delay error values of the Monte Carlo simulation are filtered with a moving average filter. By averaging 0.5 seconds of data (e.g. $L = 31$ values spaced 16 ms), an equivalent closed-loop bandwidth of about 1 Hz is obtained, as reported in the following equation:

$$B = \frac{1}{2LT_c} = \frac{1}{2 \cdot 31 \cdot 0.016} \simeq 1\ \text{Hz}. \tag{5.4}$$

In particular a GPS L1 C/A signal is considered, affected by constant Doppler frequency equal to zero for the observation period, to avoid the effect of dynamics. The following figures show the standard deviation of the code estimation error, i.e. the difference between the estimated code delay and the true one, expressed in meters (pseudo-range error standard deviation) for different C/N_0 values.



(a) $T_c = 1$ ms, $T = 16$ ms



(b) $T_c = 4$ ms, $T = 64$ ms

Figure 5.24: Comparison between code delays estimation accuracy, $B = 1$ Hz, $D = 0.2$ chip.

In particular, in Figure 5.24a, a coherent integration time equal to 1 ms and 16 non coherent sums are considered, while in Figure 5.24b a coherent integration time equal to 4 ms and 16 non coherent sums, spanning a total time $T = 64$ ms, are considered. The second case is more representative of Galileo OS signals, even if the simulation is carried out with GPS signals. In both cases the Monte Carlo simulation results are extremely good for high C/N_0. The code delay error estimate is slightly higher than its equivalent in the DLL formulation. The open-loop estimation error notably increases in the first case below 40 dBHz due to strong outliers, the probability of occurrence of which depends on the C/N_0. In fact, this effect is smoothed in the second case, where the coherent integration time is 4 times larger, thus improving the SNR.

Nevertheless, it has to be pointed out that the comparison between open-loop multi-correlation approach and closed-loop DLL is difficult and approximate, since the parameters involved are different, and the results are only qualitative.

5.3.4 Effect of the coherent integration time

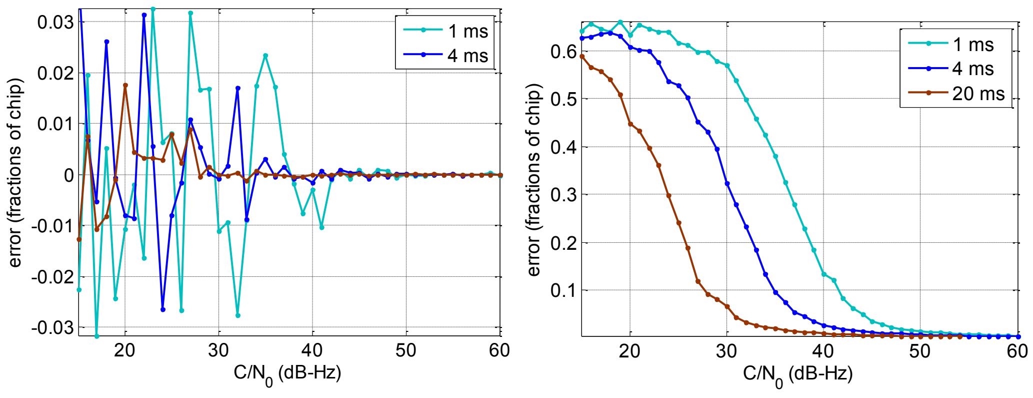
By increasing the coherent integration time the effect of noise is reduced, thanks to the averaging process, and the results improve, as proved by Figure 5.25. At 20 ms the standard deviation is lower than 0.01 chips for a C/N_0 larger than 40 dBHz.

At high integration times and in the presence of a front-end filter the results are even better. The values of the pseudo-range error of Table 5.10 lower up to 2 m for a 10 MHz bandwidth when integrating for 10 ms.

5.3.5 Effect of the chip spacing

As already said, the chip spacing controls the periodicity of the code delay estimate. Figure 5.26 shows the trend of the code delay error versus the code delay for the three different resolution modes (1/10 chip, 1/6 chip and 1/2 chip), in the absence of the front-end filter and for a signal without noise. It is evident that the larger the resolution the better the accuracy.

However, this affects only the bias. In fact, in a signal affected by noise, 50 dBHz, and with zero delay, the difference in terms of standard deviation between the modes is negligible, as depicted in Figure 5.27.



(a) Mean value **(b)** Standard deviation

Figure 5.25: Code delay accuracy versus C/N_0 for different coherent integration times.

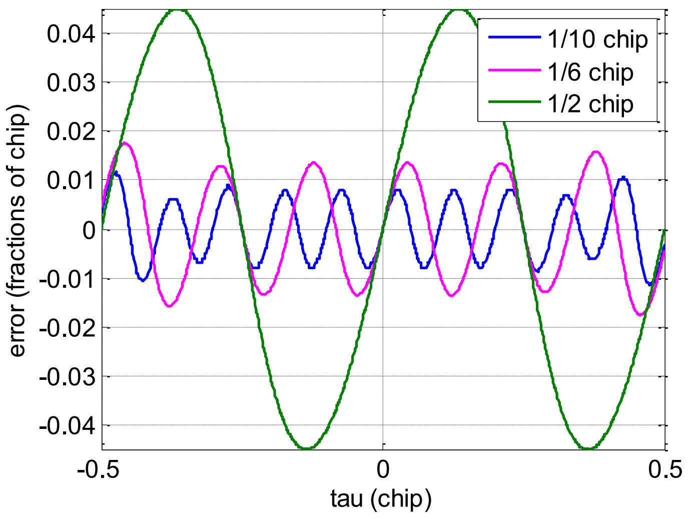


Figure 5.26: Effect of the chip spacing in the code delay accuracy.

Figure 5.28 proves that the maximum error reduces to 0.001 chip, equivalent to 30 cm, when increasing the resolution to 1/100 of a chip, but this is no longer computationally affordable, since it requires a number of correlators 10 times larger.

5.3.6 Effect of the resolution of the interpolation curve

The effect of the resolution of the interpolation is significant. Figure 5.29 reports the error in the estimate for resolutions equal to 1, 10, 100 and 1000 samples per correlation function point; in particular, 1 sample per point is equivalent to the case in which interpolation is not performed. It can be noted that for values larger than 100 samples per correlation function point the effect becomes negligible.

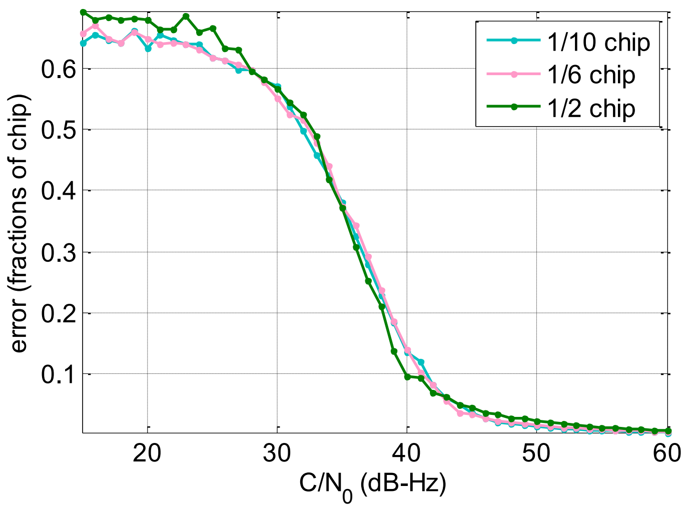


Figure 5.27: Effect of the chip spacing in the code delay accuracy, standard deviation for different resolution modes.

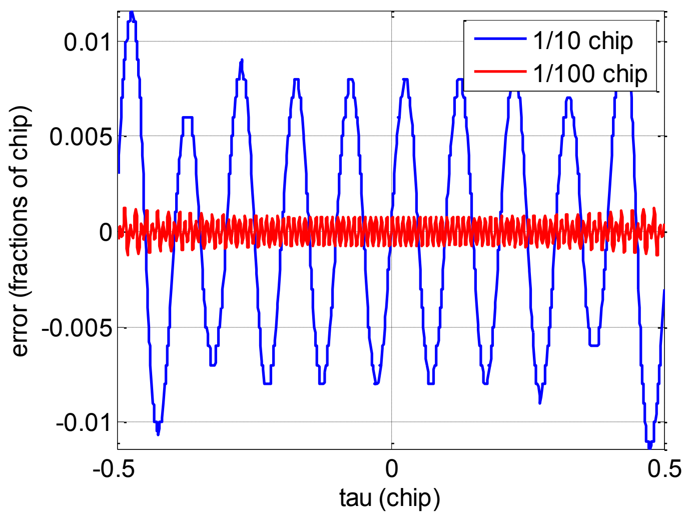


Figure 5.28: Code delay accuracy for extremely large resolution (100 samples per chip).

5.3.7 Effect of the number of points of the correlation function used to interpolate

Depending on the number of points around the peak of the correlation function used for computing the sinc interpolation, different results are achieved. Theoretically, the more points are considered, the most accurate is the sinc reconstruction. Indeed an infinite number of points would be theoretically required.

It has been proved with a semi-analytical simulation that when a few points are used there is a bias. In the case of GPS L1 signal (BPSK modulation), at high C/N_0 (80 dBHz) and at super-high resolution (10 samples per chip) at least 15 points around the correlation peak are necessary to reduce the bias, as shown in Figure 5.30a, the bias is completely removed when 21

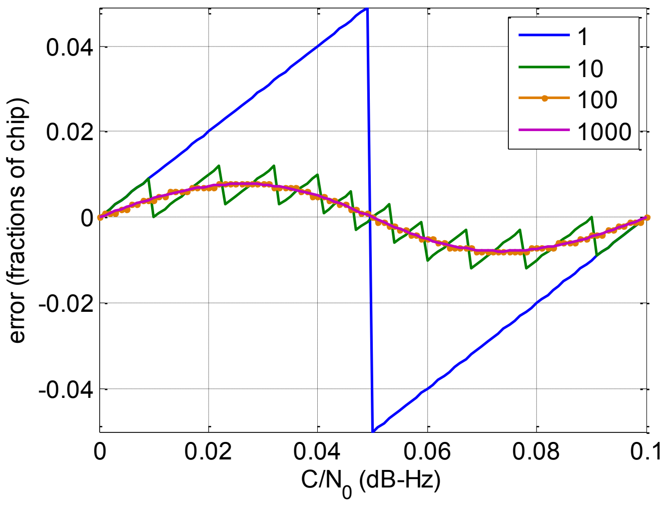
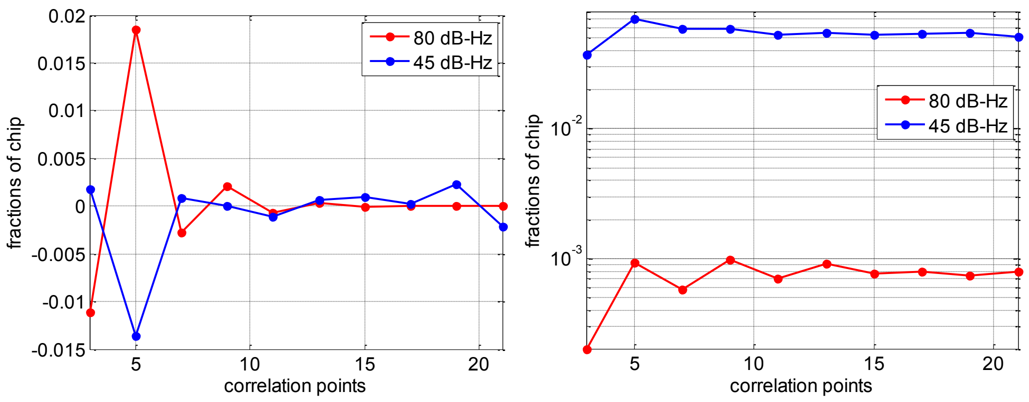


Figure 5.29: Code delay accuracy for different resolution of the interpolation curve.

points, corresponding to 2 chips, are used. At the same time, the standard deviation (shown in Figure 5.30b) increases when increasing the number of points, because of noise. Nevertheless its value is bounded to about 0.008 chips, since the noise is correlated and weighted by the sinc. For the same signal at 45 dBHz the results are worse: the bias is not removed, even for 21 points, and the variance is more than one order of magnitude larger, around 0.05 chips. It can be proved that also the interpolation resolution, i.e. the sampling frequency of the sinc curve used to interpolate, can impact on the accuracy.



(a) Mean value **(b)** Standard deviation

Figure 5.30: Accuracy of the code delay estimate vs. number of correlation points used.

5.3.8 Effect of the cross-code correlation

It is noted that also the simple intra modulation noise between the data and the pilot code in Galileo E1BC channel creates an error in the pseudo-range estimation. In particular the BOC(1,1) modulation is considered. A difference of about 14 cm in the code delay estimate is introduced with respect to the case in which only the data signal is generated with N-FUELS, as reported in Figure 5.31.

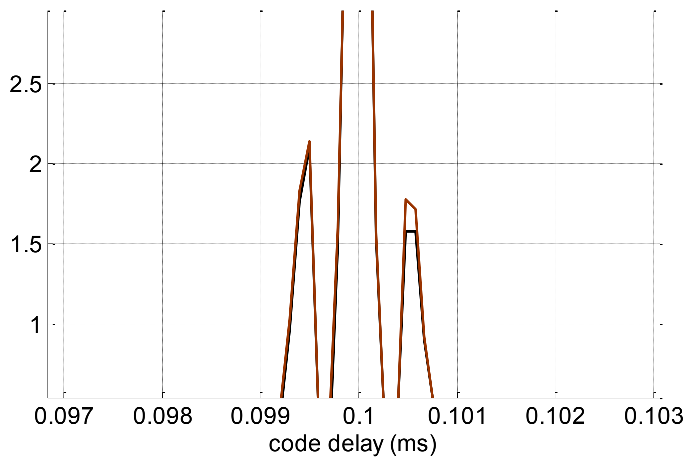


Figure 5.31: Intra-correlation error in Galileo E1BC.

5.4 Doppler frequency estimation

In this section, first some results obtained with the frequency estimation techniques described in Section 4.1.3 are reported. Then, results obtained with the innovative DFFT-based estimation technique are presented.

5.4.1 Comparison of different estimators

The frequency estimation accuracy has been computed for the different techniques described in Section 4.1.3. The most interesting results are reported hereafter, showing the frequency error with respect to the true Doppler frequency depending on different parameters, such as the initial frequency misalignment and the C/N_0.

Before presenting the results, the problem of the initial frequency misalignment is briefly recalled. In order to estimate the Doppler frequency with MLE FFT techniques, a rough estimate of the Doppler frequency has to be available. As described above, the FFT is able to represent the signal frequency components in frequency range defined in (4.12). This introduces a first level of dependence on the so called wipe-off frequency. If the difference between the wipe-off frequency and the real frequency does not lie in this range, the technique leads to wrong results. Admitted that this first requirement is satisfied, it has been shown that the performance of the algorithm changes depending on the frequency difference also within the FFT bin. So a periodicity of δf (see (4.13)) can be seen in the results. For this reason, in all the simulation reported in the following, the interval $[-15\text{ Hz}; +15\text{ Hz}]$ is considered. In order to simulate the initial frequency error, the same GNSS signal, with Doppler frequency equal to 0 Hz is considered, but the wipe-off frequency is changed between -15 Hz and $+15$ Hz, to account for any possible position inside the bin.

The Doppler frequency estimate average error, standard deviation and Root Mean Square (RMS) are computed with a Monte Carlo simulation using an N-FUELS simulated Galileo signal with the parameters reported in Table 5.11.

Table 5.11: Parameters of N-FUELS simulation for frequency estimation performance assessment.

Signal length	7 s
IF carrier frequency	4.1304 MHz
Sampling frequency	16.3676 MHz
Modulation	Galileo E1B BOC
Doppler frequency	Fixed, 0 Hz
Noise	Varying, range 20 - 50 dBHz, absent

The snapshot frequency estimation algorithms considered in the first phase of the tests are:

- FFT only, no interpolation (Section 4.1.3);
- Sinc interpolation (Section 4.1.3);
- Interpolation based on the correction term ΔP (Section 4.2.3);
- Interpolation based on nearly MLE (Section 4.1.3);
- Interpolation based on quadratic fit (Section 4.1.3)

Absence of noise

First some results in the case ideal signal conditions are presented, corresponding to open sky and good channel propagation conditions. In this case the noise is absent and only the useful signal component is generated. By exploiting results of this simulation it is possible to derive information about how the different estimators behave with respect to the initial frequency error, i.e. to the position of the Doppler frequency inside the FFT bin. In addition, bias and standard deviation of the estimators is evaluated.

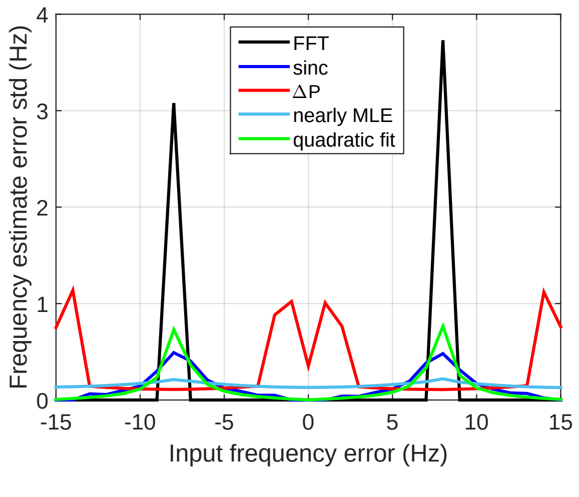


Figure 5.32: Average frequency estimate error for a signal without noise.

Figure 5.32 shows the mean error for different initial frequency errors. The black curve corresponds to the results of FFT only processing. The error confirms the theoretical expectations, it is minimum when the frequency error is equal to 0, and to multiples of the bin size Δf, because one FFT point falls exactly on the correct value of the Doppler. At the same time, it is maximum when the true Doppler falls in the middle between two FFT points, i.e. $-\Delta f/2$, $-\Delta f/2$, and so on.

The blue curve corresponds to the sinc interpolation. Results of the FFT are interpolated and some improvement is seen, as already detailed theoretically in Section 4.1.3. In particular, when the frequency misalignment is close to 0 and to multiples of Δf, the performance is similar to the case of FFT only processing. When moving far from these points the error increases, despite being lower than the FFT only case. It can be proved that this is a limit of the sinc interpolation technique. In fact, a sinc interpolation is optimal for band-limited signals, but doesn't apply perfectly on the shape of the BOC correlation function. It has to be noted that in this case 3 points of the FFT are used for computing the interpolation. In addition, increasing the resolution of the interpolating curve, the results can slightly improve, at the expenses of a larger computational load.

The same effect can be seen in the case of the quadratic fit, green curve; this approach is theoretically similar to the sinc interpolation case. The error is slightly lower, at the expense of a larger computational load.

The cyan curve, is the average error for the method denoted as nearly MLE. It is trivial to notice that the error is extremely lower than in the three cases reported above. In order to appreciate it, an enlargement is depicted in Figure 5.33. Still, it is possible to see the same trend of the FFT only technique, with smaller errors for input frequencies errors multiples of Δf.

Finally, the red curve corresponds to the case of interpolation based on the correction factor ΔP. The result is much better than the first three methods analysed and slightly better than the method based on nearly MLE. In particular, it can be seen that the error is almost constant for each initial frequency.

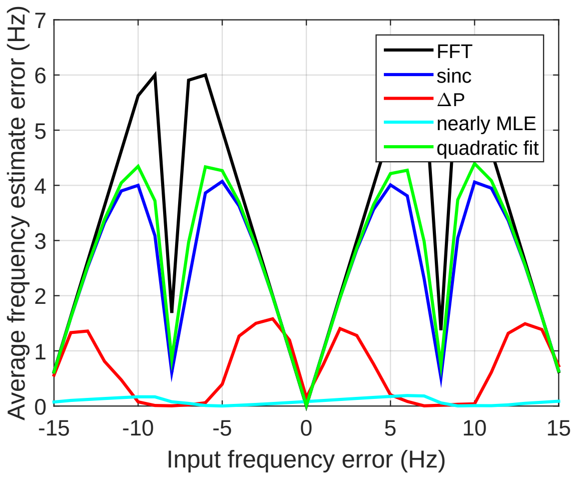


Figure 5.33: Zoom on the average frequency estimate error for a signal without noise.

According to these results, the method based on the discriminator was chosen as candidate for frequency estimation, and then successfully expanded and improved, as described in Section 4.2.

However, it is important also to have a look on the standard deviation of the estimates. Figure 5.34 shows the standard deviation of the 5 techniques analysed in the same simulation. As expected, the standard deviation of FFT only processing is null. The sinc interpolation presents a large standard deviation in correspondence to $-\Delta f/2$ and to $+\Delta f/2$, probably due to a numerical approximation problem, connected to the resolution of the interpolation curve. The quadratic fit and the nearly MLE methods exhibit a similar profile for the standard deviation, with a trend

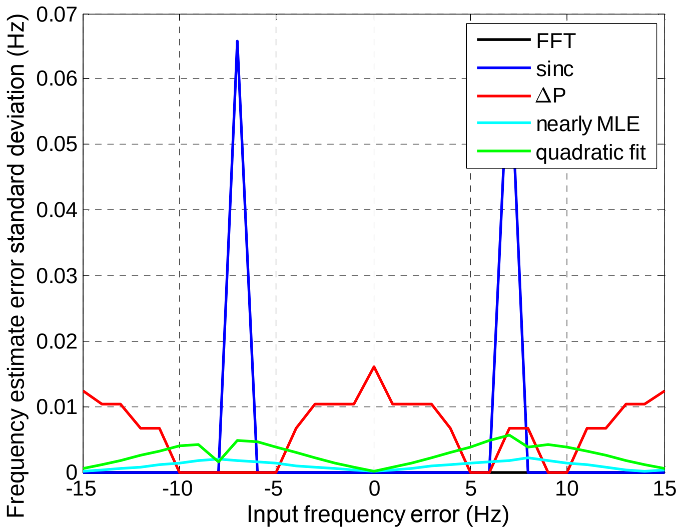


Figure 5.34: Frequency estimate error standard deviation for a signal without noise.

similar to the one of the average error. On the other hand, the correction factor method exhibits the larger standard deviation, in particular for values of the frequency misalignment close to 0 and to multiples of Δf; this fact anticipates the behaviour of the technique for low C/N_0, which will be described in the following. Better analysis of the variance will be given for the cases in the presence of noise.

Nevertheless, in order to properly analyse the performance of each technique, it is necessary to take a look at the computational load required. It is well known that the FFT has a high computational burden; then the majority of the time elapsed is used by the Matlab function performing the Discrete Fourier Transform (DFT).

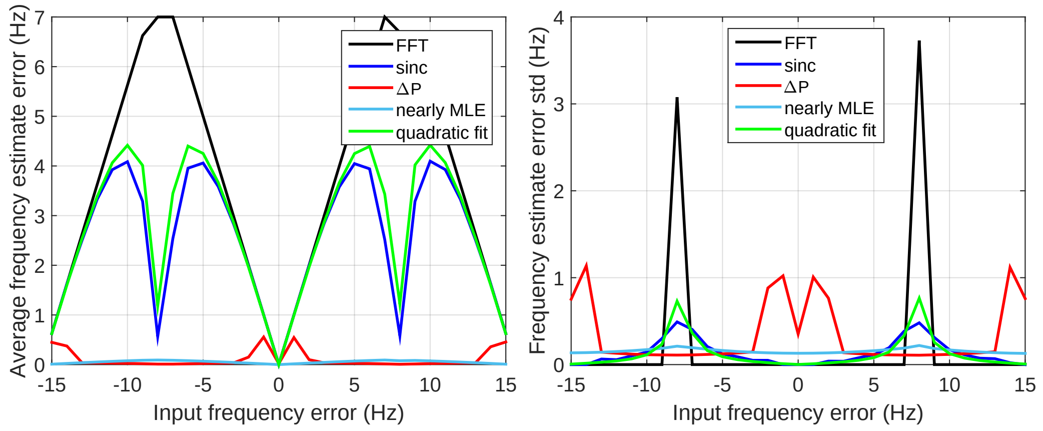
Table 5.12: Run time in s for 100 000 iterations of each single algorithm.

Technique	FFT	sinc	ΔP	Nearly MLE	Quadratic fit
Execution time	8.108 s	19.659 s	10.959 s	10.717 s	15.754 s
	-	+240%	+135%	+132%	+194%

Table 5.12 shows the elapsed time for the five methods described and considered in the simulation. Since this figure of merit strongly depends on the hardware of the PC, it is more interesting to consider the proportion between the results than the absolute values. It is evident that the fastest method is the FFT only implementation, given that the other four methods just add more processing to the first. The sinc interpolation more than doubles the execution time, as expected. The quadratic fit can be considered as a computationally simplified sinc interpolation; in fact it achieves similar results with a lower computational burden. ΔP correction factor and nearly MLE based methods have very similar execution times, worsening the FFT only method of a factor of 30-35%.

Results for $C/N_0 = 50$ dBHz

As soon as the C/N_0 decreases to 50 dBHz it is possible to see some different result. Figure 5.35a shows the average frequency error is this case. Four techniques (FFT only, sinc, nearly MLE and quadratic fit) exhibit almost the same performance as in the case of absence of noise. On the contrary, the ΔP correction factor method has some important degradation for input frequency errors of ± 1 Hz, ± 2 Hz, ± 14 Hz and ± 15 Hz. This confirms a weakness of the algorithm, as described in Section 4.2.3.



(a) Average **(b)** Standard deviation

Figure 5.35: Frequency estimate error standard deviation for a signal with $C/N_0 = 50$ dBHz.

Figure 5.35b shows the standard deviation of the 5 techniques analysed in the same simulation. The trend is similar to the one reported in Figure 5.34 for the case with no noise. The main differences are the larger standard deviation for FFT, sinc and quadratic fit techniques in correspondence of $\pm\Delta f/2$, and the larger standard deviation for the correction factor techniques around the bin centre, confirming the facts described above. It is interesting to note that nearly MLE estimator has an almost constant standard deviation.

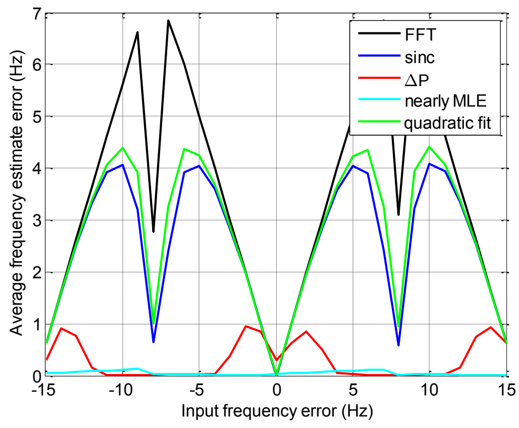
Results for $C/N_0 = 41$ dBHz

Results obtained increasing the noise to $C/N_0 = 41$ dBHz and considering 100 simulations are shown hereafter. Figure 5.36a depicts the average frequency estimate error. In all the 5 cases the error slightly increases, maintaining the same trend with respect to the input frequency error. For ΔP and nearly MLE based techniques the average error is always below 1 Hz. In particular, the ΔP technique exhibits errors lower than 0.05 Hz in the region around $\Delta f/2$. Figure 5.36b shows the standard deviation in the same case. It increases, if compared to the case at 50 dBHz, especially in the most critical zones.

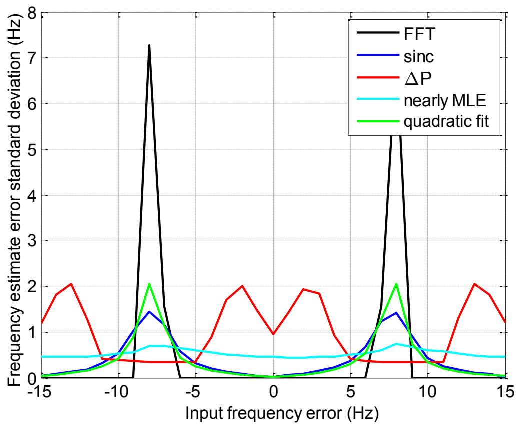
Results for $C/N_0 = 53$ dBHz

Finally results at 35 dBHz are reported (100 simulations).

Figure 5.37a shows the average error. Again, the trend is similar to the cases of higher C/N_0. ΔP and nearly MLE-based techniques clearly exhibit better results than the other techniques. The main effect of increasing the noise is that the frequency region in which the ΔP technique has a very low error decreases: while for $C/N_0 = 50$ dBHz very good results are obtained in the frequency range 3–13 Hz, this range reduces to 4–11 Hz for $C/N_0 = 41$ dBHz and to 6–10 Hz for $C/N_0 = 35$ dBHz. This is due to the fact that a larger noise increases the probability of choosing a correct second peak and then of making an error in determining the correction sign.

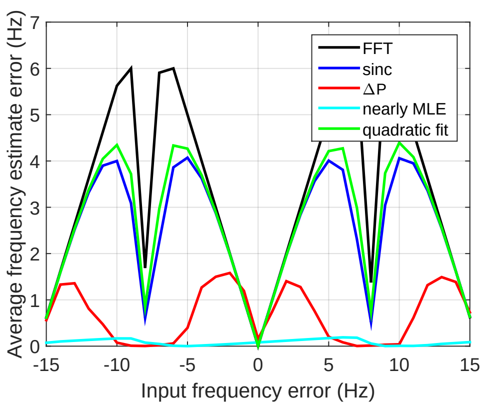


(a) Average

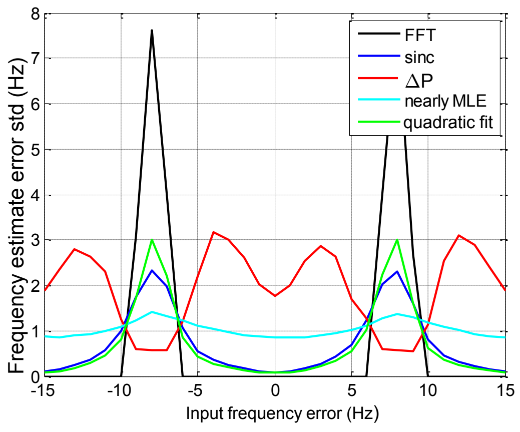


(b) Standard deviation

Figure 5.36: Frequency estimate error standard deviation for a signal with $C/N_0 = 41$ dBHz.



(a) Average



(b) Standard deviation

Figure 5.37: Average frequency estimate error for a signal with $C/N_0 = 35$ dBHz.

Figure 5.37b shows the standard deviation. It is interesting to notice that the 5 techniques have opposite behaviour when the input frequency error is around an FFT point of far from it. In general this rule applies: the smaller is the standard deviation for values around 0 and $\pm\Delta f$, the larger is the standard deviation around $\pm\Delta f/2$, as in the case of FFT only, sinc interpolation and quadratic fit. The larger is the standard deviation for values around 0 and $\pm\Delta f$, the smaller is the standard deviation around $\pm\Delta f/2$, as in the case of ΔP technique. The case of nearly MLE based estimation follows this rule, even if in a lighter way, and is the one offering the best variance in average.

Results for $f = 8$ Hz

In the following simulations, a value of the input frequency error is fixed, in order to study the dependence on the signal strength. The input frequency errors providing the best performance

(8 Hz according to the results obtained before) is chosen, assuming the fact that for other frequency misalignments similar results can be obtained with the DFFT technique.

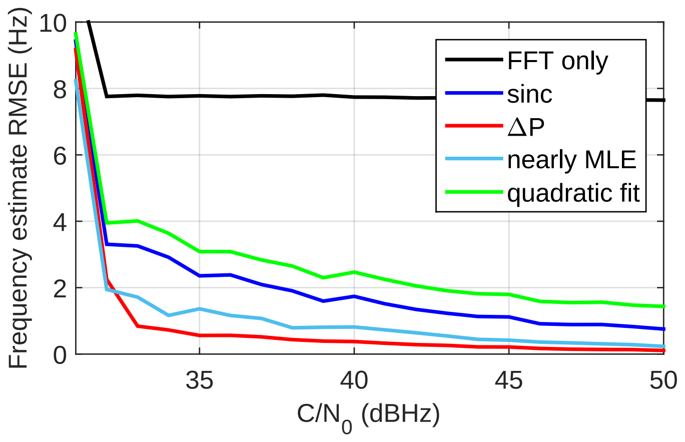


Figure 5.38: Frequency estimate RMSE for different techniques vs. C/N_0 and for a fixed input frequency misalignment, $T_c = 64$ ms.

Figure 5.38 shows a comparison of the Root Mean Square Error (RMSE) for the 5 techniques for different values of the C/N_0. It is interesting to point out how in this case the ΔP technique outperforms any other algorithm implemented, also thanks to its good behaviour for the input frequency error chosen. As expected, the higher the C/N_0, the lower the RMSE. A well-known drawback of FFT-based techniques, documented in literature and concerning all MLE based techniques, is the so called threshold effect [70, 71]. Below a certain C/N_0 the frequency estimate computed with MLE suffers of an error, because of the appearance of large noise spikes in the frequency search domain, yielding to the identification of a wrong FFT peak as correct peak. This has the effect to suddenly increase the average error and to invalidate any other interpolation method. This is evident in Figure 5.38 for $C/N_0 < 32$ dBHz A possible solution consist in increasing the integration time. Figure 5.39 shows the result for ΔP and nearly MLE based techniques, in the case of a total time equal to 128 ms and an FFT computed on 32 points. The C/N_0 threshold is reduced to 29 dBHz.

Semi-analytical estimation

At the same time a semi-analytic simulation has been performed to prove the consistency of the results of the ΔP technique. Two C/N_0 (35 dBHz and 50 dBHz) and two initial frequency errors (2.475 and 7.8135 Hz respectively the values giving the highest and lowest errors), are considered. The results for $T_c = 128$ ms are shown in Figure 5.40 and Figure 5.41 respectively. The consistency between the N-FUELS Monte Carlo simulation and the semi-analytical result is evident. Also in this case the threshold effect described before is evident; the RMSE increases with respect to the CRLB for $C/N_0 < 29$ dBHz.

5.4.2 Zero forcing and DFFT results

In order to assess the performance of the DFFT frequency estimation technique described in Section 4.2 and to compare it with the other techniques analysed, some semi-analytical and Monte Carlo simulations are carried out.

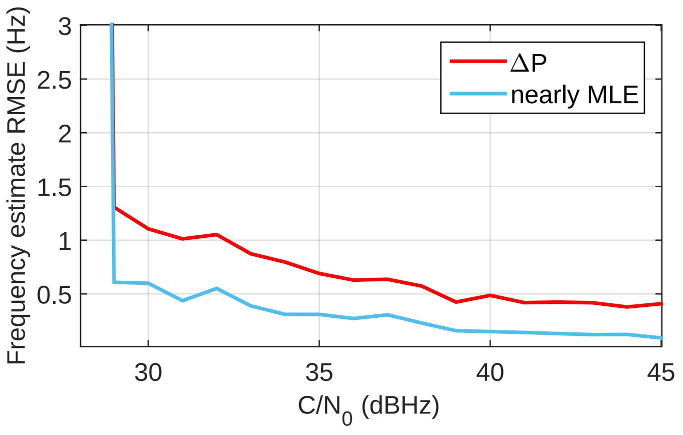


Figure 5.39: Frequency estimate RMSE for different techniques vs. C/N_0 and for a fixed initial frequency misalignment, $T_c = 128$ ms.

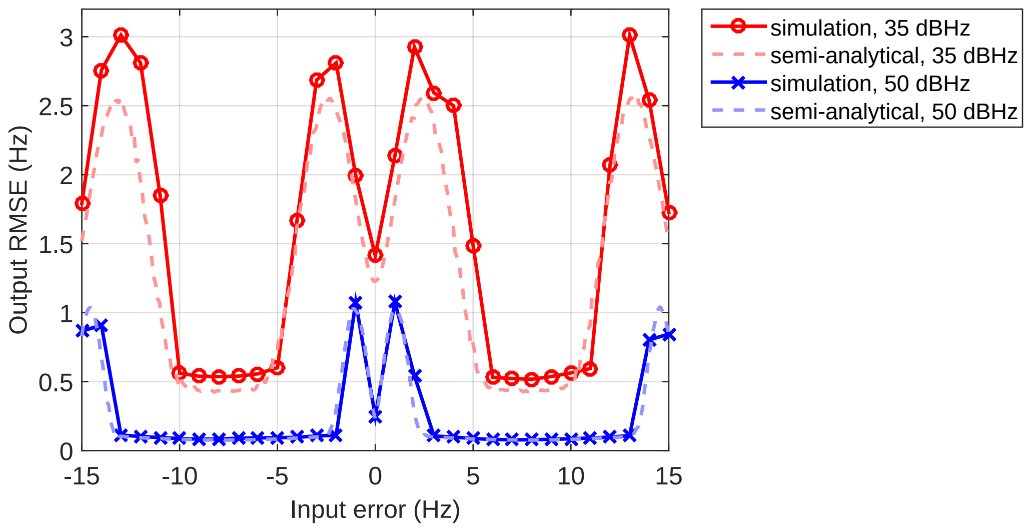


Figure 5.40: Doppler frequency estimate RMSE for different C/N_0 vs. the initial frequency error in the range $[-\Delta f + \Delta f]$. The plot is periodic for frequency values outside this range.

A few simulations are run with $C/N_0 = 45$ dBHz and 5 000 iterations, comparing the standard case ($p = 0$. i.e. the ΔP technique, with the notation used in the simulations of the previous section), the case with zero-forcing and $p = 2$ and the DFFT case with $p = 2$. Figure 5.42a shows that the error average absolute value in the DFFT case is comparable to the one of zero-forcing. On the contrary, Figure 5.42b shows that the error standard deviation in the DFFT case decreases with respect to the case with zero-forcing. In particular, it almost reaches the value of the non zero-forcing case for frequency values far from $\Delta f = 0$, whereas it clearly outperforms the other methods around $\Delta f = 0$.

The same results, but for $C/N_0 = 30$ dBHz are plotted in Figure 5.43. Similar considerations can be drawn: the DFFT technique offers the best performance in almost every condition. Concerning Figure 5.43a, it can be stated that the errors shown in the zero-forcing and DFFT cases

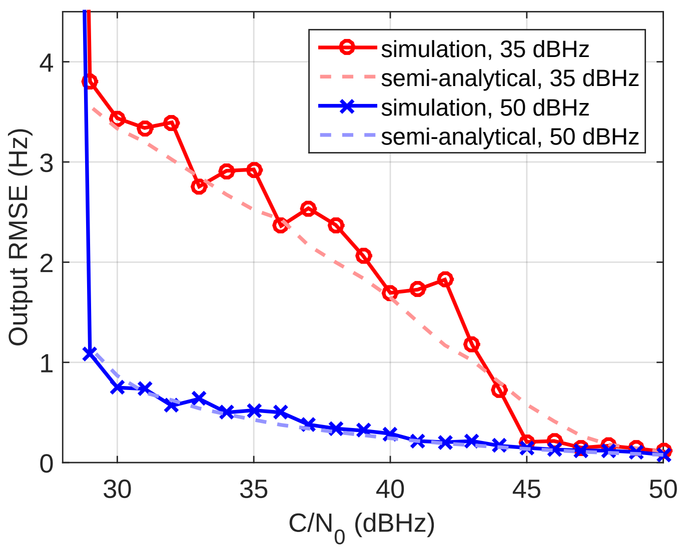


Figure 5.41: Doppler frequency estimate RMSE for different initial frequency error vs. C/N_0 in super high resolution.

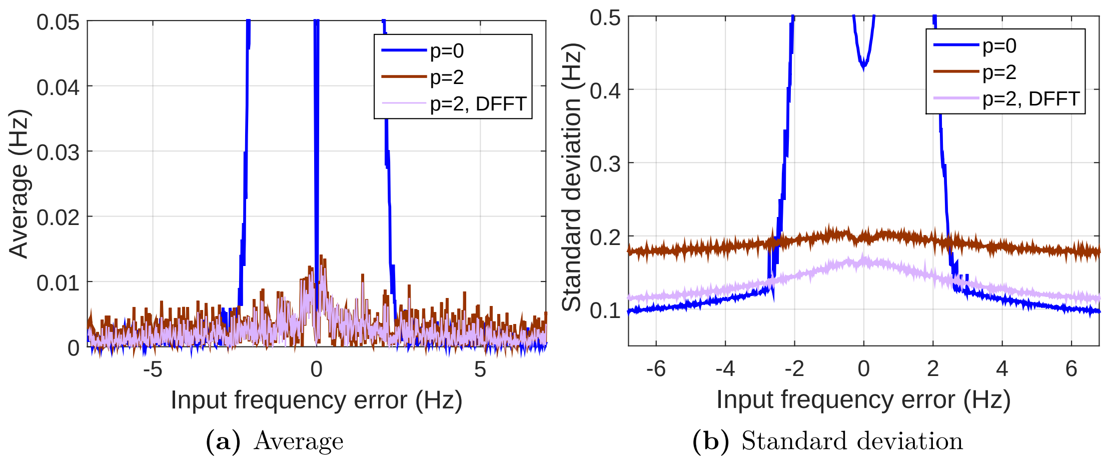
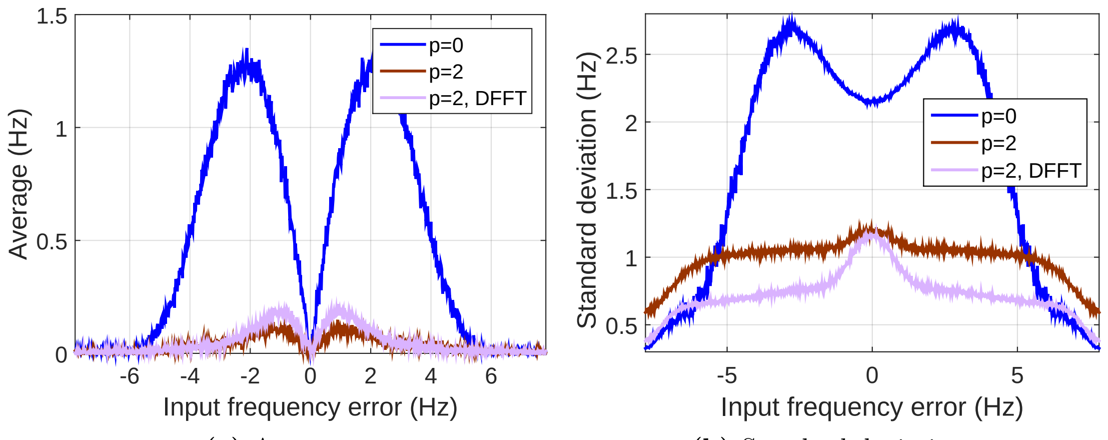


Figure 5.42: Frequency estimate error for different p and for the DFFT case, $C/N_0 = 45$ dBHz.

are due to the noise and to the shape of the *P-curve*, while in the standard case the bigger errors are driven by the wrong sign estimate.

Finally, a case for $p = 4$ is considered, but comparable results are achieved and are not reported. Summarizing, Table 5.13 reports mean and standard deviation for different cases. The solution with $p = 2$ and DFFT represents the best solution in every scenario tested. As expected, when $\Delta f = 6$ Hz, the case with no zero-forcing ($p = 0$) reaches the same performance of the case with $p = 2$ and DFFT, but clearly worsen for lower initial frequency errors. It is also confirmed that the case of zero-forcing with $p = 4$ underperforms the case with $p = 2$.



(a) Average (b) Standard deviation

Figure 5.43: Frequency estimate error for different p and for the DFFT case, $C/N_0 = 30$ dBHz.

Table 5.13: Summary of the accuracy of the different techniques analysed for two different C/N_0 values and for two different Δf, 5 000 iterations, values in Hz.

	$C/N_0 = 45$ dBHz				$C/N_0 = 35$ dBHz			
	$\Delta f = 1$ Hz		$\Delta f = 6$ Hz		$\Delta f = 1$ Hz		$\Delta f = 6$ Hz	
Technique	Mean	Std	Mean	Std	Mean	Std	Mean	Std
$p = 0$	0.59	0.98	**0.002**	**0.10**	0.80	1.52	0.01	**0.32**
$p = 2$	**0.003**	0.20	0.003	0.18	0.03	0.62	0.001	0.57
$p = 4$	0.02	0.45	0.011	0.45	0.08	1.36	0.16	1.22
DFFT ($p = 2$)	**0.003**	**0.16**	**0.002**	0.12	0.04	**0.48**	**0.001**	0.37
DFFT ($p = 4$)	0.005	0.17	0.003	0.12	**0.02**	0.53	0.008	0.38

5.4.3 Comparison with existing techniques

A comparison with existing techniques shall take into account several different aspects. In the following, the proposed technique is compared with the standard closed-loop PLL structure and with the theoretical CRLB. However, it has to be said that, given the fact that the architectures are extremely different, it is difficult to carry on a fair comparison.

Search space

The first point to compare is the size of the frequency search space. In the case of the standard PLL the search space is related to the loop bandwidth, which can assume values from a few Hz up to $20 - 30$ Hz. At the same time, the open-loop architecture proposed has a much larger search space (up to 250 Hz). This means that a much larger frequency uncertainty can be held. At the same time, a closed-loop architecture undergoes a transient, introducing a delay in the estimation procedure, which is absent in the open-loop scheme.

Performance analysis

Three techniques, namely the standard PLL (with $T_c = 64$ ms and different bandwidths), the standard FFT and the DFFT as described above are compared in terms of RMSE of the frequency estimate, using a GNSS software receiver. A normal acquisition stage is used to provide the initial rough estimates of the signal parameters; then a standard DLL is used in all cases to track the code delay. The parameters of the three techniques have been tuned in a comparable way. In addition, different initial frequency estimates, in the range $[-7\text{ Hz}; 7\text{ Hz}]$, are tested. An N-FUELS simulated Galileo E1 signal at 35 dBHz is tracked, and several iterations of the algorithm are considered. The results are reported in Figure 5.44.

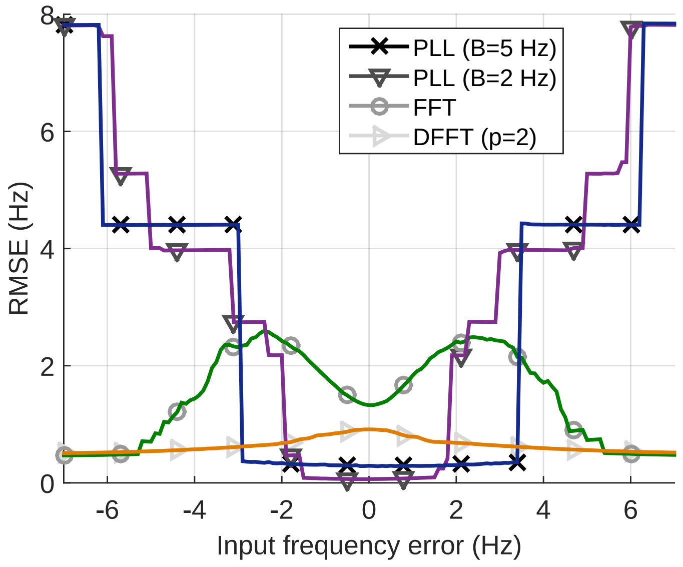


Figure 5.44: Comparison between standard PLL and DFFT technique in terms of RMSE.

It has been proved (but omitted in the figure) that for values of the PLL bandwidth larger than 6 Hz, the loop cannot track the signal, because of instabilities problems due to the large coherent integration time. When the bandwidth is reduced to 5 and 2 Hz, the PLL correctly tracks the signal, but only in a limited pull-in region, i.e. for an initial frequency error lower than about 3.4 and 1.7 Hz respectively, as depicted in the figure. Nevertheless, as expected, in this region the frequency estimate RMSE is very low. The FFT exhibits the behaviour described in Section 4.2.5 and confirms the outcome of the semi-analytical simulations. The DFFT algorithm implemented is the version with zero-forcing and $p = 2$, the results are consistent.

It can be said that the DFFT technique gives the best overall performance, considering the search space analysed. Its RMSE is always below 1 Hz, while in the case of a standard closed-loop, it is bounded to 0.3 Hz only in the central region, but reaches values larger than 4 Hz for larger input frequency errors.

Complexity analysis

The complexity can be evaluated in terms of execution time in the software implementation. The same simulation described in the previous section has been run and 10 seconds of data have been processed. The execution time in the three cases is comparable.

Robustness and sensitivity

The benefits of open-loop approach in terms of sensitivity and robustness were evaluated by the authors in [72]. In particular, the main advantage of the proposed technique is the absence of the risk of LOL. Indeed, whereas in a classic close loop scheme some signal nuisances can lead to LOL and then to a time-consuming re-acquisition stage, open-loop architectures just give an invalid measure for a brief time interval. As soon as the signal quality is restored, the DFFT scheme gives good frequency estimates, without further transients.

5.4.4 Comparison with Cramer-Rao lower bound

10 000 realizations of the semi-analytical process are considered, for a C/N_0 spanning a range from 20 to 50 dBHz; the standard case, the case with zero-forcing and $p = 2$ and the DFFT case with $p = 2$ are considered.

Moreover, the CRLB is proposed as a benchmark to compare the performance of the different algorithms. The CRLB for frequency estimation is defined as [49]

$$CRLB = \frac{12}{(2\pi)^2 \cdot (C/N_0)/B_{\text{fe}} \cdot f_s \cdot T_{\text{tot}}^3}, \tag{5.5}$$

where C/N_0 is expressed in linear units and B_{fe} is the single sided front-end bandwidth, equal to $f_s/2$ in this case. The CRLB is a useful metric to assess the performance of an estimator, in terms of lower bound of the RMSE. An algorithm able to approach this bound can be considered a valid alternative to standard receiver architectures.

The results are reported in Figure 5.45. The red curves represent the RMSE for Δf close to the FFT bin centre ($\Delta f = 1$ Hz), whereas the blue curves correspond to the case of larger frequency errors ($\Delta f = 6$ Hz). As expected, the second set of curves, is closer to the CRLB; in particular, the curve obtained by just applying a standard FFT for $\Delta f = 1$ Hz gives the worst performance, for the reasons outlined above. On the contrary, the curve obtained with the same method but for $\Delta f = 6$ Hz is closer to the theoretical bound for high values of the C/N_0, i.e. where the sign error is very unlikely. As already outlined, the frequency error for the case of $\Delta f = 1$ Hz substantially improves when introducing zero-forcing. The same trend is observed when the initial frequency error is higher. Moreover, in both cases, the Double FFT technique slightly outperforms the single FFT technique, but only for $C/N_0 > 25$ dBHz. In general, it is possible to assume that both zero-forcing and DFFT techniques, for any initial frequency error, have the same performance, which is really close to the CRLB, confirming the validity of the techniques proposed, for values of the C/N_0 higher than 28 dBHz.

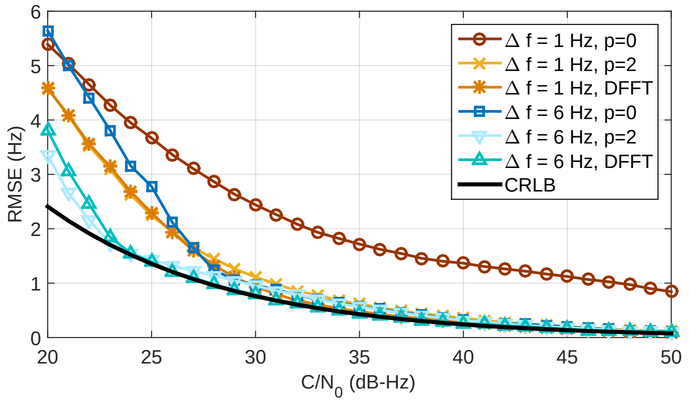


Figure 5.45: Doppler frequency estimate RMSE versus C/N_0, comparison between theoretical and simulated results.

Below this C/N_0 threshold, errors are larger, mainly because of the limit of the MLE FFT approach. The same effect arise in the code delay estimation process below 30 dBHz. It is indeed well known in literature that estimator performance degradation occurs beyond a certain noise power, because of the appearance of large noise spikes in the search domain [70]. This threshold effect can be overcome by further increasing the integration time, thus entering the framework of HS receiver. In this particular case, it is possible to increase the value of K to 32, so as to span a total accumulation interval equal to 128 ms, at the expenses of a larger computational burden.

5.5 Navigation message decoding

A software routine for the demodulation of the navigation message is present in the fully software receiver implemented. Its block scheme is depicted in Figure 5.46.

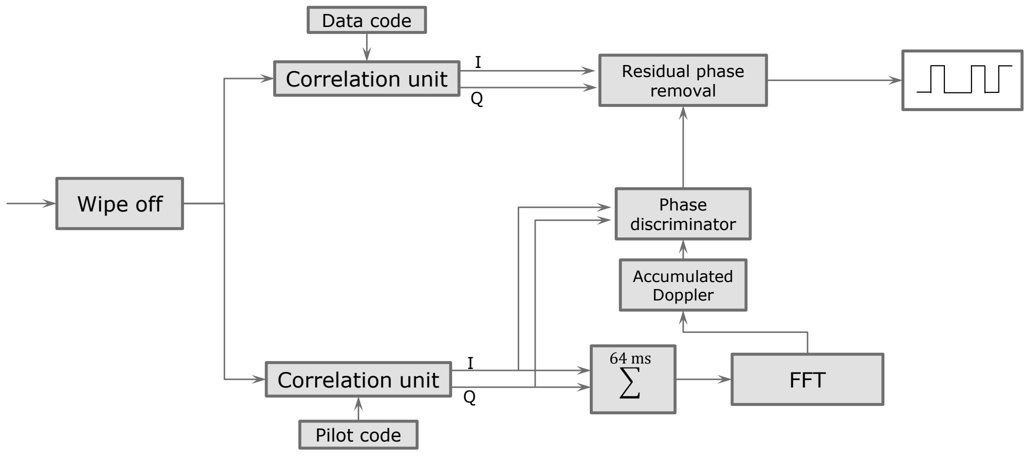


Figure 5.46: Block scheme of navigation message demodulation in the receiver architecture.

The message demodulation exploits the phase estimate computed in the data channel, as described in Section 4.1.5, using the code of Galileo E1B, to remove the phase from the signal and to estimate the message symbols. The navigation message data bits are obtained just taking the real part of I channel correlation values and multiplying them times the exponential $e^{-j\hat{\varphi}}$, in order to remove the residual phase oscillations, as shown in Figure 5.47. The bits can then be extracted with the sign operator. The bit transitions can be clearly seen, as depicted in the figure.

In the particular case considered in this example, Galileo E1C codes are used in both channels, just for test purposes, to check the validity of the algorithm, and without lacking of generality. This is evident from the data bits reported in the figure, corresponding exactly to the Galileo E1C secondary code.

5.6 Windowing results

The problem of the search space windowing has been addressed in Section 4.1.4. Briefly recalling, both code delay and Doppler frequency search spaces are reduced, in order to decrease the computational burden, provided that a good initial estimate is available. However, the true code delay and Doppler frequency can exit from the actual reduced search space, because of the signal dynamic. It is then necessary to update the search space, centring it around the true values,

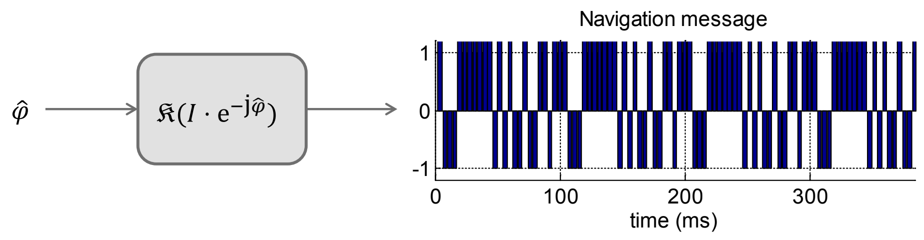


Figure 5.47: Navigation data bit extraction in Galileo E1B.

with a process denoted signal windowing. When performing windowing, a few points have to be considered.

- The update process cannot be performed at each iteration, because then most of the benefits of the open-loop approach would be loosen; moreover, it is really not necessary to update the search space at each iteration.
- Performing a wrong update is dangerous as much as not performing any update: if the estimates around which the search space is redefined are wrong, because of a wrong previous estimation, then invalid measures are obtained. It is necessary to rely on the information about the new values before applying them.
- Code delay and frequency search spaces redefinitions are independent and can be updated in different moments.
- Large dynamics or on demand processing (Section 4.3) can fast up the rate of evolution of the signal parameters and thus require a more frequent and careful windowing process.

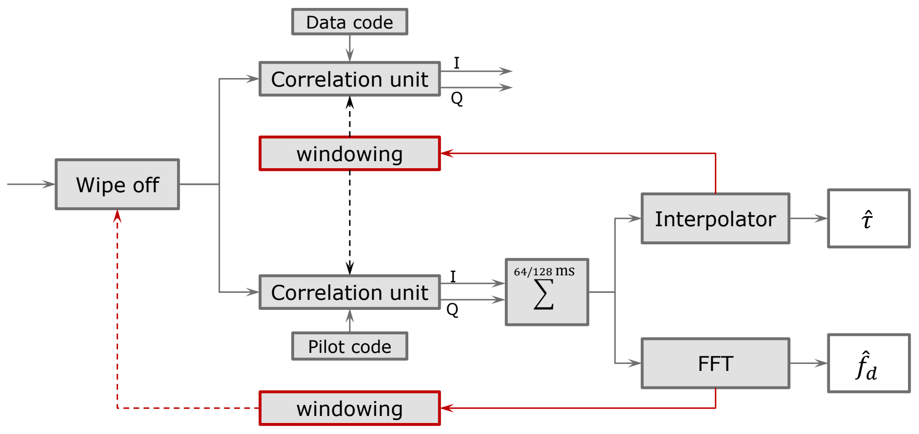


Figure 5.48: Block scheme of the windowing process.

For these reasons, windowing requires a dedicated software routine, based on the following points:

1. estimation of the quantity (frequency or code delay) at the current epoch;
2. evaluation of the reliability of the quantity;

3. average/filtering of the last quantity estimates;
4. determination of the necessity to update the search space;
   (a) definition of the new quantity around which centring the search space;
   (b) update of the information in the frequency wipe-off/code correlation blocks.

This is shown in Figure 5.48; in particular, operations 4.(a) and 4.(b) are performed only if operation 4. returns a positive answer. Only in this case the loop is closed (feedback represented by the dotted line).

5.6.1 Code delay windowing

As explained in Section 4.1, the reduced code delay search space size is equal to $KL/1023$ fractions of chip. In high resolution mode, this is equal to $12 \cdot 3/1023 = 0.0352$, i.e. about 3.5% of the full search space. The code delay search space is redefined as soon as the last estimate of the code delay, properly filtered to assure its reliability, enters the 10% area close to the border, as depicted in Figure 5.49. In this case a new value for the estimated code delay is fixed in the code correlation block.

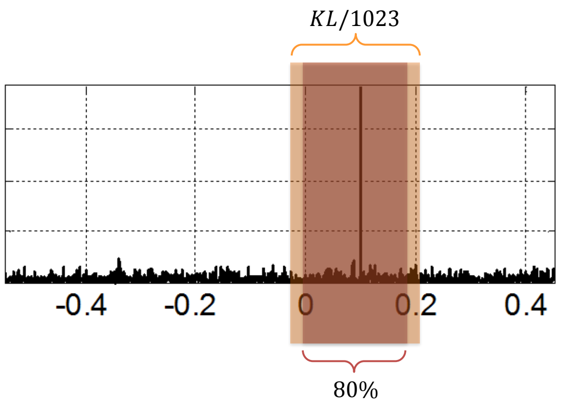
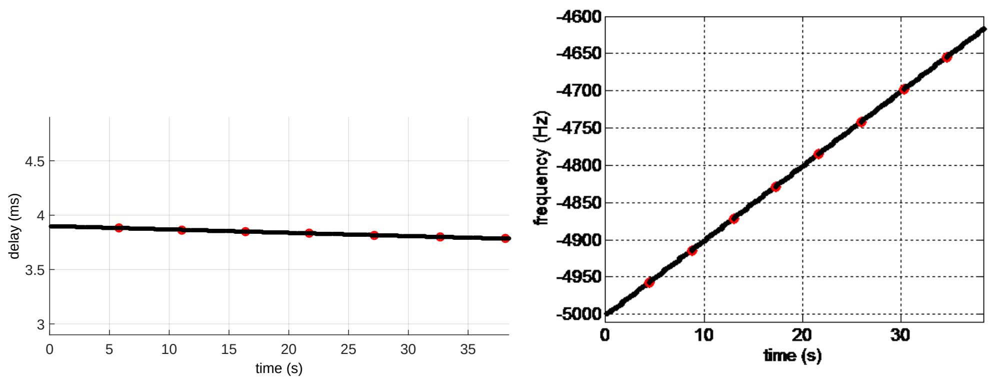


Figure 5.49: Code delay windowing limits.

In the following example an N-FUELS Galileo E1B signal without data bits is tracked for 38 s. The Doppler profile is set to a linear increase of 10 Hz/s, starting from −5000 Hz (particular values are chosen just to have a fast changing Doppler rate and code delay rate, to appreciate the effect of widowing in a small time scale). The search space is updated 6 times: after 11.0 s, 16.4 s, 21.7 s, 27.1 s, 32.6 s and 38.1 s, i.e. every about 5.4 s. Figure 5.50a shows the code delay estimated during all the simulation; the red dots correspond to the points in which the code delay used to centre the code search space has been changed.

5.6.2 Doppler frequency windowing

The Doppler frequency reduced search space size corresponds to $1/T_c$. The oscillator frequency is changed as soon as the filtered frequency estimate approaches the 20% area close to the border, as depicted in Figure 4.13. Figure 5.14 shows the frequency estimate in the same simulation as above; the red points correspond to the instants in which the wipe-off frequency is changed, as reported in Table 5.14.



(a) Code delay **(b)** Doppler frequency

Figure 5.50: Estimates in the case of signal windowing.

Table 5.14: Frequency windowing updates in a real simulation.

New f_D	-4957 Hz	-4871 Hz	-4828 Hz	-4784 Hz	-4741 Hz	-4697 Hz	-4654 Hz
time	4.4 s	13.0 s	17.3 s	21.6 s	26.0 s	30.3 s	34.7 s

5.6.3 Duty cycle windowing

When dealing with DC techniques, it is necessary that at any epoch, the true code delay and Doppler frequency lie within the reduced search space, otherwise the signal cannot be found. When tracking continuously the signal and in normal conditions this is not a big problem, because even in high dynamics the code delay rate and the frequency rate are limited and the estimates should be maintained well centred in the search space. On the contrary, when DC is employed, it is possible that during the sleep state the signal changes so much that the true code delay and/or the Doppler frequency exit from the search space. For example, assuming that the code search space covers 10 chips and that the actual code delay lies exactly in the middle of the search space, and assuming a relative velocity between user and satellite equal to 500 m/s [69], the code delay covers half the search space (towards right or towards left) in $10/R_c \cdot c/v \simeq 5.8$ s. Therefore the sleep period should be lower than 5.8 s to assure that the signal can be declared present. Otherwise it is necessary to predict the trend of the code delay and of the frequency and to define a new search space at each RTC wake-up. However this approach could decrease the algorithm robustness, since outliers in the frequency estimation would cause a wrong search space definition, potentially leading to a missed detection. Filtering techniques, with both the purposes of reducing the impact of noise and of predicting the evolution of the parameters are implemented.

5.6.4 Effect of Doppler rate

It has been proved (Appendix 7, [37]) that, for the Galileo constellation, the maximum Doppler shift rate, occurring when the satellite is at the zenith, and for a user moving at a constant speed of 100 km/h is bounded to 0.7 Hz/s. By fixing $K = 16$, a negligible frequency change within the integration period is obtained (less than 0.05 Hz). At the same time, when the user experiences accelerations, a Doppler frequency rate is introduced. Figure 5.51 depicts the average RMSE for a signal with $C/N_0 = 45$ dBHz, characterized by a different Doppler rate, from zero up to 15 Hz/s.

The DFFT technique with $p = 2$ is still able to estimate the frequency with a good accuracy even for a Doppler rate equal to 15 Hz/s.

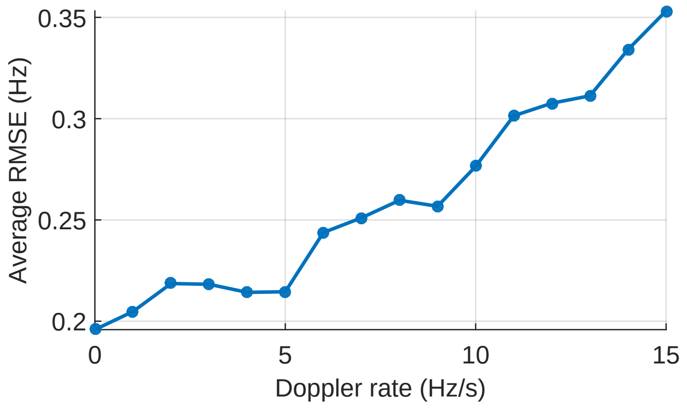


Figure 5.51: Average RMSE error for a signal at 45 dBHz and different values of the Doppler rate.

5.7 Duty cycle power saving techniques

An important driver for mass-market receivers design is represented by power consumption. In order to reduce at maximum the power consumption, the different chip manufacturers adopted different solutions, most of them are based on the concept that, contrarily to a classical GNSS receiver, a mass-market receiver is not required to constantly compute a PVT solution. In fact, most of the time, GNSS chipsets for consumer devices are just required to keep updated information on approximate time and position and to download clock corrections and ephemeris data with a proper time rate depending on the navigation message type and the adopted extended ephemeris algorithm. Then, when asked, the receiver is able to provide a position fix in a short time interval. By reducing the computational load of the device during the waiting mode, the power consumption is reduced proportionally.

In order to better understand advantages and disadvantages of power saving technologies, some of them have been studied and analysed in details in Section 4.3. At the same time a duty cycle power saving layer has been added to the fully software receiver, as explained in Section 4.3.3. The results are reported in this section: DC performance is first assessed by comparing the accuracy of code delay, Doppler frequency and C/N_0 estimates when continuously tracking a signal and after a re-activation following a sleep period, for different signal power levels. Then a full positioning solution is computed in different scenarios, in the full active state and for various DC patterns.

5.7.1 Accuracy of observables

In the following simulations a Galileo E1B signal generated with N-FUELS is considered. The data bits are not generated, to emulate the presence of assistance, but the same result could be achieved by processing the pilot code E1C instead. The Doppler profile is set to a linear increase of 1 Hz/s, starting from -1000 Hz.

First 6.4 s of data, corresponding to 100 snapshots of 64 ms, are processed in the full active mode, with the software receiver described in Section 5.1. Then, the same data are processed with a DC approach, with $T_a = 64$ ms, $T_s = 576$ ms, so that 10 snapshots of 64 ms are processed and a DC estimate is available every 10 full active estimates. The gain, in terms of execution time, resulting from the simulations, is about 87 %.

Figure 5.52 shows the code delay and Doppler frequency estimates. The black points are the result of the full active state processing, while the red crosses represent the estimates when DC is active. The figure proves that the accuracy is maintained good enough, even when the receiver is re-activated after almost one second of sleep period.

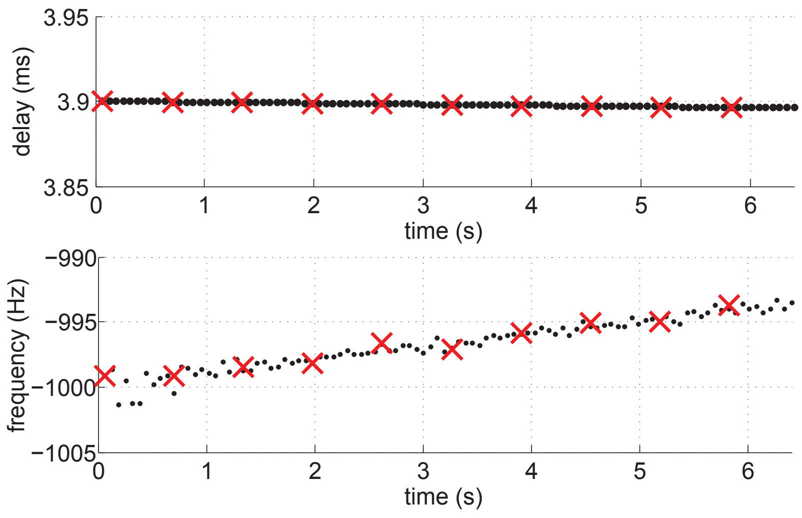


Figure 5.52: Full active (black) versus duty cycle (red) estimates.

Figure 5.53 reports the estimated C/N_0 in the same two situations, confirming once more that DC approach gives results very close to the full active standard processing.

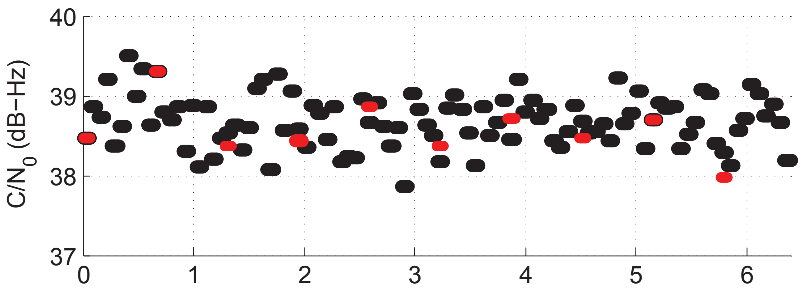


Figure 5.53: Full active (black) versus duty cycle (red) C/N_0 estimate.

Figure 5.54 shows the same results but with signals characterized by a lower C/N_0 (about 30 dBHz). Also in this latter case, the accuracy of the estimates obtained with DC techniques (red crosses) is in line with the accuracy in the full active case (black points), confirming the benefits of this approach and the absence of invalid measures.

5.7.2 The PVT solution

To prove the accuracy of the position solution, GNSS data are generated with an hardware signal generator (Section A.3.2) and stored in a memory through a SiGe front-end, with $f_s = 16.3676$ MHz and $f_{\mathrm{IF}} = 4.1304$ MHz. 4 Galileo signals are processed by the multi-correlator open-loop snapshot

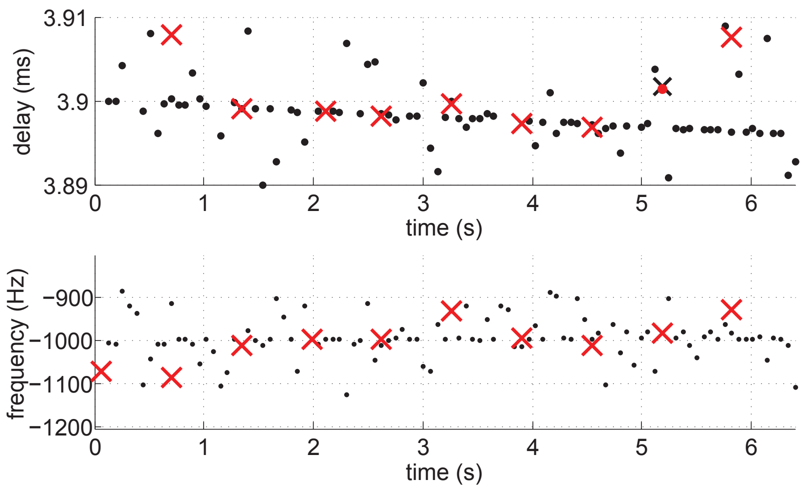


Figure 5.54: Full active (black) versus duty cycle (red) estimates with low C/N_0.

software receiver for about 45 s and for different DC patterns, reported in Table 5.15: the full active case, considered as a reference, and three different DC configurations. Successively, a PVT is computed with a rate of 1 Hz exploiting a software routine.

Table 5.15: Different configurations of the DC pattern.

	T_a	T_s	update interval	DC
Full active	–	0 ms	0.064 s	100 %
DC 1	64 ms	576 ms	0.64 s	10 %
DC 2	64 ms	960 ms	$\simeq$ 1 s	6.25 %
DC 3	64 ms	1984 ms	$\simeq$ 2 s	3.125 %

Static

First the static case is considered; results are reported in Figure 5.55. The true position corresponds to the GNSS antenna on top of ISMB NavSAS navigation lab. The full active case, obtained continuously processing 35 s of data is plotted with a white marker. Similarly, the DC cases reported in the table are drawn. As expected, as long as the sleep state increases, both accuracy and precision of the position decrease. It can be stated that all the solutions presented reach a level of accuracy which is acceptable for the majority of mass-market applications. Moreover, it has to be said that the position is obtained with basic navigation algorithms, without any kind of filtering, and just adopting a least squares solution. Proper PVT algorithms, such as a Kalman filter, are expected to improve the results.

Dynamic

A scenario emulating pedestrian and vehicular users has been exploited to test the accuracy of the results also in the dynamic case. The hardware generator allows to emulate a dynamic trajectory following a circle of radius 100 m with a certain velocity, equal to 2 m/s and to 15 m/s for the pedestrian and the vehicular case respectively.

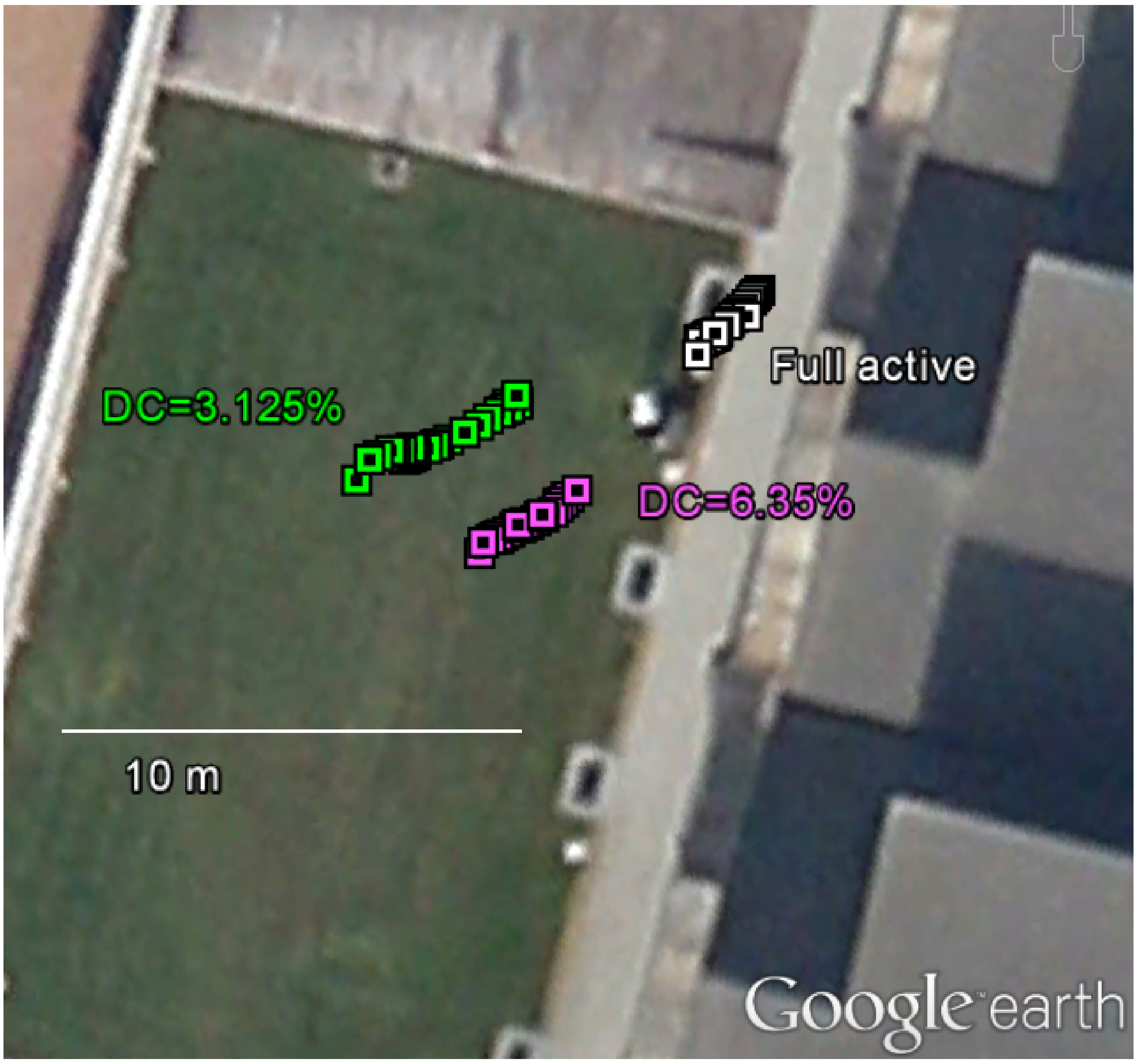


Figure 5.55: Positioning results for the static case and for different DC strategies.

Results for the pedestrian case are shown in Figure 5.56, while the vehicular case is depicted in Figure 5.57. In both cases an arc of circle is plotted for full active and DC tracking strategies. The conclusions are similar to the ones of the static case; a slight deterioration is evident in DC approaches, but the error is always kept within a few meters, confirming the validity of the technique and of the parameter propagation strategy described in the paper.

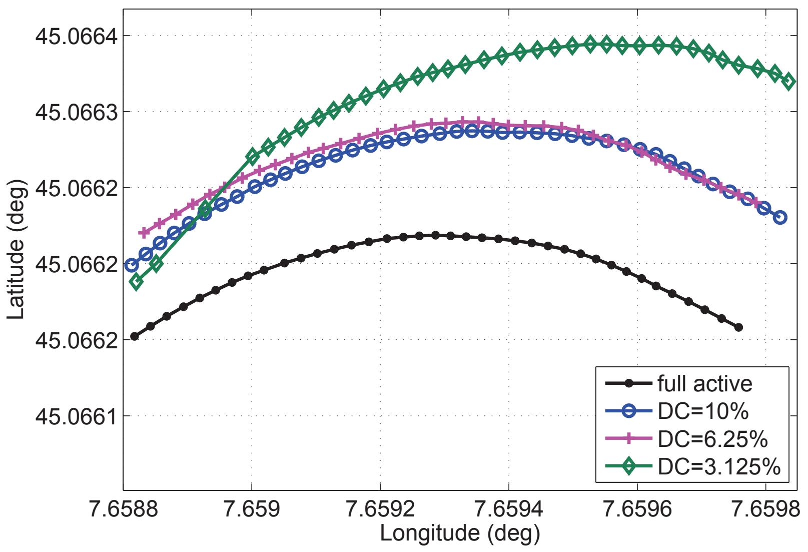


Figure 5.56: Positioning results for the pedestrian case and for different DC strategies.

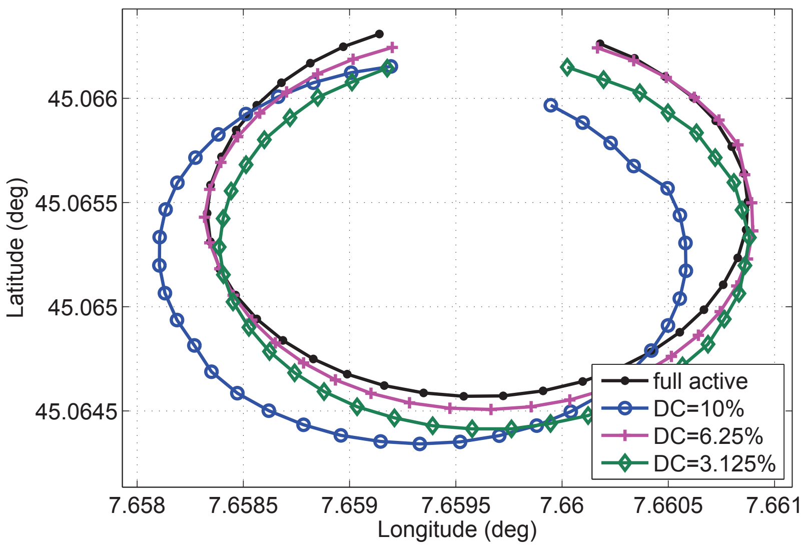


Figure 5.57: Positioning results for the vehicular case and for different DC strategies.

5.8 Real data results

In this section a few results obtained with real Galileo signals are reported, so as to prove the capability of the estimation algorithms proposed and the validity of the full open-loop snapshot processing scheme. The final architecture considered, including antenna, front-end and the software receiver, is reported in Figure 5.58.

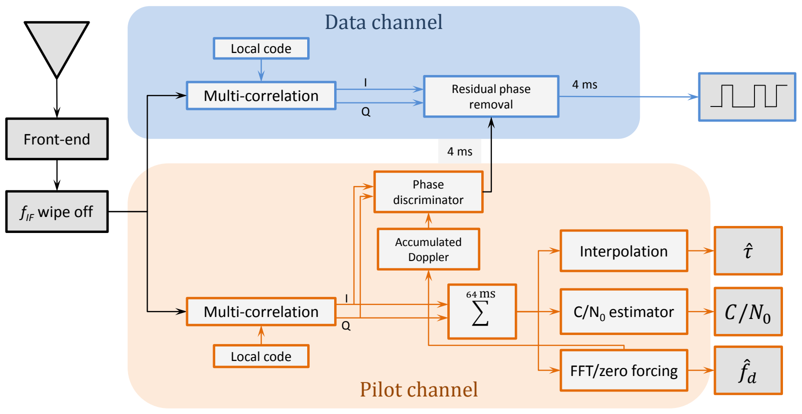


Figure 5.58: Block scheme of the open-loop receiver architecture as implemented in the fully software receiver used for Galileo IOV processing.

Summarizing, the RF signal is first captured by a patch antenna then stored on a PC memory exploiting the SiGE front-end (Section A.2.1. A wipe-off block removes the residual IF component and the signal enters two separate channels, processing respectively Galileo E1B and E1C signals. The multi-correlator snapshot processing units performs the partial multi-correlations, at $T_c = 4$ ms exploiting a local replica of both data and pilot codes. Then 16 the I and Q correlation values in the pilot channel are accumulated. Code delay, C/N_0, and Doppler are finally estimated at a rate of 64 ms. At the same time, a phase discriminator and a frequency accumulator compute the residual phase, in order to properly estimate the bits of the navigation message from I and Q correlation values in the data channel. The results are then passed to a different software routine able to compute a PVT solution.

Real Galileo IOV data were collected on June 04, 2014, 01.40 CET, in Torino city centre, Piazza Castello. The location was chosen in order to confirm the robustness of the snapshot open-loop approach in a real urban scenario, characterized by interference, multipath and signal obstruction. At that time all the four Galileo satellites were in view, the sky-plot is reported in Figure 5.59. During the first 60 s of simulation, the antenna was in a fixed position in the middle of a square (static scenario). Then, for the following 60 s, the user was moving at about 15 km/h around the square, thus experiencing signal shadowing and multipath (dynamic scenario).

Nevertheless, at that time the satellite transmitting Code Number 20 (FM4) was not transmitting any valid signal (UNAVAILABLE FROM 2014-05-27 UNTIL FURTHER NOTICE) [73]. In addition, Code Number 12 (FM2) was broadcasting only words 63. For these reasons it has not been possible to compute a full PVT solution exploiting only Galileo satellites.

Figure 5.61 reports the C/N_0 as estimated by the software routine along the two minutes of data collection. It is interesting to notice that, as expected, the average C/N_0 of satellite 11 is about 5 dBHz lower with respect to PRN 12 and 19. In fact, this SV is characterized by a lower elevation at the time of the data collection, as confirmed by the skyplot of Figure 5.59. In addition, during the second minute, in which the user is moving, the same satellite experiences very low C/N_0

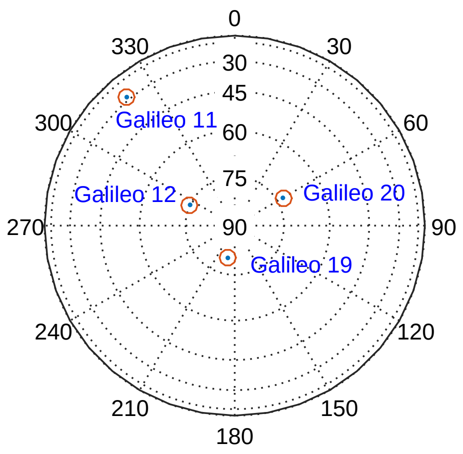


Figure 5.59: Skyplot in Torino city centre on June 04, 2014, at 01.40 CET.

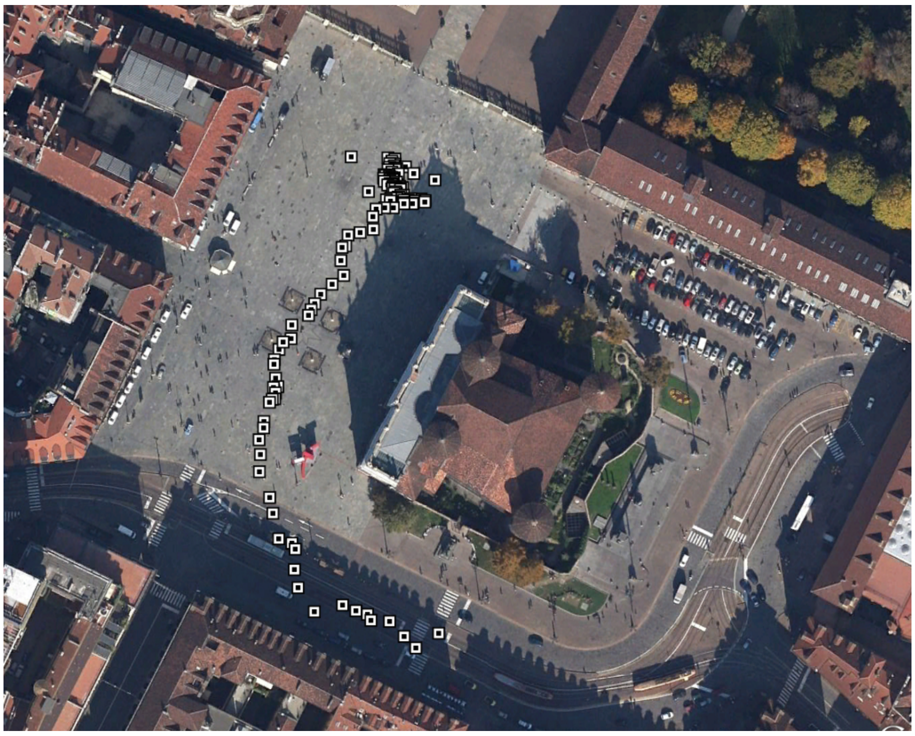


Figure 5.60: Path followed during the data collection at about 15 km/h, as a result of GPS only processing.

levels, down to 30 dBHz, probably because of multipath, signal reflection, and temporary LOS shadowing. Also satellites 12 and 19 C/N_0 estimate exhibits a larger variance in the second part of the signal analysis, for the same reasons.

Figure 5.62a reports the estimates of the Doppler frequency, computed according to the DFFT technique described in Section 4.2. In all the three cases, a proper estimate is always provided, with very few and limited outages, especially in the second part in which a dynamic scenario is

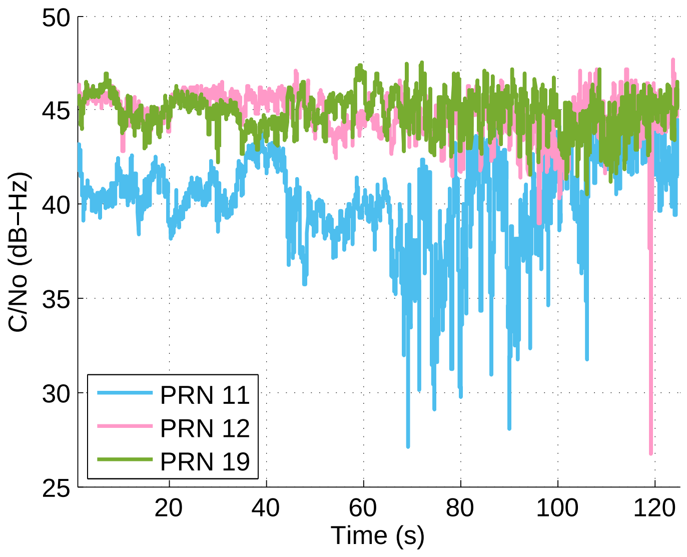
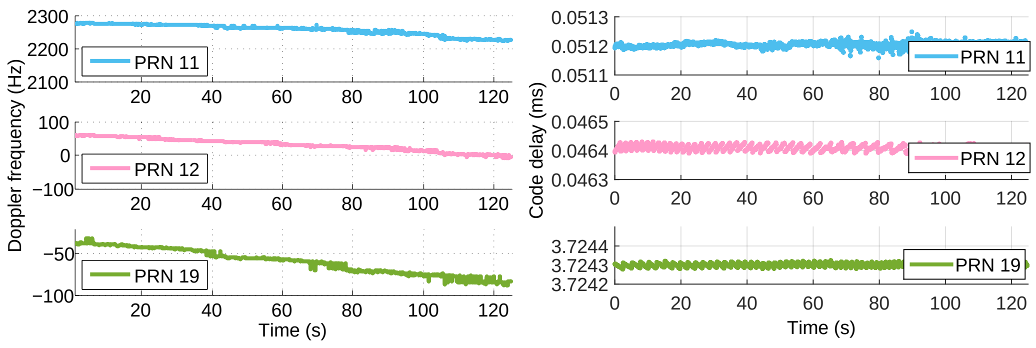


Figure 5.61: C/N_0 estimate for the three Galileo IOV satellites along two minutes of data processing.

considered. The performance deterioration for PRN 19 can be explained by comparing the sky-plot of Figure 5.59 and the track reported in Figure 5.60. It is indeed expected a signal deterioration, because PRN 19 azimuth is about 180° and the user is moving towards the south side of the square, where some buildings are present. Similarly, Figure 5.62b reports the code delay as estimated by the receiver.



(a) Doppler frequency **(b)** Code delay

Figure 5.62: Parameters estimate for the three Galileo IOV satellites along two minutes of data processing.

5.9 A real device: the STM Teseo 2

Some commercial mass-market receivers evaluation boards were available at ESA ESTEC Navigation lab. In particular, the Teseo 2, a stand-alone, single-chip, multi-constellation positioning device by ST Microelectronics (STM) has been used. The performance of different firmware releases in terms of TTFF, sensitivity and positioning accuracy has been evaluated. More details on the architecture and characteristics of the Teseo 2 receiver are given in Section A.4.

5.9.1 Time to first fix

As described in (Section 3.2), the TTFF is the time a user has to wait before his device provides a position fix since it is powered on. Some results for different C/N_0, for hot, warm and cold start, and for different constellation combinations have been obtained using the Teseo 2. First some test with only GPS and GLONASS constellations were carried out; then Galileo signals are added, and the receiver performance verified.

GPS and Glonass

Hot and warm start tests were carried out by sending a proper National Marine Electronics Association (NMEA) command to the device. In total 500 trials are considered. Results are obtained with real GNSS data, captured by a static rooftop antenna at ESTEC premises.

Table 5.16 reports the results. As expected, hot start assures a much lower TTFF. It is worth recalling that hot start means that all the parameters (such as ephemeris, time and estimated user position) are already present at power on, so the receiver only has to measure the pseudo-ranges [34]. It is interesting to notice that while in warm start, the combined use of GPS and Glonass gives a lower TTFF, in cold start the effect is adverse.

Table 5.16: TTFF in seconds for different cases.

	Hot start	Warm start		Cold start	
	GPS + Glonass	GPS	GPS + Glonass	GPS	GPS + Glonass
50%	2.3	32.3	20.6	35.2	35.5
Max	3.4	39.8	37.9	42.4	42.8
Min	1.6	19.3	13.5	23.7	26.3
Std	0.4	5.5	5.8	3.9	4.1

GPS and Galileo

Figure 5.63 reports the hot start TTFF for different C/N_0 values, in the range $25 - 53$ dBHz, computed using the Teseo 2 and hardware generated GNSS data. The receiver is configured in dual constellation mode (GPS and Galileo) and carries out 40 TTFF trials, with a random delay between 15 and 45 seconds. In standard AWGN scenario and in hot start conditions, the results mainly depend on the acquisition strategy and on the receiver availability of correlators and acquisition engines. In an ideal case with open sky conditions and variable C/N_0, the introduction of a second constellation only slightly improves the TTFF and this result cannot be generalized since it mainly depends on the acquisition threshold of the receiver, which can be different with different constellation signals. In real world conditions, the situation can be quite different.

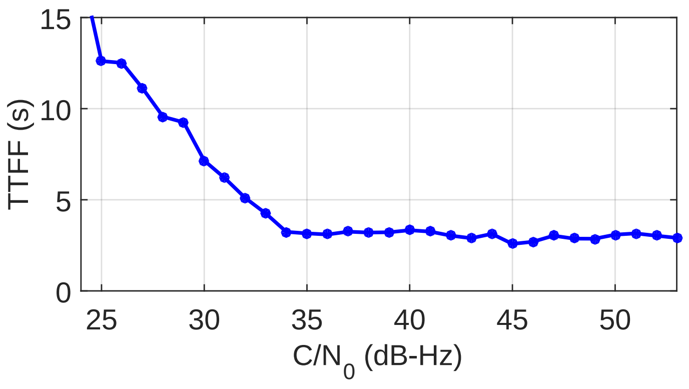


Figure 5.63: Hot start TTFF for Galileo and GPS configuration vs. C/N_0 using the Teseo 2 receiver.

Confirming benefits of I-NAV message

It is interesting to analyse the difference in the TTFF due to the different structure of the GPS and Galileo navigation message. Galileo E1 signal I/NAV message and GPS L1 C/A signal navigation message contain data related to the satellite clock and ephemeris and to the GNSS time. These parameters are important for computing the position, since they describe the position of the satellites in their orbit, their clock error and the signals transmission times.

Some results in the particular case of cold start, with an ideal open sky AWGN scenario, computed with the Teseo 2 receiver, are reported in Table 5.17. The TTFF is significantly lower when using also Galileo satellites: while the mean TTFF when tracking only GPS satellites is equal to about 31.9 s, it decreases to 24.7 s when considering only Galileo satellites, and to 22.5 s in the case of dual constellation. Similarly, the minimum and maximum TTFF values are lower when tracking also Galileo satellites. The 95% probability values confirm the theoretical expectations [74]. Again, in the ideal case with open sky conditions, the results with two constellations are quite similar to the performance of the signal with faster TTFF. However, as detailed below, in non-ideal conditions the usage of multiple constellations represents a big advantage and underlines once more the importance of developing multi constellation mass-market receivers.

Table 5.17: Comparison between TTFF (in seconds) in cold start for different constellation combination obtained with the STM Teseo 2 receiver.

	min	Max	Mean	95%
GPS	22.2	40.1	31.9	36.2
Galileo	18.6	36.6	24.7	32.3
GPS+Galileo	19.6	35.4	22.5	31.9

TTFF in harsh environments

The TTFF in harsh environments was estimated exploiting an hardware signal generator (Section A.3.1) along with the multi satellite LMS model described in Section 3.4.3. About 50 tests, in hot, warm and cold start, were carried on, first using both GPS and Galileo satellites, and then using only one constellation. It must be noted that in the second case only the 2D fix is considered, since, according to the scenario described, at maximum three satellites are in view. Table 5.18

reports the results for the double constellation case: in hot start the average TTFF is about 8 s, and it increases to 36 s and 105 s respectively for the warm and cold cases. It is straightforward that the results are much worse than in the case reported above, where full open sky AWGN conditions are considered. In this scenario only 6 satellites are available at maximum; moreover, the presence of multipath and fading affects the results, and they exhibit a larger variance, because of the varying conditions of the scenario.

Table 5.18: TTFF (in seconds) exploiting GPS and Galileo constellations in harsh environments obtained with the Teseo 2 receiver.

	min	Max	Mean
HOT start	4.5	12.4	7.9
WARM start	31.5	67.2	36.3
COLD start	40.5	265.7	105.0

At the same time, in Table 5.19 similar results, but for the GPS only case, are reported. In this case the Teseo 2 has been configured to track only GPS satellites. The mean TTFF increases both in the hot and in the warm start case, whereas in cold start it is not possible compute a 2D fix with only three satellites; the ambiguity of the solution cannot be solved if an approximate position solution is not available. It can seem unfair to compare a scenario with three satellites with a scenario with six satellites. However, it can be assumed that this is representative of what happens in limited visibility conditions, such as an urban canyon, where a second constellation theoretically double the number of satellites in view.

Table 5.19: TTFF (in seconds) exploiting only GPS constellations in harsh environments obtained with the Teseo 2 receiver.

	min	Max	Mean
HOT start	4.7	38.0	11.8
WARM start	31.6	109.7	51.9
COLD start	N.A. (*)	N.A. (*)	N.A. (*)

* 4 SVs required for cold start

These results confirm the benefits of dual constellation mass-market receivers in harsh environments, especially in urban canyons, where the number of satellites in view can be really low. Making use of the full constellation of Galileo satellites will allow mass-market receivers to substantially increase performance in these scenarios.

5.9.2 GGTO

Time is a crucial feature in satellite-based radio navigation systems. The elapsed time between the transmission of a GNSS signal and its reception by a receiver, multiplied times the speed of light, provides the basis for calculating the pseudo-range. However, GPS and Galileo are using different reference time systems, and thus a time difference arises: the GGTO [75].

To be more precise the pseudo-ranges determined with Galileo are referenced to the GST, while the ones from GPS use the GPST as a reference. The GST is steered to a prediction of UTC, modulo one second, obtained through an external time service provider. GST will be kept to within 50 ns (95%) of UTC, modulo one second, over any one-year time interval. The offset between UTC and GST (respectively modulo one second) will be known with a maximum uncertainty of at least 28 ns (2-σ).

Different implementation solutions

When computing a PVT solution with mixed signals, three solutions are possible:

1. **GGTO determination at user level**: the GGTO can be estimated from the navigation equations. Then, five unknowns need to be determined: the user's 3D position, the time bias between the user and Galileo (or GPS) and the GGTO. This means at least five pseudo-range measurements.

2. **GGTO determination at system level**: the signals broadcast by GPS and Galileo satellites include the GGTO in the navigation message. So the receiver can apply this broadcasted GGTO to account for the time offset. This way, it only has to determine the 3D position and the time bias between the receiver and the navigation system. The navigation solution is obtained with at least four pseudo-range measurements, but the navigation message has to be decoded.

3. **Hybrid solution**: for users who do not want to use the broadcast GGTO, but still need to deal with restricted visibility condition, it is in fact possible to just determine the GGTO with the fifth satellite on their own, when enough visibility conditions are good. Afterwards, when satellite visibility conditions worsen, the receiver can switch to the four parameter solution, utilizing the last computed GGTO estimate, opportunely smoothed to reduce its noise. This exploits the assumption that the constellation time-offset slowly drifts over time: indeed the inherent variations of stochastic nature are assumed to cause an error around 0.2 ns (RMS) in one hour time interval [76].

The Teseo 2 exploits the second solution. The driving performance requirements associated to GGTO can be summarized as [77]:

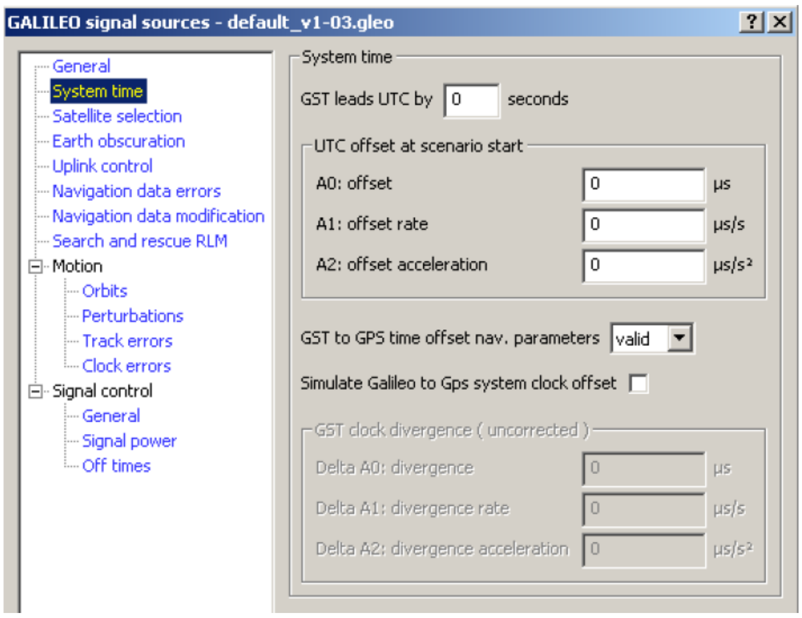
- GGTO validity: the validity period of the GGTO shall be a minimum 24 consecutive hours;
- GGTO offset accuracy: the accuracy of the offset between GST and GPS Time (modulo 1 s) shall be less than 5 ns with a 2-σ confidence level over any 24 hours;
- GGTO Stability: The stability of the GGTO, expressed as an Allan deviation, shall be better than 8×10^{-14} over any 1 day.

GGTO parameters computation

In order to test the Teseo 2 receiver and to evaluate the performance of the receiver computing the PVT with a combination of constellations, a Spirent scenario with GGTO dedicated configuration has been set up, starting from the nominal GPS+Galileo scenario. The GGTO can be simulated in the Spirent signal generator (see Figure 5.64) by setting the GST to GPS time offset. The true GGTO, that is basically the difference of the A_0 (offset), A_1 (offset rate) and A_2 (offset acceleration) parameters between GPS and Galileo, is used to model the time difference and is broadcast in the navigation message. The user is also given the opportunity to generate clock divergence terms that are not broadcasted as correction terms but apply only in the RF generation: the ΔA_0, ΔA_1, ΔA_2 terms are divergence errors, thus applying at the signal level but not included in the navigation message.

In order to compute the A_0 parameter let's assume that the difference between GPS and Galileo system time is in between 10 and 20 m, corresponding respectively to $10/(3 \times 10^8)$ and $20/(3 \times 10^8)$ s, approximately equal to 30 and 60 ns. So

$$A_0 = 30 \text{ ns} = 3 \times 10^{-2} \ \mu\text{s}\,. \tag{5.6}$$



GALILEO signal sources - default_v1-03.gleo

System time

GST leads UTC by 0 seconds

UTC offset at scenario start

A0: offset 0 µs

A1: offset rate 0 µs/s

A2: offset acceleration 0 µs/s²

GST to GPS time offset nav. parameters valid

Simulate Galileo to Gps system clock offset ☐

GST clock divergence (uncorrected)

Delta A0: divergence 0 µs

Delta A1: divergence rate 0 µs/s

Delta A2: divergence acceleration 0 µs/s²

Figure 5.64: GGTO determination in the Spirent simulator.

A_1 is chosen in such a way that the maximum drift in the simulation time is less than 60 ns. Considering a simulation time of 1 day, we have:

$$A_1 | A_0 + A_1 T_{\mathbf{sim}} = 60 \text{ ns}\,, \tag{5.7}$$

that is

$$A_1 = \frac{60 \text{ ns} - A_0}{T_{\mathbf{sim}}} = \frac{60 \text{ ns} - 30 \text{ ns}}{86\,400 \text{ s}} = 3.47 \times 10^{-7} \text{ µs/s}\,. \tag{5.8}$$

From the requirements (Allan deviation) we know that the stability shall be better than 8×10^{-14} over any 1 day. Supposing that the ΔA_2 is equal to 0 µs/s^2, then it is possible to compute ΔA_1 in microseconds, and we have:

$$\Delta A_1 = 8 \times 10^{-8} \text{ µs/s}\,. \tag{5.9}$$

ΔA_1 is computed in such a way that the maximum deviation does not exceed 5 ns $\pm$ 2-σ,

$$\Delta A_0 | \Delta A_0 + \Delta A_1 T_{\mathbf{sim}} < 5 \text{ ns}\,. \tag{5.10}$$

So

$$\Delta A_0 < 5 \text{ ns} - 8 \times 10^{-8} \text{ ns/s} \cdot 4\,600 \text{ s} = 4.7 \text{ ns}\,. \tag{5.11}$$

It is possible to choose for example $\Delta A_2 = 2$ ns. The equation to add the ΔA_2 parameter is:

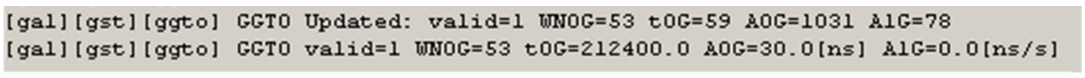
$$\Delta A_1 + \Delta A_2 T_{\mathbf{sim}} < 5 \text{ ns}\,. \tag{5.12}$$

Two scenarios have been defined. They are identical concerning constellations, signal power and standard parameters. The difference is only in GGTO definition.

Scenario GGTO 1

$$\begin{cases} A_0 = 0.03\ \mu\text{s} \\ A_1 = 3.4722 \times 10^{-7}\ \mu\text{s/s} \\ \Delta A_0 = 0\ \mu\text{s} \\ \Delta A_1 = 0\ \mu\text{s/s} \end{cases}. \tag{5.13}$$

This is a nominal scenario, in which the same corrections on the GGTO are transmitted in the navigation message and used in the signal generation. Therefore all the divergence terms are set to 0. It is useful to test with the Teseo 2 the correct decoding of the A_0 and A_1 parameters of the I-NAV, as done in some of the tests reported below. On the right window of Figure 5.65 some information confirm the correct demodulation of the parameters.



```
[gal][gst][ggto] GGTO Updated: valid=1 WN0G=53 t0G=59 A0G=1031 A1G=78
[gal][gst][ggto] GGTO valid=1 WN0G=53 t0G=212400.0 A0G=30.0[ns] A1G=0.0[ns/s]
```

Figure 5.65: GGTO demodulation with the Teseo 2.

Scenario GGTO 2

$$\begin{cases} A_0 = 0.03\ \mu\text{s} \\ A_1 = 3.4722 \times 10^{-7}\ \mu\text{s/s} \\ \Delta A_0 = 0.002\ \mu\text{s} \\ \Delta A_1 = 8 \times 10^{-8}\ \mu\text{s/s} \end{cases}. \tag{5.14}$$

This is a more realistic scenario, in which there are some divergence terms, as computed above. The GGTO transmitted in the navigation message is different from the GGTO used to generate the signal. In this case some problems at the receiver can be expected.

Both scenarios have been tested at ESTEC navigation lab, confirming the correct implementation of the GGTO decoding and GPS+Galileo mixed solution in the latest releases of the Teseo 2 firmware analysed.

GGTO and TTFF

Furthermore, it is interesting to analyse more in details the case of GPS and Galileo joint solution TTFF, which is strictly related to GGTO. As reported above, when computing a PVT solution with mixed signals, three solutions are possible: either to estimate it as a fifth unknown, or to read it from the navigation message, or to use pre-computed value. In the first case it is not necessary to rely on the information contained in the navigation message, eventually reducing the TTFF. However, five satellites are required to solve the five unknowns, and this is not always the case in urban scenario or in harsh environments, as it will be proved below. On the contrary, in the second case, it is necessary to obtain the GGTO information from the navigation message, and since it appears only once every 30 seconds, in the worst case it is necessary to correctly demodulate 30 seconds of data. Both approaches show benefits and disadvantages, depending on the environment. The Teseo 2 exploits the second solution: in this case, it is possible to see an increase in the average TTFF when using a combination of GPS and Galileo, due to the demodulation of more sub-frames of the broadcast message.

Advantages and disadvantages of using the broadcast GGTO when computing a mixed GPS and Galileo position can be evaluated also in harsh conditions. During some tests, the scenario described in Section 3.4.4, where the 3 GPS and 3 Galileo signals, affected by fading and multipath (LMS Multi-SV model), are considered, is tested with the Teseo 2 receiver.

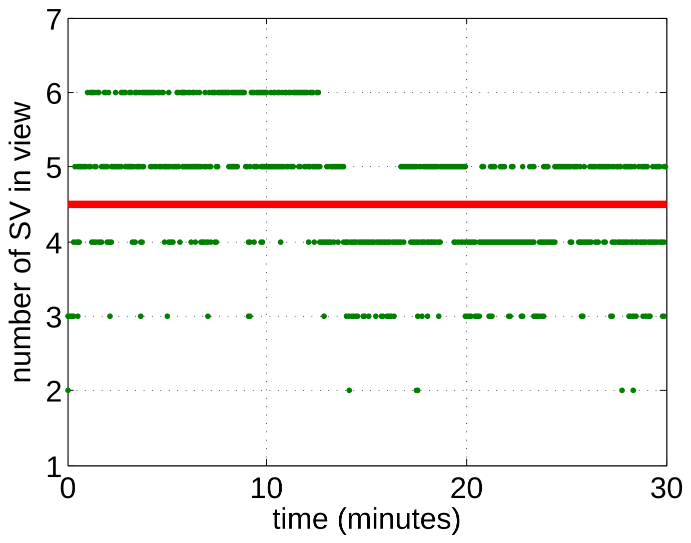
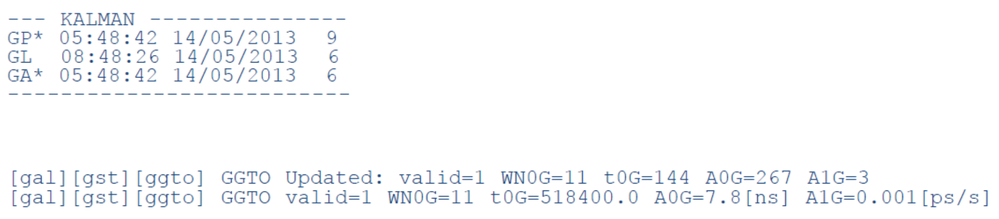


Figure 5.66: Number of satellites tracked by the Teseo 2 receiver in the Multi-SV LMS 3GPP simulation.

Figure 5.66 shows the number of satellites in tracking state for 30 minutes of test. When the LMS channel conditions are good, all the six SVs in view are in tracking state. However, when the fading becomes important, the number is reduced to only two satellites. If the receiver is designed to extract the GGTO from the navigation message, then a PVT solution is possible also when only four satellites are in tracking state, i.e. for 90% of the time in this specific case. On the contrary, if the GGTO has to be estimated, one more satellite is required, and this condition is satisfied only 57% of the time, strongly reducing the probability of having a fix. Nevertheless, estimating the GGTO requires the correct demodulation of the navigation message, and this is possible only if the signal is good enough for a sufficient amount of time.

Real data results

Since May 2013 Galileo IOV satellites started the transmission of a valid GGTO in the navigation message. The Teseo 2 combined fix has been tested with real GPS and Galileo data, correctly demodulating the broadcast GGTO, as reported in Figure 5.67.



```
--- KALMAN ---------------
GP* 05:48:42 14/05/2013  9
GL  08:48:26 14/05/2013  6
GA* 05:48:42 14/05/2013  6
--------------------------


[gal][gst][ggto] GGTO Updated: valid=1 WN0G=11 t0G=144 A0G=267 A1G=3
[gal][gst][ggto] GGTO valid=1 WN0G=11 t0G=518400.0 A0G=7.8[ns] A1G=0.001[ps/s]
```

Figure 5.67: Screenshot of the GPS testing tool showing the correct demodulation of the GGTO.

5.9.3 Sensitivity in harsh environments

As firstly reported in Section 3.2, a receiver capability to work in hostile environments is measured in terms of sensitivity. Sensitivity can be measured by verifying the device performance when artificially decreasing the received signal strength or adding multipath effects.

Tracking tests

Results in terms of TTFF in harsh environments are reported in Section 5.9.1.

A 30 minutes tracking test has been carried out with the Teseo 2 and compared with the results of the fully software receiver, exploiting the data of the Multi-SV LMS 3GPP scenario (Section 3.4.4). Both the receivers were able to process the signals, even with some LOLs due to fading and multipath reflections. Figure 5.66 reports the number of satellites in tracking state in the Teseo receiver at every second, while Figure 5.68 reports the HDOP as computed by the receiver. It is interesting to note that when all the six satellites are in tracking state the HDOP lies in the range 1.4 – 2.1, as expected from the simulation scenario definition; on the contrary in correspondence with a LOL it increases.

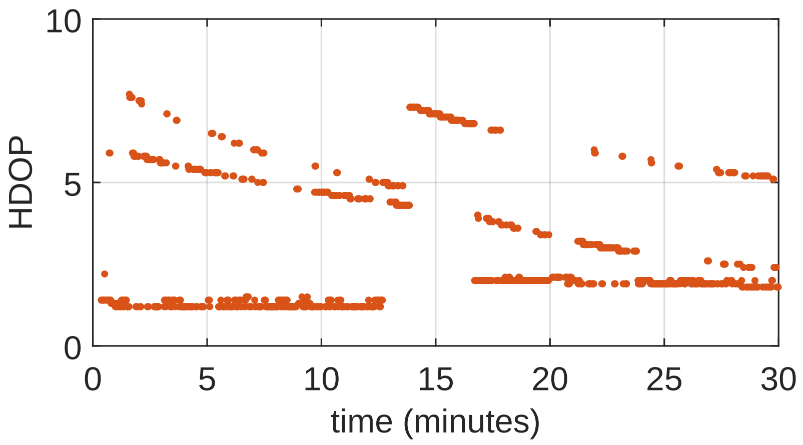


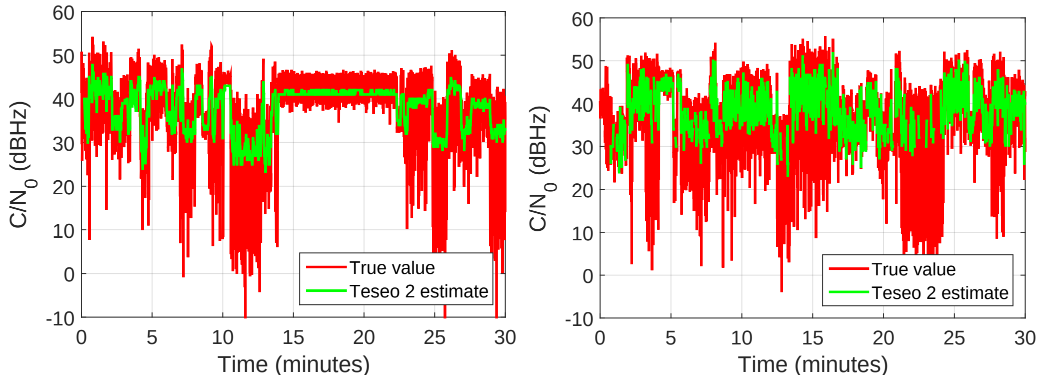
Figure 5.68: HDOP computed by the Teseo 2 receiver in the Multi-SV LMS 3GPP simulation.

Figure 5.69 shows a comparison between the power of the signal generated by the simulator and the power estimated by the Teseo 2, in the case of GPS PRN 7 and Galileo code number 23. This proves the tracking capability of the receiver also for high sensitivity. In order to deal with low power signals, the integration time is extended both for GPS and for Galileo, using the pilot tracking mode in the latter case.

Finally Figure 5.70 and Figure 5.71 show respectively the position and the velocity solution. In the first case latitude, longitude and altitude are plotted, while in the second case the receiver speed estimate in km/h is reported.

5.9.4 Concept of loss of lock

The main effect of signal shadowing, multipath and interference nuisances in legacy tracking loops is represented by the LOL. A LOL occurs any time degradations in the received signal prevent the tracking loops to lock into a stable lock point. It usually happens upon the exit of the loop discriminators from their linearity zone and it leads to errors in the estimates of Doppler frequency, phase and delay. The signal is declared lost and the tracking stage ends: in most receivers, it is



(a) GPS PRN 7 **(b)** Galileo PRN 23

Figure 5.69: C/N_0 estimate computed by the receiver in harsh environments and compared with the signal power.

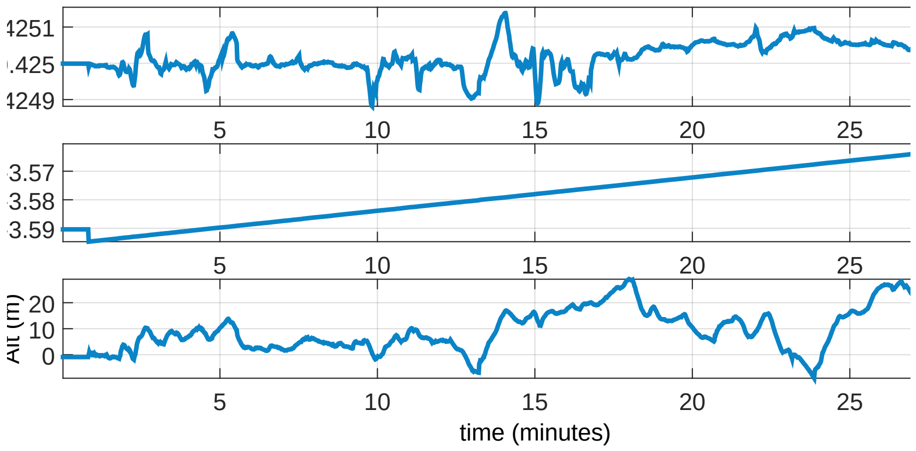


Figure 5.70: Teseo 2 position solution in LMS scenario.

signalled by setting the C/N_0 estimate to 0 dBHz, as reported in Figure 5.72a. At the same time pseudo-range and frequency estimation (reported in Figure 5.72b) are not available.

Because of the limited bandwidth of the loops, when a severe LOL occurs it is necessary to reacquire the signal, i.e. to go back to the acquisition stage. This process is time and resources consuming; it has been proved and reported in Figure 5.72a that a receiver adopting the standard acquisition and tracking scheme can take up to 3 or 5 seconds to reacquire the signal after a LOL, even if the signal degradation lasts only a fraction of second, unless specific techniques are implemented. This time interval is not only due to the reacquisition process itself and to the availability of acquisition engines, but also to the time needed to detect and declare a LOL and to the channel characteristics at that time. As depicted already in Figure 5.69, it can happen that for a few seconds the receiver stops tracking the signal.

This problem is absent when adopting open-loop snapshot techniques. In fact, since no feedback mechanisms are present, the signal nuisance affects only one processing epoch, exactly for the time duration of the nuisance. As soon as the signal degradation ends the receiver operates again, without the need to reacquire the signal. It has to be pointed out that the concept of LOL is not

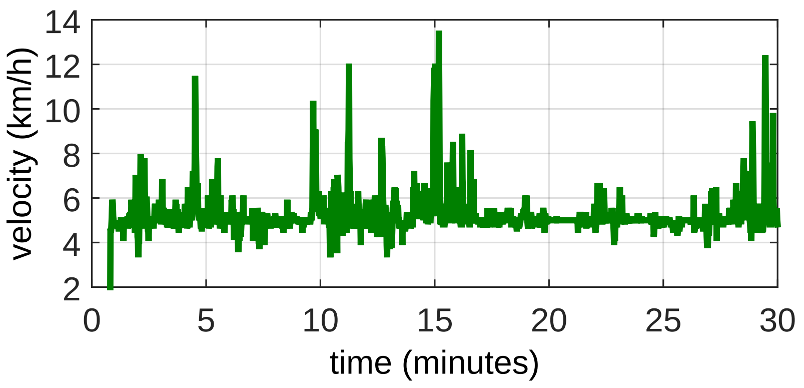
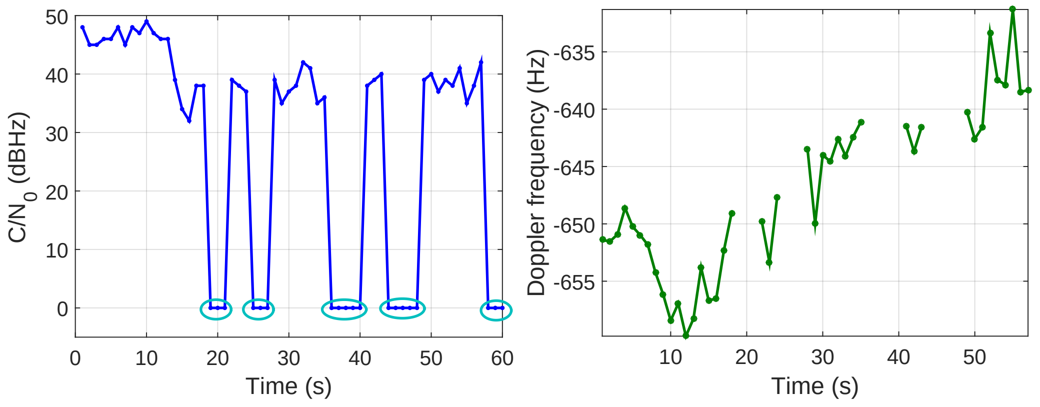


Figure 5.71: Teseo 2 velocity solution in LMS scenario.



(a) C/N_0 **(b)** Doppler Frequency

Figure 5.72: Teseo 2 estimates when tracking signals affected by LMS.

applicable to open-loop schemes; the expression *invalid measure* is indeed more appropriate. To prove this fact, a comparison between a standard receiver adopting a DLL and a FLL and the snapshot software receiver is carried out in the case of LMS affected signal. In particular, the Scenario described in Section 3.4.4 with the LMS model of Section 3.4.2 is set up on an hardware signal simulator. First the Teseo 2 receiver is connected to the signal generator, and a tracking test in run; NMEA data are stored. Secondly, the same amount of data is grabbed with the bit grabber described in Section A.2 and post-processed with the software receiver described in Section 5.1.

It has been proved that the Teseo 2 receiver exhibits several LOLs. In a 10 minutes simulation with the following results are reported:

- average outage period: 12.6%
- average duration of a LOL: 7.96 s
- LOL probability: 0.8%

It is clear that the robustness of the receiver is strongly reduced when the LMS time series is added to the signal: during almost 1/6 of the total time the receiver is not providing any valid result. On the contrary, when the same GNSS signal is processed with the fully software receiver the results are significantly better. When adopting a total integration time of 64 ms the effects of the signal degradation are significantly limited, and they are totally transparent when increasing it to 128 ms.

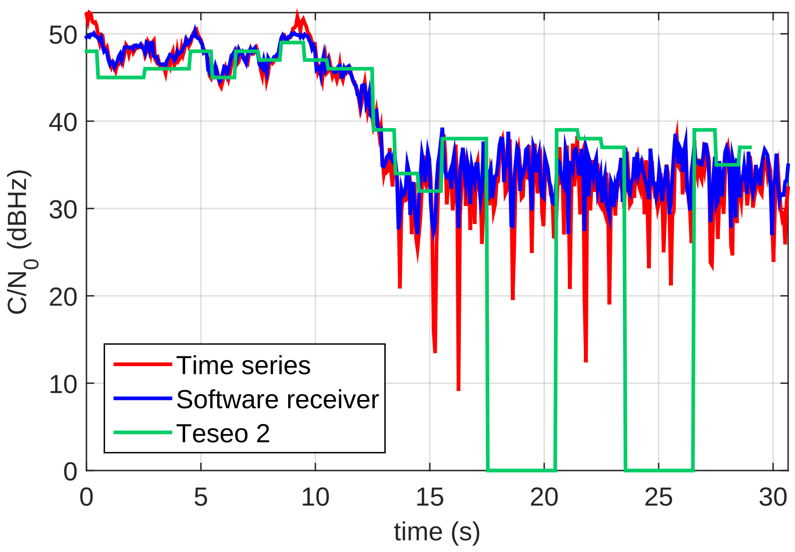
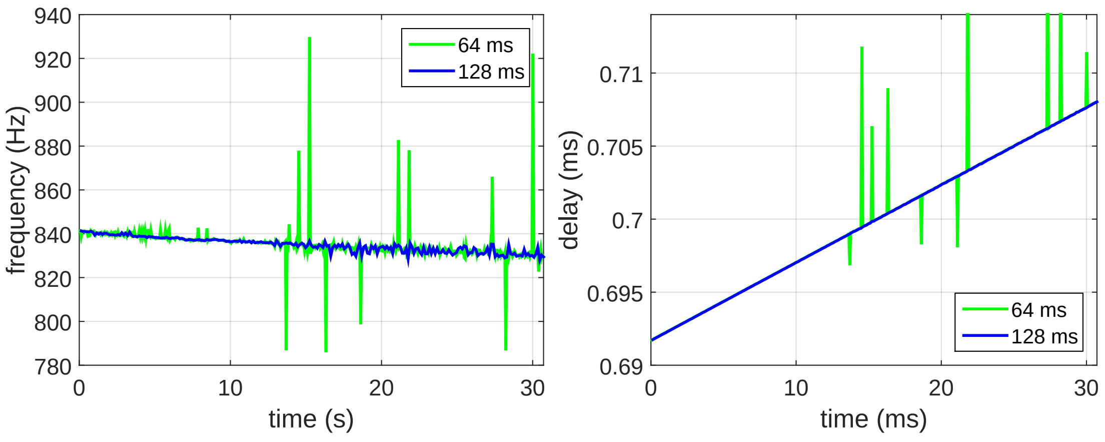


Figure 5.73: C/N_0 estimation comparison in a signal affected by LMS nuisances.

Figure 5.73 shows a comparison of the C/N_0 estimated by the Teseo 2 and by the snapshot software receiver, along with the reference C/N_0 time series from the hardware signal generator, over a time interval of 30 s. It has to be said that the C/N_0 is a good parameter to analyse the quality of the receiver, despite the low rate of this measure available from the Teseo NMEA messages. The red curve corresponds to the true signal power, as generated by the signal generator. The green curve, corresponding to the Teseo 2 tracking loops output, exhibits 2 drops to 0 dBHz, corresponding to 2 LOLs, lasting about 3 seconds. During these intervals the signal is not tracked and no measures are available, even if the degradation occurs only for a few milliseconds. On the contrary the estimate of the software receiver is much closer to the true C/N_0 value, proving that the open-loop snapshot solution performs better. This is confirmed by the plots of the code delay and of the frequency reported in Figure 5.74b, where some outliers are present. In these cases, some estimates are clearly wrong, due to invalid measures in the multi correlation stage and in the FFT estimation. However, as soon as the fade ends the open-loop software receiver immediately provides a valid measure. In the case of 128 ms the trend of the estimates is very smooth, and no invalid measures occur. It is interesting to note that the outliers present in the case of 64 ms integration correspond exactly to the time instants in which the C/N_0 is low in Figure 5.73, as reported in Table 5.20.

Table 5.20: C/N_0 measures corresponding to LOL events.

Time (s)	13.7	14.53	15.23	16.32	18.62	21.12	21.82	27.33	28.22	30.02
C/N_0 (dBHz)	20.8	23.4	13.4	9.1	19.5	20.8	12.4	23.5	24.6	23.9



(a) Doppler frequency **(b)** Code delay

Figure 5.74: Elevation and azimuth of Galileo IOV satellites as computed from navigation message.

5.10 Galileo IOV results

On March 12, 2013, for the first time, the four Galileo IOV satellites were switched on at the same time, broadcasting a valid navigation message [16]. From 9.02 CET, all the satellites were visible at ESTEC premises at the ESA's technical site, in Noordwijk, The Netherlands, allowing a team of researchers of ESA and myself to perform the first Galileo only position fix. The achieved accuracy of better than 10 meters met expectations, taking into account the limited infrastructure deployed so far [16].

5.10.1 First Galileo IOV dynamic PVT

At the same time, some results were obtained tracking Galileo satellites with the Teseo 2 receiver. In particular, thanks to its small size and portability, it was installed on a mobile test platform, embedded in the ESA's Telecommunications and Navigation Test-bed vehicle (Figure 5.75). Then, exploiting a network connection, it was possible to follow, from the navigation lab, the real time position of the van moving around ESTEC. The receiver was also saving NMEA messages to a local memory; the results reported in the following have been obtained post-processing these data. To the best of the authors' knowledge, this represents the first Galileo only mobile navigation solution.

An important consideration has to be done: the results obtained with the Teseo 2 have to be considered preliminary, since its firmware supporting Galileo was in an initial test phase (for example the absence of ionospheric model and the tracking of the E1B data channel only). With the latest releases of the Teseo 2 Galileo firmware even better results are expected.

Sky-plot

Figure 5.76 shows the sky-plot of the Galileo constellation around 10.30 CET at ESTEC premises: all the four IOV satellites are in view. However the Code Number 11 (311) has a very low elevation, although it is rising. Moreover some buildings around the van parking slot were obstructing the line of sight path to the satellite. Only after about 35 minutes the mobile test unit was moved and

Figure 5.75: ESA's mobile test bed unit.

the Teseo 2 was able to acquire and track all the four signals, and thus to pass from a 2D fix to a 3D fix. It is noted that the geometry is not optimal, as proved by the high Position Dilution Of Precision (PDOP).

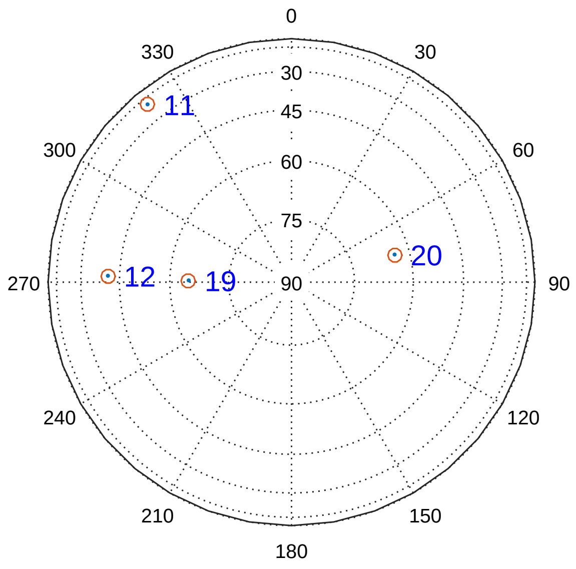
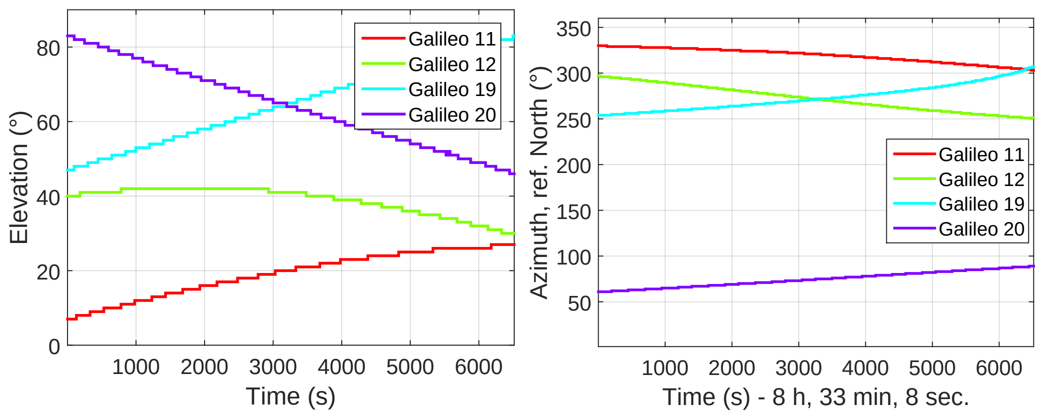


Figure 5.76: Galileo constellation sky-plot around 10.30 CET at ESTEC premises.

Satellites elevation and azimuth

Elevation and azimuth for each satellite have been computed according to the content of the Navigation message demodulated by the receiver and saved in the NMEA messages. Results are reported in Figure 5.77.



(a) Elevation **(b)** Azimuth

Figure 5.77: Elevation and azimuth of Galileo IOV satellites as computed from navigation message.

C/N_0 estimate

The C/N_0 is estimated by the receiver with rate 1 Hz, and the result is reported in Figure 5.78. From this figure it is possible to see that the SV Code Number 11 has been tracked only after about 30 minutes, i.e. when its elevation was higher and it was in line of sight with the receiver antenna. Moreover, its average C/N_0 is lower with respect to Code Number 12, 19 and 20, as expected from its lower elevation angle. The average C/N_0 of each satellite is proportional to its elevation: Code Number 20, characterized by the highest C/N_0, oscillating in the range 45 to 50 dBHz, was in fact the one with highest elevation. Also the rising and setting trend is respected.

Pseudo-range estimate

Estimate of the pseudo-range of the four satellites are also reported in NMEA messages at 1 Hz rate and depicted in Figure 5.79. It is noted that the Teseo 2 gives as output raw range measurements, which need to be turned into pseudo ranges. Data are also affected by a nominal drift of about 9 000 m/s due to the nominal clock offset. This is clear from Figure 5.80, where a zoom of the third plot of Figure 5.79 is reported.

Doppler frequency estimate

The Doppler frequency estimate by the FLL of the Teseo 2 is reported in Figure 5.81; results are consistent.

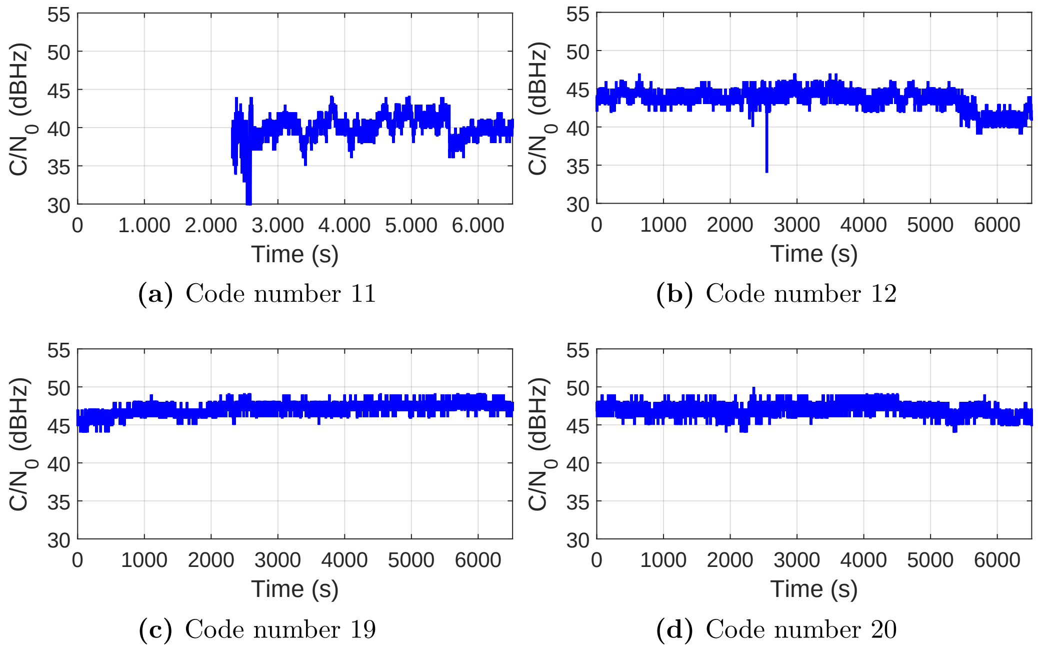


Figure 5.78: C/N_0 of Galileo IOV satellites as estimated by the Teseo 2.

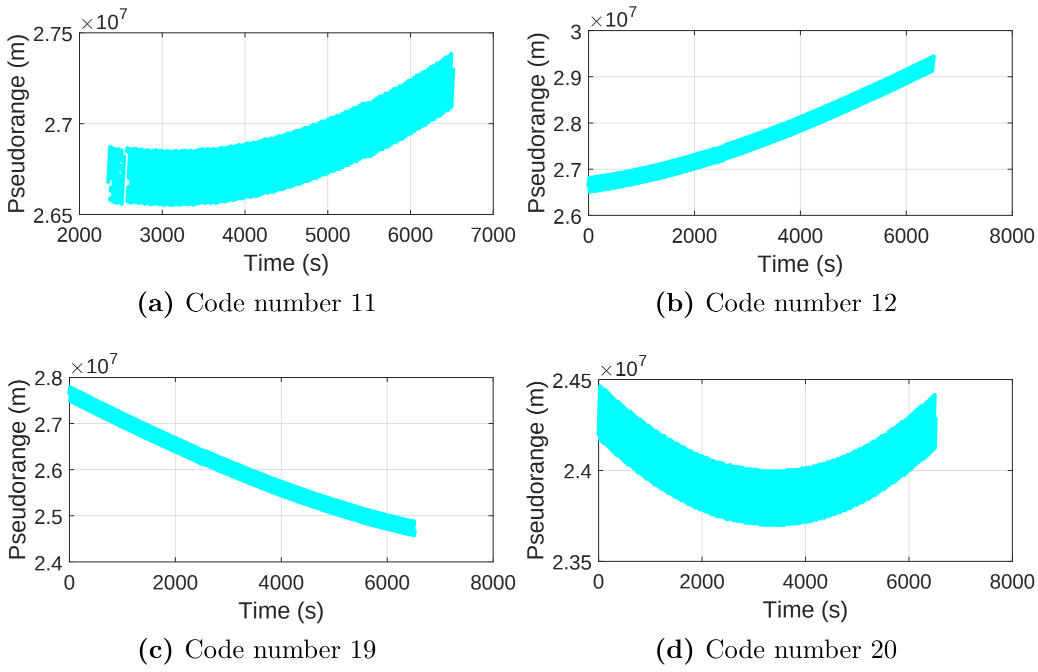


Figure 5.79: Pseudo-ranges of Galileo IOV satellites as estimated by the Teseo 2.

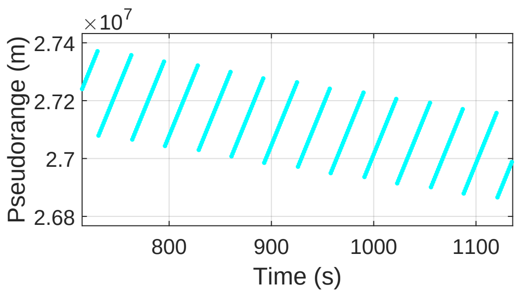
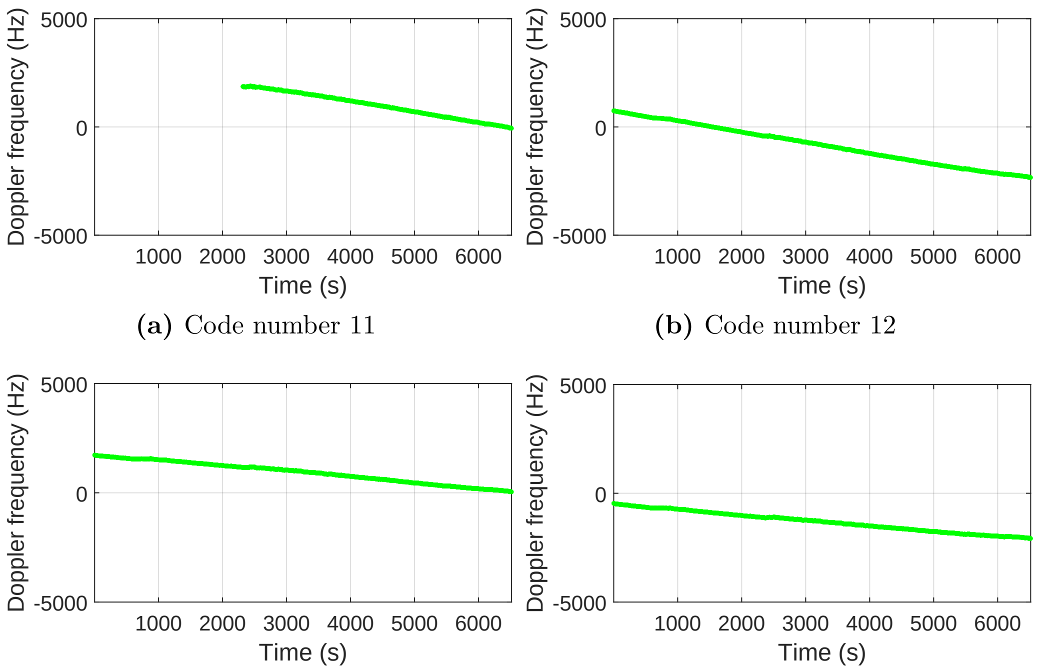


Figure 5.80: Zoom on the pseudo-range estimate of satellite 19.



(a) Code number 11

(b) Code number 12

(c) Code number 19

(d) Code number 20

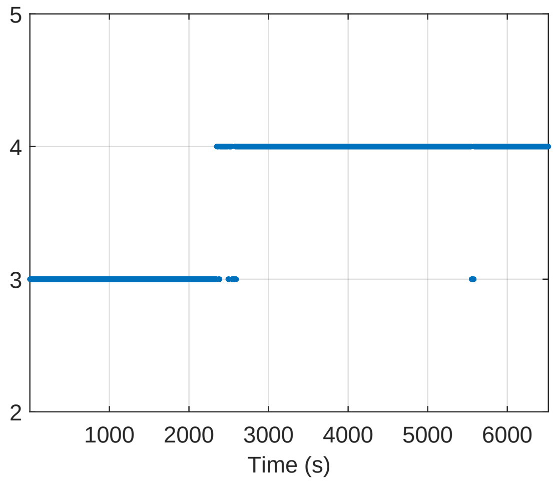
Figure 5.81: Doppler frequency of Galileo IOV satellites as estimated by the Teseo 2.

Satellites in tracking

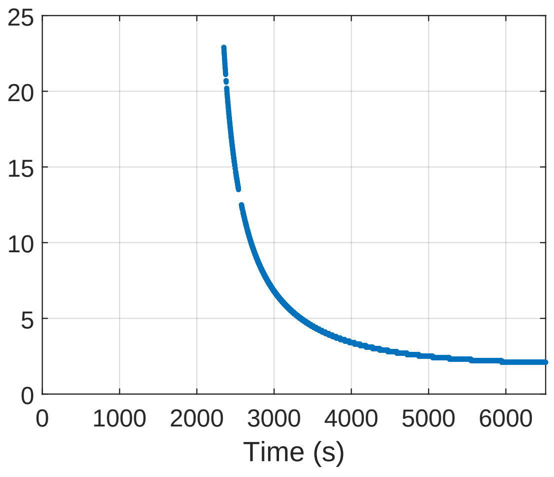
The number of satellites in tracking state is reported in Figure 5.82a. For the first 30 minutes they are only 3, namely Code Numbers 12, 91 and 20. Code number 11 was low on the horizon, and shadowed by buildings. Around 2 300 s after the test started, the van started to move, and for a few seconds the SV 19 is acquired, even if several LOLs are experienced. Again, around about 5 600 s, one on the four signals exhibits a LOLs.

Horizontal Dilution of Precision

It is also interesting to analyse the HDOP, as expected from the sky-plot. It is noted that a reliable value of the HDOP is obtained only when 4 or more satellites are in tracking. As already



(a) Satellites in tracking



(b) HDOP

Figure 5.82: Teseo 2 results.

outlined this happens only after about 30 minutes. At about 5 600 seconds the HDOP explodes again, because of a LOL of one satellite, as seen in Section 5.10.1. Excluding the outliers due to the presence of only 3 SV, a general decreasing trend of the HDOP is evident from the figure, sign that the geometry is improving.

Position solution

The position solution in latitude, longitude and altitude is computed by the Teseo 2. Results are reported in Figure 5.83.

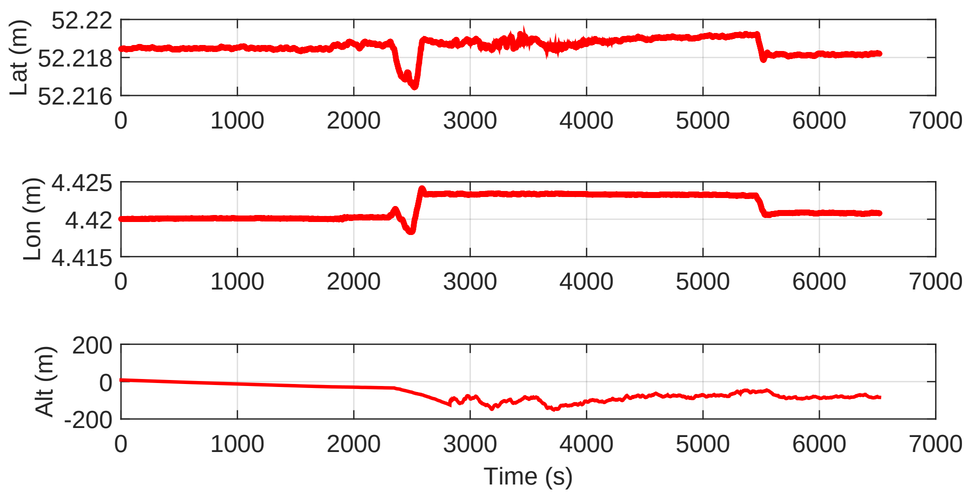


Figure 5.83: Teseo 2 position solution.

The same results are plotted in Google Earth, as shown in Figure 5.84 Once more it is outlined that the results obtained have to be considered preliminary. First because it was the first test with

IOV satellites and with the limited infrastructure deployed so far. Second because the firmware supporting Galileo was in an initial test phase.

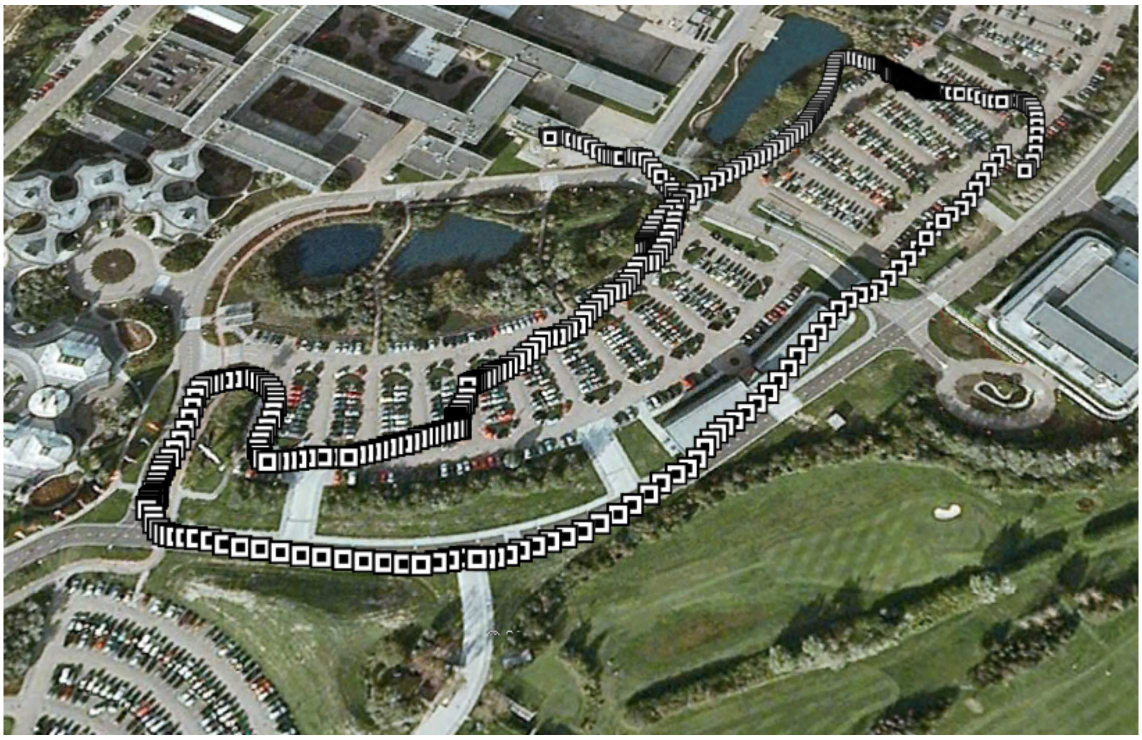


Figure 5.84: Galileo only Teseo 2 mobile fix, computed on March 12, 2013.