POLITECNICO DI TORINO

Department of Electronics and Telecommunications

Master's degree programme in ICT for Smart Societies

Master Thesis

Improved Positioning techniques for positioning based on raw GNSS measurements from smartphones

Supervisor:
Prof. Fabio Dovis

Candidate:
Giampaolo Marinaro

July 2019

Acknowledgements

I would like to thank my supervisor prof. Fabio Dovis who has guided me through these months.
I am grateful to Neil Gogoi for his help and for his boundless availability.
Finally, I would like to thank my family and my girlfriend Virginia for their continuous emotional support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

User's location is a crucial information for a large number of applications installed on our smartphones. It is no coincidence that Google is constantly working on strategies to improve the position on its Android Operative System (OS) and that today manufacturers are mounting modern GNSS chipsets with increasing performances from this point of view. Access to Android GNSS raw measurements are the most interesting tool among those introduced as they allow developers to implement their own strategies to get location. This is a very important innovation because it allows to calculate the position based on the data collected directly from the smartphone's GNSS receiver without having to rely on the final position provided by the operating system. Android GNSS raw measurements were introduced with Android 7 (Nougat) and During Google I/O '17 and Google I/O '18 positioning supervisors explained how the new Application Programming Interfaces (APIs) and devices can potentially achieve very high accuracy and precision, in particular they said that it is possible to move from 5-meters to 1-meter accuracy in outdoor applications. Moreover, some smartphone models can decode the navigation message from the satellites broadcast signal, meaning that it could be possible to evaluate the user's position independently from the network.

The theoretical capabilities of these new technologies can be compared to those of expensive high-precision GNSS receivers and the applications could be various. The goal of this thesis is to analyse in depth if the promised results can be achieved in real applications and if nowadays smartphones can really act as GNSS receivers. In particular the possible strategies to achieve high performances in terms of Position, Velocity and Time (PVT) estimation are analysed comparing the Google strategies with others suitable for the context. Only data from the GNSS receiver are taken into account and other sensors are not considered.

This thesis is composed by seven chapters. In the second one the main concepts of GNSS are explained; the third provides a detailed analysis of the Android GNSS raw measurements needed to evaluate and improve the PVT estimation. In

the fourth part several strategies and algorithms to improve measurements quality and positioning performance are implemented and tested, with a detailed description of the developed strategy to decode the GPS Navigation Message coming from the Android API. In the fifth chapter some tests were performed to implement a multi-GNSS constellation system, the results of which are presented. The sixth chapter presents a case study in which some of the algorithms previously developed are applied to Cooperative Positioning, in particular two smartphones share GNSS information to calculate the geometric distance between them. In the last chapter there are the conclusions which remark the main findings.

Chapter 2

GNSS basics

2.1 Global Navigation Satellite Systems

A Global Navigation Satellite Systems (GNSS) is a system that provide autonomous geo-spatial positioning by means of electromagnetic signals sent by a constellation of satellites orbiting around the earth. There are 4 GNSSs: GPS (USA), GLONASS (RUSSIA), BeiDou (China) and Galileo (Europe). As June 2019 GPS and GLONASS are fully operational whereas BeiDou and Galileo are scheduled to reach full operational capability by 2020. They all aim at providing a global coverage on the Earth surface. There are also some systems that offer regional coverage, in particular it is important to mention NAVIC, which aims to cover the Indian state, and QZSS which instead wants to increase the quality of GPS in Japan and the surrounding area. A GNSS has 3 segments: space, control and user. The space segment is composed by a satellite constellation placed in Low or Medium Earth Orbit; the control segment consists of several stations placed on the earth, they manage and control the system; the receivers that process the signals broadcasted by the satellites are the user segment. Among the different GNSS systems, the most popular is GPS which allows sufficient accuracy to most applications. Nevertheless, there is still a large potential for improvement and for this reason GPS is implementing a modernisation plan.

Apart from the position, the receiver outputs a very high precision time, so GNSSs can be used for applications that need synchronization. In Section 2.2 we are going to briefly describe how GNSSs work. More details and information about GNSS functional principles can be found in [2] and [5].

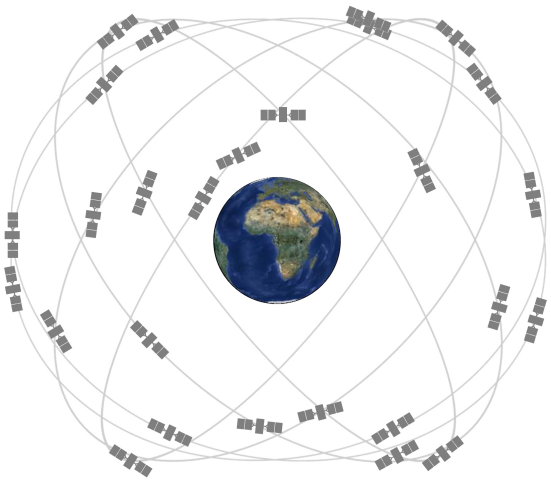


Figure 2.1: Example of constellation: GPS expandable 24 Slots constellation [4]

2.2 Functional principles

In general, radionavigation systems are based on trilateration: the position is obtained by the intersection of geometrical loci, named Line of Position. In GNSS the geometrical loci are spheres whose centres are the satellites. The radii of these spheres are the estimated distances between the receiver and the satellites. These distances are evaluated by means of the propagation time. Since GNSS uses a one-way approach (the users only receive the signals), all the satellites must be synchronized. The position of the satellites is transmitted with the code (actually, the user receives the ephemeris which contains the orbits' parameters through which it is possible to calculate the position), so, theoretically, 3 satellites should be enough to evaluate the user position, because they intersect at 2 two points, and only one is located close to the Earth surface. Since the receiver clock is not synchronous with the GNSS clock, there is one more unknown: the time difference between the system and the receiver; that's why at least 4 satellites in line of sight are needed to estimate the position. The basic operation of GNSS and the main elements needed to understand the algorithms used in the tests are described in this Section.

The GNSS receiver has to estimate the distance user-satellites; this estimation is the pseudorange, that can be defined as

$$\rho = c\tau + c\delta t_u, \tag{2.1}$$

where c is the speed of light, τ is the transmission time and δt_u is the user clock bias. (2.1) is the code phase raw pseudorange equation and it does not take into account non-deterministic impairments (ionosphere and troposphere contribution,

ephemeris errors, relativistic effect, etc.) which are discussed in Section 2.4. If the receiver measures four psudoranges with respect to four satellites, it can determine four unknowns: the user coordinates (x_u,y_u,z_u) and the bias of the clock with respect to the GNSS time scale (δt_u) [18]. The set of equations that the receiver has to solve is

$$\begin{cases} \rho_1 = \sqrt{(x_1 - x_u)^2 + (y_1 - y_u)^2 + (z_1 - z_u)^2} - b_{ut} \\ \rho_2 = \sqrt{(x_2 - x_u)^2 + (y_2 - y_u)^2 + (z_2 - z_u)^2} - b_{ut} \\ \rho_3 = \sqrt{(x_3 - x_u)^2 + (y_3 - y_u)^2 + (z_3 - z_u)^2} - b_{ut} \\ \rho_4 = \sqrt{(x_4 - x_u)^2 + (y_4 - y_u)^2 + (z_4 - z_u)^2} - b_{ut} \end{cases} \tag{2.2}$$

where x_j, y_j and z_j are the satellites coordinates and b_{ut} is the bias introduced by the clock misalignment that is common to all the measurements:

$$b_{ut} = c \cdot \delta t_u \tag{2.3}$$

Generally the solution of the system is computed through an iterative Least Mean Square approach, but the pseudoranges' equations are first linearised through a Taylor expansion around an initial approximation of the location ($\hat{x}_u$,$\hat{y}_u$,$\hat{z}_u$). The closer the approximated position is to the user' position, the shorter will be the time needed to get a good result. In general, the receiver uses the last known position, if the receiver memory is empty, it will start from the earth's centre. The first order approximation of the pseudorange equation for j^{th} satellite is

$$\Delta\rho_j = a_{xj}\Delta x_u + a_{yj}\Delta y_u + a_{zj}\Delta z_u - \Delta b_{ut} \tag{2.4}$$

where a_{xj}, a_{yj} and a_{zj} are the Taylor coefficients, while Δx_u, Δy_u and Δz_u are the increments in user position:

$$a_{xj} = \frac{x_j - \hat{x}_u}{\hat{r}_j}, \qquad a_{yj} = \frac{y_j - \hat{y}_u}{\hat{r}_j}, \qquad a_{zj} = \frac{z_j - \hat{z}_u}{\hat{r}_j}, \tag{2.5}$$

where $\hat{r}_j$ is the geometric range between the linearization point and the satellite:

$$\hat{r}_j = \sqrt{(x_1 - \hat{x}_u)^2 + (y_1 - \hat{y}_u)^2 + (z_1 - \hat{z}_u)^2}. \tag{2.6}$$

The linearized system can be written as

$$\mathbf{\Delta}\rho = \mathbf{H} \cdot \mathbf{\Delta x} \tag{2.7}$$

where $\mathbf{H}$ is a 4 x 4 Jacobian matrix that contains the coefficients, so the unitary vectors steering from the approximation point towards the satellites and $\mathbf{\Delta x}$ is the searched solution:

$$\mathbf{H} = \begin{bmatrix} a_{x1} & a_{y1} & a_{z1} & 1 \\ a_{x2} & a_{y2} & a_{z2} & 1 \\ a_{x3} & a_{y3} & a_{z3} & 1 \\ a_{x4} & a_{y4} & a_{z4} & 1 \end{bmatrix} \tag{2.8}$$

$$\mathbf{\Delta x} = \begin{bmatrix} \delta x_u \\ \delta y_u \\ \delta z_u \\ \delta b_{ut} \end{bmatrix} \tag{2.9}$$

If four satellites are used, the solution is

$$\mathbf{\Delta x} = \mathbf{H^{-1}} \mathbf{\Delta} \rho \tag{2.10}$$

If more than 4 satellites are used, the dimension of the H matrix is N x 4, where N is the number of satellites. In general, by using more satellites the geometry improves and the accuracy of the solution increases. In order to calculate the solution with more than four rows in the H matrix, in general approach, a Least Square solution (LS) is used, the gradient of the LSE is set to zero.

In LS all the measurements are treated in the same way. Nevertheless, in a real scenario not all the measurements have the same quality, meaning they don't have the same error [18]. A symmetric, positive-definite weighting matrix **W** is then introduced [18]. Then the equation that the GNSS receiver has to solve to get the searched solution is

$$\mathbf{\Delta x} = \left(\mathbf{H^T W H}\right)^{-1} \mathbf{H^T W} \mathbf{\Delta} \rho. \tag{2.11}$$

W can be taken as the inverse of the covariance matrix of the position error **R**, in which the off-diagonal elements indicate the level of cross-correlation between the variables; (2.11) becomes

$$\mathbf{\Delta x} = \left(\mathbf{H^T R^{-1} H}\right)^{-1} \mathbf{H^T R^{-1}} \mathbf{\Delta} \rho. \tag{2.12}$$

This particular solution corresponds to the Best Unbiased Minimum Variance Estimator (BLUE) [18]. Unfortunately the values of the measurement errors and the covariance matrix are usually unknown, so in general the measurements from different satellites are considered uncorrelated [18]. Then, the **W** matrix becomes

$$\mathbf{W} = \begin{bmatrix} \frac{1}{{\sigma_1}^2} & & \\ & \ddots & \\ & & \frac{1}{{\sigma_n}^2} \end{bmatrix}, \tag{2.13}$$

where ${\sigma_i}^2$ is the uncertainty (UERE) of the i^{th} satellite, given by the sum of the contributions of each error source [18], as mentioned in Section 2.4. This parameter has to be estimated because it depends on different contributions. In this work three techniques for the ${\sigma^2}_{UERE}$ estimation are considered and they are discussed in Section 4.1.

2.3 GNSS Signals

GNSS satellites continuously transmit navigation signals at two or more frequencies in L band(range of frequencies from 1 to 2 gigahertz). These signals contain ranging codes and navigation data to allow users to compute the pseudorange and the satellite's position at any epoch [18]. The signal is mainly composed by three elements, the carrier, the ranging code (PRN) and the navigation data, as it is shown in Figure 2.2, kindly taken from [2]. The carrier is a sinusoidal signal at a given

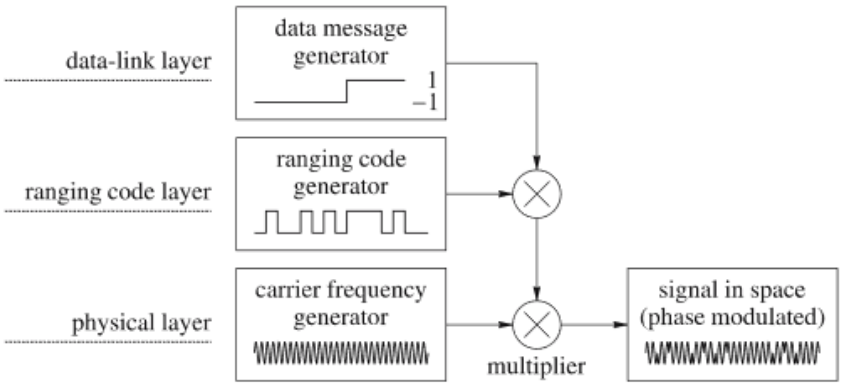


Figure 2.2: Composition of the navigation satellite signal [2]

frequency, the ranging code is a binary sequence different for each satellite through which the receiver can estimate the pseudorange, and lastly the navigation data is a message providing all the information that the receiver has to know about the satellite (ephemeris, clock parameters, almanac, satellite health status)[16].

The GNSS signal-in-space (SIS) can generally be represented as

$$s_{RF}(t) = \sqrt{2P_T}c_b(t)d(t)cos(2\pi f_{RF}t) \tag{2.14}$$

where P_T is the power associated to the channel, $d(t)$ is the data signal which might be unavailable in some channels, f_{RF} is the carrier frequency, t is the time instant and

$$c_b(t) = c(t) \cdot s_b(t) \tag{2.15}$$

is the product of the PRN $c(t)$ and a subcarrier frequency $s_b(t)$. The subcarrier is usually a sinusoidal square wave whose frequency is a multiple of the carrier and it is currently present in certain channels only. The concept of subcarrier has been introduced because thanks to an agreement signed in 2004 between the USA and the European Commission (EC), the common signal baseline structure for the

open access GPS and Galileo signals at the L1/E1 band is based on Binary Offset Modulation (BOC), in particular BOC(1,1). The BOC modulation is also used in other Galileo signals; it is usually denoted as BOC(m,n), where m and n are two integer numbers, the former represents the subcarrier frequency in multiples of 1.023 MHz while the latter gives the chip rate in multiples of 1.023 Mega chips per second (Mcps). The generic BOC(m,n) signal can be written as

$$s(t) = \sum_{n=-\infty}^{\infty} \left\{ c_i d_i r\left(t - i\frac{T_R}{n}\right) sign\left[sin\left(2\pi \frac{m}{T_R} t\right)\right]\right\} \tag{2.16}$$

where $f_R = \frac{1}{T_R} = 1.023MHz$, c_i is the sequence of the code chips values, d_i is the data and $r(t)$ is a rectangular pulse shape of unitary amplitude and duration $\frac{T_R}{m}$. The signal

$$s_{sin}(t) = sign\left[sin\left(2\pi \frac{m}{T_R} t\right)\right] \tag{2.17}$$

is a sinusoidal square wave representing the subcarrier. This type of modulation is denoted as $BOC_{sin}(m,n)$; in literature also the $BOC_{cos}(m,n)$ modulation is present and the subcarrier is given by a cosinusoidal square wave. Since the USA/EC agreement stressed the possibility to improve the signal, thanks to a new agreement signed in 2006, the new concept of Multiplexed-BOC (MBOC) has been introduced. More information about this technique and all the kinds of BOC used in GNSS can be found at [26].

GPS provides two services, the Standard Positioning System (SPS), a free service for users worldwide, and the Precise Positioning Service (PPS), a restricted service for military purposes. The frequencies used by GPS are Link 1 (L1), Link 2 (L2) and the newer Link 5 (L5). On these frequencies, several PRN codes and messages are modulated, in particular it is important to mention the Coarse Acquisition (C/A) code, a sequence containing 1023 bits and repeated every milliseconds, and the Navigation Message, modulated at 50 bps reporting on ephemeris and satellite clock drifts, ionospheric model coefficients etc [18]. For a complete overview on frequency allocation and PRN codes, see [18]; in this work, civil signals broadcast on L1 and L5 are considered. GPS uses the Code Division Multiple Access (CDMA) in order to allow the satellites to transmit on the same frequencies and to identify the satellites without ambiguity. Moreover codes are mutually orthogonal in order to permit the receiver to separate the signal of a satellite from the others. The L5 frequency is the result of the GPS modernisation plan started in 2005 with the launch of the first IIR-M satellite. This plan also aims to the introduction of a pilot channel (a channel without data transmission) on both, L1 and L5 frequency, in order to help the receiver work [4]. Table 2.1 shows an overview of the GPS signals, frequencies and applied modulations.

Link	**Carrier freq.** (MHz)	**Channel or sig. comp.**	**Modulation type**	**Code rate** (Mcps)	**Data rate** (bps)	**Services**
L1	1575.420	C/A	$BPSK(1)$	1.023	50	Civil
		P	$BPSK(1)$	10.23	50	Military
		M	$BOC_{sin}(10,5)$	5.115	N/A	MIlitary
		L1C-I data	$MBOC(6,1,1/11)$	5.115	50	Civil
		L1C-Q pilot			-	
L2	1227.600	P	$BPSK(10)$		50	Military
		L2C M L	$BPSK(1)$	10.23	25	Civil
					-	
		M	$BOC_{sin}(10,5)$		N/A	Military
L5	1176.450	L5-I data	$BPSK(10)$	10.23	50	Civil
		L5-Q pilot			-	

Table 2.1: Courtesy of [18]: Summary of the current and future GPS signals,frequencies and applied modulations. The ranging code rate and data rateare also given in the table.

Galileo system offers more services than GPS: Open Service (OS), a free service generally used combined with GPS, Public Regulated Service (PRS), intended for security authorities, High Accuracy Service (HAS) ,previously known as Commercial Service (CS), which promise higher performance by providing an additional navigation signal (encrypted), Search and Rescue (SAR), that contributes to international COSPAS-SARSAT programme, which provides accurate and reliable alert and location data to help authorities assist people in distress [18]. For more information on services and frequency allocation see [18] and [27]. Galileo uses CDMA technique, and the frequency dedicated to civilian uses are currently E1, E5a and E5b. Table 2.2 presents an overview of Galileo signals, frequencies and applied modulations. The signal component sent by Galileo satellites include a data channel and a pilot channel, which does not contain data and, since no bit transition occurs, makes the signal travelling time easier to be estimated by the receiver in the tracking stage. For example, on E1 frequency, the data channel is E1-B and the pilot channel is E1-C. Both provide the ranging code, but only the former contains the navigation message, which is different for each service provided; in particular the Free accessible Navigation Message (F/NAV), the Commercial Navigation Message (C/NAV) and the Governmental Navigation Message (G/NAV)[18] are broadcast. In this work only GPS and Galileo are considered, for information about other systems, see [18].

Band	**Carrier freq.** (MHz)	**Channel or sig. comp.**	**Modulation type**	**Code rate** (Mcps)	**Data rate** (bps)	**Services**
E1	1575.420	E1-A data	BOC_{cos}(15,2.5)	2.5575	N/A	PRS
		E1-B data	$MBOC$(6,1,1/11)	1.023	125	OS,HAS
		E1-C pilot			-	
E6	1278.750	E6-A data	BOC_{cos}(10,5)	5.115	N/A	PRS
		E6-B data	$BPSK$(5)		448	HAS
		E6-C pilot			-	
E5a	1176.450	E5a-I data	$BPSK$(10)	10.23	25	OS
		E5a-Q pilot			-	
E5b	1207.140	E5b-I data	$BPSK$(10)	10.23	125	OS,HAS
		E5b-Q pilot			-	

Table 2.2: Courtesy of [18]: Galileo navigation signals. The two signals located in the E5a and E5b bands respectively are modulated onto a single E5 carrier frequency of 1191.795MHz using the AltBOC technique: AltBOC(15,10)

As mentioned above, satellites transmit the navigation massage, providing all the information that the receiver has to know about the system and the satellite itself. The GPS navigation message (NAV) is modulated at 50 bps, the whole message contains 25 frames of 30 seconds each; the master frame takes 12.5 minutes to be transmitted. Every frame is composed by 5 subframes of 6s (10 30-bits words) [18]. Every subframe starts with the telemetry word, that is necessary for synchronization. Table 2.3 provides a brief description of each subframe. Subframe 1, 2 and 3 are repeated every 30 seconds, Subframes 4 and 5 contain different pages, but data are the same for each satellite. For more information about the GPS Navigation message, see [21]. Since Galileo navigation message structure is beyond the scope of this work, its details are not described here, more information can be found at [18].

2.4 Error sources

So far, the signal sent by the satellite has been considered as travelling through the vacuum in a static scenario. Actually, the earth and the satellites are moving at very high speeds and the signal is subject to the effect of the atmosphere. So, there are some errors which must be considered. Some contributions are predictable and have to be corrected. After the corrections though there are still errors in the pseudorange. So residual contributions are then modelled as independent Gaussian random variables with zero mean and variance σ_j^2, identically distributed on the

Subframe number	Content
1	Parameters for clock correction (polynomial coefficients) and satellite health condition
2-3	Ephemerides
4	Parameters for ionospheric model, UTC information and part of the almanac
5	Data from almanac and constellation status

Table 2.3: GPS:Data broadcast by subframes [18]

different pseudoranges, and the total pseudorange error can be defined as

$$\sigma_{UERE} = \sqrt{\sum_j \sigma_j^2}[m] \tag{2.18}$$

where $UERE$ stands for User Equivalent Range Error.

The errors affecting the pseudorange measurements have different sources, the main ones are misleading information broadcasted by the GNSS (i.e. on-board clock, ephemeris, payload failures), atmosphere effects (ionosphere and troposphere), multipath (unpredictable reflections of the signal on obstacles), receiver noise, uncompensated relativistic effects [18]. Each receiver applies corrections for the expected contributions of the bias of the satellite clock, the tropospheric and ionospheric delay and the relativistic effect. The satellite clock's bias with respect to the system is sent in the navigation message, but there is a small relativistic correction caused by the orbital eccentricity that has to be modelled [18]. The troposphere is the layer of the atmosphere that extends in the first 60 km from the Earth. Its effect on the signal is an extra delay that depends on temperature, pressure and humidity as well as the transmitter and receiver antenna locations. The delay is modelled using the refractivity parameter that is composed by hydrostatic and wet components. The former is caused by the dry gases in the troposphere, the latter is caused by water vapour [18]. The ionosphere extends from 60 km up to to 2000 km, it contains a partially ionised medium. Its contribution is strongly dependent on the Total Electron Content (TEC), defined as the number of electrons in a tube of $1m^2$ cross section from receiver to satellite:

$$I_p = \frac{40.3 \cdot TEC}{f^2}, \tag{2.19}$$

where I_p is the ionospheric contribution and f is the carrier frequency. TEC parameter changes depending on the location of the receiver, the hour of the day and the intensity of the solar activity. Dual frequency receivers can completely remove the ionosphere effect as shown in (2.20), but the contribution of the other sources increases:

$$\rho^{\star} = \frac{{f_1}^2 \rho_1 - {f_2}^2 \rho_2}{{f_1}^2 - {f_2}^2}, \tag{2.20}$$

where $\rho^{\star}$ is the iono-free pseudorange, ρ_1 is the pseudorange evaluated on frequency f_1 and ρ_2 is the pseudorange evaluated on frequency f_2 [18].

Single-frequency receivers have to apply a ionospheric correction using a model. GPS satellites broadcast the parameters needed to apply the Klobuchar model. This model follows an empirical approach, it is assumed that the electron content is concentrated in a thin layer at 350 km in height [18]; the main advantages of the Klobuchar model are the low computational power required and the low number of parameters that have to be transmitted. Galileo system suggest to use NeQuick model to compute ionospheric corrections. It is a three-dimensional and time-dependent ionospheric electron density model, which provides the electron density in the ionosphere as a function of position and time. Hence, it allows ionospheric delays to be computed as the integrated electron density along any ray path [18]. It is important to remember that after applying the corrections a residual error for each contribution remains and only this residual is modelled as a Gaussian random variable. Table 2.4 reports the error range for each contribution.

Contributing source	**Error Range** (m)
Satellite clocks	±2
Orbit errors	±2.5
Ionospheric delays	±5
Tropospheric delays	±0.5
Receiver noise	±0.3
Multipath	±1

Table 2.4: GNSS System Errors [24]

2.5 The GNSS receiver

GNSS receiver is the tool that process the signals coming from the satellites. User receivers search for the presence of these radio signals that travel through space, and

try to synchronize with them. This way, a GNSS receiver can be seen as a radio-navigation user device that aims at tracking the GNSS signals, in order to correctly demodulate and extract the measurements and navigation information needed to calculate te user position [12].

As it is possible to see in Figure 2.3, a generic receiver is mainly composed by four blocks, the Antenna, the Front End, the Baseband Signal Processing and the Application Processing. GNSS antennas aim at catching signals in L-band, they are the entry point from the space to the user segment. The front end receives the RF inputs form the antenna and, after a down-conversion, performs filtering and digitization. The baseband processing block performs the signal processing tasks, in particular acquisition and tracking of each signal [28]; the acquisition stage is an initial rough estimate of the delay between the incoming code and the local replica and the doppler shift on the carrier, the traking keep the codes syncronized to dynamically recover the delay between sequences, refinement of the doppler shift and of the phase. In the application block the information arriving from the baseband are combined in order to satisfy the requirements of a specific application (e.g. PVT computation)[28].

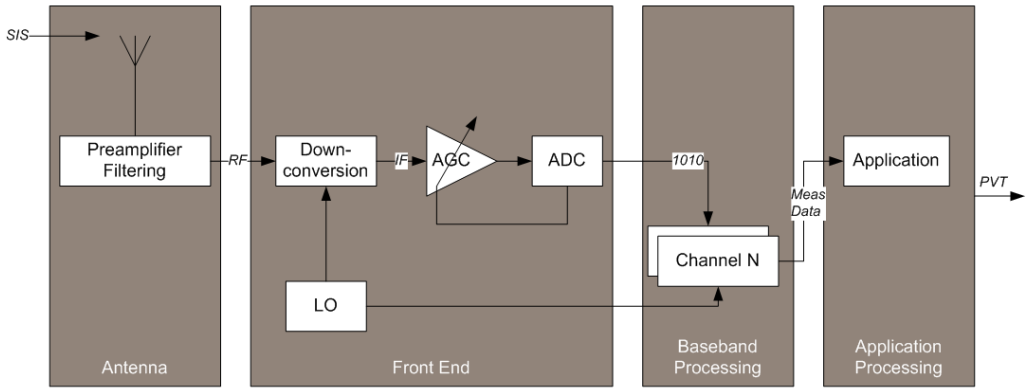


Figure 2.3: Block diagram of a typical GNSS receiver, kindly taken from [28]

In a smartphone the last two blocks in Figure 2.3 are software based and, in general, they receive external data in order to simplify the acquisition and tracking stages and improve the PVT solution. The GNSS/navigation chip can be considered as a black box that outputs the PVT and limited information from the tracked satellites [8]. The Android GNSS raw measurements, that are the main argument of this thesis, come from the baseband processing and can be used to find new algorithm for PVT calculation in mass market devices [8].

Chapter 3

Overview on Android raw measurements

3.1 State of the art

User position is an important information that nowadays smartphones allow to get using several strategies. In general, the main idea of Android devices is to obtain a good enough position for the common users' purposes as finding the fastest path to reach points of interest using Google Maps. Battery consumption has even a significant role and location processes must use less power as possible. The smartest method that an Android developer can follow to obtain a good result for commercial applications is to use the *fused location provider*, a location API in Google Play services that intelligently combines data coming from different sensors (mainly GNSS, cell signal strength, Wi-Fi RSSI) [9]. This API gives the final computed position without showing the followed steps, but starting from Android 7 (Nougat) new strategies are possible for both indoor and outdoor position evaluation. Three important improvements were added to Android smartphones positioning system: 802.11mc protocol support for indoor applications, Raw GNSS measurements and dual frequency receivers. This Section aims to briefly describe these systems.

As mentioned, position calculated by Google libraries is based on different signals coming from several components located inside the smartphone. One of the main components for this purpose is Wi-Fi card. In particular, it is possible to evaluate the RSSI, received signal strength indication, through which the smartphone can be assumed as located to a certain distance from a referenced Access Point. The higher is the RSSI value, the stronger is the received signal and the lower is the distance from the AP. This approach is simple, but it has some limitations and a new method based on Wi-Fi Round Trip Time (RTT) is now available on Android P systems in developer mode. This technique needs IEEE 802.11mc protocol on both, the phone

and the AP [29]. Wi-Fi RTT evaluation works as follows:

1. Throw beacons and probe response the phone scans searching for RTT capable AP;
2. The AP sends an FTM (Fine Timing Measurements) packet;
3. The phone reply with an ACK;
4. The AP sends one more packet with the timestamps related to the exchanged packets.

Using timestamps, the mobile can evaluate the time interval passed between the FTM dispatch and the ACK delivery and then the distance from the AP:

$$distance = \frac{RTT}{2}c, \tag{3.1}$$

where c is the speed of light. This distance is the range from the AP. If only one AP is present, it is possible to suppose that the phone is located on a point of a circumference having as centre the AP and as radius the calculated range. If more compatible APs are available, there are more constraints and the position could be better estimated. Several techniques can be used to improve the quality of the position estimation and the velocity (e.g. WLS and Kalman Filter).

The previous approach is very useful in indoor applications and it will be integrated in fused location provider. The second improvement is the support for the GNSS raw measurements. They are discussed in Section 3.2. Not all the Android smartphones models support raw measurements API and different compatible phones may be able to manage only some of the different information. Android's developers website provides a table with the common smartphone models supporting raw measurements [1]. In this table there is a field called L5 Support, this field says if a particular mobile supports the dual frequency system that is the third important improvement in positioning system in the smartphone market. Until the arrival of Xiaomi MI8, all the smartphones available in the market were equipped with L1-only GPS chipset, meaning that only the signal carried by L1 frequency were kept into account during position estimation. In order to understand the improvements that L5 frequency enabled chipsets can introduce, it is important to mention that GPS position evaluation can be performed through two different approaches, code phase measurements and carrier phase measurements. The first one is easier and it is based on the C/A code carried by satellites signal, the second one is based on the carrier, it is more complex but it allows more accurate results. This second technique can be performed thanks to the newest GNSS chipsets and L5 frequency can help a lot. From code-phase point of view, L5 signal can be used to provide

iono-free pseudoranges. Since ionosphere is a notable error source, this fact could seem a very success, but unfortunately removing iono-effect implies to intensify the effect of other error sources. In this work code-phase pseudorange is considered and then several tests using iono-free pseudoranges have been performed.

3.2 Android GNSS raw measurements API

The location APIs have changed a lot as the Android operating system has been updated. Although today the recommended API to get the position is *fused location provider*, in this thesis we are interested in the new version of *android.location*. In fact starting from Android 7 Nougat (API level 24), this library has been enriched with classes that allow to explore new algorithms to calculate the position obtained through GNSSs and to collect more information about satellites. The new classes that give the requested values to evaluate the pseudorange are *GNSS Clock* and *GNSS Measurements*; in addition we can rely on *GNSS Navigation message* that provides the elements required to decode the navigation message coming from the satellites. Figure 3.1, kindly taken from GSA's white paper [8], shows the main evolutions in *location* library starting from API level 24. In order to encourage

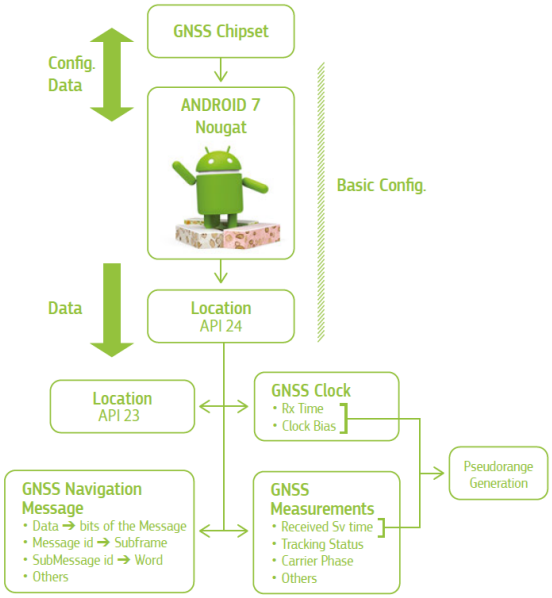


Figure 3.1: Location API starting from API Level 24 [8]

developers to work with GNSS raw measurements, Android has distributed some very useful tools: GNSS Logger, GNSS analysis and GPS Measurements Tool Project

open source MATLAB code [13]. The first is an Android application that collects measurements, the second is an application for Windows, Linux and macOS that analyses data collected through GNSS Logger, the third is a code that provides the basic functionalities to study the collected data. GNSS Logger is the APP that we used to collect data and the open source code is the starting point for our tests. Thanks to GNSS raw measurements, it is not needed to rely on Google evaluated position and it is possible to perform the estimation using different algorithms. This perspective opens a very wide range of possibilities meaning that a simple smartphone can act as real GNSS receiver. This fact is the main point of this research in which we try to understand how to make the best use of raw measurements and which level of accuracy and precision can be reached by common commercial devices using only GNSS. As mentioned in Section 2.5, the Navigation chipset inside the smartphone acts as a black box: since it is a proprietary hardware, it is not possible to know how the data is collected and processed. The main measurements provided by *GNSSClock* and *GNSSMeasurement* classes are reported in Table 3.1, published in [8]. The Table also provides a brief description of each field; Section 3.4 and Section 3.5 explain how they can be used to evaluate the pseudorange and perform some analysis.

ANDROID CLASS	**FIELD**	**DESCRIPTION**
GNSSClock	TimeNanos	GNSS receiver internal hardware clock value in nanoseconds
GNSSClock	BiasNanos	Clock's sub-nanosecond bias
GNSSClock	FullBiasNanos	Difference between TimeNanos inside the GPS receiver and the true GPS time since 0000Z, 6 January 1980
GNSSClock	DriftNanosPerSecond	Clock's drift
GNSSClock	HardwareClockDiscontinuityCount	Count of hardware clock discontinuities
GNSSClock	LeapSecond	Leap second associated with the clock's time
GNSSMeasurement	ConstellationType	Constellation type
GNSSMeasurement	Svid	Satellite ID
GNSSMeasurement	State	Current state of the GNSS engine
GNSSMeasurement	ReceivedSvTimeNanos	Received GNSS satellite time at the measurement time
GNSSMeasurement	AccumulatedDeltaRangeMeters	Accumulated delta range since the last channel reset

GNSSMeasurement	Cn0DbHz	Carrier-to-noise density
GNSSMeasurement	TimeOffsetNanos	Time offset at which the measurement was taken in nanoseconds
GNSSMeasurement	CarrierCycles	Number of full carrier cycles between the satellite ans the receiver
GNSSMeasurement	CarrierFrequencyHz	Carrier frequency at which codes and messages are modulated
GNSSMeasurement	PseudorangeRatemeterperSecond	Gets the pseudorange rate at the timestamp
GNSSMeasurement	ReceivedSvTimeUncertaintyNanos	Gets the error estimate for the received GNSS time

Table 3.1: Main Android raw measurements [8]

3.3 Android Smartphones comparison

In this thesis, the analysed datasets were collected using Xiaomi Mi8 and Xiaomi Mi8 Pro. These smartphones are very interesting from GNSS point of view, in fact they put in as standard the first dual frequency chipset available in smartphone market: Broadcom *BCM47755*. In general, this chipset is very performing compared with other models mounted in TOP-level smartphones, as demonstrated in [22]. Figure 3.2 is kindly taken from this paper and it shows the higher quality of MI 8 with respect to other smartphone models. Despite the support of double frequency, the calculation of the position using phase measurements has not been taken into account in this thesis, but some tests have been made extrapolating the iono-free pseudorange when possible. The results of these analysis are described in the following chapter. Broadcom *BCM47755* is a complete chip capable of receiving signals from GPS, Galileo, Beidu, Glonass and QZSS. As well as on smartphones, it can be mounted on tablets and wearables, thanks also to low power consumption. Despite the excellent performance of this chip, there are some measurements on which it is not always possible to rely, in particular problems have been found in carrier and doppler measurements. The main issues are discussed in Section 3.5. It is important to underline that all the anomalies found are linked to the use of the aforementioned smartphones, and they are not documented in other phone models.

An important consideration that has to be made is that among the prerogatives of the vendors stands out the importance of energy saving. If in conditions of daily use the GNSS chipset worked continuously the battery would have an abnormal

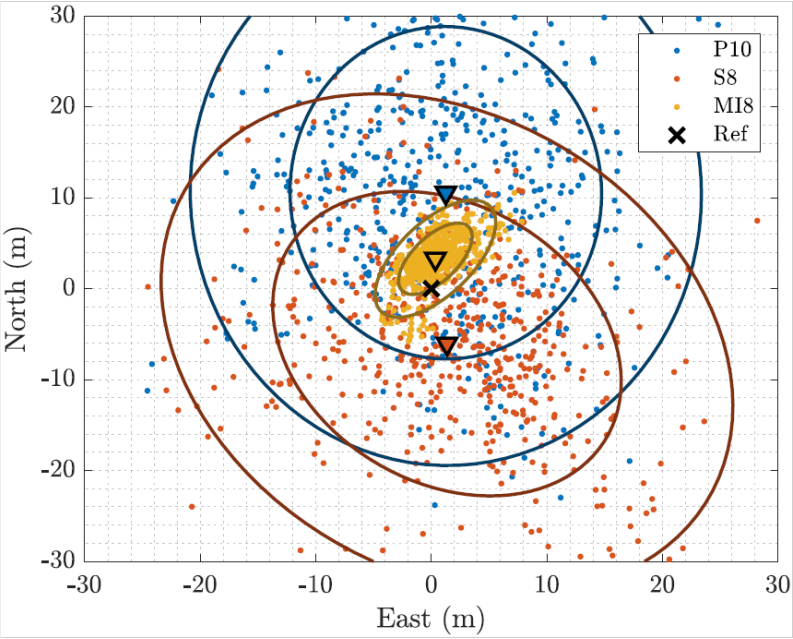


Figure 3.2: PVT comparison among smartphones

consumption, then the duty cycle is introduced. This strategy requires the chipset to enter the tracking phase only in a few fractions of time for each epoch. Duty cycle may cause problems in carrier phase measurements because between two consecutive epochs, several cycle slips can occur. Generally the duty cycle can be deactivated, and it has been noted that Xiaomi Mi8 has no limitations in this respect.

3.4 Code pseudorange computation strategies

As mentioned in Section 2, the main element that a GNSS receiver has to estimate is the pseudorange. The strategy that smartphones adopt to obtain it is based on common reception time; this technique is usually used in commercial receivers and allows pseudoranges to be obtained at any time. More information on the operation can be found in [20]. Android does not provide pseudorange value, but all the elements needed to evaluate it. The GSA's white paper [8] and the Google Matlab code [13] help to take the first steps with raw measurements. Before calculating it, some data coming from the *GNSSMeasurement* class have to be analysed. First of all, the constellation type of each measurement has to be detected, then it is necessary to find the tracking status. This value is given as *State*. In order to obtain full pseudoranges, for GPS-only measurements flagged as *TOW Decoded* can be used. It needs to be checked using a bitwise AND operation, then it is possible to proceed with

pseudorange evaluation. The value of the field *State* is very important to interpret the satellite time at the measurement time (*ReceivedSvTimeNanos*), if Time of Week (TOW) is decoded, it means that the satellite time can assume any value from 0 to 1 week (in nanoseconds).

It is possible to consider the pseudorange as

$$\rho = \frac{t_{Rx} - t_{Tx}}{1e9} \cdot c \tag{3.2}$$

where c is the speed of light, $1e9$ is needed to convert nanoseconds in seconds, t_{Rx} is the measurement time and t_{Tx} is the transmitted time, so the time at which the signal left the satellite transmitter. t_{Tx} is given by the filed *ReceivedSvTimeNanos*, and it is follows its system time. t_{Rx} is provided in GNSS time, so the reference time of the receiver. It is evaluated thanks to three fields of the class *GNSS Clock*: *TimeNanos*, that gives the internal receiver's clock value, *FullBiasNanos*, which provides the time difference between the internal time and the real GPS time starting from the beginning of GPS time and *BiasNanos*, that is the clock's sub-nanosecond bias. It is clear that among the measurements provided by the operating system there is a value that should be available only after the PVT evaluation, in fact the time difference between the internal clock and the system clock is one of the unknowns in (5.3). This would suggest that the GNSS chipset makes some initial calculation of the clock bias. Having said that, the Google MATLAB code suggests to calculate t_{Rx} using *FullBiasNanos*. The first estimation in absolute time of the measurement time can be reconstructed as

$$Time_{GNSS} = TimeNanos - (FullBiasNanos(1) + BiasNanos(1)), \tag{3.3}$$

the index "1" indicates that only the first value of the whole dataset has to be used. $Time_{GNSS}$ has to be differently managed for each system. In particular, for GPS t_{Rx} is obtained by removing the number of nanoseconds occurred between the beginning of GPS time and the beginning of the current week. This value is evaluated as

$$weekNNanos = floor(\frac{-FullBiasNanos}{NNanoSecondsWeek})NNanoSecondsWeek, \tag{3.4}$$

where *NNanoSecondsWeek* is the number of nanoseconds in a week. Then it is possible to evaluate the measurement time in GPS time:

$$t_{rx} = Time_{GNSS} - weekNNanos. \tag{3.5}$$

In the first part of this thesis only GPS is used and only in the last part Galileo has been added. More details about Galileo pseudorange can be found in the next chapters. Since the described approach rely on a measurement that should be available only after the PVT computation, t_{rx} can be obtained following a different strategy.

The typical propagation time for a signal from a satellite to a user receiver on the earth can be approximated to seventy milliseconds ($propagationT$), and then it is possible to initialise the clock value:

$$t_{rx} = TimeNanos - TimeNanos(1) + ReceivedSvTimeNanos(1) + propagationT. \tag{3.6}$$

The results obtained using these two approaches are substantially identical in terms of position, as shown by the Figures 3.3 and 3.4. Since the obtained results are

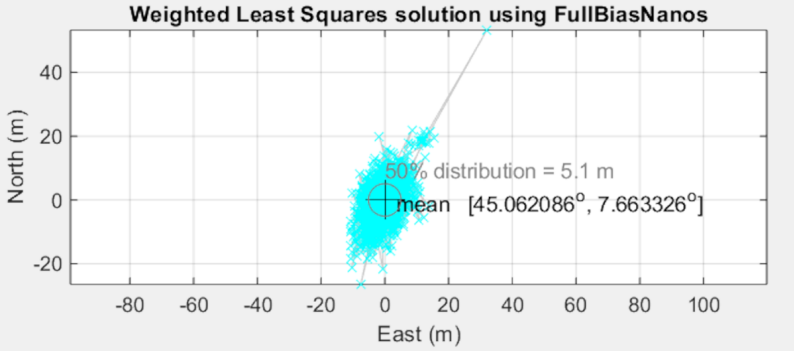


Figure 3.3: PVT with pseudoranges obtained using FullBiasNanos

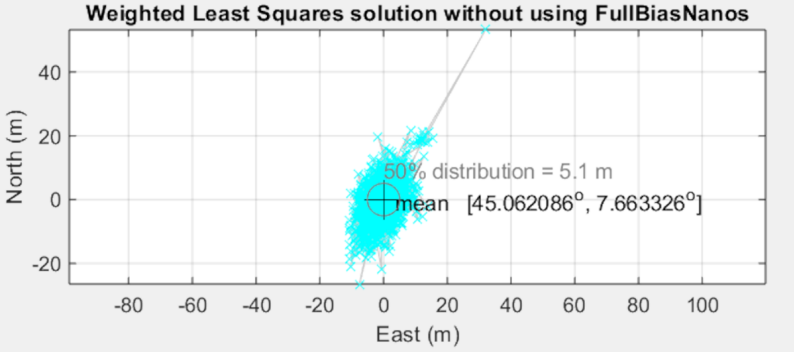


Figure 3.4: PVT with pseudoranges obtained without using FullBiasNanos

identical, the Google approach was chosen for this work because of a greater simplicity in view of a possible application in real time. However, a change was made, in particular, since the receiver clock is not accurate, instead of using only the first value of *FullBiasNanos* it has been chosen to use the corresponding value for each epoch. The evaluated position is the same for both the approaches, but different values in the clock bias are obtained because the second approach already takes

into account part of the bias. After explaining which values are used to calculate the pseudorange, it is important to make some considerations about *BiasNanos* and *TimeOffsetNanos*. In collected measurements they are always equal to zero. The use of the former has already been explained, the latter is defined in Android Developers guide as "the time offset at which the measurement was taken in nanosecond", its reference time is *TimeNanos* and should be always added to its reference time in order to have sub-nanoseconds accuracy. Google suggests to add *TimeOffsetNanos* also in t_{Tx} evaluation.

3.5 Analysis of other important measurements

In addition to the basic data for code based stand alone positioning, Android provides other useful measurements as well. *AccumulatedDeltaRangeMeters* and *PseudorangeRateMetersperSecond* are the values needed to evaluate respectively phase and doppler. Since in general these measurements are of reasonable quality as shown in Figure 3.5, they can be used to smooth the pseudoranges, but unfortunately sometimes they both present some problems.

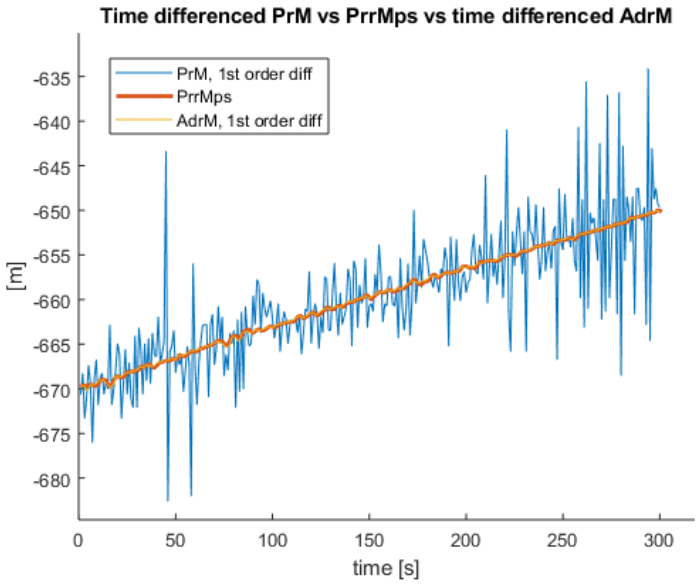


Figure 3.5: Pseudorange rate vs time-differenced Accumulated delta rage vs time-differenced pseudorange (300 epochs)

PseudorangeRateMetersperSecond gives the Pseudorange rate at timestamps, its

value is defined in (3.7) and it can be used to calculate the dopplershift:

$$PseudorangeRateMetersperSecond = -k \cdot dopplershift \tag{3.7}$$

where k is a constant defined as

$$k = \frac{c}{carrierFrequency}, \tag{3.8}$$

where c is the speed of light. In performed tests, *PseudorangeRateMetersperSecond* values are always within the 10^3 order, for surveys lasting approximately 10 minutes, as shown in Figure 3.6. But some measurements are affected by "jumps" in the order of 10^5 as it is possible to see in Figure 3.7. Since the magnitude is very high

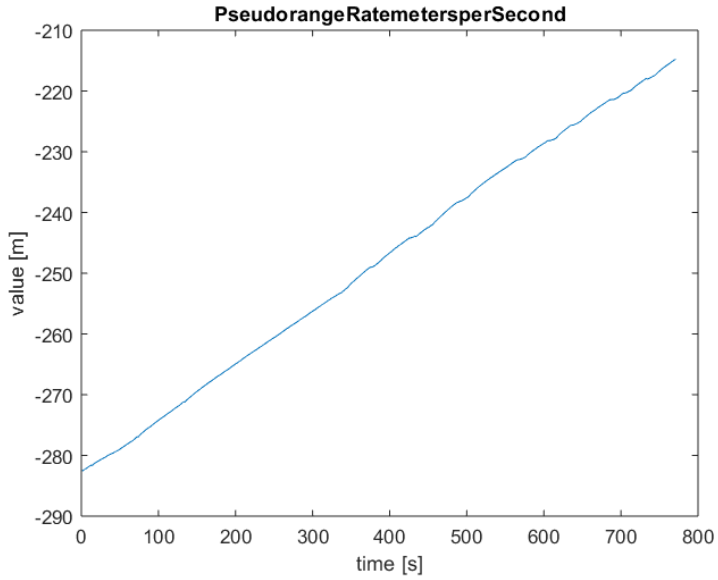


Figure 3.6: Normal measurements of pseudorange rate

and the duration is of a single period, this kind of jumps is easy to detect. In the following chapter it is described how the pseudorange rate is used to smooth the calculated pseudoranges and which strategies are used to overtakes the anomalies in measurements.

A different point should be made for *AccumulatedDeltaRangeMeters*. It is the accumulated delta range since the last channel reset, and it is defined as

$$AccumulatedDeltaRangeMeters = -k \cdot carrierphase, \tag{3.9}$$

where k is defined in (3.8). The measurements of this field depend on *AccumulatedDeltaRangeState*. Since accumulated delta range is unknown without the time information, if a cycle slip occurs the receiver loses this count. So it is important to use

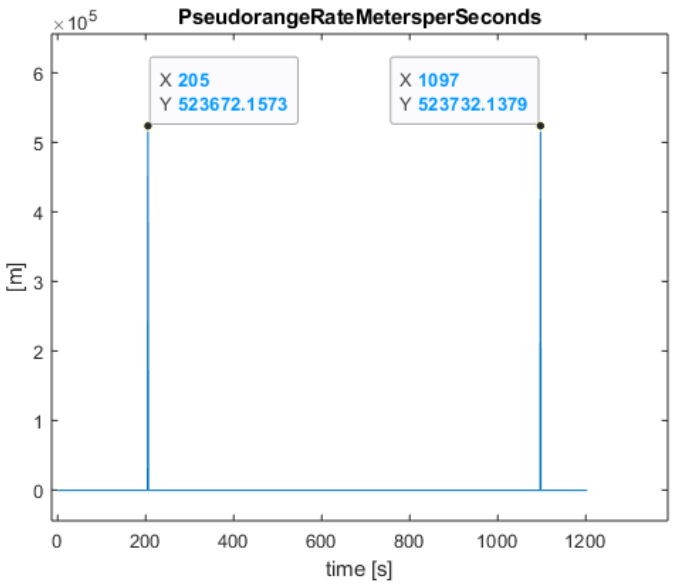


Figure 3.7: Measurements of pseudorange rate affected by anomalies

only measurements flagged as "good" in state field, as described in [8](pp 22-23). Unfortunately the state parameter is not reliable in collected datasets because it often assumes values outside of those expected. In addition, *AccumulatedDeltaRangeState* also presents jumps in measurements carried on L5 frequency. An example of this behaviour is shown in Figure 3.8. The accumulated delta range value can still be used to smooth the collected pseudoranges, and possible strategies are described in the following chapter. Figure 3.9 shows the de-trended values for this measurement on L1 and L5 frequencies for satellite 10 shown in Figure 3.8, it is possible to notice that L5 measurements are affected by anomalies. The magnitude of the jumps is very strong and deteriorates the quality of the measurements. Where there are not problems in the measurements, the values recorded follow the same trend for both frequencies, as shown in Figure 3.10. The jumps are due to the fact that when the receiver can not evaluate the value of *AccumulatedDeltaRangeState*, it gives the values estimated for L1 frequency, so the trend is still the same, but it is misaligned
.

Another important measurement coming from *GNSS Measurements* class is *ReceivedSvTimeUncertaintyNanos*, that in Android developers guide is defined as "the error estimate for the received GNSS time, in nanoseconds". In [13], Google suggests to use this field to build the weighting matrix in a particular epoch. The σ suggested to use as weight is

$$\sigma_{Google} = ReceivedSvTimeUncertaintyNanos \cdot 1e^{-9} \cdot c, \tag{3.10}$$

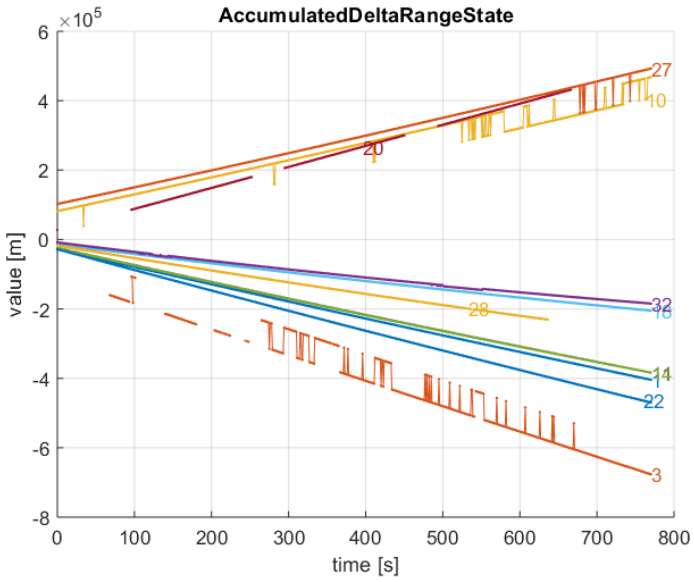


Figure 3.8: Jumps affecting satellites 3,10,27,32 in AccumulatedDeltaRangeState

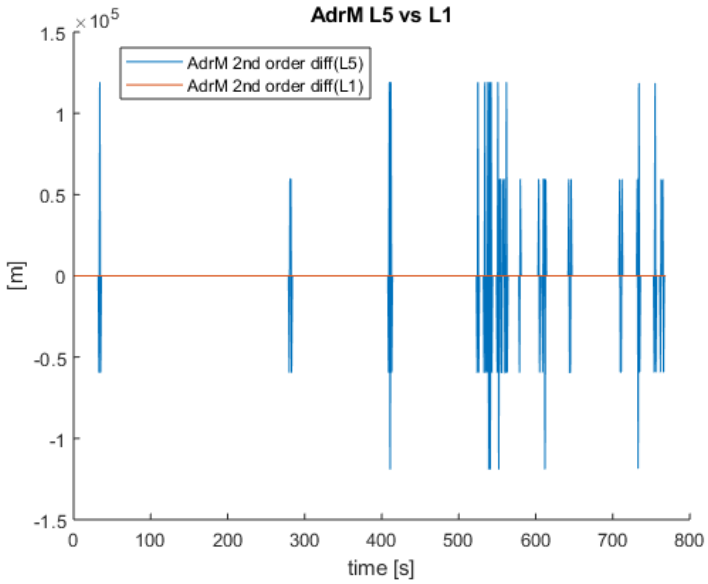


Figure 3.9: Detrented AdrM measurements on L5 and L1 frequencies

where c is the speed of light. The found value has to be inverted in the weighting

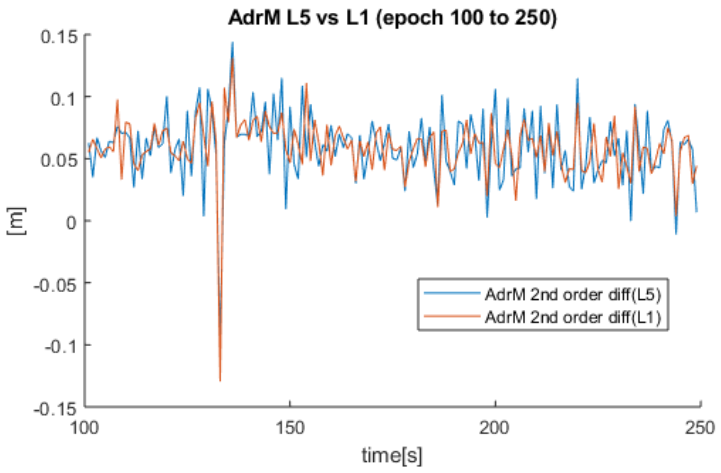


Figure 3.10: Detrented AdrM measurements on L5 and L1 frequencies in no-jumps epochs

matrix: the higher is the value of σ_{Google}, the smaller is the weight of the measurement. The explanation of how *ReceivedSvTimeUncertaintyNanos* is calculated is not present either in the Android developers guide or in the GSA's white paper. The σ_{Google} has been compared with another interesting field, *Cn0DbHz* of *GNSS Measurements* class, which gives the carrier-to-noise density. The comparison plot is shown in Figure 3.11. The correlation between the two measurements is evident, but it is not always clear as in this picture.

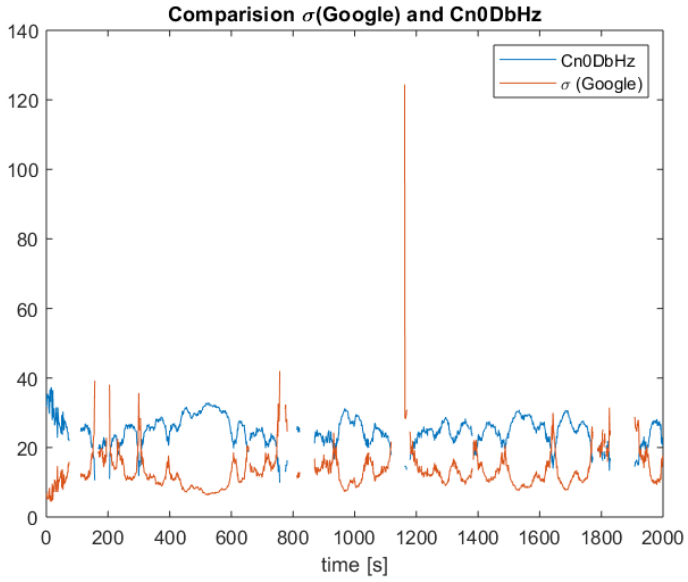


Figure 3.11: Comparison between Google sigma and Carrier-to-noise density

Chapter 4

PVT Computation

4.1 The weighting matrix

As mentioned in Section 2.2, in this work the algorithms used to compute the PVT are based on the Weighted Least Squares (WLS) method. In literature there are different methods to calculate the weighting matrix; for example it is possible to rely on the elevation of the satellites, where the higher the elevation the more the error is considered small [18]. In Google MATLAB code [13], the proposed method is based on the value of the raw measurement *ReceivedSvTimeUncertaintyNanos*, as mentioned in Section 3.5. Since no information is provided on how this parameter is estimated by the GNSS chipset, it is interesting to try to understand how this method behaves compared to others; in particular, in this work two "run-time" methods based on the value of the variance of the de-trended pseudorange measurements are proposed.

The Google MATLAB code [13] suggests to evaluate σ_i using *ReceivedSvTimeUncertaintyNanos* measurements. Unfortunately this value is not well explained neither in the code nor in the Android Developers guide which defines it as "the error estimate (1-sigma) for the received GNSS time". As already mentioned in Section 3.5 this value is strictly related with the *Cn0DbHz*, but by performing more detailed analysis, it is possible to better understand the value of σ_i, evaluated as in (3.10). Figure 4.1 shows the σ value compared with the de-trended pseudorange series. It is clear that there is a connections between the two elements. It is important to remark that the de-trended values are evaluated through a digital differentiation and they are then calculated in the aftermath. The σ value seems to be an upper-bound of the de-trended pseudorange measurements. As already mentioned, there is not documentation about the technique that Android uses to estimate this value and all the considerations about it in this work are based on the performed test; unfortunately there is no way to have access to the inside of the GNSS chipset and it is only

possible to comment and manage its output. The σ_{Google} is updated at each epoch

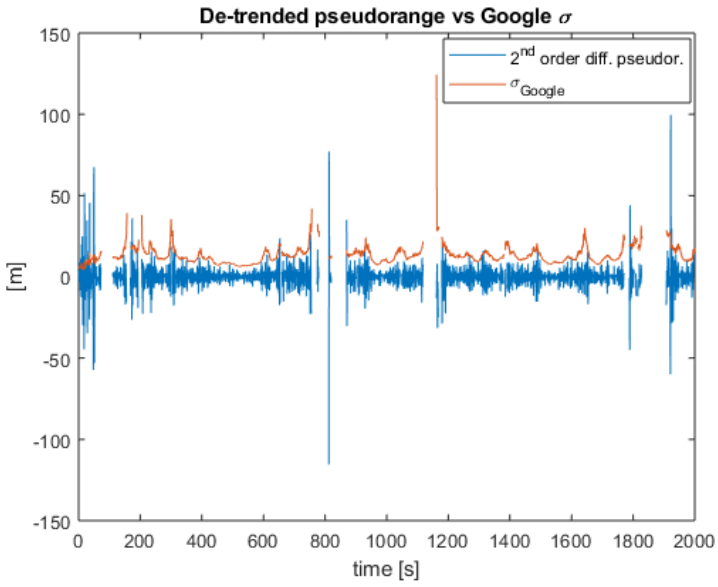


Figure 4.1: De-trended pseudorange series for satellite 28 and σ_{Google} value for each epoch

and provides a real time estimate of the error. The quality of this measurement can not be evaluated, but several tests using all the described approaches have been performed. Another strength of this parameter is that measured values are always comparable with one another, even if measurements come from different frequency channels.

Let's now consider the weighting matrixes built using the de-trended pseudorange measurements. This procedure is described in [7]. It is possible to de-trend the pseudorange measurement using a differentiation of the value series. Being $\rho_i[n]$ the value of the measured pseudorange of the i^{th} satellite at the n^{th} epoch, the first order differentiation is

$$\rho_i^{(1)}[n] = \frac{1}{\sqrt{2}}(\rho_i[n+1] - \rho_i[n]). \tag{4.1}$$

The second order differentiation is enough to remove the parabolic behaviour of the pseudorange measurements and to obtain a zero mean time series, as shown in Figure 4.2. On the differentiated data is now possible to estimate the variance. In this work, a first rough estimation of the σ_i was performed using this approach.

However the normalization is valid only if the samples are uncorrelated; since the pseudorange measurements are provided at a rate of one per second, it is likely that the measurements have correlated error contributions and the differentiation may

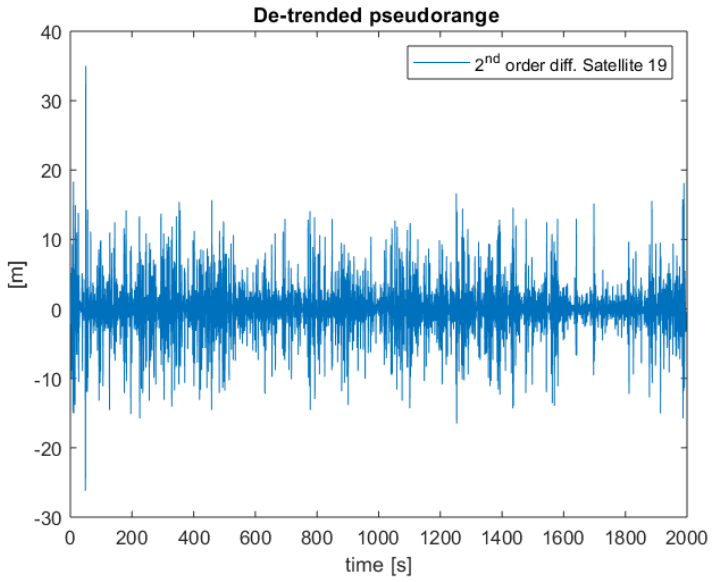


Figure 4.2: De-trended pseudorange series for satellite 19 (2^{nd} order differentiation)

remove some error components [7], so the previous approach gives an overoptimistic estimation of σ_i. A second method is later introduced, in particular it is possible to perform an under sampling operation in order to reduce the cross-correlation between the measurements [7]. As suggested in [7], the under-sampling rate N_s^i is given by the maximum rate that ensures that the mean of the third-order differentiation does not overcome the set value 2.5. The median between all the sampling rates is chosen as sampling interval (N_s) for all the satellites. It is important to notice that if the under-sampling operation is performed, a second order differentiation is not enough to obtain a zero-mean series, so a third-order differentiation is used. The main limitation of the described methodologies is that they are not compatible with real-time applications, or they need a window of measurements large enough to estimate a first value of σ_i. In the tests performed in this work, the weighting matrix is evaluated in the aftermath, considering the whole series of collected pseudoranges.

The described approaches give different solution in terms of weighting matrix because the estimated values of the error are different. Nevertheless, for each method, the values are comparable and even if the estimated error is not truthful, it can still be used as a weight. In order to compare the estimated σ through the different method, four satellites have been selected in a 300 seconds time interval. Satellites have different elevation, as it is possible to see in Figure 4.4 and two of them, SV 10 and SV 24, transmit on both L1 and L5. All the signals are available for the whole time interval. A simple error estimation method based on elevation would consider

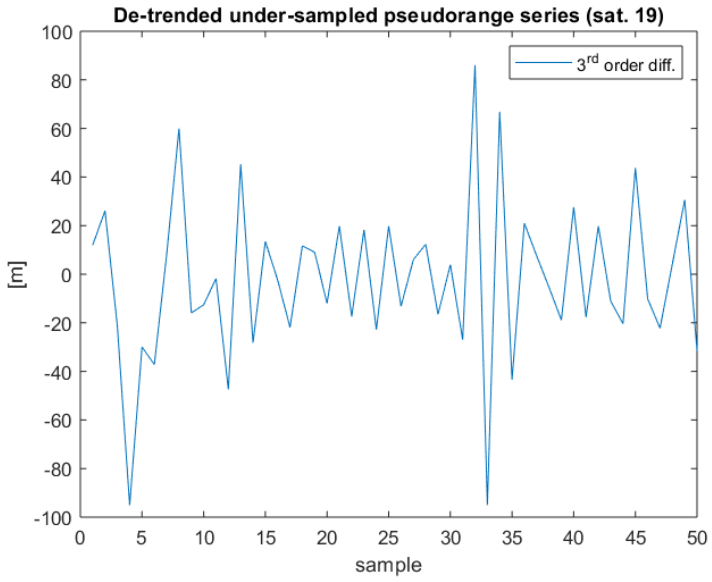


Figure 4.3: De-trended under-sampled pseudorange series for satellite 19 (3^{rd} order differentiation, N_s=38)

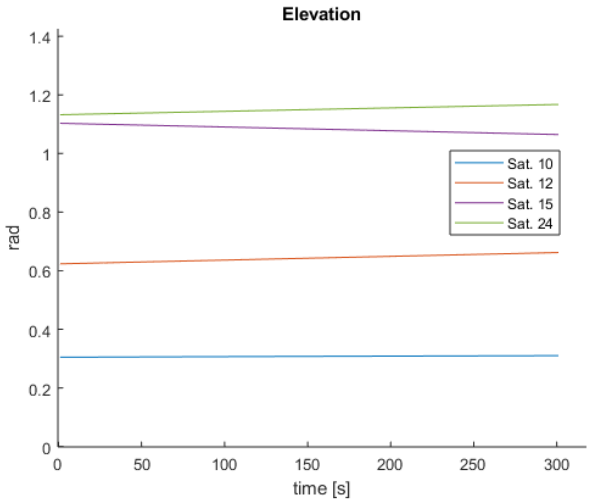
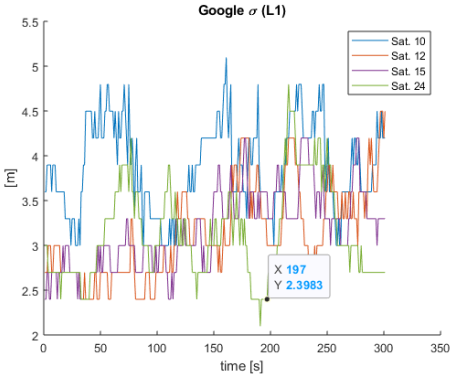
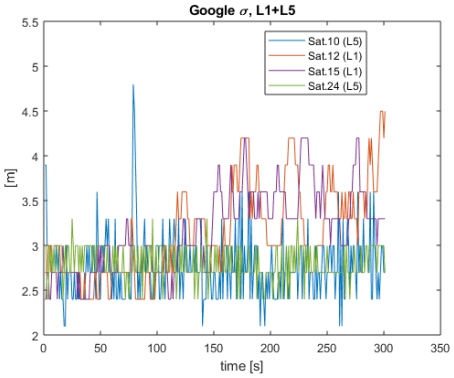


Figure 4.4: Satellite elevation, 300 epochs

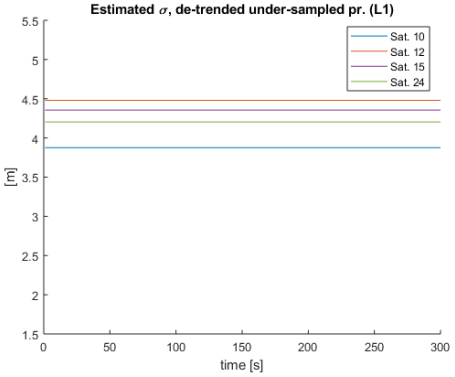
satellites 24 and 15 as the best, then satellite 12 would be classified and only last satellite 10. The proposed methods provide different results, as it is shown in Figure 4.5.



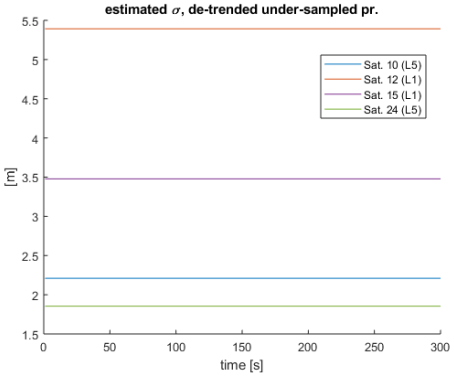
(a) *Google σ (L1)*



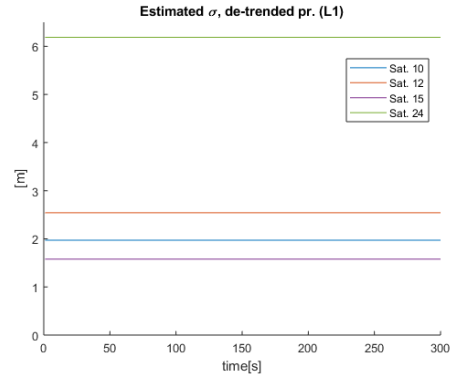
(b) *Google σ (L1+L5)*,



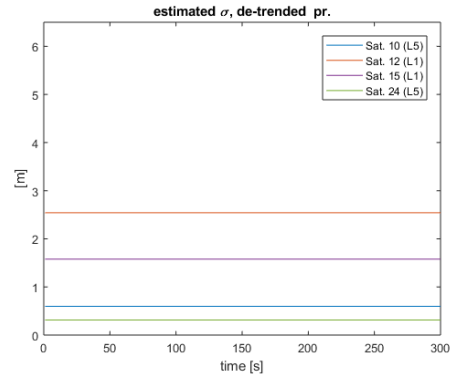
(c) *σ evaluated with de-trended under-sampled pr. measurements (L1)*,



(d) *σ evaluated with de-trended under-sampled pr. measurements (L1+L5)*,



(e) *σ evaluated with de-trended pr. measurements (L1)*,



(f) *σ evaluated with de-trended pr. measurements (L1+L5)*

Figure 4.5: σ comparison between different methods

The Google approach is the only dynamic method that provides a new weighting matrix at each epoch. The values obtained are always comparable with each other even if both L1 and L5 are used, in this case the σ values obtained on L5 seem to be more stable. A different point should be made for run-time methods, where the standard deviation of L5 pseudorange measurements is lower than the same value calculated in L1; this fact could distort the weighting matrix because satellites with good elevation, and in general with a good quality and whose signal is transmitted only on L1, could be considered worse than mediocre satellites transmitting on L5. This is evident in Figure 4.5e and Figure 4.5f where Satellite 24 has very different σ estimation. This behaviour is also common in Figure 4.5c and Figure 4.5d with a lower difference. The Google approach to weighting matrix construction overtakes this problem, in fact all the measurements are always comparable. This fact is important when PVT is evaluated using both L1 and L5 frequencies. The values obtained through the simple de-trended pseudorange series are generally lower with respect the other run-time methods, this is because consecutive samples are likely to be correlated and part of the error is removed. The standard deviation estimate changes in L1-only satellites in Figure 4.5c and Figure 4.5d because the under-sampling interval is different if the signals change.

By looking at Figure 4.5a and Figure 4.5b it is possible to say that the estimation is partially based on the elevation, as shown by Satellite 15 that increases its σ value as the elevation decreases. The error is also influenced by the frequency, as demonstrated by Satellite 10 whose error values decrease in L5 frequency, but still remain close to the others.

4.2 Frequency usage

The first *GPS IIF* satellite was launched in May 2010, this space vehicle is the first satellite with a full L5 transmitter. As June 2019, twelve of these satellites are orbiting around the Earth and L5 signals can often reach GNSS receivers. L5 signal was introduced to meet demanding requirements for safety-of-life transportation and other high-performance applications [23]. The main innovations introduced are: a higher power with respect to the legacy L1 signal, a greater bandwidth and an advanced design [23], that's why in Section 4.1 we mentioned that variance on de-trended pseudorange series is lower for L5 frequency. With the introduction of the new signals, which will become four with future satellites, GPS will be likely to improve accuracy through ionosphere correction and trilaning, a technique that by combining three GPS frequencies can enable sub-meter accuracy without augmentation [23].

As mentioned in Section 3.3, the tests in this work were performed with Xiaomi MI8, which is the first smartphone able to receive both L1 and L5 signals. Because of

these capabilities, three ways of using frequencies have been developed. In the first approach only L1 frequency has been used and the smartphone acted as a normal single-frequency receiver. The second approach is to use the measurements collected on the L5 frequency when they are available. The evaluated pseudorange is treated exactly like those on the L1 frequency. The last approach involves the use of the iono-free pseudorange, evaluated combining the measurements received on both L1 and L5, as explained in Section 2.4.

4.3 Smoothing strategies

Despite the problems that have arisen with carrier and Doppler measurements, as already mentioned in Section 3.5, some methods have been developed to overcome the anomalies and carry out the pseudorange smoothing.

All the strategies are based on the Hatch filter, a simple algorithm that allows to smooth the noisy code pseudorange measurements with the precise, but ambiguous, carrier phase measurements [17]. In the Hatch filter algorithm, the smoothed code for a given satellite s at epoch n, is

$$\hat{R}(s,n) = \frac{1}{n}R(s;k) + \frac{n-1}{n}[\hat{R}(s;k-1) + (\Phi(s,k) - \Phi(s;k-1))], \tag{4.2}$$

where $n = k$ when $k < N$ and $n = N$ when $k \leq N$; in this work, the smoothing window N is set equal to 50. The Hatch filter has to be initialised, in particular $\hat{R}(s,1) = R(s,1)$. Every time that a cycle slip occurs, the algorithm must be initialised [17].

As mentioned in Section 3.5, the *AccumulatedDeltaRangeMeters* measurements present anomalies. In many cases it is not possible to apply the Hatch Filter without making corrections or filtering measurements if the L5 frequency is also used. Several strategies have been implemented in order to use carrier phase measurements to smooth pseudoranges. The first approach consists in finding the points where anomalies are present and not smoothing the respective pseudoranges. Unfortunately the size of jumps is very variable and the assumed values are always comparable with correct values, so a threshold can not be set. Moreover it is not possible to rely on the status (*AccumulatedDeltaRangeState*) that does not filter out bad measurements. The second approach is to identify the satellites with anomalies in measurements and not to apply the smoothing on respective pseudoranges. This "ad-hoc" strategy can be applied by checking the norm of the de-trended *AccumulatedDeltaRangeMeters* measurements series. The mean can not be used because it is close to zero even in case of jumps. This approach has several limitations; the first one is the maximum value of norm allowed to consider a satellite as "good", the second is that the norm can be evaluated only in the aftermath, and last is the

smoothing is not applied on several satellites and the improvements coming from a smoothing strategy could not be achieved. A robust strategy is then introduced. The main idea is to consider the first-order differentiation of the pseudorange series as guideline of the expected carrier phase measurements. Figure 3.5 shows that the trend of regular measurements should be the same. It is then possible to set a threshold based on this value and consider a value as good only if it is within a given range. The smoothing can be then applied everywhere jumps are not detected. Although this strategy allows to smooth in a fairly effective way also the pseudoranges whose carrier phase measurements present problems, it is however not recommended to apply a smoothing strategy based on the carrier phase in this scenario. This method can still be applied to measurements obtained on L1; Figure 4.6 shows the high decrease in the variance of smoothed psudorange measurements.

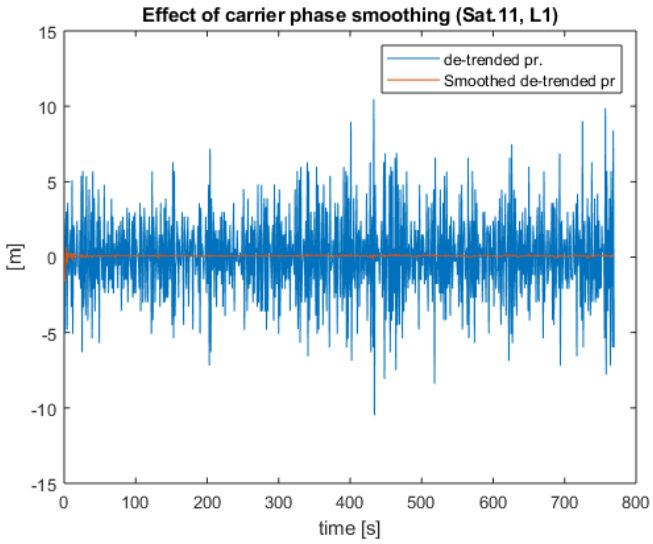


Figure 4.6: Comparison between smoothed and no-smoothed de-trended pseudorange series (carrier phase)

Despite Hatch filter is usually used for carrier phase smoothing, it is also possible to apply it using Doppler measurements, as reported in [30]. In fact, as reported in Figure 4.7, the *PseudorangeRateMetersperSeconds* follows the same trend of the first order differentiation of the *AccumulatedDeltaRangeMeters*. Moreover the GNSS Doppler, as an instantaneous measurement, is robust and immune to cycle slips and proven useful in GNSS-challenged environments [30].

As mentioned in Section 3.5, unfortunately *PseudorangeRateMetersperSeconds* often present anomalies given by "jumps". Since all the surveys have been performed in a static scenario, where no jumps are present, the first order differentiation of

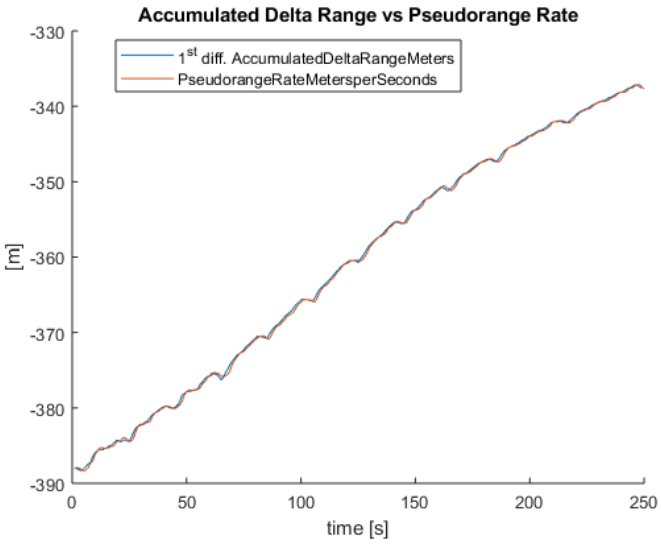


Figure 4.7: Comparison between 1^{st} order diff. of AccumulatedDeltaRangeMeters and PseudorangeRateMetersperSeconds (250 epochs)

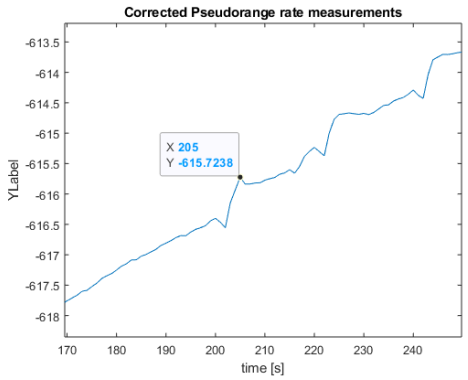
the *PseudorangeRateMetersperSeconds* has quasi-zero mean and very low variance. So a method to remove jumps has been implemented, in particular the corrected *PseudorangeRateMetersperSeconds* measurement in epoch n ($PrrMps_c(n)$) is

$$PrrMps_c(n) = PrrMps(n-1) + (PrrMps(n-1) - PrrMps(n-2)), \qquad (4.3)$$

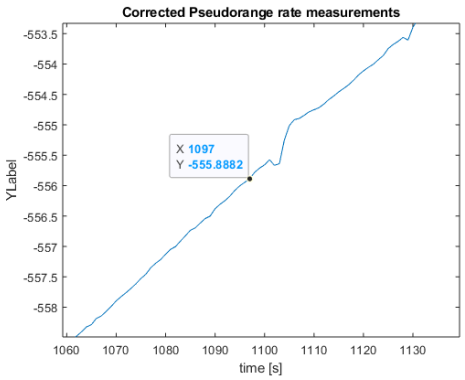
where $PrrMps(n)$ is the measured value at the n^{th} epoch.

Considering Figure 3.6, it is clear that the anomalies were detected at epochs 205 and epoch 1097. Figure 4.8 shows how the anomalies have been corrected using the described method. The anomalies that were found all had a duration of 1 epoch and a very high value. So it is possible to detect them using a threshold, in particular a possible value is 100000.

Since "jumps" in Doppler measurements last only one epoch, another possible approach to overcome the problem could be to use the previous smoothed pseudorange as smoothed pseudorange in the epoch that presents the jump. This is a very simple approach that could simplify a lot the operations especially in real time application. As reported in the GSA white paper [8], it is highly recommended to use Doppler measurements to smooth the pseudoranges as they are more reliable and there is no need to pay attention to cycle slips. Moreover, the results are very similar to those obtained using the carrier phase in epochs without anomalies, as shown in the Figure 4.9.



(a) *Corrected measurement at epoch 205*



(b) *Corrected measurement at epoch 1097*

Figure 4.8: Corrected measurements of Figure 3.7

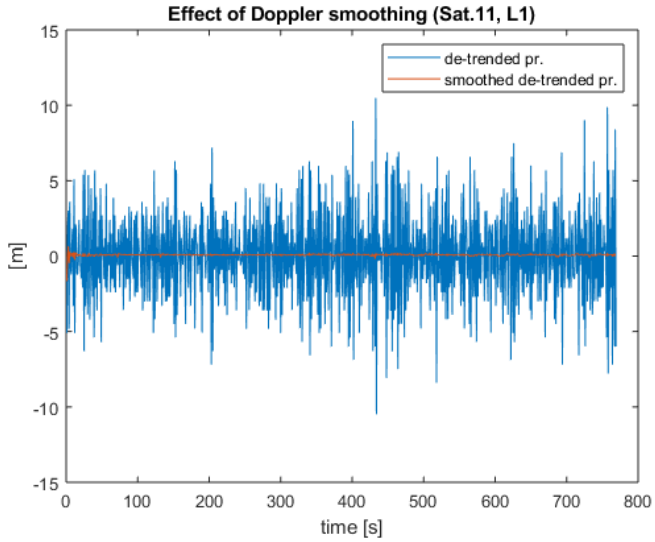


Figure 4.9: Comparison between smoothed and no-smoothed de-trended pseudorange series (Doppler)

4.4 The collected datasets

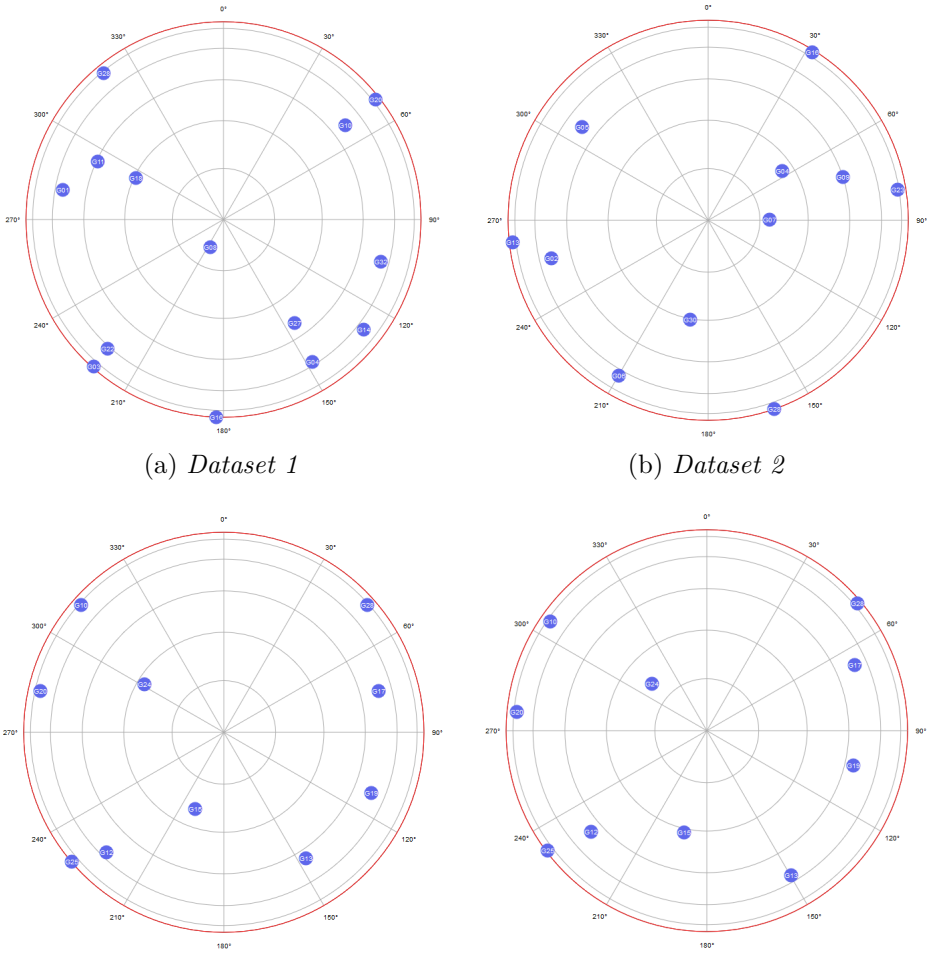
During the thesis work, the APP used to collect GNSS raw measurements is GNSS Logger, a free and open source APP provided by Google [13]. A lot of surveys have been performed, but unfortunately some of them were unusable or unreadable by the Google MATLAB code, due to invalidity of some measurements. Among the

good quality log files, four were chosen to perform analysis; all of them were collected in geo-referenced locations. The first one (Dataset 1) was collected on the 19^{th} October 2018 on the roof top of DET Departement of Politecnico di Torino; the geo-referenced position is [45.063304798, 7.660465262, 306.7413] and the survey duration was 771 second. The other measurements were collected in *Piazzale Duca d'Aosta*, where an IGT point is present; the coordinates of the point are [45.0620986111111, 7.66333444444445, 295.070]. Two were collected on 24 January 2019, one of these was collected in a morning with snow on the ground (Dataset 2), this condition is interesting because of the possible presence of multipath, the other one (Dataset 3) was collected in the afternoon and by then snow had completely melted. The last log file dates back to 28 January 2019 and it was collected with a different smartphone than the other three (Dataset 4). Table 4.1 summarises all the information relating to the datasets and it also provides the number of L5 signals detected for each dataset.

Dataset	**Location**		**Starting Time** day, hour	**Duration** (s)	**Satellites** total, (L5)	**Notes**
1	N	45.063304798	2018-10-19, 11.15 AM	771	11,(5)	/
	E	7.660465262				
	H	306.7413				
2	N	45.0620986111111	2019-01-24, 10.43 AM	1938	12,(3)	Snow on the ground
	E	7.66333444444445				
	H	295.070				
3	N	45.0620986111111	2019-01-24, 3.34 PM	1999	12,(2)	/
	E	7.66333444444445				
	H	295.070				
4	N	45.0620986111111	2019-01-28, 03.43 PM	1203	9,(2)	/
	E	7.66333444444445				
	H	295.070				

Table 4.1: Datasets information

The sky plot of each dataset is provided in Figure 4.10. There are misalignments with respect to the number of satellites given in the Table 4.1 because only the detected signals are considered in the table. The sky plots are provided in open sky condition, it is possible to see that several satellites have very low elevation, these can hardly be detected by the receiver.



(a) *Dataset 1* (b) *Dataset 2*

(c) *Dataset 3* (d) *Dataset 4*

Figure 4.10: Sky plot of the datasets

4.5 Ephemeris acquisition and decoding the GPS navigation message

Since not every smartphone is able to decode the GPS navigation message, the easier approach to collect the satellite ephemeris is to connect to the Crustal Dynamics Data Information System (CDDIS) [3] database that contains all the data collected by International GNSS Service(IGS) stations around the world. In order to download the daily ephemeris, it is only needed to obtain a rough estimation of the UTC time (the day is enough to access the database), and it is possible to obtain it by using *FullBiasNanos* and *TimeNanos*, described in 3.4; in particular an estimation of the

reception time, in seconds, for all the measurements could be performed as

$$RxTime = (TimeNanos - FullbiasNanos) * 1e9, \tag{4.4}$$

it is then necessary to convert it in UTC time, Google MATLAB code provides a method for this purpose [13]. The *.zip* containing the daily ephemeris can be the downloaded at the following address:

ftp://cddis.nasa.gov/gnss/data/hourly/YYYY/ddd/hourddd0.YYn.Z,

where *YYYY* is the full year number, *ddd* is year day number and YY is the short year number (e.g. 19 for year 2019) [3]. The obtained file is in Receiver Independent Exchange Format (RINEX) format, a very common format used in GNSS world for data exchanging, all the information about this are available at [15]. Once the daily ephemeris file has been downloaded it is possible to find the closest ephemeris for each satellite. This can be performed comparing the field *Time of ephemeris* with the *RxTime* field. All the operations that have to be performed to evaluate the satellite position, essential for PVT calculation, are available in [18].

An important feature of Xiaomi MI8 is that it can decode the navigation message, so a MATLAB function has been developed to extract the ephemeris directly from the received signal. All the information about the Navigation message are operated by Android through the class *GnssNavivagtionMessage*. Table 4.2 gives all the parameters provided by this class.

PARAMETER	**DESCRIPTION**
Data	Gets all the data of the reported GPS message
Message Id	Gets the value of the frame id
Status	Gets the status of the navigation message
Submessage Id	Gets the value of the subframe number
Svid	Satellite ID
Type	Gets the type of the navigation message

Table 4.2: Fields of GnssNavivagtionMessage class [8]

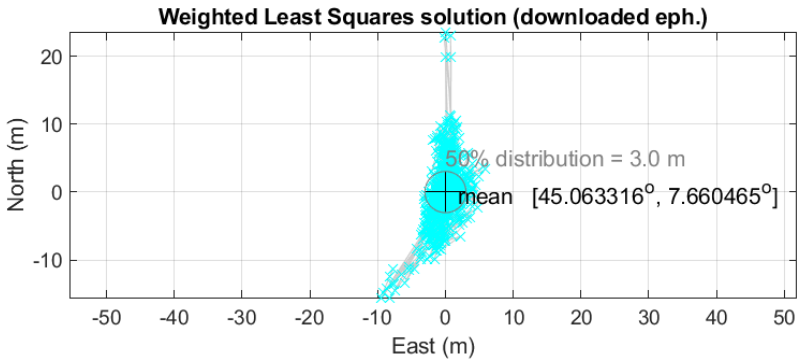
For GPS, the *Message Id* field gives a value in the range of [1,25] only for subframe 4 and 5, since subframes 1, 2 and 3 do not contain the "frame id", this value is set to -1. The field that has to be decoded is *Data*, it contains the data, so also the ephemeris parameters. For GPS L1 C/A, this field is provided by Android as a list of values in *int8* format, so a list of 40 integer numbers in the range of [-128,127], this list is a single subframe. This format is unusual and requires some

passages to be correctly decoded. First of all it is important to filter the log file and select only the data coming from GPS, this can be performed checking the satellite ID, GPS satellites have ID in the range of [1,32]. The subframes are not always consecutive, but to correctly decode the navigation message, a full 1500-bits message is needed. Therefore the MATLAB function only keeps subframes where they are received in correct sequence from 1 to 5. As mentioned in Section 2.3, each subframe is composed by 10 30-bit words, so in order to obtain a word, four 8-bit numbers have to be decoded. But by simply converting the list of numbers (the subframe) in binary format using two's complement, a list of 320 bits is obtained; since the subframe length is 300 bit, some bits have to be discarded. In particular, for each word (4 numbers), bit 31 and bit 32 have to be skipped, this corresponds to drop the first two bits of the word. Once 5 consecutive 300-bit subframes are obtained for a specific satellite, it is possible to decode the navigation message using a function extracted by SoftGNSS v.3, a free software provided by Darius Plausinaitis and Kristin Larson, released under GNU General Public License. For now only the ephemeris are preserved and they are saved into a Matlab variable, keeping exactly the same format as the downloaded ephemeris, in order to preserve the code structure. The used approach is clearly post-processing based;real time applications could be tried, but an initial time to download the full ephemeris for each satellite is required before evaluating the PVT. It is important to mention that on 6^{th} April 2019 the GPS week number rollover happened [11], it is then necessary to correct the ephemeris week number value (adding 1024). This fact could be seen also as a proof that the data provided by Android are the real navigation messages read from the signal, because downloaded ephemeris do not need the correction.

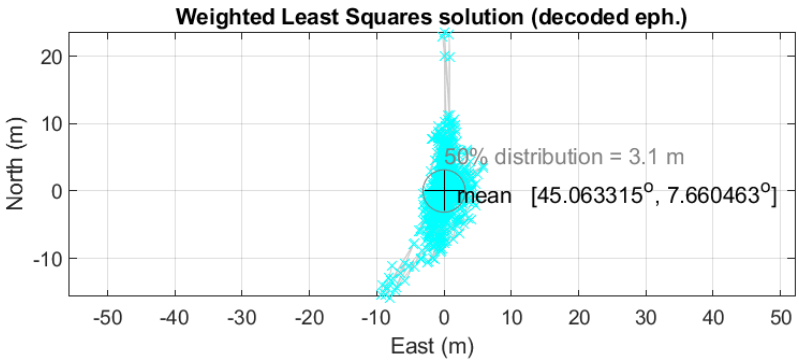
Figure 4.11 shows the position solution of the dataset collected on the DET rooftop using the downloaded ephemeris and the decoded ones. The results are very similar, but there are small differences, in the order of centimetres, given by the fact that IGS stations always provide corrected ephemeris, while the navigation message may contain some small errors. The results are anyway comparable and Xiaomi Mi8 may act as a GNSS receiver that does not need the network to download data for PVT computation.

4.6 PVT results analysis

This section aims to compare the results obtained using the strategies mentioned in the previous section. The first part reports the PVT solutions, the second part shows the changes introduced with the smoothing. Before starting it is important to remember the differences between accuracy and precision. By comparing the results from many epochs of data, we might see that the coordinated values agree amongst themselves quite closely, they have high precision. But, due to some remaining bias,



(a) *PVT with downloaded ephemeris*



(b) *PVT with decoded ephemeris*

Figure 4.11: PVT comparison using downloaded and decoded ephemeris, no pseudorange smoothing applied

they are offset from the true value, their accuracy is low [25].

4.6.1 PVT comparison

Let's now analyse the results obtained using the different frequencies and different ways to find the weighting matrix. In this Section all the results have been obtained with no-smoothed pseudoranges. The Tables report some fields that have to be introduced, *Accuracy* is given by the mean distance from the reference point to the mean of the positions at each epoch in every direction, North (N), East (E), and Height (H); *Precision*, is the standard deviation of the above mentioned distances in N,E,H direction; */Mean-true/* is the two-dimensional distance (N,E) between the mean of the found positions at each epoch and the true position; the *50% distribution* is the median of the distances from the mean point. */Mean-true/* is then a parameter connected with the accuracy, while *50% distribution* gives information about the precision, this value is used in this work because it is the default value chosen by

Google.

The description and the names used for the datasets are available in Section 4.4. The methods used to calculate the weighting matrix are described in Section 4.1; in order to simplify the analysis, in this section we will call Method 1 the strategy that builds the weighting matrix using the Android estimated error (σ_{Google}), Method 2 is the one that builds the weighting matrix with the variance of the de-trended under-sampled pseudorange value series and Method 3 the one that utilizes the variance of the de-trended pseudorange series (not sub-sampled).

Let's start with the data collected on L1. Table 4.3 shows the results obtained with Method 1. It is possible to notice that the values change a lot in the different datasets, and it is difficult to define a general trend. */Mean-true/* assumes values from 1.8 meters to 7.8 meters, while *50% distribution* have more regular values. The elevation is always the worse estimated parameter, North and East directions are in general acceptable and it is not possible to say that one direction has more critical aspects than the other. Only North direction in accuracy measurements for Dataset 1 presents a value higher than expected. It is curious that the best result in terms of accuracy is in the dataset theoretically more critical due to the presence of snow on the ground (Dataset 2). Table 4.4 gives the results obtained by building the

Parameter		**Dataset 1** (m)	**Dataset 2** (m)	**Dataset 3** (m)	**Dataset 4** (m)
Accuracy	N	7.5965	0.3481	-1.7401	-3.4863
	E	1.5944	-3.6952	-0.6274	-4.1019
	H	-33.029	-25.3308	-12.6673	-15.6176
Precision	N	8.7202	6.7557	5.7302	6.1532
	E	3.6861	6.5644	4.1329	6.5138
	H	17.8148	18.4926	10.1584	9.552
|Mean-true|		7.8	3.7	1.8	5.4
50% distribution		7.2	7.3	5.2	6.4

Table 4.3: L1-only PVT: **W** built with σ_{Google}

weighting matrix with Method 2. Again, the values concerning the elevation are not good and it is difficult to define a general trend. The values of */Mean-true/* are in the range of 2 meters, with the lowest value reached of 0.6 meters in dataset 4, with the exception of Dataset 2, in which the assumed value is 5.1 meters. *%50 distribution* assumes larger valuer with respect to the previous method, with particular changes in Dataset 2 and Dataset 3. There is a peak in the precision measurements for East direction in Dataset 2. Table 4.5 closes the measurements collected using only L1, it shows the results obtained with Method 3. In general the results are comparable

Parameter		**Dataset 1** (m)	**Dataset 2** (m)	**Dataset 3** (m)	**Dataset 4** (m)
Accuracy	N	-1.4916	1.7584	-0.8503	-0.2991
	E	-1.3796	-4.7604	0.6639	0.5656
	H	-13.1394	-19.7656	-11.0594	-11.3446
Precision	N	9.8973	8.6250	14.3219	6.9564
	E	5.0326	19.0278	7.2111	7.4878
	H	15.1357	39.7325	21.1068	9.8510
|Mean-true|		2	5.1	1.1	0.6
50% distribution		9.1	14	10.8	7

Table 4.4: L1-only PVT: **W** built with the variance of the de-trended under-sampled pseudoranges

with the previous approach. */Mean-true/* assumes similar values in Dataset 2 and Dataset 3, there are significant improvements in Dataset 1 and Dataset 4. The *50% distribution* parameter increases for Datasets 2 and 3, while it decreases in Datasets 1 and 4. Also here, there is a peak in the precision measurements for East direction, for which the obtained value is 23.997 meters. The overall results show a better

Parameter		**Dataset 1** (m)	**Dataset 2** (m)	**Dataset 3** (m)	**Dataset 4** (m)
Accuracy	N	-0.3716	1.8566	0.1220	-2.4506
	E	-0.8954	-4.5799	1.0090	-1.2728
	H	-13.8081	-19.3408	-11.3876	-9.3062
Precision	N	9.8973	9.7159	12.2306	7.6616
	E	5.0326	23.997	6.7676	7.0346
	H	15.1357	46.6465	18.7539	12.0715
|Mean-true|		1	4.9	1	2.8
50% distribution		8.2	16.6	7.9	7.4

Table 4.5: L1-only PVT: **W** built with the variance of the de-trended pseudoranges

performance of Method 1 from the precision point of view, but accuracy is better in the other approaches. There is a particular deterioration in measurements of Dataset 2 using Method 2 and Method 3 with respect to Method 1. This may be a clue to the fact that Android seems to better estimate error values under harsh conditions as with snow on the ground. Despite this fact, it is however evident that Method 1 on L1 presents some limits in terms of accuracy, as demonstrated by the results

obtained in the other datasets, in particular Dataset 1 and Dataset 4 where the other methods have higher performance.

The second way to use the frequencies is to use L5 measurements, when available, to evaluate the pseudoranges and L1 measurements otherwise, as described in Section 4.2. From the accuracy point of view, the results obtained through Method 1 are always better except for Dataset 3, in which more than 3 meters have been lost. The higher improvement is in Dataset 1 where more than six meters have been gained. *50% distribution* has better values, and also here, the greatest improvements have been reached in Dataset 1. Also using both, L1 and L5 frequencies, elevation measurement are not good. Table 4.6 shows these results. The results reported in

Parameter		**Dataset 1** (m)	**Dataset 2** (m)	**Dataset 3** (m)	**Dataset 4** (m)
Accuracy	N	1.2477	5.2865	-1.3754	-1.2479
	E	-0.0543	-2.9963	-0.6320	-4.3684
	H	-26.2861	-23.7896	-11.7797	-12.9049
Precision	N	4.7095	3.8645	5.8723	5.4539
	E	1.8735	4.3447	3.9769	6.4866
	H	15.7574	13.4250	7.6149	6.4078
|Mean-true|		1.2	6.1	1.5	4.5
50% distribution		3	4.1	5.1	6

Table 4.6: L1+L5 PVT: **W** built with σ_{Google}

Table 4.7 show an improvement in terms of precision by using Method 2 with L1 and L5 with respect L1-only, in particular, the most noticeable improvement is the *50% distribution* in Dataset 2, where 10.9 meters have been gained. On the other hand, accuracy values are of lower quality in Dataset 2 and Dataset 4, while it does not change a lot in the other datasets. Table 4.8 gives results obtained using L1 and L5 measurements and Method 3. Again the precision improves in all the datasets; accuracy is generally a bit worse with respect to the results obtained using only L1 measurements, except for Dataset 4 in which there is an improvement of 1.6 meters in the */mean-true/* field. The problem in */mean-true/* for Dataset 2 is given by the accuracy in North direction, in which the value is 7 meters.

In general the solutions are good if we consider that they come from a smartphone using only GNSS receiver, but the calculated value of the elevation always deviates a lot from the real value. The introduction of the second frequency tends to bring improvements in the use of all methods for calculating the weighting matrix in terms of precision, in fact the *50% distribution* field is always better. From the point of view of the accuracy, the results are contrasting, in fact, it is not possible to say

Parameter		**Dataset 1** (m)	**Dataset 2** (m)	**Dataset 3** (m)	**Dataset 4** (m)
Accuracy	N	1.7321	6.9956	0.979	1.5247
	E	-1.1276	-3.1358	0.7103	0.9024
	H	-8.6186	-17.1484	-10.6991	8.986
Precision	N	3.9498	3.8035	10.8284	5.1906
	E	1.8577	3.708	6.6768	7.0733
	H	8.1139	4.9638	8.986	4.1599
|Mean-true|		2.1	7.7	0.7	1.8
50% distribution		2.3	3.1	8.5	6.1

Table 4.7: L1+L5 PVT: **W** built with the variance of the de-trended under-sampled pseudoranges

Parameter		**Dataset 1** (m)	**Dataset 2** (m)	**Dataset 3** (m)	**Dataset 4** (m)
Accuracy	N	1.4864	7.006	0.3824	0.5237
	E	-1.7671	-3.1376	1.0768	-1.0379
	H	-6.8485	-17.0547	-10.558	-8.1066
Precision	N	4.0802	3.803	9.9952	5.1560
	E	1.7929	3.6992	6.7787	6.7623
	H	7.5844	4.9023	7.9293	3.7821
|Mean-true|		2.3	7.7	1.1	1.2
50% distribution		2.3	3.1	7.2	5.9

Table 4.8: L1+L5 PVT: **W** built with the variance of the de-trended pseudoranges

that there are large improvements or deteriorations in most cases. For all 3 methods there is a deterioration in */Mean-true/* of about 2.5 meters in Dataset 2, this fact is given by uncommon values of the mean of the distances in North direction. Dataset 2 anyway is difficult to analyse because the best results on it are obtained using Method 1 on L1-only; this combination is, in general, the worse among the tried ones. In Dataset 1 the best value of */Mean-true/* has been obtained through Method 3 by using only L1, the best value of *50% distribution* is given by Method 2 and Method 3 using both, L1 and L5 measurements. Considering Dataset 2 the best level of accuracy was achieved by Method 1 using only L1, while the best precision has been obtained again with Method 2 and Method 3 (L1+L5). The higher quality of accuracy and precision for Dataset 3 have been obtained respectively by Method 2 and Method 1, Both by using L1 and L5. If Dataset 4 is considered, the best

/Mean-true/ value has been obtained through Method 2 using only L1 frequency, while the best *50% distribution* value has been output using Method 3 and both L1 and L5. Then there is not an overall better strategy, in general accuracy is better using Method 2 and Method 3 with only L1, but precision increases a lot using the dual-frequency approach. It is important to say that all the methods give comparable results, especially in L1+L5 scenario. Therefore also the Google approach is valid, especially if we consider that the value of σ necessary for the calculation of the weighting matrix is calculated in real time.

The last implemented method to use the frequencies is to obtain the iono-free pseudoranges where possible. The followed procedure is available in Section 2.4. As June 2019, there are only 12 GPS satellites transmitting on both L1 and L5 frequencies [4], so the results obtained using this mode are only indicative. The satellites that transmit on both the frequency in the collected datasets are in the range of 2-4.

Table 4.9 shows the results obtained using Method 1, the σ_{Google} chosen to build the weighting matrix is the one available for L5 signal. Dataset 1 presents very low quality values of both accuracy and precision, present in North direction, while E direction has acceptable values, especially in terms of accuracy. Dataset 2 is also affected by anomalies, especially in North direction. Dataset 3 and Dataset 4 results are instead comparable with the results obtained without iono-free pseudorange. Table 4.10 reports the results obtained using Method 2. It is clear that with this

Parameter		**Dataset 1** (m)	**Dataset 2** (m)	**Dataset 3** (m)	**Dataset 4** (m)
Accuracy	N	26.6517	-8.4529	-1.2165	-4.3807
	E	0.957	-5.7651	-0.789	-3.9973
	H	-20.9873	-20.4429	-12.4952	-18.5013
Precision	N	30.8078	21.6242	9.3513	9.3563
	E	11.8339	16.2525	5.2182	6.6973
	H	31.8474	35.992	22.9191	19.1525
|Mean-true|		26.7	10.2	1.4	5.9
50% distribution		20.7	14.5	7.8	8.7

Table 4.9: iono-free PVT: **W** built with σ_{Google}

approach the results relating to Dataset are much better than in Method 1, moreover this is the best obtained result in terms of accuracy, in fact the */Mean-true/* parameter is equal to 0.5 meters. Unfortunately the quality of precision is not up to that of variance. Dataset 3 and Dataset 4 results are also good in terms of accuracy if compared with the previous ones. Also in this case the problems with the d2 are

evident. Table 4.11 closes the analysis of the PVT strategies, it shows the result

Parameter		**Dataset 1** (m)	**Dataset 2** (m)	**Dataset 3** (m)	**Dataset 4** (m)
Accuracy	N	0.1387	-6.1991	-0.9556	-2.9493
	E	0.5055	-13.5896	0.684	0.1284
	H	-22.3055	-32.0535	-11.6698	-13.363
Precision	N	15.2222	17.429	17.608	10.9104
	E	7.5535	41.8352	8.4424	7.9231
	H	19.537	97.5431	19.4299	13.575
|Mean-true|		0.5	14.9	1.2	3
50% distribution		12.5	22.8	12.8	9

Table 4.10: iono-free PVT: **W** built with the variance of the de-trended under-sampled pseudoranges

obtained with Method 3 and the iono-free pseudoranges. In general they are similar to the ones achieved through Method 2. The main differences are the lower quality of the accuracy in Dataset 1 and Dataset 4 and the higher precision in Dataset 3. In general, the results obtained using the iono-free pseudorange are quite good from

Parameter		**Dataset 1** (m)	**Dataset 2** (m)	**Dataset 3** (m)	**Dataset 4** (m)
Accuracy	N	3.1516	-6.1526	-0.4095	-4.7485
	E	1.7255	-13.2843	1.0674	-1.2047
	H	-18.8057	-31.5586	-11.766	-12.6025
Precision	N	14.7587	17.699	12.865	9.7630
	E	8.1836	41.1843	7.0614	7.0600
	H	20.2063	97.9701	17.6361	13.2999
|Mean-true|		3.6	14.6	1.1	4.9
50% distribution		12.3	22.0	8.3	8.4

Table 4.11: iono-free PVT: **W** built with the variance of the de-trended pseudoranges

the point of view of accuracy, the main problems arise in Dataset 2. This is probably due to the fact that measurements on Dataset 2 are noisier, because of the snow on the ground. In general by applying the iono-free combination the noise and the error contributions increase, that's why this solution for code-phase pseudorange is often applied in high-level receivers. From the precision point of view, the results are not comparable with the ones achieved without the iono-free pseudorange. In

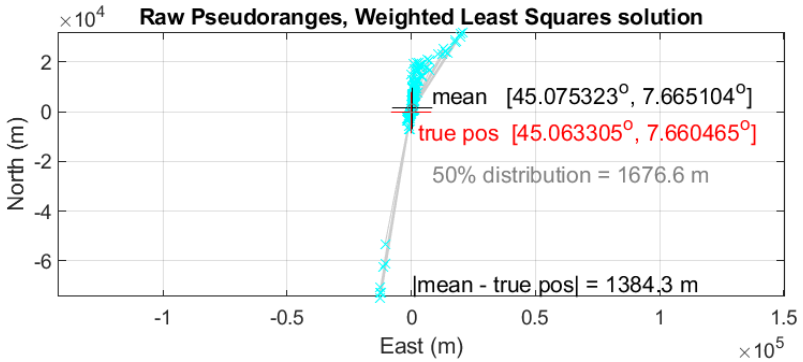
this scenario, Method 1 seems to be the least appropriate, this is probably given by the fact that σ_{Google} is specifically connected to the received signal and not to the quality of the calculated pseudorange. Although the performances are not exciting, this can still be a starting point for more in-depth analysis. On the other hand it is important to remember that during Google I/O 2018, Google GNSS managers talked about smartphones' dual frequency chipsets as innovative tools for calculating position by carrier phase, while the possibility of being able to perform the iono-free combination was never mentioned.

4.6.2 Smoothing strategies analysis

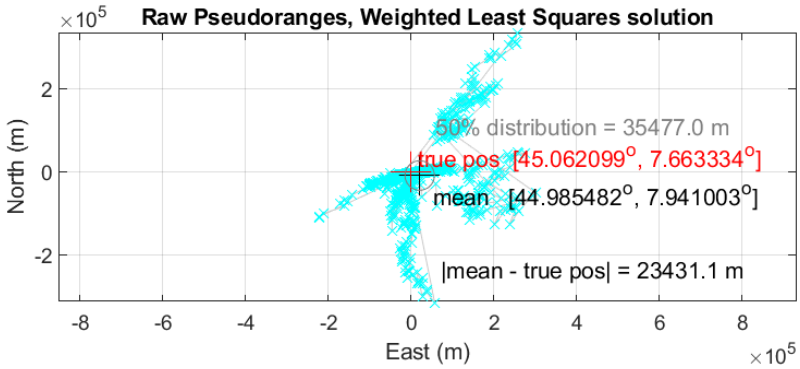
As mentioned in Section 4.3, a standard smoothing technique is not usable for a lot of datasets, because of the anomalies in Doppler and carrier phase measurements. Figure 4.12 shows the PVT results obtained if a standard Hatch filter algorithm is applied. It is clear that the anomalies cause serious deteriorations in the PVT. Let's then analyse the effect of the developed smoothing strategies described in Section 4.3 while using the PVT methods described in Section 4.6.1. The first one is the robust strategy that detects and corrects the jumps in the *PseudorangeRatemetersperSeconds* measurements. Table 4.12 shows the differences in terms of precision and accuracy. For simplicity only *50% distribution* and */Mean-true/* parameters have been taken into account. The table shows a general improvement of precision, in particular the largest improvements have occurred in the use of Method 2 and Method 3. On the other hand the accuracy has more diversified variations, in general smoothing brings improvements, but in some cases */Mean-true/* parameter is worse; the main worsening happens in Dataset 1 using Method 1 and Method 3, and Dataset 3 using Method 2. The improvements in precision for Dataset 2 using Method 2 and Method 3, and Dataset 3 using Method 3 are noteworthy.

Table 4.13 gives a comparison between the PVT solutions obtained through the smoothed and no-smoothed pseudoranges using L1 and L5 when available, without the iono-free combination. The *50% distribution* parameter is always improved. The precision is higher if compared with L1-only solutions, that always present worse values. There are notable improvements in several datasets, the greatest have been registered in Dataset 3 and Dataset 4, using all the methods. Again, the */Mean-true/* values are more fluctuating, but the changes are generally small, in the order of 0.5 meters. The highest differences are present in Dataset 1 using Method 2 and Method 3, where the accuracy is however improved, Dataset 3 for Method 2 and Dataset 4 using Method 1. Problems in Dataset 2 are still present; although there is a good precision, the accuracy is still much worse than the other datasets.

In general, it is possible to say that Doppler smoothing improves the obtained solutions and performs well especially using both L1 and L5 frequencies. There are some cases in which the quality of accuracy decreases, but in general they are



(a) *Standard Hatch filter using carrier phase, Dataset 1, L1+L5 frequency*



(b) *Standard Hatch filter using Doppler, Dataset 4, L1+L5 frequency*

Figure 4.12: Standard smoothing strategies on anomalies-affected measurements

very small variation, while the improvements in precision are notable. The obtained results say that, by using also L5 frequency, this strategy works well; it is also important to remark that Doppler is not affected by cycle slips, so it is more reliable, especially if we consider the problems of Android measurements of *AdrState*.

For the reasons given in Section 4.3 and in Section 3.5, it is recommended to apply carrier phase smoothing only if data are collected exclusively on the L1 frequency. In the performed tests, the robust strategy based on the first-order differentiation of the pseudorange series is used, so that any instability does not affect the smoothing output. Table 4.14 shows the comparison between the results obtained using smoothed and not smoothed pseudoranges. As in the previous approach the smoothing highly increases the quality of the precision. The only case in which the precision decreases is Dataset 1 using Method 3. On the other hand, accuracy is not always improved. In general variations are in the range of [0.5,1.5] metres, but the anomaly recorded in Dataset 1 using Method 3 is evident: more than 13 meters of accuracy have been lost.

Method	**Dataset**	**50% distribution** (m)		**|Mean-true|** (m)	
		Smoothing	No smoothing	Smoothing	No Smoothing
1	Dataset 1	5.9	7.2	9.7	7.8
	Dataset 2	4.1	7.3	2.9	3.7
	Dataset 3	2.7	5.2	1.4	1.8
	Dataset 4	5.3	6.4	3.1	5.4
2	Dataset 1	4.5	9.1	1.6	2
	Dataset 2	5.1	14	4.9	5.1
	Dataset 3	5.1	10.8	1.8	1.1
	Dataset 4	2.1	7	0.7	0.6
3	Dataset 1	4.5	8.2	2.2	1
	Dataset 2	6.2	16.6	4.7	4.9
	Dataset 3	6.3	7.9	1.2	1
	Dataset 4	3.2	7.4	1.3	2.8

Table 4.12: L1-only: Accuracy and precision for each method with and without Doppler smoothing

Method	**Dataset**	**50% distribution** (m)		**|Mean-true|** (m)	
		Smoothing	No smoothing	Smoothing	No Smoothing
1	Dataset 1	2,6	3	1,3	1,2
	Dataset 2	2,7	4,1	6,2	6,1
	Dataset 3	2,6	5,1	1,4	1,5
	Dataset 4	4,5	6	2,8	4,5
2	Dataset 1	1,3	2,3	1,4	2,1
	Dataset 2	1,8	3,1	7,7	7,7
	Dataset 3	3,4	8,5	1,4	0,7
	Dataset 4	1,7	6,1	1,5	1,8
3	Dataset 1	2,2	2,3	1,2	2,3
	Dataset 2	1,8	3,1	7,6	7,7
	Dataset 3	3,8	7,2	1,4	1,1
	Dataset 4	1,7	5,9	1,5	1,2

Table 4.13: L1+L5: Accuracy and precision for each method with and without Doppler smoothing

Method	**Dataset**	**50% distribution** (m)		**|Mean-true|** (m)	
		Smoothing	No smoothing	Smoothing	No Smoothing
1	Dataset 1	6.2	7.2	8.4	7.8
	Dataset 2	4.4	7.3	2.7	3.7
	Dataset 3	4.7	5.2	1.5	1.8
	Dataset 4	4.0	6.4	5.2	5.4
2	Dataset 1	4.3	9.1	1.9	2
	Dataset 2	5.0	14.0	3.9	5.1
	Dataset 3	8.4	10.8	1.5	1.1
	Dataset 4	2.9	7.0	1.6	0.6
3	Dataset 1	8.3	8.2	14.2	1
	Dataset 2	6.0	16.6	3.5	4.9
	Dataset 3	7.0	7.9	1.4	1
	Dataset 4	3.4	7.4	2.3	2.8

Table 4.14: L1-only: Accuracy and precision for each method with and without Carrier phase smoothing

By comparing the results obtained through the two different smoothing strategies using the L1 frequency only, it is possible to notice that they behave in a similar manner, but Doppler smoothing gives better results in both accuracy and precision, in more datasets. The only dataset that gain quality using carrier phase is Dataset 2, especially using Method 2 and Method 3.

Comparing all the results it is possible to state that basically the best solutions are provided by the use of the pseudoranges obtained on both frequencies and smoothed with the Doppler measurements. For future works it is therefore advisable to use this startegy also because it is simple from the computational point of view as no complex operations are necessary to clean up the measurements obtained from the GNSS chipset. The fact that the Doppler is not affected by cycle slips is an extra strength because it is more reliable and does not require any reset in the Hatch filter.

Chapter 5

Multi-constellation Implementation

5.1 Galileo and GPS

As mentioned in Section 2, the minimum number of satellites in line of sight to get the position is four, but to get a a reliable position, at least 8-10 satellites are needed. If the receiver is in a critical condition in terms of sky visibility, as in urban areas, this number is hardly reached by GPS alone. That's why more than one GNSS can be used to improve the positioning performance [8]. In this work it has been decided to use Galileo to try to achieve a better quality of PVT. Despite the fact that other systems can also be used in a multi-constellation receiver, Galileo benefits from complete interoperability with GPS [14]. In addition to sharing the L1/E1 at 1575.420 MHz and L5/E5a at 1176.450 MHz frequencies, both systems keep their time close to the world's reference time (UTC) as seen Figure 5.1 which reports the time difference between reference times. Since the two internal times are derived independently of one another, there is still a small bias between them, the GPS-Galileo Time Offset (GGTO). This parameter has to be taken into account and it is precisely calculated on a continuous basis by the Precision Time Facility (PTF) and then transmitted within the Galileo's navigation message [10]. The accuracy of the GGTO should be less then 5ns with 2-sigma confidence level over any 24 hours [19].

The partnership between the USA and European Union in space-based positioning area started in 2004 with an agreement establishing the cooperation between GPS and Galileo programs. The agreement aimed to increase satellite availability and improved resistance to signal interference for civil users around the world. The US and the EU are also cooperating to enable manufacturers to produce dual-system

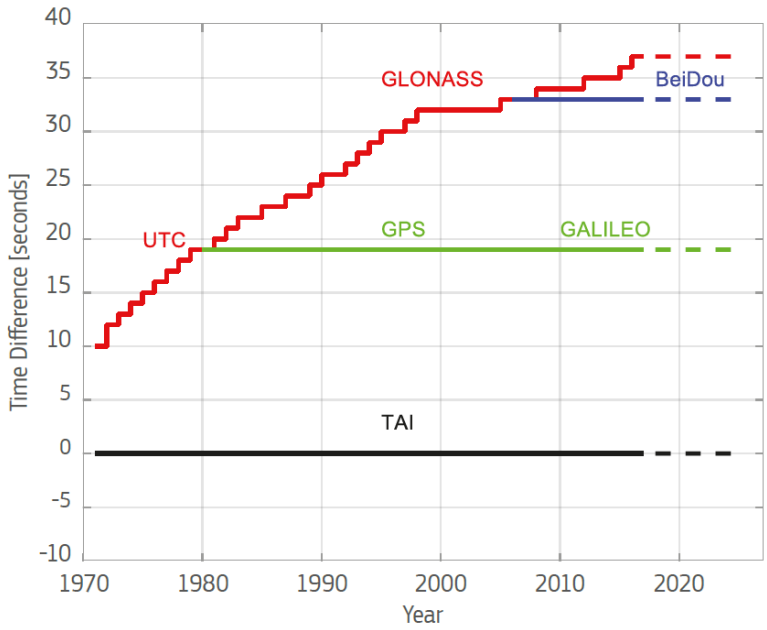


Figure 5.1: Time difference between reference times [8]

receivers, as private competition in the market can lead to the success of GNSS technology.

5.2 Galileo code pseudorange generation

In Section 3.4 explained how the GPS pseudorange could be calculated. With Galileo instead, things are a bit different: once the constellation has been verified to be correct, it is necessary to carefully check the tracking status given by *State* parameter. In particular, the pseudorange is calculated differently depending on whether the tracking status is flagged as *TOW Decoded* or *E1C 2nd Code lock*. The former says that the valid range of the reception time is within a week and it is possible to proceed in the same way as GPS, as described in Section 3.4; the latter says that the valid range is 0-100ms, so a new strategy to obtain the pseudorange is needed. Since practical tests show that after a short period of time Galileo measurements go from *TOW Decoded* to *E1C 2nd Code lock* status, in GSA white paper [8], authors strongly encourage developers to use only *E1C 2nd Code lock* flagged measurements. This state indicates that the GNSS chipset is tracking the second *E1C pilot component* whose length is 100ms. As already mentioned, the received satellite time at the measurement time given by *ReceivedSvTimeNanos* depends on the status. It is

then necessary to bring the measurement time in the same range. The measurement time is in GNSS time, that in general in the receivers is the GPS time, and it is

$$Time_{GNSS} = TimeNanos - (FullBiasNanos(1) + BiasNanos(1)); \tag{5.1}$$

it is then necessary to move it in the same range of the transmission time (*ReceivedSvTimeNanos*):

$$Time_{Galileo} = mod(Time_{GNSS}, NNanoSecond100Milli), \tag{5.2}$$

where *NNanoSecond100Milli* is the number of nanoseconds within 100ms and mode operator gives the remainder of the division [8]. At this point the pseudorange can normally be calculated, as in (3.2). Unfortunately during the performed test, the *State* parameter does not always work, in fact sometimes it happens that even the measurements collected on E5 are flagged as *E1C 2nd Code lock*. This fact is arbitrary and when it happens the introduction of Galileo worsens the performance of the PVT, even if E5 measurement are filtered out. When *State* is reliable and the measurements are correctly filtered, some improvements are obtained.

5.3 Collecting the ephemeris and the GGTO

As with GPS, Galileo receiver needs to know the positions of the satellites to be able to compute the PVT and they are calculated through the ephemeris transmitted in the navigation message. In the performed tests no Galileo messages have been registered, that's is probably due the fact that GNSS chipset starts tracking the second E1C pilot code as soon as it has decoded the TOW. It is therefore necessary to download the navigation message from the network. CDDIS database, explained in Section 4.5, provides this information. The procedure is similar to the one used for GPS, but there is a substantial difference: it is not distributed as unified version, it is necessary to download a Rinex file of a specific IGS station that also provides the Galileo data. In particular the selected station is the GRAC00FRA, located in Caussols (France) because the provided Rinex file of the navigation message also contains the GGTO parameters in the field *GPGA*. Galileo message rinex message files usually contain *GAUT* parameters through which it is possible to convert Galileo time to UTC time and then to GPS time, but *GPGA* simplifies the process. The link that allows the navigation message to be downloaded is

ftp://cddis.gsfc.nasa.gov/gnss/data/daily/YYYY/ddd/YYl/GRAC00FRA_R_YYYYddd0000_01D_EN.rnx.gz,

where *YYYY* is the full year number, *ddd* is year day number (e.g. 1^{st} February is 032), YY is the short year number (e.g. 19 for year 2019) and *l* is an identification for Galileo Navigation message folder [3]. The following steps are similar to those

described for GPS, moreover things are simplified by the fact that inside the rinex file the parameters relating to the Time of the ephemeris (Toe) and the week number are reported in GPS time [15].

5.4 Multi-constellation PVT strategy

As in GPS, the PVT algorithm is based on Least Squares. Combining signals from different constellations is possible, but different systems use different times and this leads to the introduction of a new unknown, the bias between the time of the receiver and the time of the system introduced. In case of Galileo and GPS, two biases have to be found, then (5.3) becomes

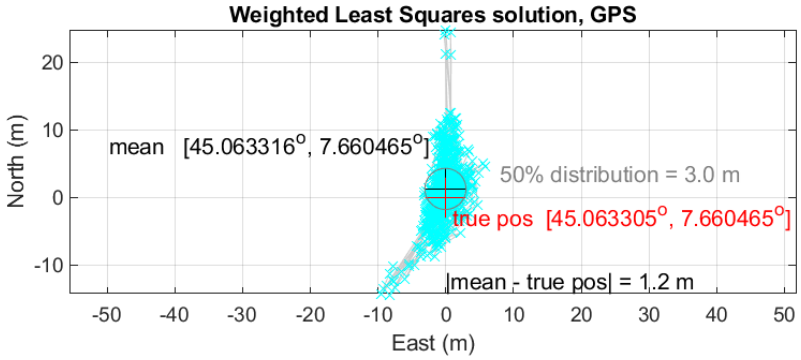
$$\mathbf{\Delta x} = \begin{bmatrix} \delta x_u \\ \delta y_u \\ \delta z_u \\ \delta b_{GPS} \\ \delta b_{Galileo} \end{bmatrix} \tag{5.3}$$

That means that theoretically 4 satellites are not enough anymore to obtain the solution because the unknowns become 5. This problem can be overcome by considering a priori the bias between systems, the Inter-System Bias (ISB). The GGTO, is transmitted in the navigation message, as reported in the previous sections. Thanks to this, it is possible to adjust the times in a single reference system and solve the PVT with four unknowns. As mentioned in Section 4.5, in our tests GGTO has been downloaded from the network because no navigation messages have been decoded by the GNSS chipset. In this work the multi-constellation algorithm uses the Method 1 described in Section 4; moreover, since GSA strongly encourage to use only *E1C* 2^{nd} *code* status, only E1 frequency has been considered.

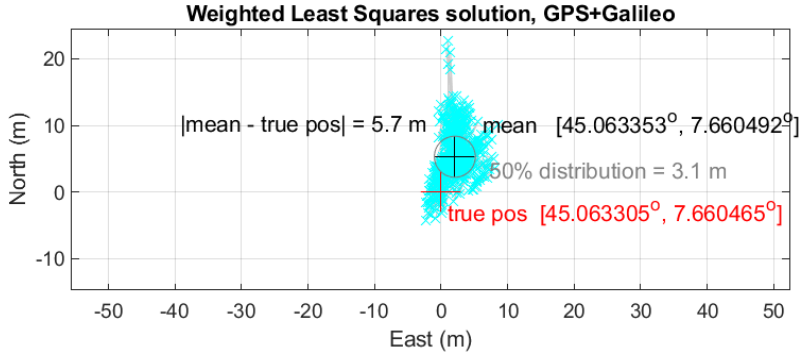
5.5 Analysis of PVT solutions

The Galileo measurements collected during our tests often present some anomalies in terms of status, as already mentioned in Section 5.2. This problems affect the final PVT. Moreover in several datasets, Galileo measurements are only obtained on E5 frequency and following GSA suggestions they are not used. Figure 5.2 shows a comparison between a single-constellation PVT and a multi-constellation PVT in a dataset in which the E5 measurements were not correctly filtered by *E1C 2nd Code lock* state. It is clear that using also the Galileo satellites, which in this specific dataset are three, the solution worsens considerably in terms of accuracy. Despite the fact that the value of *50% distribution* is substantially unchanged, it can be seen

that in the multi-constellation graph, there are fewer outliers and the dispersion is lower. In this specific plot pseudoranges are not smoothed and GPS signals were collected on both L1 and L5.



(a) *GPS-only PVT*

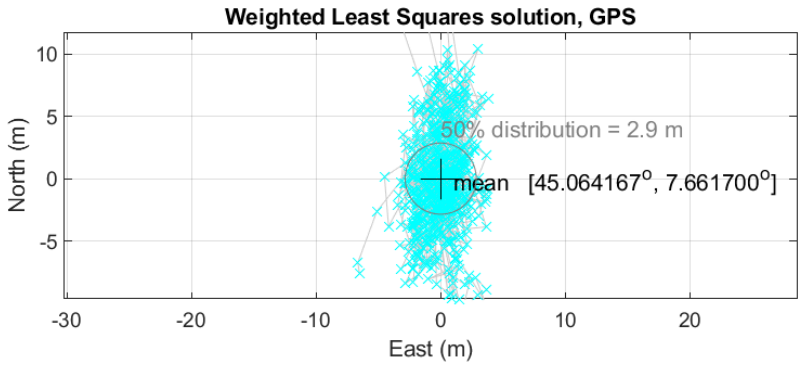


(b) *GPS+Galileo PVT*

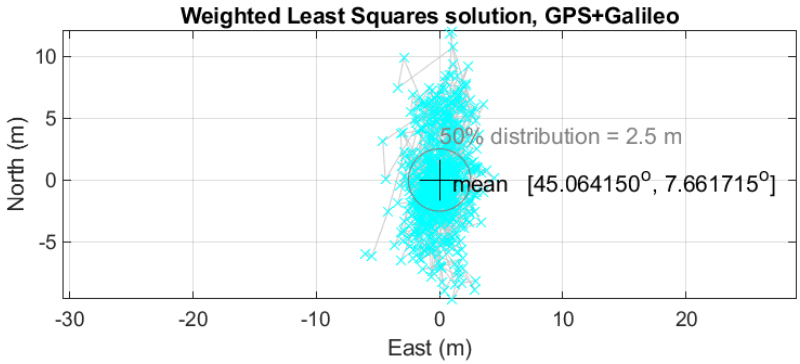
Figure 5.2: Comparison between multi-constellation and single-constellation PVT in anomalies-affected dataset

Figure 5.3 gives the PVT solutions using Galileo and GPS and only GPS in a dataset that does not present anomalies. In this case the flag *E1C 2nd Code lock* works and all data have been filtered correctly. Unfortunately there are no geo-referenced dataset among the ones not affected by anomalies. The results are similar, but the introduction of the Galileo satellites (3) produces less outliers and the *50% distribution* improves by 0.5 meters. It is seen that the distance between the mean of the two solutions is within one meter. Since the test location is not geo-referenced, it is difficult to analyse the performance in terms of accuracy hence for a rough estimate Google Earth is used. It is not a perfect tool to perform this type of analysis, but comparing the mean results with the approximate Google Earth static coordinates of the test, it is noticed that both the solutions lie within one meter.

In general, from what has been obtained through our tests, it can be said that



(a) *GPS-only PVT*



(b) *GPS+Galileo PVT*

Figure 5.3: Comparison between multi-constellation and single-constellation PVT in good dataset

the raw measurements do not have an excellent compatibility with Galileo. However, it is important to remember that Android updates its libraries monthly and it is likely that these issues will be fixed soon. In addition, the GSA is very active in this respect and in other contexts good results have been achieved using the European constellation [8].

Chapter 6

Case study: cooperative positioning

6.1 Introduction

GNSS performances are not always enough: there are some applications that need both very high accuracy and availability. Several augmentation strategies are present in literature, for example Differential GNSS could help a lot in reducing measurements errors, but the costs and the infrastructure requirements make this approach not applicable in several environments. Another possible solution is the Peer-to-Peer (P2P) localization, where the main idea is that a group of receivers can cooperate to improve their position. In order to achieve this goal, a communication channel is needed because receivers have to share with each other several information. Since nowadays smartphones have different communication technologies such as wi-fi direct and 4G, peer-to-peer localization could be a very interesting field for Android devices. This cooperative technique is very useful with different conditions and applications. For example, if a user were in a limited sky visibility condition, another device could send some data to improve the constellation geometry. Another possible scenario could be found in a vehicular environment: nowadays cars are equipped with different ranging sensors (Lidar, UWB, Ultrasound), but they cannot work in presence of occlusions, so peers could share positioning information to evaluate the baseline.

Thanks to the collaboration with the NavSAS and the PIC4Ser groups of Politecnico di Torino, which, among other things, are engaged in the study of possible methods of cooperative positioning, a practical experiment has been carried out to test the developed algorithms in the field of cooperative ranging. In particular a zero-baseline test using GNSS pseudorange double differences has been performed. The idea behind the experiment carried out is the calculation of the baseline between

two smartphones, which share GNSS raw measurements with each other. As for the previous tests, the rage calculation has been performed in the aftermath using data collected by two Xiaomi Mi8 Pro.

6.2 Pseudorange Double Differences ranging

Through a raw GNSS differential approach, the correlated errors are expected to be cancelled out in two receivers sharing the GNSS raw measurements. The pseudorange differencing approach for the range evaluation is based on the double differentiation of pseudorange measurements [6].

The pseudorange between the receiver a and the the satellite j can be considered as

$$\rho_a^j = R_a^j + \epsilon \tag{6.1}$$

where R_a^j is the real range between the receiver and the satellite while ϵ is the error introduced by the various sources described in Section 2.4. By differentiating the pseudoranges evaluated by two receivers a and b for satellite j at the same time epoch, the error related to the satellite clock is completely eliminated. Whereas if the receivers are close enough to each other, the satellite ephemeris error and the atmospheric error contributions are also mitigated [6](around 100 Km for ionosphere and 10 km for troposphere) [18]. The single difference is then

$$\Delta\rho_{ab}^j = \Delta R_{ab}^j + \Delta\epsilon \tag{6.2}$$

where $\Delta\epsilon$ is composed by the non-common error contributions $\Delta\epsilon_{ab}$ and the addition of the receiver clock offset Δc_{ab}:

$$\Delta\epsilon = \Delta c_{ab} + \Delta\epsilon_{ab} \tag{6.3}$$

If a pair of satellites (i, j) is visible to both receivers, it is possible to make a double difference by subtracting the two single differences. This leads to the cancellation of the individual receiver clock offsets, hence negating the term Δc_{ab} while the uncorrelated errors are statistically doubled (ϵ_{dd}) increasing the noise of the variance. The double difference result is

$$\Delta\rho_{ab}^j - \Delta\rho_{ab}^i = [\vec{e}^{\,i} - \vec{e}^{\,j}] \cdot \vec{r}_{ab} + [\Delta\epsilon_{ab}^i - \Delta\epsilon_{ab}^j] \tag{6.4}$$

where $[\vec{e}^{\,i} - \vec{e}^{\,j}]$ is the steering vector in the projection of the baseline vector $\vec{r}_{ab}$. After selecting an appropriate satellite used as a reference, $\vec{r}_{ab}$ can be calculated solving the equation

$$\mathbf{d}_{ab} = \mathbf{H}\vec{r}_{ab} + \epsilon_{dd} \tag{6.5}$$

where **d** is a column vector of double differences between the reference and all the other available satellites and **H** is the matrix containing the differentiation of the steering vectors, for N satellites. It is defined as

$$\mathbf{H} = \begin{bmatrix} [\vec{e}^{1} - \vec{e}^{0}]^T \\ \vdots \\ [\vec{e}^{N-1} - \vec{e}^{0}]^T \end{bmatrix} \tag{6.6}$$

where Satellite 0 is used as reference. $\vec{r}_{ab}$ is than evaluated as

$$\vec{r}_{ab} = (\mathbf{H}^T \mathbf{R}_d \mathbf{H})^{-1} \mathbf{H}^T \mathbf{R}_d \mathbf{d}_{ab} \tag{6.7}$$

where $\mathbf{R}_d$ is a weighting matrix, built in this work using the σ_{Google} described in Section 3.5. Since $\vec{r}_{ab}$ is a vector of coordinates, the range is given by its norm.

6.3 Zero-baseline test

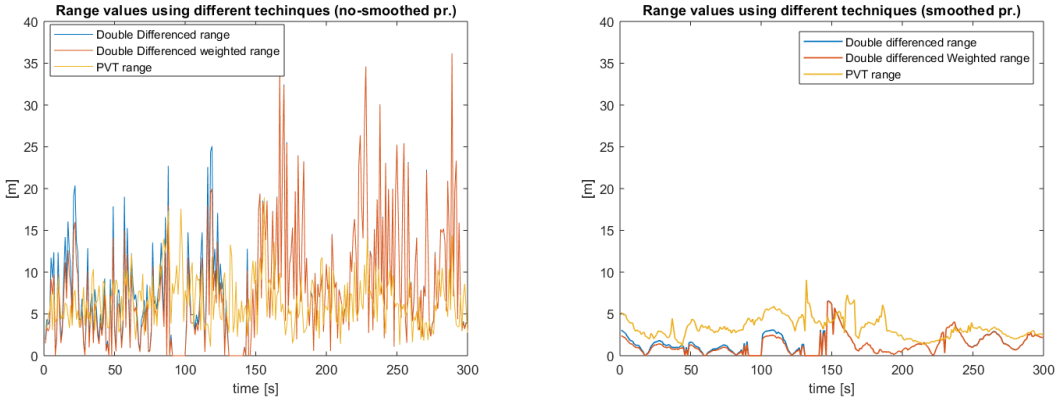
The static zero-baseline test was performed on the 15^{th} of March 2019 on the campus of Politecnico di Torino, Torino, Italy (45.063780°N, 7.662003°E) with two Xiaomi Mi8 Pro placed in contact with each other under the same reception conditions and there GPS L1 measurements were processed. Theoretically a zero baseline test should be performed connecting two receivers to the same antenna, but for an overall view of the smartphones' capabilities the followed approach was considered sufficient. After the data collection, the measurements coming from the two smartphones have to be synchronised. In particular the method explained in [6] has been followed in order to estimate the right value of the pseudorange. This proposes to use a linear extrapolation of the Doppler measurements at the receiver:

$$\rho(t_0 + \Delta t) = \rho(t_0) + \Delta t \cdot \lambda \cdot \phi(t_0) \tag{6.8}$$

where t_0 is the measurement time, $t_0 + \delta t$ it the time at which it is needed to estimate the pseudorange, λ is the carrier wavelength and ϕ is the Doppler measurement. In order to perform this adjustment it is important to have corrected values of Doppler, possibly using the strategy described in Section 4.3. Once the measurements are synchronised it is possible to apply the procedure explained in Section 6.2. It is important to underline that before evaluating the range, at each epoch there is a filtering stage at which only the measurements considered as good are used in the ranging procedure. In this test the σ_{Google} described in Section 3.5 is used to grade the satellites.

Figure 6.1 shows the results obtained in the zero-baseline test compared with the range evaluated through the PVT results. In particular the graphs show the range

values obtained without using the weighting matrix (Double Differenced range) and using it (Double Differenced Weighted range). The PVT is evaluated using Method 1 explained in Section 4.1. It is evident that all the approaches have an high improvement using smoothed pseudoranges. The used smoothing strategy is the one explained in Section 4.3. In general while both Double Differences ranges are comparable with the PVT-based ones in a non-smoothing scenario they always assume closer to zero values. Introduction of the weighting matrix brings minimal benefits in both cases.



(a) *Range evaluation using non-smoothed pseudoranges*

(b) *Range evaluation using smoothed pseudoranges*

Figure 6.1: Range comparison using smoothed and non-smoothed pseudoranges

Thanks to the fact that the smoothing strategy decreases the noise in the pseudorange measurements, the uncorrelated errors contribution is lower. This is very important in the double differences methodologies because these errors are theoretically quadrupled.

Chapter 7

Conclusions and possible improvements

In this work, an overview of the capabilities and the effective performance of the Android GNSS raw measurements has been provided. Every test has been performed using only the data coming from the smartphones' GNSS chipset without any aid from external sensors and the network. It is important to underline some concepts related to the quality and reliability of the measurements received by the smartphone receiver, which acts as a black box and on which it is not possible to operate. The first questions arise from the fact that the received data includes the difference between the time of the receiver and the GPS' one. This should be estimated through the PVT computation. It is clear that the receiver makes an initial calculation of the bias. Although, as has been described in Section 3.4, the pseudorange can be calculated even without the bias, this data can be used to simplify the process. Another important measurement that remains ambiguous and deserves to be investigated is the one related to the estimation of the error; Android does not provide any explanation on how it is calculated, but it would be interesting to have more information in order to analyse and possibly improve the used method. In addition, other measurements often have anomalies and sometimes are not available.

From the point of view of PVT solutions, good results have been achieved, especially considering that they have been obtained using the data collected from a smartphone and therefore from a very low-cost GNSS chipset. The algorithms introduced, both for the construction of the weighting matrix and for the pseudorange smoothing, have brought considerable improvements. The use of the L5 frequency has contributed significantly to the increase in overall quality. Unfortunately, the signals received by the Galileo satellites have not always been usable. Moreover, the number of available satellites is always limited; therefore, the implementation of the multi-constellation system has not led to the desired improvements.

As for the decoding of the GPS Navigation message, after understanding the

standard used by Android to provide the data it was possible to obtain the ephemeris of the satellites. This makes the smartphone completely independent from the positioning point of view. On the other hand, Galileo has also shown shortcomings in this respect, since no message has been received from the European constellation. However, it is important to stress that the Google APP has been used for data collection and this omission could be due to that.

The results obtained in the zero-baseline test for cooperative ranging give hope for future applications based on cooperative positioning. The smoothing strategy makes it possible to greatly improve the estimate of the baseline. Due to time constraints, it was not possible to carry out the test using also the L5 frequency. However, there are potentials for improvement and the addition of other constellations could provide even more accurate results.

There are aspects of this work that can be the starting point for future analysis. This work has been done completely in post-processing, it would be interesting to test the developed strategies directly on the smartphone in a real-time. In addition, all the tests were carried out in a static scenario, the current condition could be enriched with the analysis of dynamically collected data. Since the smartphone is a tool that has among its main objectives the access to the network and given the continuous developments in dual-frequency GNSS chipsets, a further idea to improve the position is to use augmentation systems. In addition more advanced techniques such as Real Time Kinematics (RTK) and Precise Point Positioning (PPP) could be implemented.

Bibliography

[1] *Android developers guide on Raw GNSS Measurements*. URL: https://developer.android.com/guide/topics/sensors/gnss#log.

[2] Bernhard Hofmann-Wellnhof, Herbert Lichtenegger, Elmar Walse. *GNSS-Global Navigation Satellite Systems: GPS, GLONASS,Galileo and more*. Ed. by Springer Wien New York. 2008.

[3] *CDDIS website*. URL: https://cddis.nasa.gov/Data_and_Derived_Products/GNSS/GNSS_data_and_product_archive.html.

[4] *Constellation Arrangement*. URL: https://www.gps.gov/systems/gps/space/.

[5] Elliot D. Kaplan, Christopher J. Hegarty. *Understanding GPS: principles and applications*. Ed. by Artech House. 2006.

[6] Fabian de Ponte Muller, Alexander Steingass and Thomas Strang. "Zero-Baseline Measurements for Relative Positioning in Vehicular Environments". In: *Inside GNSS* (2013).

[7] Fabio Dovis, Bilal Muhammad, Ernestina Cianca, Khurram Ali. "A Run-Time Method Based on Observable Data for the Quality Assessment of GNSS Positioning Solutions". In: *InsideGNSS* (November 2015). URL: https://www.researchgate.net/publication/277341499_A_Run-Time_Method_Based_on_Observable_Data_for_the_Quality_Assessment_of_GNSS_Positioning_Solutions.

[8] The GSA GNSS Raw Measurements Task Force. "Using GNSS raw measurements on Android devices". In: (2018). URL: https://www.gsa.europa.eu/system/files/reports/gnss_raw_measurement_web_0.pdf.

[9] *Fused Locaion Provider API*. URL: https://developers.google.com/location-context/fused-location-provider/.

[10] *Galileo and GPS 'synchronise watches': new time offset helps working together*. URL: https://m.esa.int/Our_Activities/Navigation/Galileo_and_GPS_synchronise_watches_new_time_offset_helps_working_together.

[11] Galileo GNSS. "GPS Week Number Rollover. April 6, 2019". In: (29 March 2019). URL: https://galileognss.eu/gps-week-number-rollover-april-6-2019/.

[12] *Generic Receiver Description*. 2011. URL: https://gssc.esa.int/navipedia/index.php/Generic_Receiver_Description.

[13] *GPS measurements tool*. URL: https://github.com/google/gps-measurement-tools.

[14] Gunter Hein. "GNSS Interoperability: Achieving s Global System of Systems or "Does Everything Have to be the Same?"" In: *Inside GNSS* (2006). URL: http://insidegnss.com/auto/0106_Working_Papers_IGM.pdf.

[15] International GNSS Service (IGS), RINEX Working Group and Radio Technical Commission for Maritime Services Special Committee 104 (RTCM-SC104). "RINEX, The Receiver Independent Exchange Format". In: (2015). URL: ftp://igs.org/pub/data/format/rinex303.pdfs.

[16] J. Sanz Subirana, JM. Juan Zornoza and M. Hernandez-Pajares. *GNSS signal*. 2011. URL: https://gssc.esa.int/navipedia/index.php/GNSS_signal.

[17] J. Sanz Subirana, J.M. Juan Zornoza and M. Hernández-Pajares. *Carrier-smoothing of code pseudoranges*. 2011. URL: https://gssc.esa.int/navipedia/index.php/Carrier-smoothing_of_code_pseudoranges.

[18] J. Sanz Subirana, J.M. Juan Zornoza, M. Hernández-Pajares. *GNSSData Processing, Vol. I: Fundamentals and Algorithms. Vol. I: Fundamen-tals and Algorithms*. Ed. by ESA Communication. 2013. URL: https://gssc.esa.int/navipedia/GNSS_Book/ESA_GNSS-Book_TM-23_Vol_I.pdf.

[19] Jorg H Han and Edward Powers. "Implementation of the GPS to Galileo time offset (GGTO)". In: (2005). URL: https://www.researchgate.net/publication/4212833_Implementation_of_the_GPS_to_Galileo_time_offset_GGTO.

[20] Marco Rao, Gianluca Falco. "How can pseudorange measurements be generated from code tracking?" In: *InsideGNSS* (January/February 2012), pp. 26–33.

[21] GPS NAVSTAR. *Global Positioning System Standard Positining Service Signal Specification*. 1995. URL: https://www.gps.gov/technical/ps/1995-SPS-signal-specification.pdf.

[22] Neil Gogoi, Alex Minetto, Nicola Linty, Fabio Dovis. "A Controlled-Environment Quality Assessment of Android GNSS Raw Measurement". In: (2018). URL: https://www.researchgate.net/publication/329842633_A_Controlled-Environment_Quality_Assessment_of_Android_GNSS_Raw_Measurements.

[23] *New civil signals*. URL: https://www.gps.gov/systems/gps/modernization/civilsignals/.

[24] Charles Jeffrey for NovAtel Inc. *An Introduction to GNSS: GPS, GLONASS, Galileo and other Global Navigation Satellite Systems*. Ed. by NovAtel Inc. 2012. URL: https://www.novatel.com/assets/Documents/Books/An_Introduction_to_GNSS.pdf.

[25] Richard Langley. "Innovation: Accuracy versus Precision". In: (2010). URL: https://www.gpsworld.com/gnss-systemalgorithms-methodsinnovation-accuracy-versus-precision-9889/.

[26] J.A Ávila Rodríguez. *Binary Offset Carrier (BOC)*. 2011. URL: https://gssc.esa.int/navipedia/index.php/Binary_Offset_Carrier_(BOC).

[27] J.A Ávila Rodríguez. *Galileo Signal Plan*. 2011. URL: https://gssc.esa.int/navipedia/index.php/Galileo_Signal_Plan.

[28] *System Design Details*. 2011. URL: https://gssc.esa.int/navipedia/index.php/System_Design_Details.

[29] *Wi-Fi location: ranging with RTT*. URL: https://developer.android.com/guide/topics/connectivity/wifi-rtt.

[30] Zebo Zhou, Bofeng Li. *Optimal Doppler-aided smoothing strategy for GNSS navigation*. 2015. URL: https://www.researchgate.net/publication/291390683_Optimal_Doppler-aided_smoothing_strategy_for_GNSS_navigationhttps://www.researchgate.net/publication/291390683_Optimal_Doppler-aided_smoothing_strategy_for_GNSS_navigation.