POLITECNICO DI TORINO

Master's Degree in
Mechatronics Engineering

Master's Degree Thesis

Integration of Navigation Sensors Based on Factor Graph Optimization

Relatori
Prof. Fabio DOVIS
Dr. Alex MINETTO
Dr. Simone ZOCCA
Dr. Oliviero VOUCH

Candidato
Alessandro Pighini

Anno Accademico 2022-2023

Abstract

Global Navigation Satellite Systems (GNSS) and Inertial Navigation Systems (INS) play pivotal roles in modern navigation and positioning applications. The integration of these two technologies has become crucial for achieving high-accuracy and robust navigation solutions in various domains, such as autonomous vehicles, aviation and mobile devices. These two technologies are characterized by complementary features, with GNSS providing absolute position but suffering from external conditions and INS offering continuous, high-rate relative motion data but being affected by error accumulation over time. Their integrations aim to emphasize strengths while minimizing weaknesses for robust and accurate navigation solutions.

Several integration approaches, of different complexity, can be found in literature. In this thesis a factor graph framework has been adopted to perform such integration. Factor graphs have proven as a powerful mathematical framework for modeling and solving complex estimation and optimization problems as Simultaneous Localization and Mapping (SLAM). Recently, because of their flexibility, factor graphs have emerged as an alternative method for GNSS positioning. Factor graphs describe positioning problems in terms of optimization problems, allowing the solution to be obtained over multiple iterations, differently from other traditional navigation filters such as Extended Kalman Filter (EKF). This means that the involved functions are linearized multiple times and time correlation between consecutive epochs is better exploited. Moreover, positions related to previous instants can be kept inside the graph so that, as new positions are estimated, the gained information can be used to refine the old solutions, obtaining a better performance in post-processing. These features, along with many robust estimation techniques that have been developed for factor graphs, allow to obtain positioning solutions which can be more robust in challenging environments, such as urban scenarios.

This thesis provides a comprehensive overview of GNSS and INS technologies as well as the mathematical formulation of factor graphs. The framework of a GNSS/INS tight integration based on factor graphs is then developed. Finally, an analysis of the results obtained deploying this formulation on a MATLAB receiver is performed and compared to the solution obtained from an EKF. The dataset used for this purpose is taken from a urban scenario (città di Torino). In conclusion, the results obtained in this thesis prove the advantages coming from the integration of GNSS and INS in a factor graph framework, offering increased accuracy and robustness in complex environments.

Contents

List of Figures

List of Tables

Chapter 1

Satellite Navigation and Inertial Navigation systems

1.1 Global Navigation Satellite Systems

Global Navigation Satellite Systems (GNSS) involve a constellation of satellites, orbiting Earth, continuously transmitting signals over the globe that allow properly equipped receiver devices to determine their three-dimensional position and timing information with respect to an absolute reference frame.

The positioning is achieved solving a geometric problem involving distances (ranges) of a user to a set of GNSS satellites of which the position is known. These fundamentals information are determined by the user's receiver elaborating signals transmitted by the satellites. [1]

The ability to provide high positioning accuracy under all weather conditions has resulted in GNSS revolutionizing modern positioning and navigation. The roots of this technology date back to 1960, and since then, it has undergone significant development.

1.1.1 Fundamentals of satellite-based navigation

Time of Arrival (ToA) is the simplest and most common ranging technique. This method is based on knowing the exact time a signal was sent, the exact time the signal arrives and the speed at which the signal travels (assumed to be the speed of light). By multiplying the interval of time by the signal speed, the so called range (distance user-transmitter) is obtained. Once different ranges are computed by the receiver from a set of visible reference transmitters, in this case satellites, a multilateration problem is solved.

In order to solve the positioning problem, the receiver's clock and the satellites' clocks should be perfectly synchronized. However, having the user clock perfectly aligned to that of the satellites is an infeasible operation for cost-complexity reasons. The temporal misalignment is then typically considered as an additional unknown in the positioning context. In other words, the multilateration problem is formulated as a four-dimensional problem: three-dimensions given by the position plus the temporal dimension, represented by the user clock bias. Given that the number of unknowns is equal to four, at least four satellites must be available to perform the multilateration. In this context, the term *pseudorange* is used instead of *range*.

The problem can be represented for simplicity in a bi-dimensional set-up as shown in Figure 1.1. Once the distances have been computed, the geometrical position of the receiver can be unambiguously retrieved from the intersection of the three circles centered in the satellites positions. This computation is called trilateration.

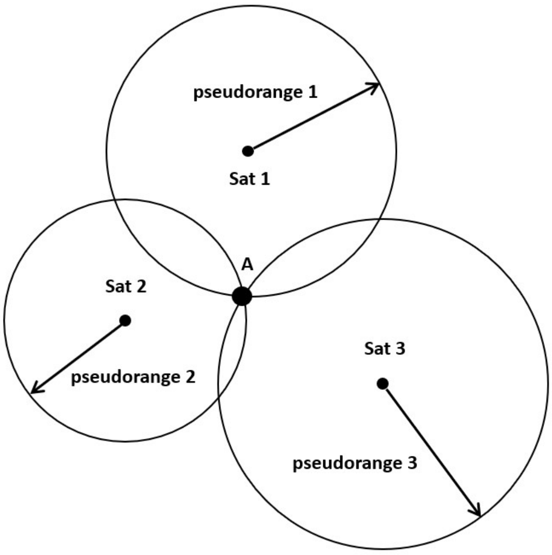


Figure 1.1: 2-D trilateration based on Time-of-Arrival (ToA). Point A is the receiver position

To extend this process to a three-dimensional formulation a new spatial component will be added (i.e. height) as well as another satellite. The distances from each satellite will now represent spheres instead of circles. Computing the intersection of these four spheres the three-dimensional location can be unequivocally retrieved [1, 2].

1.1.2 GNSS architecture

Any GNSS constellation operates through three different segments known as the Space Segment, the Ground Control Segment, and the User Segment [1].

The **Space segment** consists of the satellites themselves (enough to ensure that at least four are visible simultaneously from any point at any time) whose major function is the down-link transmission of radio-navigation signals, as well as storage and re-transmission of the navigation message sent by the control segment. These transmissions are controlled by highly stable atomic clocks onboard the satellites.

The **Ground control segment** consists of a global network of ground facilities. It utilizes earth based tracking stations around the world to manage the entire navigation system by tracking the satellites, keep the corresponding GNSS time scale updated, performing analyses and sending commands and data to space.

The **User segment** consists of the GNSS receivers. The satellites' Radio Frequency (RF) signals are tracked and decoded to determine pseudoranges (and other information), utilized in determining user's position, velocity, and time.

In figure 1.2 a schematic representation of GNSS segments and their interactions.

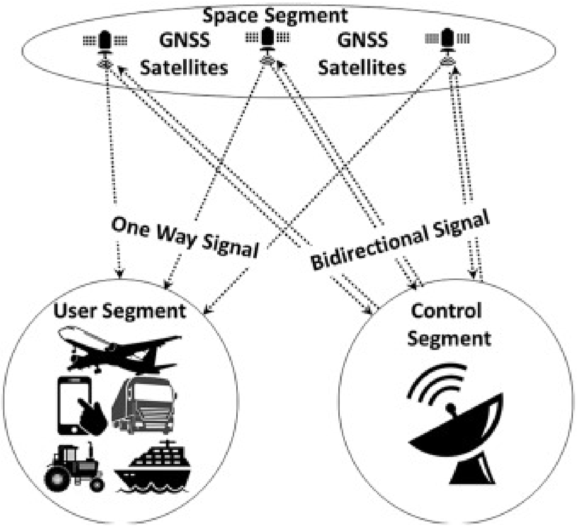


Figure 1.2: GNSS system segments [3].

1.1.3 Navigation Satellite Systems

Nowadays, several satellite systems (i.e. constellations) have been developed for both military and civilian applications. In the following section, an overview of the most important global coverage satellite systems is given, with the main purpose of examining their core features in relation to architectural structure and signal construction. However, of particular significance within this overview is the Global Positioning System (GPS), given that all the GNSS ranging data under consideration within this thesis are obtained from this constellation.

GPS

Global Positioning System (GPS) is composed by 24 Medium-Earth Orbit (MEO) satellites, circling the Earth twice a day (celestial time) at an altitude of about 20200 Km, became fully operational in 1993 [4]. At the time of writing, the number of operational GPS satellites amounts to 31 [5, 6]. The GPS signal is a spread spectrum Code Division Multiple Access (CDMA) signal modulated at two carrier frequencies in the L-band (1 GHz - 2 GHz): L1 (1575.42 MHz) and L2 (1227.60 MHz). These signals are modulated by two Pseudo-Random Noise (PRN) codes, the Coarse-Acquisition (C/A) code on L1 and the P-code on both L1 and L2, that grant orthogonality between transmissions from different satellites. P-code is restricted only to military users via its encryption by a Y code. In addition to PRN codes, the navigation message consisting of information such as satellite ephemeris, satellite clock bias and satellite status is also modulated onto the L1 and L2 carriers [1, 7].

GALILEO

Galileo constellation consists of 28 operational satellites satellites, all but two placed in three circular MEO-planes at an altitude of 23222 Km above the Earth [8]. Galileo satellites permanently transmit CDMA signals modulated on top of three different carriers: E1 (1575.420 MHz), E5 (1191.795 MHz) and E6 (1278.750 MHz); E5 signal is further sub-divided into two signals, E5a (1176.450) and E5b (1207.140) [9]. Differently from GPS signals, Galileo signals have a further modulation on top of them, a sub-carrier modulation (Binary-offset carrier - BOC), allowing interoperability with GPS system and reducing interference from other systems transmitting over the same bands.

GLONASS

Globalnaya navigatsionnaya sputnikovaya sistema (GLONASS) constellation consists of 24 operational satellites satellites placed in three circular MEO-planes at an altitude of 19130 Km above the Earth. GLONASS-M satellites transmit Frequency Division Multiple Access (FDMA) signals, characterised by the same PRN over different RF-carriers. Next generation GLONASS-K satellites are meant instead to transmit CDMA signals for both restricted and civil services [1, 7].

BEIDOU

The BeiDou Navigation Satellite System is the Chinese satellite navigation system. The first generation, known as BeiDou-1, had three satellites providing limited navigation services mainly in China and nearby regions from 2000 to 2012. The second generation, called BeiDou-2 or COMPASS, began operating in 2011 with 10 satellites and extended coverage to the Asia-Pacific region. The third generation, BeiDou-3, launched in 2015, aimed for global coverage. Nowadays, the space constellation consists of 3 Geostationary-Earth Orbit (GEO) satellites, 3 Inclined Geosynchronous Orbit (IGSO) satellites, and 24 MEO satellites. [7]

The research carried out in this thesis can be applied to all these satellite systems, allowing multi-constellation signal processing.

1.2 GNSS receiver

A GNSS receiver serves as the interface between a navigation system and a generic target end-user. The receiver will continuously acquire and track the low-power RF-signals transmitted by a group of visible satellites of a given GNSS-constellation in order to determine the information needed to obtain the PVT solution. These information are the temporal misalignment information (code delay), leading to pseudorange measurements, and the frequency-offset information (carrier frequency offset), leading to Doppler-shift measurements. The combination of these observables, then, is exploited within a processing unit to produce a PVT solution.

High-cost professional receivers, to enhance the precision of the PVT solution, also exploit carrier phase information. Adding this observable, it is possible to gain precision solving the integer-cycle ambiguity [10]. This work only includes a short overview of low-cost GNSS receiver radio-frequency front-end and, beyond analogue-to-digital conversion, of the early signal-processing stages.

1.2.1 Ranging signals structure

GNSS satellites continuously transmit synchronous navigation signals towards the Earth. These signals, named Signal-in-Space (SIS), are properly constructed to contain ranging codes and navigation data, thus allowing the receiver to compute both the satellite coordinates and the user-to-satellite temporal misalignment, in the view of constructing an estimate of the pseudorange (i.e. noisy and biased user-to-satellite range). This signal is constructed starting from the waveform at RF (carrier) on top of which the navigation message and the spreading code are modulated. The navigation message is a binary sequence which brings about relevant information including ephemeris, satellite-clock corrections, and all synchronization parameters needed to retrieve a pseudorange estimate. As in case of Galileo signal structure, a further modulation, called sub-carrier, may be present.

A schematic representation of the signal construction is represented in Figure 1.3.

While the spreading code is a binary pseudo-random-noise (PRN) ranging sequence which grants the orthogonality between signals belonging to the same constellation (Multiple Access scheme) and implements a Direct Sequence Spread Spectrum (DSSS) paradigm to mitigate inter-system interference [11].

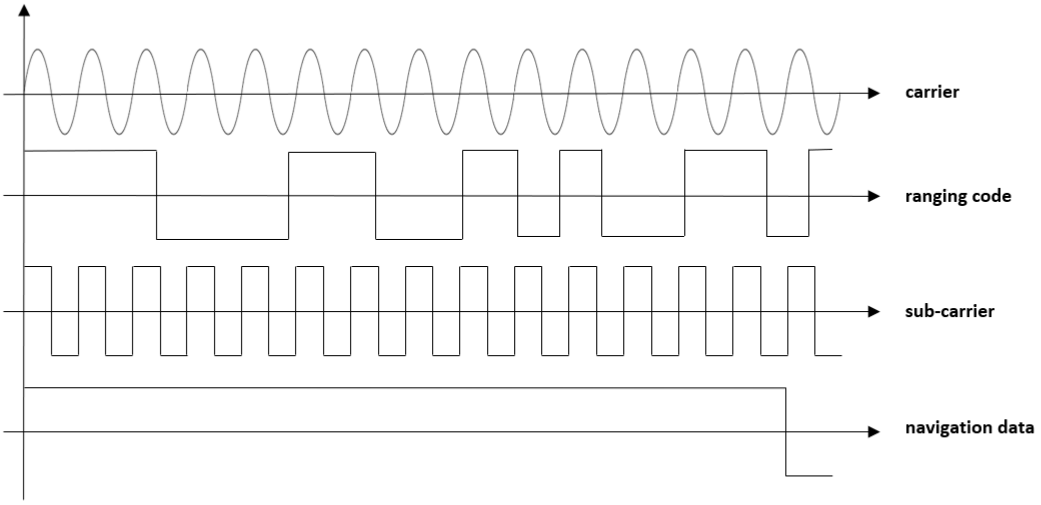


Figure 1.3: Multi-layer GNSS signal structure

The RF-SIS broadcast by satellites are affected by interference effects due to the upper (ionosphere) and lower (troposphere) layers of the atmosphere, as well as other effects such as scintillation, the presence of other GNSS systems transmitting over adjacent bands, and, possibly, intentional interference (e.g. jamming, spoofing etc.). Hence, the captured signal broadcast by a single satellite will have the structure described previously, but might be compromised in its integrity and characterized by a very low signal intensity.

1.2.2 Front-end structure

The incoming signals are captured though the receiver's antenna and then fed to the front-end section. The front-end is responsible for the elaboration of the received signals for signal processing tasks. A schematic representation of a typical front-end structure is given in Figure 1.4. The shown components, at front end architecture level, absolve interconnected tasks aimed to process and convert a RF signal to a baseband digital signal as follows [12].:

– Filtering and amplification: ensure low-noise and out-of-band rejection, as well as amplification to compensate for transmission losses.

– Down-conversion: the front end is responsible for down-converting the signal, so that the spectrum is shifted to intermediate frequencies (IF), meaning that a residual carrier modulation is still present , but close to baseband.

– Quantization: the signal is now digitized through analog to digital converters (ADC).

– Automatic Gain Control: here the front-end section gain is regularized to take benefit from the full dynamic range.

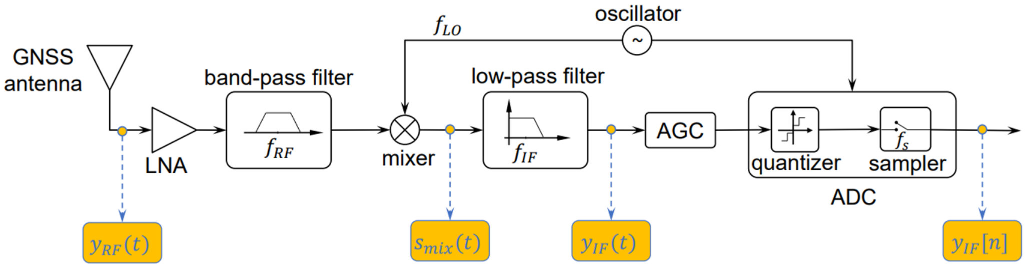


Figure 1.4: GNSS receiver analogue front-end [13].

1.2.3 Signal Acquisition and Tracking

The final receiver front-end output is a sequence of noisy samples, at intermediate frequency, $y_{IF}[n]$. This sequence is then further processed by the receiver to actually recover delay and Doppler-shift. To compute these quantities, the sequence of samples will pass through two consecutive stages: the acquisition stage and the tracking stage. In the acquisition stage, the digitized signal is correlated with local digital replicas of different PRN ranging sequences, so that the correct PRN is identified and a rough estimate of delay and Doppler-shift is obtained. In the tracking stage, starting from the acquisition stage output, the local replica is kept synchronized with the noisy IF-samples in order to dynamically retrieve an accurate estimate of delay and Doppler-shift. These quantities are eventually exploited to compute pseudorange and Doppler measurements [14].

Acquisition

The acquisition stage has the aim of estimating the arrival time τ, that is used for computing user position and clock offset, and the Doppler frequency f_d, which instead is used for computing the user velocity and clock drift. Other then estimating τ and f_d, also an estimation of the carrier phase may be performed; this aspect will not be faced in this thesis. Given the noisy samples $y_{IF}[n]$ as input, the receiver will compute the Cross-Ambiguity Function (2-D cross-correlation between the incoming signal and the local replica) as [15]:

$$S_i(\tau, f_d) = \sum_{n=0}^{L-1} y_{IF}[n] c_i(n-\tau) e^{j2\pi(f_{IF}+f_d)n} \tag{1.1}$$

where $y_{IF}[n]$ is the sequence of samples of the received signal, c_i is the PRN code of the i-th satellite, τ is the local code offset, f_d is the local Doppler shift, L is the number of samples contained in the so called integration time T_d ($T_d = LT_s$) given a sampling time T_s), and f_{IF} is the intermediate frequency (IF) of the carrier. Instead, the exponential represent the locally generated carrier, where the Doppler shift f_d is applied (the phase of the received signal is unknown and it not estimated at this stage, then a complex exponential in the local signal is used). Since the CAF is a complex function, its squared modulus is usually considered. First, the receiver has to detect which satellites are present in the received signal (each of them transmits a specific PRN): for every PRN a local replica is produced and used to compute, with the incoming signal, the corresponding CAF. It is then compared with a predetermined threshold and, if every point is below it, the satellite identified with that PRN is flagged as not present.

Finally, for the detected PRNs, the estimate of the couple of unknowns $\{\tau, f_d\}$ is performed with a Maximum-likelihood (ML) approach, maximizing the normalized squared norm of the CAF. Hence, defining a vector $p_{ML} = \{\tau, f_d\}$, the problem is to find $\hat{p}_{ML}$ such that [13]:

$$\hat{p}_{ML} = \max_{\{\tau, f_d\}} |S_i(\tau, f_d)|^2 \tag{1.2}$$

In Figure 1.5 a typical CAF is represented. The location of the peak is detected by the software receiver so that the corresponding value $\hat{p}_{ML} = \{\hat{\tau}, \hat{f}_d\}$ can be retrieved. Moreover, the noise-floor height is proportional to the noise level affecting the incoming signal, causing the peak to be not so identifiable if the satellite signal is too degraded.

Tracking

Once the acquisition has been performed, the estimate $\hat{\tau}$ and $\hat{f}_d$ will be passed to the tracking stage as input. Here, the codes are kept synchronized so that a fine code alignment is obtained through a dynamic recovering of delay and phase between the sequences. Doppler and pseudorange measurements are then continuously obtained. The most common architecture is designed with an outer closed feedback control loop which, inside, contains code and carrier tracking loops [16].

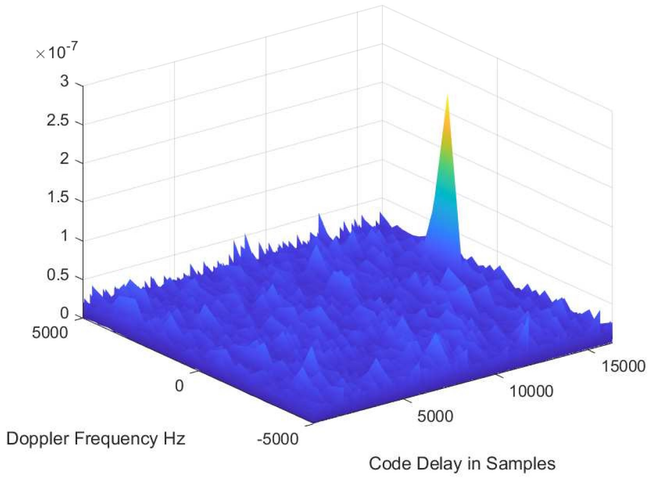


Figure 1.5: Cross Ambiguity Function for a noisy GPS signal.

The code delay is followed by the Delay Lock Loops (DLL), while the carrier can be tracked in two ways: following the phase, using Phase Lock Loops (PLL), or following the Doppler frequency, using the Frequency Lock Loops (FLL). In some applications PLL and FLL are used simultaneously (see Figure 1.6). With the adjustments obtained by these tracking loops, the final estimation of $\hat{\tau}$ and $\hat{f}_d$ is obtained and, other than that, the carrier and code wipe-off is performed, allowing the navigation data demodulation [17, 18].

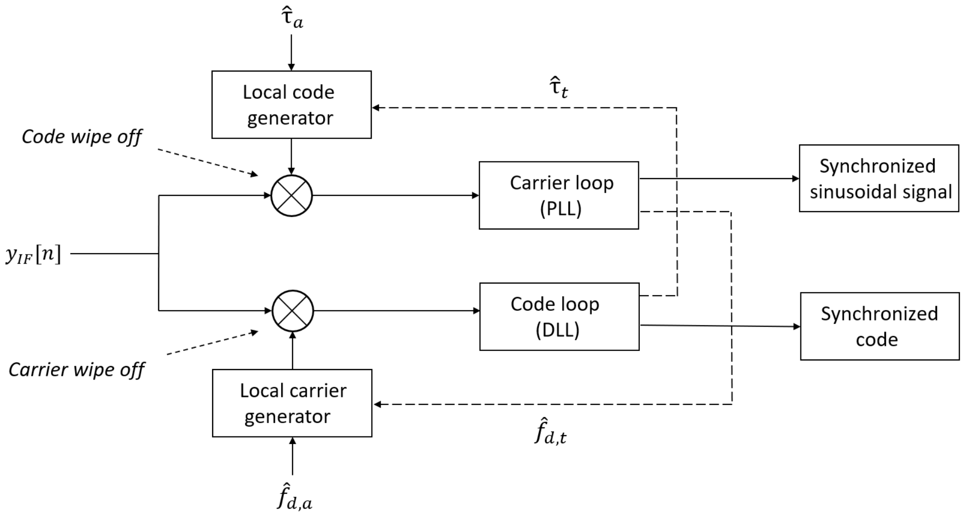


Figure 1.6: High-level tracking loop (PLL+DLL) architecture. Subscripts a and t stand for acquisition and tracking respectively.

1.3 GNSS PVT solution

Once carrier and code tracking loops (PLL and DLL) are locked, the final outputs of the receiver can be obtained: pseudorange and Doppler frequency. Pseudorange measurements, as anticipated in Section 1.1.1, can be obtained as the difference between the moment the receiver detects the signal, called Time of Arrival (ToA), and the time the signal was transmitted by the satellite, as indicated in the signal's navigation message (that has been previously decoded), multiplied by the speed of light. This difference will lead to the computation of a pseudorange instead of the geometrical range (i.e. user-satellite distance) due to the inherent presence of a bias. In fact the satellite and the receiver have their own independent clocks, making the synchronization unfeasible for cost reasons. Because of that, as anticipated previously, the localization problem is described by four unknowns: three, due to the geometrical 3-D position, plus one, due to the receiver's clock bias (the satellite's clock bias instead can be compensated for, since these clocks are continuously monitored by the control segment). Hence, at least four satellites have to be in view leading to the availability of four independent pseudoranges to be processed. In this section, an overview of the most common measurement models of pseudorange and Doppler-shift is given, leading to the estimation of the receiver's position, velocity and time (PVT). The following discussion is done referring to [1, 2]

1.3.1 Pseudorange equations and position computation

At each epoch, the pseudorange measurement associated with the j-th satellite can be described mathematically as:

$$\rho_j = r_j + c(\delta t_u - \delta t_j) + I_j + T_j + \epsilon_j \tag{1.3}$$

where r_j is the true range between the receiver antenna at signal reception time and the satellite antenna at transmission time; δt_u and δt_k are the receiver clock bias and the satellite clock bias respectively; I_j and T_j represent the delays induced on the signal passing through the ionosphere and troposphere respectively, ϵ_j accounts for residual errors. That being said, in the extracted navigation data there are also present information useful to compensate for known bias-error components such as the satellite clock offset, relativist effects and the ionospheric and tropospheric induced delays. These compensations will affect only the deterministic components of these errors. Hence, the corrected pseudorange ρ_j can be re-written as:

$$\rho_j = r_j + b_u + \epsilon_{\rho_j} \tag{1.4}$$

Here, all the residual errors affecting the pseudorange measurement have been collected in the term ϵ_{ρ_j}, usually called User Equivalent Range Error (UERE), while the receiver clock-bias term $c \cdot \delta t_u$ has been replaced by its range-equivalent term b_u, in meters. Furthermore, the expression of the geometric range r_j between user and j-th satellite is the Euclidean distance between the satellite itself $\boldsymbol{x_j} = (x_j, y_j, z_j)$ and receiver antenna phase centre coordinates $\boldsymbol{x_u} = (x_u, y_u, z_u)$:

$$r_j = \sqrt{(x_j - x_u)^2 + (y_j - y_u)^2 + (z_j - z_u)^2} \tag{1.5}$$

For each j-th satellite, an equation as (1.4), that is non-linear in the term $\boldsymbol{x_u}$, will be available. These nonlinear equations can be solved for the unknowns by employing iterative techniques based on linearization. Given an approximation point $\boldsymbol{\hat{x}_u} = (\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{b}_u)$, we can linearize them around that point, obtaining for each $j - th$ equation:

$$\rho_j = \hat{\rho}_j - \frac{x_j - \hat{x}_u}{\hat{r}_j}\Delta x_u - \frac{y_j - \hat{y}_u}{\hat{r}_j}\Delta y_u - \frac{z_j - \hat{z}_u}{\hat{r}_j}\Delta z_u + b_u \tag{1.6}$$

where

$$\hat{\rho}_j = \hat{r}_j + \hat{b}_u \tag{1.7}$$

$$\hat{r}_j = \sqrt{(x_j - \hat{x}_u)^2 + (y_j - \hat{y}_u)^2 + (z_j - \hat{z}_u)^2} \tag{1.8}$$

and

$$\begin{aligned} x_u &= \hat{x}_u + \Delta x_u \\ y_u &= \hat{y}_u + \Delta y_u \\ z_u &= \hat{z}_u + \Delta z_u \\ b_u &= \hat{b}_u + \Delta b_u \end{aligned} \tag{1.9}$$

To simplify the equations these new variables are introduced:

$$\begin{aligned} \Delta\rho_j &= \hat{\rho}_j - \rho_j \\ a_{x,j} &= \frac{x_j - \hat{x}_u}{\hat{r}_j} \\ a_{y,j} &= \frac{y_j - \hat{y}_u}{\hat{r}_j} \\ a_{z,j} &= \frac{z_j - \hat{z}_u}{\hat{r}_j} \end{aligned} \tag{1.10}$$

The $a_{x,j}$, $a_{y,j}$ and $a_{z,j}$ terms in (1.10) denote the direction cosines of the unit vector pointing from the approximate user position $\boldsymbol{\hat{x}_u}$ to the $j - th$ satellite. Now the equation (1.6) can be rewritten as:

$$\Delta\rho_j = a_{x,j}\Delta x_u + a_{y,j}\Delta y_u + a_{z,j}\Delta z_u - \Delta b_u \tag{1.11}$$

Supposing that N_s satellites are in view, a system of equations in matrix form can be written:

$$\boldsymbol{\Delta\rho} = \boldsymbol{H}\boldsymbol{\Delta x} \tag{1.12}$$

where the used terms are:

$$\boldsymbol{\Delta\rho} = \begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \vdots \\ \Delta\rho_{N_s} \end{bmatrix} \boldsymbol{H} = \begin{bmatrix} a_{x,1} & a_{y,1} & a_{z,1} & 1 \\ a_{x,2} & a_{y,2} & a_{z,2} & 1 \\ \vdots & & & \vdots \\ a_{x,N_s} & a_{y,N_s} & a_{z,N_s} & 1 \end{bmatrix} \boldsymbol{\Delta x} = \begin{bmatrix} \Delta x_u \\ \Delta y_u \\ \Delta z_u \\ -\Delta b_u \end{bmatrix} \tag{1.13}$$

(1.12) is a system of linear equations. Hence a Least Squares solution can be retrieved as:

$$\boldsymbol{\Delta x} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{\Delta\rho} \tag{1.14}$$

Moreover, the errors affecting pseudoranges (indicated as ϵ_{ρ_j}) in equation (1.4)) are usually assumed to be Gaussian, zero-mean and equally distributed, described by a covariance called σ^2_{UERE}. Hence the covariance matrix of ϵ_ρ, that includes all the pseudoranges, can be written as:

$$cov(\epsilon_\rho) = \sigma^2_{UERE} \cdot \boldsymbol{I}_{N_s \times N_s} \tag{1.15}$$

where $\boldsymbol{I}_{N_s x N_s}$ is the identity matrix of dimension $N_s \times N_s$. This error, affecting pseudoranges, will propagate in the solution according to:

$$cov(\boldsymbol{\Delta x}) = (\boldsymbol{H}^T\boldsymbol{H})^{-1} \cdot \sigma^2_{UERE} \tag{1.16}$$

Finally, defining the so called Geometric Dilution of Precision ($GDOP$):

$$GDOP = \sqrt{trace((\boldsymbol{H}^T\boldsymbol{H})^{-1})} \tag{1.17}$$

the standard deviation of the positioning error can be expressed as:

$$\sigma_x = GDOP \cdot \sigma_{UERE} \tag{1.18}$$

1.3.2 Doppler equations and velocity computation

The tracking loop (precisely the PLL) continuously provide as output the Doppler-shift between the received signal with respect to the nominal one. This shift is due to the relative motion between satellite and user (Doppler effect). Since the satellite's velocity can be retrieved from the information contained in the navigation data, an estimation of the user's velocity is then possible. According to the Doppler equation, yields:

$$f_{r,j} = f_{c,j}(1 - \frac{\boldsymbol{v}_{u,j} \cdot \boldsymbol{a}_j}{c}) \tag{1.19}$$

where $f_{r,j}$ is the received (hence affected by the Doppler) frequency related to the j-th satellite, $f_{c,j}$ is its nominal carrier frequency, $\boldsymbol{v}_{u,j}$ is the user-satellite relative velocity, $\boldsymbol{a}_j = (a_{x,j}, a_{y,j}, a_{z,j})$ is the unit steering vector pointing from user to satellite and finally c indicates the speed of light.

Given that the relative velocity user-satellite can be written as difference between the satellite ($\boldsymbol{v}_j = (v_{j,x}, v_{j,y}, v_{j,z})$) and user ($\boldsymbol{v}_u = (v_{u,x}, v_{u,y}, v_{u,z})$) velocity:

$$\boldsymbol{v}_{u,j} = \boldsymbol{v}_j - \boldsymbol{v}_u \tag{1.20}$$

and substituting (1.20) into (1.19), Doppler-shift is obtained as:

$$\delta f = f_{r,j} - f_{c,j} = -f_{c,j}\frac{(\boldsymbol{v}_j - \boldsymbol{v}_u) \cdot \boldsymbol{a}_j}{c} \tag{1.21}$$

Moreover, the receiver's clock is affected by a frequency bias offset, causing the actually received frequency $f_{r,j}$ to be different from the measured signal frequency $f_{u,j}$. This offset can be related to the receiver's clock drift rate $\delta\dot{t}_u$ as:

$$f_{r,j} = f_{u,j}(1 + \delta\dot{t}_u) \tag{1.22}$$

Now, substituting (1.22) into (1.21), after some steps:

$$\frac{c(f_{u,j} - f_{c,j})}{f_{c,j}} + \boldsymbol{v}_j \cdot \boldsymbol{a}_j = \boldsymbol{v}_u \cdot \boldsymbol{a}_j - \frac{c f_{u,j}\delta\dot{t}_u}{f_{c,j}} \tag{1.23}$$

and expanding the dot products:

$$\frac{c(f_{u,j} - f_{c,j})}{f_{c,j}} + v_{j,x}a_{j,x} + v_{j,y}a_{j,y} + v_{j,z}a_{j,z} = v_{u,x}a_{j,x} + v_{u,y}a_{j,y} + v_{u,z}a_{j,z} - \frac{c f_{u,j}\delta\dot{t}_u}{f_{c,j}} \tag{1.24}$$

Regarding the left side, $f_{c,j}$ is known, as well as $f_{u,j}$ and $\boldsymbol{a}_j$ (the latter from the position computation), $\boldsymbol{v}_j$ is computed from the satellite navigation message. On the other hand, the right side includes the velocity components of $\boldsymbol{v}_u = (v_{u,x}, v_{u,y}, v_{u,z}) = (\dot{x}_u, \dot{y}_u, \dot{z}_u)$ that has to be estimated, the term $f_{u,j}/f_{c,j}$ that is very close to unity (hence the approximation $f_{u,j}/f_{c,j} \simeq 1$) and the term $c\delta\dot{t}_u = \dot{b}_u$. Hence, in order to simplify (1.24), the variable d_j is introduced as:

$$d_j = \frac{c(f_{u,j} - f_{c,j})}{f_{c,j}} + v_{j,x}a_{j,x} + v_{j,y}a_{j,y} + v_{j,z}a_{j,z} \tag{1.25}$$

doing so, the equation (1.24) can be rewritten as:

$$d_j = \dot{x}_u a_{j,x} + \dot{y}_u a_{j,y} + \dot{z}_u a_{j,z} - \dot{b}_u \tag{1.26}$$

Finally, considering that N_s satellites are in view, N_s equations as (1.26) will be available. Compacting all of these equations in matrix form, it yields:

$$\boldsymbol{d} = \boldsymbol{H}\dot{\boldsymbol{x}} \tag{1.27}$$

where $\dot{\boldsymbol{x}} = (\dot{x}_u, \dot{y}_u, \dot{z}_u, -\dot{b}_u)$. The velocity estimation can be expressed as Least Square solution as:

$$\dot{\boldsymbol{x}} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{d} \tag{1.28}$$

Note that the matrix $\boldsymbol{H}$ obtained here is the same matrix $\boldsymbol{H}$ that arises in the resolution of the positioning problem and $\dot{\boldsymbol{x}}$ is the time derivative of $\boldsymbol{x}$. Because of that, even though this formulation is linearized, it is useful to re-formulate the velocity estimation problem to be solved in an iterative way. It means that the correction $\boldsymbol{\Delta}\dot{\boldsymbol{x}} = (\Delta\dot{x}_u, \Delta\dot{y}_u, \Delta\dot{z}_u, -\Delta\dot{b}_u)$, that is the time derivative of $\boldsymbol{\Delta x}$, has to be estimated and applied to the linearization point $\hat{\dot{\boldsymbol{x}}} = (\hat{\dot{x}}_u, \hat{\dot{y}}_u, \hat{\dot{z}}_u, \hat{\dot{b}}_u)$, given that:

$$\begin{aligned}
\dot{x}_u &= \hat{\dot{x}}_u + \Delta\dot{x}_u \\
\dot{y}_u &= \hat{\dot{y}}_u + \Delta\dot{y}_u \\
\dot{z}_u &= \hat{\dot{z}}_u + \Delta\dot{z}_u \\
\dot{b}_u &= \hat{\dot{b}}_u + \Delta\dot{b}_u
\end{aligned} \tag{1.29}$$

Doppler measurement equation (1.26) can then be re-written substituting (1.29) in, while the nominal Doppler measurement equation can still be retrieved from (1.26), substituting the linearization point $\hat{\dot{\boldsymbol{x}}}$ in, obtaining:

$$\begin{gathered}
d_j = (\hat{\dot{x}}_u + \Delta\dot{x}_u)a_{j,x} + (\hat{\dot{y}}_u + \Delta\dot{y}_u)a_{j,y} + (\hat{\dot{z}}_u + \Delta\dot{z}_u)a_{j,z} - (\hat{\dot{b}}_u + \Delta\dot{b}_u) \\
\hat{d}_k = \hat{\dot{x}}_u a_{j,x} + \hat{\dot{y}}_u a_{j,y} + \hat{\dot{z}}_u a_{j,z} - \hat{\dot{b}}_u
\end{gathered} \tag{1.30}$$

Taking the difference of the two expressions in (1.30), the following expression can be written:

$$\Delta d_j = \Delta\dot{x}_u a_{j,x} + \Delta\dot{y}_u a_{j,y} + \Delta\dot{z}_u a_{j,z} - \Delta\dot{b}_u \tag{1.31}$$

where $\Delta d_j = d_j - \hat{d}_j$. Recalling the definition of d_j (1.25), it can be noticed that the first term is equal to the opposite of the pseudorange-rate (it holds:

$\dot{\rho}_j = -\dfrac{c(f_{u,j} - f_{c,j})}{f_{c,j}}$). Then, taking the difference of d_j and $\hat{d}_j$, the satellite velocity is simplified, obtaining $\Delta d_j = -\Delta\dot{\rho}_j = \dot{\rho}_j - \hat{\dot{\rho}}_j$ (from now on the pseudorange-rate $\dot{\rho}_j$ will be considered as the GNSS observable, together with the pseudorange ρ_j). Instead, $\hat{\dot{\rho}}_j$ can be computed from the linearization point $\hat{\dot{\boldsymbol{x}}}$ as [19]:

$$\hat{\dot{\rho}}_j = (\boldsymbol{v}_j - \hat{\dot{\boldsymbol{x}}}) \cdot \boldsymbol{a}_j + \hat{\dot{b}}_u \tag{1.32}$$

Again, compacting all of these N_s (number of satellites) equations in matrix form and having $\Delta\dot{\rho}_j = \hat{\dot{\rho}}_j - \dot{\rho}_j$, it yields:

$$\boldsymbol{\Delta\dot{\rho}} = \boldsymbol{H\Delta\dot{x}} \tag{1.33}$$

from which the estimation of the velocity correction to be applied to the linearization point can be obtained as:

$$\boldsymbol{\Delta\dot{x}} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{\Delta\dot{\rho}} \tag{1.34}$$

1.4 Inertial Navigation System

In this section an overview about Inertial Navigation Systems (INS) will be provided, covering the mathematical foundations for explaining the principles of navigation systems and their integration, leading to the estimation of position, velocity and attitude, as well as the overall architecture structure.

1.4.1 Inertial Navigation System architecture

An Inertial navigation system (INS) is a navigation system based on the effect of gravity, hence it is governed by Newton laws of physics. It involves a blend of inertial measurements, mathematics, control system design and geodesy. It comprises a set of inertial sensors, known as an inertial measurement unit (IMU), together with a navigation processor. These sensors will measure accelerations generated on the object by external forces and integrating these, the change in velocity and then in position can be retrieved [20]. A stand-alone INS will then provide just relative information about position and velocity, as consecutive variations of these quantities. Hence, it can provide an absolute solution only if some a-priori information, together with an initial reference frame, are available. This concept is known as Dead-Reckoning [21]. As a consequence of this, the errors affecting an inertial navigation solution grow with time, due to the consecutive summations in the Dead-Reckoning method, as in Figure 1.7.

A conventional IMU consists of three gyroscopes for measuring angular rates and three accelerometers for measuring accelerations. They are mounted in triad,

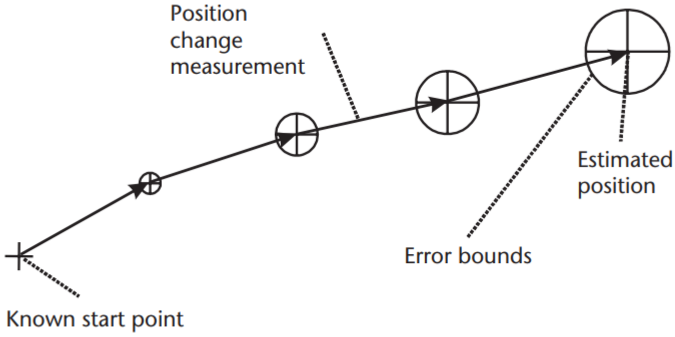


Figure 1.7: Dead-Reckoning approach [21].

respectively, so that their three axes are mutually orthogonal, setting up a Cartesian reference frame.

Note that accelerometers, as they are conceived, will sense the specific force (i.e. the non-gravitational force per unit mass on a body, sensed with respect to an inertial frame.) at which they are subject to. For example, a non-rotating accelerometer which is in free fall, and then subject only to gravity, will sense nothing. On the contrary, under zero acceleration, it will sense the gravitation, and the measured specific force will be equal and opposite to the acceleration due to gravitation. The used term *gravitation*, that will be indicated as γ, is the fundamental mass attraction force; it does not incorporate any centripetal components. Moreover, in the e-frame, a stationary accelerometer will sense also the centrifugal acceleration due to the earth rotation. The composition of gravitation and centrifugal acceleration is the so called *gravity*, hereinafter indicated as $\boldsymbol{g}$ (usually approximated to $9.81ms^{-2}$). For this reason, when using an accelerometer, the user must know the attitude of the accelerometer itself in order to compensate the effect of gravity. Hence the use of gyroscopes, which can sense the angular rates with respect to a reference frame, such as in an ECEF frame, is needed. Figure 1.8 shows a typical strapdown inertial sensors assembly for an IMU The term *strapdown* is used to refer to an IMU-configuration in which the inertial sensors are directly embedded in the body and rotate with it. Nowadays sensors as gyros and accelerometers can be found in a wide range of different designs, varying in cost, size and performance. In this thesis the adopted gyros and accelerometers are manufactured using micro-electromechanical systems (MEMS) technology offering the advantages of low cost, size, and mass, and a high shock tolerance, but give relatively poor performance. They can be included in the automotive-grade and are typically employed in the mass-market sector [21].

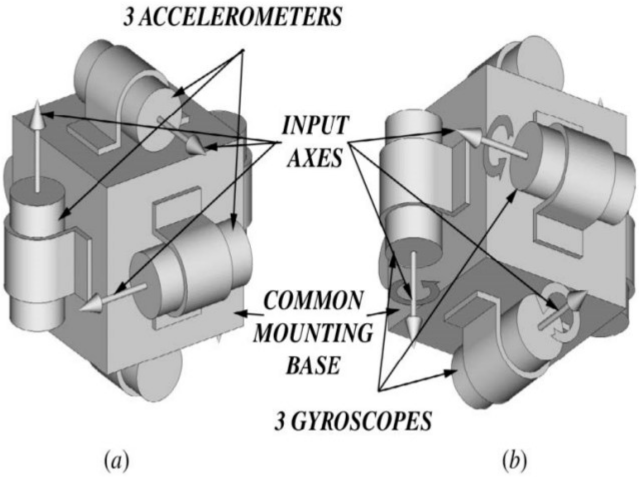


Figure 1.8: IMU assembly for of strapdown inertial sensors [22].

1.4.2 Reference frames

Here, the reference frames that will be adopted in the following discussion are presented, as well as the mathematical set up to apply coordinates-transformation from one frame to another [20].

Inertial Frame (i-frame)

Given a frame, it is considered to be inertial if it is non-rotating and non-accelerating with respect to far-off stars, at least as far as the accuracy of the measurement instruments used can sense. The definition of the i-frame is then the following;
Origin : Earth's centre of mass
Z-Axis : Parallel to the Earth's instantaneous spin axis
X-Axis : Pointing towards the mean equinoctial colure in the equatorial plane
Y-Axis : Orthogonal to the X and Z axes to complete a right-handed frame

Earth Centred Earth Fixed Frame (ECEF or e-frame)

The Earth-fixed frame is defined as
Origin : Earth's centre of mass
Z-Axis : Parallel to the Earth's mean spin axis
X-Axis : Pointing towards the mean meridian of Greenwich
Y-Axis : Orthogonal to the X and Z axes to complete a right-handed frame

Body Frame (b-frame)

The body frame represents the orientation of the IMU axes. In a strapdown inertial system as used herein, the IMU is rigidly mounted to the vehicle and thus can have arbitrary orientation. For convenience, the body frame (i.e. IMU axes) is assumed to be aligned with the frame of the vehicle with the following convention. An example is given in Figure 1.9
Origin : Centre of IMU
X-Axis (Roll) : Pointing towards the front of the vehicle
Y-Axis (Pitch) : Pointing towards the right of the vehicle
Z-Axis (Yaw) : Orthogonal to the X and Y axes to complete a right-handed frame.

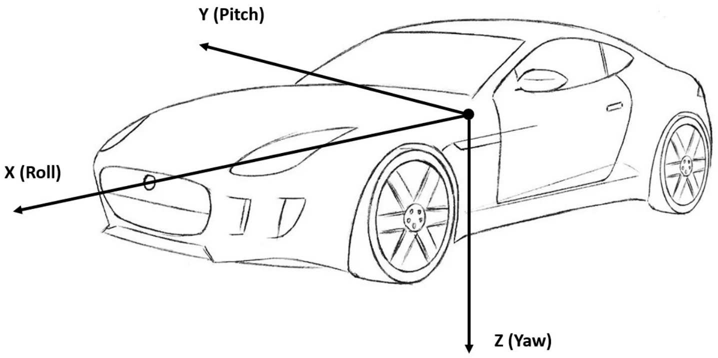


Figure 1.9: IMU axes aligned with vehicle body frame.

Coordinate Transformation Matrix

There are different ways to pass from one frame to another (e.g. quaternions, Euler angles etc.). In this thesis, a matrix representation of this type of transformation is adopted, called coordinate transformation matrix. The coordinate transformation matrix (also called Direction Cosine Matrix (DCM)) is a 3 × 3 matrix, indicated as $\boldsymbol{C}$, used to change vector representation coordinates from one frame to another. The lower index represents the from-coordinate frame and the upper index the to-frame (i.e. $\boldsymbol{C}_{from-frame}^{to-frame}$). They are obtained expressing the versors defining the final frame axes in the starting frame [21]. For example, given a vector $\boldsymbol{v}$ expressed in the starting frame $1-frame$, that will be indicated as $\boldsymbol{v}^1$, the coordinate transformation of vector $\boldsymbol{v}$ from $1-frame$ to $2-frame$, called $\boldsymbol{v}^2$, can be done as follows:

$$\boldsymbol{v}^2 = \boldsymbol{C}_1^2 \boldsymbol{v}^1 \tag{1.35}$$

note that, as a property of DCM matrices, $\boldsymbol{C}_1^2 = \boldsymbol{C}_2^{1^T}$.

1.4.3 Strapdown Mechanization equations

The navigation processor, given some inertial sensor measurements, proceed with the computation of the inertial position, velocity and attitude. To do so, a system of equations, called mechanization equations, are used. Here, they will be first represented in continuous time form. This dissertation makes reference to [21, 23, 20].

Attitude computation

Given the gyro-measurements vector ω_{gyro}^b, where b indicates the $b - frame$, first the estimated earth angular velocity $\omega_{i,e}^e$ (where the subscript i, e indicates that it is the $e - frame$ velocity computed w.r.t. the $i - frame$) has to be subtracted, hence:

$$\boldsymbol{\omega}_{e,b}^b = \boldsymbol{\omega}_{gyro}^b - \boldsymbol{C}_b^{eT} \boldsymbol{\omega}_{i,e}^e \tag{1.36}$$

obtaining then the angular velocity of the $b-frame$ w.r.t. the $e-frame$, expressed in the $b - frame$. The matrix $\boldsymbol{C}_b^{eT}$ is still an unknown in equation (1.36), but it can be described using the differential equation:

$$\dot{\boldsymbol{C}}_b^e = \boldsymbol{C}_b^e \boldsymbol{\Omega}_{e,b}^b \tag{1.37}$$

where $\boldsymbol{\Omega}_{e,b}^b$ is the skew-symmetric matrix of $\boldsymbol{\omega}_{e,b}^b$ which is described by equation (1.36).

Velocity computation

First of all, as anticipated before, an accelerometer will measure, referring to an inertial frame $i - frame$, the specific force $\boldsymbol{f}$, that is the acceleration at which it is subject to ($\boldsymbol{a}_{i,b}$) minus the gravitational acceleration $\boldsymbol{\gamma}$. Hence, expressing them in the $e - frame$:

$$\boldsymbol{f}^e = \boldsymbol{C}_b^e \boldsymbol{f}^b = \boldsymbol{a}_{i,b}^e - \boldsymbol{\gamma}^e \tag{1.38}$$

that can be re-written in terms of $\boldsymbol{a}_{i,b}^e$:

$$\boldsymbol{a}_{i,b}^e = \boldsymbol{C}_b^e \boldsymbol{f}^b + \boldsymbol{\gamma}^e \tag{1.39}$$

instead, the gravity vector $\boldsymbol{g}$ in the $e - frame$ is expressed as:

$$\boldsymbol{g}^e = \boldsymbol{\gamma}^e - \boldsymbol{\omega}_{i,e}^e \times [\boldsymbol{\omega}_{i,e}^e \times \boldsymbol{r}_{e,b}^e] \tag{1.40}$$

where $\boldsymbol{\gamma}^e$ is the gravitational acceleration and the term $\boldsymbol{\omega}_{i,e}^e \times [\boldsymbol{\omega}_{i,e}^e \times \boldsymbol{r}_{e,b}^e]$ is the centrifugal acceleration.

Then, writing the expression of the body velocity with respect to the $e-frame$, $\boldsymbol{v}_{e,b}$, expressed in the $e-frame$, it yields:

$$\boldsymbol{v}^e_{e,b} = \boldsymbol{v}^e_{i,b} - \boldsymbol{\omega}^e_{i,e} \times \boldsymbol{r}^e_{e,b} \tag{1.41}$$

where $\boldsymbol{r}_{e,b}$ is the position of the body w.r.t the $e-frame$. Differentiating:

$$\boldsymbol{a}^e_{e,b} = \boldsymbol{a}^e_{i,b} - 2\boldsymbol{\omega}^e_{i,e} \times \boldsymbol{v}^{\boldsymbol{e}}_{\boldsymbol{e,b}} - \boldsymbol{\omega}^e_{i,e} \times [\boldsymbol{\omega}^e_{i,e} \times \boldsymbol{r}^e_{e,b}] \tag{1.42}$$

To obtain this equation (1.42), the earth rotation $\boldsymbol{\omega}_{i,e}$ was considered constant. Finally, substituting (1.39) and (1.40) in (1.42)

$$\boldsymbol{a}^e_{e,b} = \boldsymbol{C}^e_b \boldsymbol{f}^b - 2\boldsymbol{\omega}^e_{i,e} \times \boldsymbol{v}^e_{e,b} + \boldsymbol{g}^e \tag{1.43}$$

The term $2\boldsymbol{\omega}^e_{i,e} \times \boldsymbol{v}^e_{e,b}$ indicates the Coriolis acceleration.

The relationship between the acceleration $\boldsymbol{a}^e_{e,b}$ and the velocity $\boldsymbol{v}^e_{e,b}$ is:

$$\dot{\boldsymbol{v}}^e_{e,b} = \boldsymbol{a}^e_{e,b} \tag{1.44}$$

substituting (1.43) into (1.44) and integrating, the body velocity $\boldsymbol{v}^e_{e,b}$ with respect to the $e-frame$, expressed in the $e-frame$ itself, can be computed.

Position computation

Once the velocity $\boldsymbol{v}^e_{e,b}$ is obtained, it is trivial to compute the position $\boldsymbol{r}^e_{e,b}$ since:

$$\dot{\boldsymbol{r}}^e_{e,b} = \boldsymbol{v}^e_{e,b} \tag{1.45}$$

hence a second integration is needed.

1.4.4 Attitude and position computation

Obviously the equations previously showed, even though they are theoretically sufficient to describe the system evolution, are not suitable for a software implementation. There are several methods to handle these equations by numerical integration. Three integration functions are involved, attitude, velocity and, consequentially, position, and the inherent non-linearities require these integrations to occur at very high rates. To minimize the system computational requirements, most inertial navigation systems output what are known as coning and sculling integrals which are integrated internally and can then be used at lower rates for full state integration. This approach is commonly referred to as: two-speed integration algorithm. It is composed by a more complex exact algorithm for moderate speed updating fed by a simplified high-speed algorithm. The high-speed algorithm contains a simple summing operation of angular rate and specific force sensors inputs, plus an approximate coning and sculling motion integration function. Under conditions of constant angular rate and specific force vectors (i.e., zero-coning and zero-sculling), the coning and sculling terms become zero and the simple summing operation becomes analytically exact. In the high-speed algorithm, the coning integral refers to the non-linear high-rate contribution in the attitude update, while the sculling integral to the non-linear high-rate velocity update contribution. While, at lower speed, the integration of the velocity and angular rate outputs is performed, reducing the amount of bandwidth needed to process the data. With this approach, given an arbitrary time step size, the algorithm can provide orientation, velocity and position solution with higher accuracy with respect to the computations done just at medium-rate. Moreover the two-speed algorithm provides a lower computational complexity with respect to a one-speed algorithm run at higher rate, without loosing in accuracy. For a more in-depth discussion and technical details, refer to [24, 25].

Chapter 2

Integration of Inertial Navigation and Satellite-based positioning

There are several sensors which can be employed for positioning and navigation, leading to a different formulation of the PVT problem. Different sensors are characterized by different properties, leading to advantages and disadvantages of one over the other. The integration of inertial and satellite navigation systems arises exactly for this reason: coupling different sensors in order to exploit their complementary characteristics, emphasizing strengths and minimizing weaknesses.

2.1 Complementary nature of GNSS and INS

In case of GNSS and INS, their integration can reach superior system performance with respect to either system in a stand-alone mode. In fact their characteristics, showed in the previous chapter and here summarized, make them suitable for integration. INS is characterized by a relatively high rate in delivering navigation information (e.g. 100 Hz), while GNSS, in a low cost receiver, provides information at lower rate (e.g. 1 Hz) [26]. However, IMU are intrinsically affected by deterministic errors, such as bias, and stochastic errors, such as bias drift or noise. Even if a stochastic modelling describing these errors can be formulated, it is not possible to completely remove them. Hence, due to the integrations needed to retrieve the solution, these errors are integrated too, causing the level of accuracy to decrease over time, as explained in Section 1.4.1. On the contrary, GNSS offers a good long-term stability, involving errors effectively time invariant with homogeneous accuracy (bounded within few meters). Moreover, while INS will provide information in any condition, being self-contained, GNSS is strongly affected by environmental conditions: it needs at least four satellites in view to provide a PVT solution and the RF-signals are subject to reflection/diffraction as well as degradation due to the atmospheric layers (troposphere and ionosphere) [20].

In GNSS/INS integration, GNSS can update the inertial navigation solution avoiding it to drift in the long term; while INS can be used to interpolate GNSS trajectory and bridge outages, allowing a fine tracking of the body dynamics. This approach provides an overall improving of the navigation system performance in terms of accuracy, continuity and reliability.

In the following, an overview of different possible approaches for implementing a GNSS/INS integration is given, as well as a deeper insight on the technique that has been adopted in this thesis.

2.2 Integration architectures

As anticipated, there are different approaches that can be used to perform GNSS/INS integration. Their main difference is the amount of information the two systems are meant to share and at which the information is fused together. These are the most common ones [27]:

- Uncoupled integration
- Loose integration
- Tight integration
- Ultra-tight integration

The uncoupled integration is the integration with the minimum amount of information shared by the two systems. In this architecture, GNSS and INS elaborate satellite observables and IMU measurements independently. Once GNSS receiver output a PVT solution, it is used to initialize again the INS Dead-Reckoning algorithm, so that the accumulated error is reset to zero. However the INS error increasing rate is not affected (i.e. there is no feedback of estimated INS errors into the navigation algorithm), leading to a rapid decay of the solution accuracy between two consecutive GNSS epochs (even worse in presence of GNSS outages) [28, 29, 30]. The Ultra-tight integration instead is the deepest possible integration between the two systems. In fact in this setup the GNSS solutions are used to re-calibrate the INS, which is used directly inside the GNSS receiver: it allows maintaining tracking loops even when GNSS outages occur or in high dynamics applications. Unfortunately, for this implementation, the GNSS receiver source code should be accessible but it is usually available only for receiver manufacturers. That being said, a more detailed discussion about the two remaining solutions is given in the following sections [13, 30, 31].

2.2.1 Loose integration

The Loosely-coupled hybridization strategy, with respect to the uncoupled one, has some information shared between the independent navigation units. Here, as the uncoupled one, the GNSS unit independently process the satellites signals in order to obtain an estimation of user position and velocity, as well as the INS unit. These are given as input to a navigation filter, which will provide a blended navigation solution. The navigation filter, additionally, provides feedback information about biases affecting inertial sensors, allowing a mitigation of the IMU errors drift [13, 32].

With respect to architectures with deeper integrations, this solution is characterized by a lower level of complexity, being the integration performed at high-level and having the navigation state with fewer dimensions. Hence also the processing time will be shorter. Anyway some drawbacks are present. Having the navigation filter independent from the GNSS unit implies that it is unaware of statistical properties characterizing the navigation state, leading to sub-optimal solutions. Two different filters (INS-filter and navigation filter) also imply that process noise must be added twice. Moreover, all the information acquired by the receiver are lost whenever it is not able to provide a navigation solution (e.g. not enough satellites are in view) [33].

A schematic representation of loose integration architecture is showed in Figure 2.1.

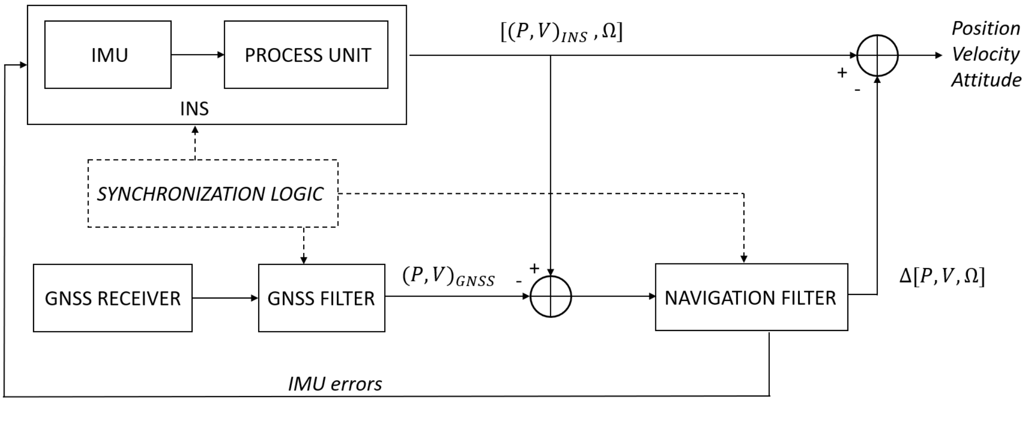


Figure 2.1: INS/GNSS loose integration.

2.2.2 Tight integration

In this architecture, the noisy ranging observables (pseudorange and pseudorange-rate), obtained by the GNSS receiver, are not processed by an independent GNSS-filter but are directly passed to a GNSS/INS common filter. This filter then will process the raw satellite measurements along with the position, velocity and attitude updates provided at high-rate by the INS process unit mechanization. Hence, in this approach, the inertial system is providing predictions on position, velocity and attitude at high-rate. These estimates are used, along with GNSS satellites position and velocity, obtained from the ephemerides, to provide also the nominal pseudorange and pseudorange-rate measurements to the navigation filter. Once the navigation filter is fed with the observed pseudorange and pseudorange-rate from the GNSS receiver too, the difference with the nominal ones is used to refine the INS solution as well as the bias affecting the IMU sensors [13, 27, 32]. The block-diagram for a tight integration is provided in Figure 2.2.

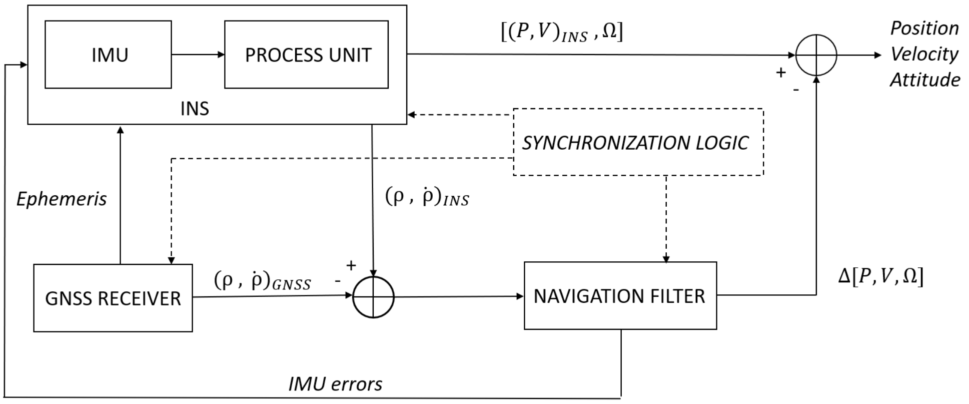


Figure 2.2: INS/GNSS tight integration ($(\rho, \dot{\rho})$ refer to pseudorange and pseudorange-rate respectively).

Tight integrations bring some advantages with respect to the loose one. As anticipated previously, since GNSS observables are fed directly to the navigation filter, those measurements are here elaborated with a statistical characterization of noise sources and modelling of states correlation, resulting in a better accuracy in the solution [13, 33, 34]. Moreover in this approach there is no constraint on the minimum satellites in view needed. In fact, even if less then four satellites are available, the GNSS receiver provides anyway pseudoranges and pseudorange-rates to the navigation filter, which process them with the information given by the INS unit. This characteristic provides robustness and continuity to the overall system in challenging environments such as urban canyons or high-dynamics applications. Finally the process noise is applied only once to the navigation filter, differently from the loose integration. As the loose solution brought with itself the advantage of a lower design complexity and a lower number of states, this architecture instead has the disadvantage of greater computational load and enhanced system complexity [34].

2.3 Tight integration framework

This thesis will exploit the GNSS/INS tight integration, hence that will be the structure considered hereinafter. In this section, the mathematical formulation of a discrete-time state space model, able to handle both GNSS observables and INS dynamics in an integrated environment, will be given. The final part will cover details about the actual software implementation, managing the flow of data coming from GNSS and INS unit.

2.3.1 INS/GNSS state-space model

As discussed previously, in a tight integration the centralized filter has to handle measurements coming from the INS and GNSS units and, through some filtering algorithm, provide corrections to the navigation state. To do so, first an appropriate state-space model formulation is needed. Moreover it has to include the mathematical formulation of the navigating body inertial motion dynamics as well as the model converting the GNSS-observables into a useful aid, in order to refine the system navigation state.

There are two different navigation modes for implementing a GNSS/INS integration unit that can be found in literature: direct and indirect navigation mode. In the direct navigation mode the absolute body position, velocity and attitude together with the biases affecting INS sensors are estimated by the filter. While in the indirect navigation mode it is the error-state to be estimated and, consequently, applied to the corresponding total state quantities, refining their predictions. The indirect (i.e. error-based) mode will be the one adopted in this thesis to develop the navigation filter. First the state-space model will be formulated in continuous-time (as for INS mechanization), then the discrete-time form, needed to be implemented in the software, will be presented.

The error-state $\boldsymbol{\delta x}$, vector of dimension 17, can be defined as [32]:

$$\begin{aligned}\boldsymbol{\delta x^e} = [&\delta r_x^e\ \delta r_y^e\ \delta r_z^e\ \delta v_x^e\ \delta v_y^e\ \delta v_z^e\ \delta \epsilon_x^e\ \delta \epsilon_y^e\ \delta \epsilon_z^e \\ &\delta b_{a,x}^b\ \delta b_{a,y}^b\ \delta b_{a,z}^b\ \delta b_{g,x}^b\ \delta b_{g,y}^b\ \delta b_{g,z}^b\ \delta b_u\ \delta \dot{b}_u]^T \end{aligned} \tag{2.1}$$

where the superscripts e and b refer, as in the previous chapter, to the ECEF-frame ($e - frame$) and body-frame ($b - frame$) respectively. The terms in (2.1) are listed below:

- $\boldsymbol{\delta r}^e = [\delta r_x^e \; \delta r_y^e \; \delta r_z^e]^T$ is the position error vector.
- $\boldsymbol{\delta v}^e = [\delta v_x^e \; \delta v_y^e \; \delta v_z^e]^T$ is the velocity error vector.
- $\boldsymbol{\delta \epsilon}^e = [\delta \epsilon_x^e \; \delta \epsilon_y^e \; \delta \epsilon_z^e]^T$ is the misalignment error vector.
- $\boldsymbol{\delta b}_a^b = [\delta b_{a,x}^b \; \delta b_{a,y}^b \; \delta b_{a,z}^b]^T$ is the accelerometer bias error vector.
- $\boldsymbol{\delta b}_g^e = [\delta b_{g,x}^b \; \delta b_{g,y}^b \; \delta b_{g,z}^b]^T$ is the gyroscope bias error vector.
- $\boldsymbol{\delta b}_u = [\delta b_u \;\; \delta \dot{b}_u]^T$ is the error vector collecting receiver clock bias δb_u and receiver clock drift $\delta \dot{b}_u$ corrections.

Note that the vector $\boldsymbol{\epsilon}^e$ is directly related to the rotation matrix C_b^e used in Section 1.4. In fact, the vector collects Roll (that refers to the $x - axis$), Pitch (that refers to the $y - axis$) and Yaw (that refers to the $z - axis$) angles, which define the orientation of the $b - frame$ with respect to the $e - frame$. Choosing some convention (intrinsic/extrinsic rotation and angles order), the matrix C_b^e can be obtained as a non-linear combination of these angles. The three chosen angles would be sufficient to described the body orientation and are, in fact, included in the state vector. Anyway, the matrix representation (even if redundant, having 9 entries instead of 3) is more suitable for mathematical manipulations (also quaternions could have been chosen at this regard) [35].

2.3.2 System process model

How the sensor errors affect the navigation state is described by the error-dynamics equations. These equations are derived by applying a differential operator to the navigation equations. That is, the variables of the navigation equations are perturbed differentially and the differentials are then interpreted as small differences, or errors. The following discussion makes reference to [32, 29]

INS error-dynamics equations

In Section 1.4.3 the following differential equations were obtained (assuming the same conventions of the previous chapter):

$$\dot{\boldsymbol{C}}_b^e = \boldsymbol{C}_b^e \boldsymbol{\Omega}_{e,b}^b \tag{2.2}$$

$$\boldsymbol{a}_{e,b}^e = \boldsymbol{C}_b^e \boldsymbol{f}^b - 2\boldsymbol{\omega}_{i,e}^e \times \boldsymbol{v}_{e,b}^e + \boldsymbol{g}^e \tag{2.3}$$

$$\dot{\boldsymbol{r}}_{e,b}^e = \boldsymbol{v}_{e,b}^e \tag{2.4}$$

taking equation (2.4) and differentiating, it is simply:

$$\boldsymbol{\delta}\dot{\boldsymbol{r}}^e_{e,b} = \boldsymbol{\delta}\boldsymbol{v}^e_{e,b} \tag{2.5}$$

instead, differentiating (2.2):

$$\boldsymbol{\delta}\dot{\boldsymbol{C}}^e_b = \boldsymbol{\delta}\boldsymbol{C}^e_b\boldsymbol{\Omega}^b_{e,b} + \boldsymbol{C}^e_b\boldsymbol{\delta}\boldsymbol{\Omega}^b_{e,b} \tag{2.6}$$

the term $\boldsymbol{\delta}\boldsymbol{C}^e_b$ represent the perturbation from the exact rotation $\boldsymbol{C}^e_b$. Given the relatively small rotation angles $\boldsymbol{\delta}\boldsymbol{\epsilon}^e$, we can represent the perturbed rotation $\hat{\boldsymbol{C}}^e_b$ as the composition of the exact rotation and the perturbation. Mathematically:

$$\hat{\boldsymbol{C}}^e_b = (\boldsymbol{I} - S(\boldsymbol{\delta}\boldsymbol{\epsilon}^e))\boldsymbol{C}^e_b \tag{2.7}$$

where $S(\cdot)$ is the skew-symmetric operator and $(\boldsymbol{I} - S(\boldsymbol{\delta}\boldsymbol{\epsilon}^e))$ is expression for infinitesimal rotations. Hence the perturbation can be written as the difference between the perturbed rotation and the exact one:

$$\boldsymbol{\delta}\boldsymbol{C}^e_b = \boldsymbol{C}^e_b - (\boldsymbol{I} - S(\boldsymbol{\delta}\boldsymbol{\epsilon}^e))\boldsymbol{C}^e_b = -S(\boldsymbol{\delta}\boldsymbol{\epsilon}^e)\boldsymbol{C}^e_b \tag{2.8}$$

taking the derivative with respect to time of (2.8):

$$\boldsymbol{\delta}\dot{\boldsymbol{C}}^e_b = -S(\boldsymbol{\delta}\dot{\boldsymbol{\epsilon}}^e)\boldsymbol{C}^e_b - S(\boldsymbol{\delta}\boldsymbol{\epsilon}^e)\boldsymbol{C}^e_b\boldsymbol{\Omega}^b_{e,b} \tag{2.9}$$

where the equation (2.2) has been substituted in.

Equaling the right-hand side of equations (2.6) and (2.9):

$$-S(\boldsymbol{\delta}\dot{\boldsymbol{\epsilon}}^{\boldsymbol{e}})\boldsymbol{C}^e_b - S(\boldsymbol{\delta}\boldsymbol{\epsilon}e)\boldsymbol{C}^e_b\boldsymbol{\Omega}^b_{e,b} = \boldsymbol{\delta}\boldsymbol{C}^e_b\boldsymbol{\Omega}^b_{e,b} + \boldsymbol{C}^e_b\boldsymbol{\delta}\boldsymbol{\Omega}^b_{e,b} \tag{2.10}$$

substituting (2.8) into (2.10):

$$-S(\boldsymbol{\delta}\dot{\boldsymbol{\epsilon}}^{\boldsymbol{e}})\boldsymbol{C}^e_b + \boldsymbol{\delta}\boldsymbol{C}^e_b\boldsymbol{\Omega}^b_{e,b} = \boldsymbol{\delta}\boldsymbol{C}^e_b\boldsymbol{\Omega}^b_{e,b} + \boldsymbol{C}^e_b\boldsymbol{\delta}\boldsymbol{\Omega}^b_{e,b} \tag{2.11}$$

and solving with respect to $S(\boldsymbol{\delta}\dot{\boldsymbol{\epsilon}}^{\boldsymbol{e}})$, the following expression is obtained:

$$S(\boldsymbol{\delta}\dot{\boldsymbol{\epsilon}}^{\boldsymbol{e}}) = -\boldsymbol{C}^e_b\boldsymbol{\delta}\boldsymbol{\Omega}^b_{e,b}\boldsymbol{C}^b_e \tag{2.12}$$

that can be re-written in vector form as:

$$\boldsymbol{\delta}\dot{\boldsymbol{\epsilon}}^{\boldsymbol{e}} = -\boldsymbol{C}^e_b\boldsymbol{\delta}\boldsymbol{\omega}^b_{e,b} \tag{2.13}$$

Recalling that $\boldsymbol{\delta}\boldsymbol{\omega}^b_{e,b}$ was described in Section 1.4.3 by the equation (1.36):

$$\boldsymbol{\omega}^b_{e,b} = \boldsymbol{\omega}^b_{gyro} - \boldsymbol{C}^{eT}_b\boldsymbol{\omega}^e_{i,e} \tag{2.14}$$

it can be differentiated to obtain:

$$\boldsymbol{\delta\omega}^b_{e,b} = \boldsymbol{\delta\omega}^b_g - \boldsymbol{C}^{eT}_b S(\boldsymbol{\delta\epsilon})\boldsymbol{\omega}^e_{i,e} - \boldsymbol{C}^{eT}_b \boldsymbol{\delta\omega}^e_{i,e} \tag{2.15}$$

where the equality $\boldsymbol{\delta C}^{eT}_b = \boldsymbol{C}^{eT}_b S(\boldsymbol{\delta\epsilon})$ (that is the transposed of the equation (2.8)) has been used.

Finally, substituting (2.15) into (2.13), assuming the earth rotation as a constant ($\boldsymbol{\delta\omega}^e_{i,e} = 0$) and after some manipulation:

$$\dot{\boldsymbol{\delta\epsilon}}^e = -\boldsymbol{C}^e_b\boldsymbol{\delta\omega}^b_g - \boldsymbol{\Omega}^e_{i,e}\boldsymbol{\delta\epsilon} \tag{2.16}$$

Instead, differentiating equation (2.3) (assuming again $\boldsymbol{\delta\omega}^e_{i,e} = 0$):

$$\boldsymbol{\delta a}^e_{e,b} = \boldsymbol{\delta\dot{v}}^e_{e,b} = \boldsymbol{\delta C}^e_b\boldsymbol{f}^b + \boldsymbol{C}^e_b\boldsymbol{\delta f}^b - 2\boldsymbol{\Omega}^e_{i,e}\boldsymbol{\delta v}^e_{e,b} + \boldsymbol{\delta g}^e \tag{2.17}$$

Assuming that $\boldsymbol{\delta g}^e = \boldsymbol{N}\boldsymbol{\delta r}^e_{e,b}$, meaning that the perturbation of the gravity vector $\boldsymbol{g}^e$ is due to the displacement $\boldsymbol{\delta r}^e_{e,b}$ mapped into the gravity domain by the tensor $\boldsymbol{N}$, and recalling that a skew-symmetric matrix times a vector is associated to the cross-product (hence its properties are valid), the following equation is obtained:

$$\boldsymbol{\delta\dot{v}}^e_{e,b} = S(\boldsymbol{C}^e_b\boldsymbol{f}^b)\boldsymbol{\delta\epsilon} + \boldsymbol{N}\boldsymbol{\delta r}^e_{e,b} - 2\boldsymbol{\Omega}^e_{i,e}\boldsymbol{\delta v}^e_{e,b} + \boldsymbol{C}^e_b\boldsymbol{\delta f}^b \tag{2.18}$$

The equations (2.5), (2.16) and (2.18) can be compacted in matrix form as:

$$\begin{bmatrix} \boldsymbol{\delta\dot{r}}^e_{e,b} \\ \boldsymbol{\delta\dot{v}}^e_{e,b} \\ \boldsymbol{\delta\dot{\epsilon}}^e \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{N} & -2\boldsymbol{\Omega}^e_{i,e} & \boldsymbol{F} \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{\Omega}^e_{i,e} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta r}^e_{e,b} \\ \boldsymbol{\delta v}^e_{e,b} \\ \boldsymbol{\delta\epsilon}^e \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{C}^e_b\boldsymbol{\delta f}^b \\ -\boldsymbol{C}^{eT}_b\boldsymbol{\delta\omega}^b_g \end{bmatrix} \tag{2.19}$$

where $\boldsymbol{F} = S(\boldsymbol{C}^e_b\boldsymbol{f}^b)$.

In the INS model (2.19), the terms $\boldsymbol{\delta f}^b$ and $\boldsymbol{\delta\omega}^b_g$ represent the inertial sensors error-vectors. This model would fit in the navigation filter if those errors were characterized by a zero-mean normal distribution. However the inertial sensors are also affected by deterministic errors (i.e. biases). Hence, they are modeled with a first-order approximation as sum of a time-varying bias component ($\boldsymbol{\delta b}^b_a$) and a noise component ($\boldsymbol{\delta b}^b_g$) as in (2.20)

$$\begin{aligned} \boldsymbol{\delta f}^b &= \boldsymbol{b}^b_a + \boldsymbol{\omega}^b_f \\ \boldsymbol{\delta\omega}^b_g &= \boldsymbol{\delta b}^b_g + \boldsymbol{\omega}^b_\omega \end{aligned} \tag{2.20}$$

The bias components errors ($\boldsymbol{\delta b}^b_a$) and ($\boldsymbol{\delta b}^b_g$) are included in the state error-vector (as showed in (2.1)) in order to estimate and compensate them. Given this sensor bias error-model, the INS error-dynamics model (2.19) can be re-written as:

$$\begin{bmatrix} \boldsymbol{\delta}\dot{\boldsymbol{r}}^{e}_{e,b} \\ \boldsymbol{\delta}\dot{\boldsymbol{v}}^{e}_{e,b} \\ \boldsymbol{\delta}\dot{\boldsymbol{\epsilon}}^{e} \\ \boldsymbol{\delta}\dot{\boldsymbol{b}}^{b}_{a} \\ \boldsymbol{\delta}\dot{\boldsymbol{b}}^{b}_{g} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{N} & -2\boldsymbol{\Omega}^{e}_{i,e} & \boldsymbol{F} & \boldsymbol{C}^{e}_{b} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{\Omega}^{e}_{i,e} & \boldsymbol{0} & \boldsymbol{C}^{e}_{b} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -diag(\boldsymbol{\alpha}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -diag(\boldsymbol{\beta}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta r}^{e}_{e,b} \\ \boldsymbol{\delta v}^{e}_{e,b} \\ \boldsymbol{\delta \epsilon}^{e} \\ \boldsymbol{\delta b}^{b}_{a} \\ \boldsymbol{\delta b}^{b}_{g} \end{bmatrix} + \\ + \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{C}^{e}_{b} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}^{e}_{b} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}^{b}_{f} \\ \boldsymbol{w}^{b}_{\omega} \\ \boldsymbol{w}^{b}_{a} \\ \boldsymbol{w}^{b}_{g} \end{bmatrix} \tag{2.21}$$

where $\boldsymbol{w}^{b}_{a}$ and $\boldsymbol{w}^{b}_{g}$ are driving noises, included to model the bias drift as an appropriate stochastic process (first-order Gauss-Markov process in this case), while $diag(\boldsymbol{\alpha})$ and $diag(\boldsymbol{\beta})$ are diagonal matrices of time constants for the accelerometer and gyro bias models respectively.

GNSS/INS error-dynamics equations

In order to complete the dynamics error-model, the receiver clock bias δb_u and the receiver clock drift $\delta\dot{b}_u$ have to be included in the error-state vector. Considering the vector $\boldsymbol{\delta b}_u = [\delta b_u \delta\dot{b}_u]$, the dynamics of these quantities is simply written as:

$$\boldsymbol{\delta}\dot{\boldsymbol{b}}_u = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{\delta b}_u \tag{2.22}$$

Hence, the final dynamics error-model can be formulated augmenting the state in equation (2.21) with $\boldsymbol{\delta b}_u$, which dynamics is described by (2.22), obtaining:

$$\begin{bmatrix} \boldsymbol{\delta \dot{r}}^e_{e,b} \\ \boldsymbol{\delta \dot{v}}^e_{e,b} \\ \boldsymbol{\delta \dot{\epsilon}}^e \\ \boldsymbol{\delta \dot{b}}^b_a \\ \boldsymbol{\delta \dot{b}}^b_g \\ \boldsymbol{\delta \dot{b}_u} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{N} & -2\boldsymbol{\Omega}^e_{i,e} & \boldsymbol{F} & \boldsymbol{C}^e_b & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{\Omega}^e_{i,e} & \boldsymbol{0} & \boldsymbol{C}^e_b & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -diag(\boldsymbol{\alpha}) & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -diag(\boldsymbol{\beta}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta r}^e_{e,b} \\ \boldsymbol{\delta v}^e_{e,b} \\ \boldsymbol{\delta \epsilon}^e \\ \boldsymbol{\delta b}^b_a \\ \boldsymbol{\delta b}^b_g \\ \boldsymbol{\delta b}_u \end{bmatrix} + \\ + \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{C}^e_b & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}^e_b & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}^b_f \\ \boldsymbol{w}^b_\omega \\ \boldsymbol{w}^b_a \\ \boldsymbol{w}^b_g \end{bmatrix} \tag{2.23}$$

Equation (2.23) can be written compactly as:

$$\boldsymbol{\delta \dot{x}} = \boldsymbol{\Phi \delta x} + \boldsymbol{G\omega} \tag{2.24}$$

where $\boldsymbol{\delta x}$ is the error-state defined as (2.1), $\boldsymbol{\Phi}$ is the 17×17 state-transition matrix, which define mathematically the time-evolution of the errors affecting the state and $\boldsymbol{G}$ is the 17×12 shaping matrix, used to characterize the white noise $\boldsymbol{\omega}$ in input, matching the actual system characteristics.

The dynamics error-model defined in (2.21) is still the theoretical formulation of the dynamics of the errors affecting the system, In order to be implemented in the navigation filter it has to be reformulated, from continuous-time form, in discrete-time form. Given the time-step Δt, the discrete-time state-transition matrix at time k, indicated as $\boldsymbol{F}_k$, can be directly derived from the continuous-time state-transition matrix $\boldsymbol{\Phi}$ as:

$$\boldsymbol{F}_k = e^{\boldsymbol{\Phi}\Delta t} = \sum_{k=0}^{\inf} \frac{1}{k!}(\boldsymbol{\Phi}\Delta t)^k \tag{2.25}$$

This expression can be approximated to the first order as:

$$\boldsymbol{F}_k = e^{\boldsymbol{\Phi}\Delta t} \simeq \boldsymbol{I} + \boldsymbol{\Phi}\Delta t \tag{2.26}$$

The error due to approximation in (2.26) (also known as forward Euler method) can be considered negligible because of the small sampling time Δt (i.e. the INS update rate: $0.1s$).

Instead the discrete-time process-noise covariance matrix at time k, indicated as $\boldsymbol{Q}_k$, can be obtained solving the following integral:

$$\boldsymbol{Q}_k = \int_{t_k}^{t_{k+1}} \boldsymbol{F}(t_{k+1}, \tau)\boldsymbol{G}(\tau)\boldsymbol{Q}(\tau)\boldsymbol{G}^T(\tau)\boldsymbol{F}^T(t_{k+1}, \tau)d\tau \tag{2.27}$$

where $\boldsymbol{F}(t_{k+1}, \tau)$ is the discrete-time state-transition matrix computed between times τ and t_{k+1}, while the matrix $\boldsymbol{Q}(\tau)$ is the continuous-time spectral density matrix of the noise (derived form the sensors specifications).

The integral in (2.27) is solved numerically with the following approximation:

$$\boldsymbol{Q}_k = (\boldsymbol{F}_k\boldsymbol{G}(t_k)\boldsymbol{Q}(t_k)\boldsymbol{G}^T(t_k) + \boldsymbol{G}(t_k)\boldsymbol{Q}(t_k)\boldsymbol{G}^T(t_k)\boldsymbol{F}_k^T)\frac{\Delta t}{2} \tag{2.28}$$

In conclusion, the discrete-time dynamics error-model can be formulated compactly as:

$$\boldsymbol{\Delta x}_{k+1} = \boldsymbol{F}_k\boldsymbol{\Delta x}_k + \boldsymbol{Q}_k\boldsymbol{\omega}_k \tag{2.29}$$

where δ has been substituted with the symbol Δ, remarking the fact that the infinitesimal perturbation $\boldsymbol{\delta x}$ becomes a discrete correction (according to the time-step Δt) to be applied to the state. This discrete-time model will be the one employed as process model in the integrating navigation filter for the GNSS/INS tightly coupled architecture.

2.3.3 System observation model

The system observation model is a mathematical formulation responsible for mapping the incoming raw measurements into the state domain, allowing the filter to combine and process this new set of raw data and obtain a refined update in terms of error-state vector. In this context the measurements consist in pseudoranges and pseudorange-rates. Since the tightly coupled framework with indirect configuration has been chosen, the measurement vector fed to the navigation filter is itself in error-form, obtained as the difference between the INS-predicted measurements and the raw GNSS measurements provided by the GNSS-receiver. The INS can compute these quantities processing information about the estimated receiver position and velocity, together with satellites position and velocity (obtained from the GNSS-receiver). The difference between these two groups of measurements embodies the information about the correction to be applied to the state, that has to be retrieved with some computation [20, 32].

Hence, the observation vector $\boldsymbol{z}_{j,k}$, where j indicates the $j-th$ satellite and k the $k-th$ epoch, can be formulated mathematically as:

$$\boldsymbol{z}_{j,k} = \hat{\boldsymbol{\zeta}}_{j,k} - \boldsymbol{\zeta}_{j,k} \tag{2.30}$$

where:

- $\boldsymbol{\zeta}_{j,k} = (\rho_{j,k}\ \dot{\rho}_{j,k})^T$ is the raw measurement, composed of pseudorange and pseudorange-rate associated with the $j-th$ satellite, provided by the GNSS-unit at the $k-th$ epoch.
- $\hat{\boldsymbol{\zeta}}_{j,k} = (\hat{\rho}_{j,k}\ \hat{\dot{\rho}}_{j,k})^T)$ is the nominal measurement, composed of predicted pseudorange and pseudorange-rate associated with the $j-th$ satellite, provided by the INS-unit at the $k-th$ epoch.

Assuming that N_s satellites are in view, the observation vector, compacting the measurements associated of all the satellites, is defined as:

$$\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{z}_{1,k} \\ \vdots \\ \boldsymbol{z}_{N_s,k} \end{bmatrix} = \left[\begin{array}{c} \hat{\rho}_{1,k} - \rho_{1,k} \\ \vdots \\ \hat{\rho}_{N_s,k} - \rho_{N_s,k} \\ \hline \hat{\dot{\rho}}_{1,k} - \dot{\rho}_{1,k} \\ \vdots \\ \hat{\dot{\rho}}_{N_s,k} - \dot{\rho}_{N_s,k} \end{array}\right] \tag{2.31}$$

In Section 1.3, an extensive derivation of the Least Square solution of the non-linear PVT problem, for a standalone GNSS positioning, was provided. In particular, the analysis of the linearized pseudorange model was provided in Section 1.3.1, while the pseudorange-rate model was analyzed in Section 1.3.2. Both formulations were given in a state-space form, in which the linearization was performed around an approximation point $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{v}}$ (containing also the user clock bias and clock drift respectively), and the corrections $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{v}$ were obtained. The respective final models are: (1.12) for position (and clock bias), and (1.33) for velocity (and clock drift). The combination of the aforementioned models leads to the formulation of a INS/GNSS linearized model as follows [20]:

$$\left[\begin{array}{c} \hat{\rho}_{1,k} - \rho_{1,k} \\ \vdots \\ \hat{\rho}_{N_s,k} - \rho_{N_s,k} \\ \hline \hat{\dot{\rho}}_{1,k} - \dot{\rho}_{1,k} \\ \vdots \\ \hat{\dot{\rho}}_{N_s,k} - \dot{\rho}_{N_s,k} \end{array}\right] = \left[\begin{array}{ccccccc} \boldsymbol{a}_{1,k}{}^T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{a}_{N_s,k}{}^T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 1 & 0 \\ \hline \boldsymbol{0} & \boldsymbol{a}_{1,k}{}^T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{a}_{N_s,k}{}^T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 0 & 1 \end{array}\right] \begin{bmatrix} \boldsymbol{\Delta x}_k^e \\ \boldsymbol{\Delta v}_k^e \\ \boldsymbol{\Delta \epsilon}_k^e \\ \boldsymbol{\Delta b}_{a,k}^b \\ \boldsymbol{\Delta b}_{g,k}^b \\ \Delta b_{u,k} \\ \Delta \dot{b}_{u,k} \end{bmatrix} + \boldsymbol{v}_k \tag{2.32}$$

or, in compact form:

$$\boldsymbol{z}_k = \boldsymbol{H}_k \boldsymbol{\Delta x}_k + \boldsymbol{v}_k \tag{2.33}$$

where $\boldsymbol{a}_{j,k} = [(\frac{x_{j,k} - \hat{x}_k}{\hat{r}_{j,k}})\ (\frac{y_{j,k} - \hat{y}_k}{\hat{r}_{j,k}})\ (\frac{z_{j,k} - \hat{z}_k}{\hat{r}_{j,k}})]^T$ is the unit vector, expressed in ECEF-coordinates, pointing to the $j-th$ satellite from the approximation point

$\hat{\boldsymbol{x}}_k$, at the $k - th$ epoch. This linearization point is obtained as the last INS-update applied to the absolute state. $\boldsymbol{H_k}$ represent the system observation matrix of size $2N_{sat} \times 17$. This matrix is formed from the Jacobian $\boldsymbol{H}$, defined in (1.13), that links the position and clock bias $\boldsymbol{\Delta x}$ to pseudoranges $\boldsymbol{\Delta \rho}$, as well as the velocity and clock drift $\boldsymbol{\Delta v}$ to the pseudorange-rates $\boldsymbol{\Delta \dot{\rho}}$, as showed in (1.33), accurately filled with zeros. Finally, $\boldsymbol{v}_k$ accounts for the residual errors affecting the measurement vector $\boldsymbol{z}_k$. The statistical description of these errors, affecting both pseudoranges and pseudorange-rates, is provided by the matrix $\boldsymbol{R}_k$, called observation-noise covariance matrix. This matrix can be estimated in different ways (in this thesis the methodology derived in [36] is adopted).

2.3.4 INS/GNSS tightly-coupled implementation

The previous section was devoted to the definition of the theoretical framework for the GNSS/INS tight integration. Here, instead, the analysis of the software implementation of such architecture will be developed. The first problem to be handled is the different rates at which the INS and GNSS units operate. In fact in the adopted integration software, which aims to simulate a realistic operating scenario, the GNSS-rate, at which the GNSS-receiver provides the observables, is set to $1Hz$, the INS-rate to $10Hz$ and the IMU-rate to $100Hz$.

As explained in Section 1.4.4, the INS adopt a two-speed integration algorithm: the IMU-rate is the rate at which the sensors (accelerometers and gyros) provide measurements and at which coning and sculling integrals are performed; while the INS-rate is the rate at which the INS process unit performs the actual integration, providing position, velocity and attitude updates. Note that the GNSS-rate, differently from the other two, is the rate at which the GNSS-receiver would provide the observables if satellite signals were constantly available. Because of external environmental conditions (e.g interference etc.), these signals can be interrupted or severely disrupted. Hence, the aforementioned periodicity of $1Hz$ is just an upper-bound, and this emphasizes the importance of the integration with inertial sensors, used to interpolate GNSS-solutions providing continuity in the positioning [29]. Regarding the integration filter timing, it is dictated by the instants at which groups of GNSS-observables are provided by the GNSS-receiver.

To keep this flow of data synchronized, all the units involved in the integrated module are administrated by a Finite-State-Machine (FSM). The FSM is a scheduling unit used to properly manage inertial navigation unit and navigation filter operations. This is done assigning timestamps to inertial data (accelerometers and gyros measurements) and GNSS-observables, then the correct synchronization is achieved following the order dictated by these timestamps, without needing any common-clock.

For clarity, a summary of the operations performed by the different units is reported here.

Inertial Navigation System

- High-rate ($100Hz$): IMU-measurements are provided and coning and sculling integrals are performed.
- Medium-rate ($10Hz$): Mechanization equations are solved providing position, velocity and attitude updates, which are used to refine the current absolute solution. Moreover, the INS provides the estimates of state-transition matrix $\boldsymbol{F}_k$ and process-noise covariance $\boldsymbol{Q}_k$, used to describe the system dynamics.
- Low-rate: when the navigation filter enters in the integration chain, the INS unit has to be fed with satellites position and velocity so that the nominal GNSS-observables $\hat{\boldsymbol{\rho}}$ and $\hat{\dot{\boldsymbol{\rho}}}$ can be computed (according to (1.7) and (1.32)).

Navigation filter

- Whenever a new set of GNSS-observables is available, the integration filter steps in the integration chain. It takes as inputs pseudoranges and pseudorange-rates, together with satellites information (used to compute observation matrix $\boldsymbol{H}_k$ and observation-noise covariance matrix $\boldsymbol{R}_k$), from the GNSS-receiver, while nominal pseudoranges and pseudorange-rates, as well as matrices $\boldsymbol{F}_k$ and $\boldsymbol{Q}_k$, from INS-unit. All these information, coming from both INS and GNSS units, are processed together with a filtering algorithm so that a low-rate integrated state correction $\boldsymbol{\Delta x}$ is provided.

As last observation, it has to be noticed that the FSM has to be initialized. In fact, since the INS provides only relative solutions in terms of updates, a first GNSS stand-alone solution (first-fix) is needed. This is done waiting a set of observables composed of at least four satellites and computing a PVT solution. For example, a simple recursive Least Squares (LS) on these observables can be performed, as in (1.14) and (1.34), starting from an arbitrary linearization point.

2.3.5 Latency management

Since this implementation has the purpose of simulate a real-time application, GNSS-latency issues must be accounted for. In fact, as explained before, due to different causes (GNSS-outages, ranging signal losses etc.), GNSS-observables are characterized by an unstable arrival frequency. In this thesis, to manage this problem, a replay-buffer solution is implemented inside the FSM. Whenever an expected set of GNSS-observables is missing, the data provided by the INS-unit are also stored in the buffer. Then, when the delayed GNSS-observables arrive, the buffered data are restored and GNSS-update performed. In case the temporal delay is larger then a certain threshold, the GNSS-outage is declared [27]. A schematic representation of this solution is shown in Figure 2.3.

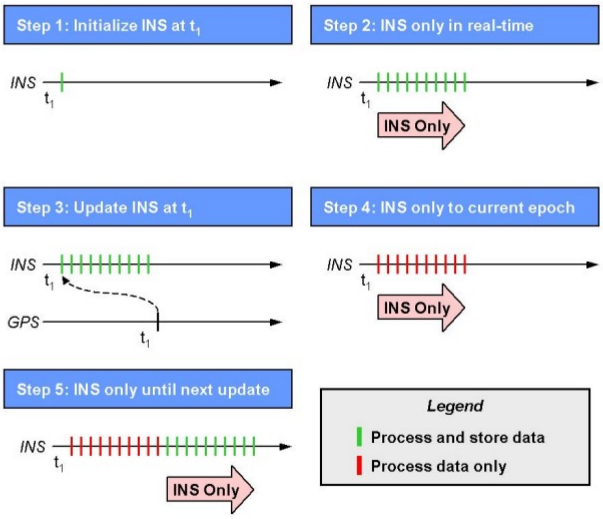

Step 1: Initialize INS at t_1
INS
t_1
Step 2: INS only in real-time
INS
t_1
INS Only
Step 3: Update INS at t_1
INS
t_1
GPS
t_1
Step 4: INS only to current epoch
INS
t_1
INS Only
Step 5: INS only until next update
INS
t_1
INS Only
Legend
🟩 Process and store data
🟥 Process data only

Figure 2.3: Buffer-replay latency management (picture taken from [30]).

Chapter 3

Factor Graphs

This chapter provides an overview of factor graphs for modeling and solving inference problems, including GNSS/INS integrated positioning. Factor graphs belong to the family of probabilistic graphical models, which are tools used to model and analyze complex systems involving probabilistic relationships and dependencies. They provide an intuitive and structured way to represent the interconnections between variables, allowing the design of flexible and modular software to retrieve the solution. In family are also Bayesian networks, the most known and simple graphical models; they are first introduced and then used to formulate factor graphs, that are a more convenient representation. Factor graphs are in fact just Bayesian networks conditioned on sensors data, that are typically known. The inference problem will be then formulated as an optimization problem on factor graphs. Finally, they will be adapted and applied to the GNSS/INS integration problem.

3.1 Probabilistic modeling

To introduce probabilistic modeling, the Simultaneous Localization and Mapping (SLAM) problem, typical in robotics, will be used as example. In fact, as a generalization of the localization problems, landmarks' (reference points used to retrieve information by sensors) position may not be known a-priori, and the unknown map has to be reconstructed while localizing with respect to it. In case their position is known, this can be reformulated as a trilateration problem.

A simplified representation of a small SLAM problem (that keeps a relative generality) is showed in Figure 3.1. The unknown consequential states x_1 x_2 and x_3 are linked to landmarks l_1 and l_2 through some measurements, given by the sensors' perception of the landmarks themselves (these variables can be vectors in general, the bold notation is not used in this chapter for simplicity).

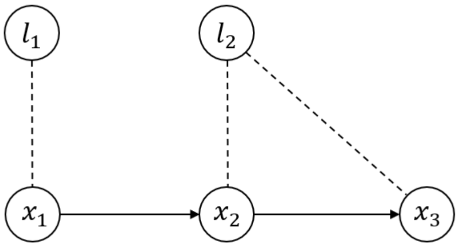


Figure 3.1: A small SLAM problem example. Arrows describe the evolution of states over time while dotted lines represent relationship between states and measurements

3.1.1 Bayesian networks

The involved unknowns in a positioning problem can be handled with a probabilistic description in the Bayesian framework, meaning that a belief over continuous, multivariate random variables $x \in \mathbb{R}^n$ have to be modeled. For this purpose, probability density functions (PDFs) $p(x)$ are used, for which it holds [37]:

$$\begin{gathered} \int p(x)dx = 1 \\ p(x) \geq 0 \end{gathered} \tag{3.1}$$

Given a set X of random variables together with a set of measurements Z, the actual interest is in the probability distribution of the set of variables X given the knowledge of the set of measurements Z, that is the conditional density [38]:

$$p(X|Z) \tag{3.2}$$

where the vertical-bar notation is used to indicate the term given. Probabilistic graphical models are adopted to obtain an expression for (3.2) over complex systems, as will be showed.

Bayesian networks (in short Bayes net) are directed graphical models in which variables θ_j are represented by nodes. The set of all the random variables $\Theta = \{\theta_1 ... \theta_n\}$, that includes both states X and measurements Z, is associated, through a Bayes net, with the joint probability density $p(\Theta)$, that is described as the product of conditional densities of each variable. Mathematically:

$$p(\Theta) = \prod_{j=1}^{n} p(\theta_j|\pi_j) \tag{3.3}$$

where π_j is an assignment of values to the parents of θ_j. What defines the node-parents relationships is the structure of the Bayes net itself (arrows' direction). For clarity, in Figure 3.2 a Bayes net is constructed on the example in Figure 3.1,

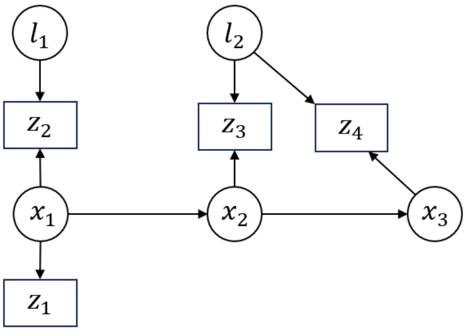


Figure 3.2: Bayes net retrieved from example in Figure 3.1. The variables associated with measurements are represented with rectangles, since they are observed variable

The random variables involved in the Bayes net in Figure 3.2 are object's and landmarks' positions X as well as measurements Z, forming the set $\Theta = \{X, Z\}$. Given the definition in (3.3), it yields [38]:

$$\begin{aligned} p(\Theta) = p(X, Z) = {} & p(x_1)p(x_2|x_1)p(x_3|x_2) \\ & \times p(l_1)p(l_2) \\ & \times p(z_1|x_1) \\ & \times p(z_2|x_1, l_1)p(z_3|x_2, l_2)p(z_4|x_3, l2) \end{aligned} \tag{3.4}$$

where the different factors, corresponding to a Bayes net's node each, have been divided into four groups. They can be distinguished in:

- Markov chain $p(x_1)p(x_2|x_1)p(x_3|x_2)$.
- Prior densities on landmarks $p(l_1)p(l_2)$.
- A conditional density on the absolute measurement of the first position $p(z_1|x_1)$.
- Three conditional densities on relative measurements of the three positions with respect to landmarks $p(z_2|x_1, l_1)p(z_3|x_2, l_2)p(z_4|x_3, l_2)$.

Note that the node-parents relationships are defined by the arrows' direction represented in the Bayes net: x_2 is an element of the parents set of z_3 while its parent set coincides with x_1.

Through a Bayes net it is now possible to model any system as a product of probabilistic relationships. As anticipated before, measurement variables (z_1, z_2 and z_3 in the example) are usually assumed to be given, and the interest is to estimate the set of variables X given that knowledge. The most common estimator in these problems is the so called Maximum a Posteriori (MAP) estimate, that is finding the maximum of the posterior density $p(X|Z)$. Mathematically [39]:

$$X^{MAP} = \operatorname*{argmax}_{X} p(X|Z) \tag{3.5}$$

Exploiting the Bayes's law [39], equation (3.5) can be re-written as:

$$X^{MAP} = \operatorname*{argmax}_{X} \frac{p(Z|X)p(X)}{p(Z)} \tag{3.6}$$

In (3.6) the posterior $p(X|Z)$ is expressed as the product of the measurement density $p(Z|X)$ and the prior $p(X)$, normalized by the factor $p(Z)$. Since the set of measurements Z is assumed to be known and the maximization of the function is performed, the normalization factor $p(Z)$ can be dropped. Moreover, the factor $p(Z|X)$ is usually formulated as a Gaussian distribution in Z. Considering Z just as a parameter, $p(Z|X)$ can be seen as a function of X, but, in general, it will not be a Gaussian distribution in this new variable. The equation (3.6) will then be reformulated equivalently as [38]:

$$X^{MAP} = \operatorname*{argmax}_{X} l(X;Z)p(X) \tag{3.7}$$

where $l(X;Z)$ is the likelihood function of the set X given the measurement set Z, defined as:

$$l(X;Z) \propto p(Z|X) \tag{3.8}$$

where it has been emphasized the fact that it is any function proportional to $p(Z|X)$ but in the variable X, with Z considered just as a parameter. Factor graphs will be adopted as probabilistic graphical models to perform a distinct division between the states X and the measurements Z, together with the necessity of handling non-Gaussian descriptions of the likelihood function $l\{X;Z\}$.

3.1.2 Factor Graphs for Inference

Similarly to Bayes nets, factor graphs describe a joint density as a product of factors. These factors can be any function ϕ of variables in the set X, without being constricted to be probability densities.

The MAP estimate of the example problem is described by (3.5). Taking (3.4), conditioning on Z and exploiting the Bayes' law, it holds:

$$\begin{aligned} p(X|Z) \propto\ & p(x_1)p(x_2|x_1)p(x_3|x_2) \\ & \times p(l_1)p(l_2) \\ & \times l(x_1;z_1) \\ & \times l(x_1,l_1;z_2)l(x_2,l_2;z_3)l(x_3,l_2;z_4) \end{aligned} \tag{3.9}$$

that is a factored unnormalized probability density over the set of variables X. This factorization, in which the measurement set Z is considered just a set of parameters, can be conveniently represented by a factor graph as in Figure 3.3

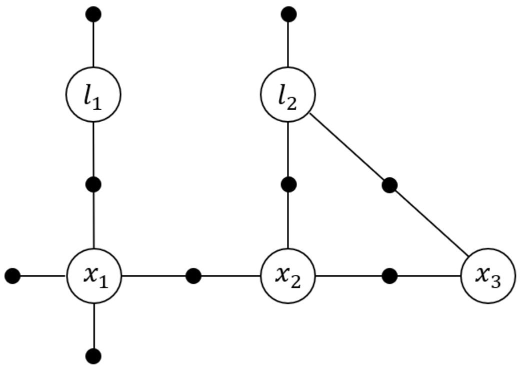


Figure 3.3: Factor graph formulation of the small SLAM example. It is obtained conditioning the Bayes' net in Figure 3.2

A factor graph, unlike Bayes nets, does not associate a conditional density function to each variable. Instead, the factors appearing in (3.8) are represented with a new node type (black dots in Figure 3.3). If a factor is function of a subset of the variables set X, then the corresponding black dot in the graph will be linked to only those variables. The 9 factors appearing in (3.8) can then be easily associated to the corresponding factors in the factor graph of Figure 3.3.

Formally, a factor graph is a bipartite graph referred to as $F \in (\mathcal{U}, \mathcal{V}, \mathcal{E})$, with two types of nodes: factors $\phi_i \in \mathcal{U}$ and variables $x_j \in \mathcal{V}$. Factor nodes and variable nodes are linked by edges $e_{i,j} \in \mathcal{E}$. A factor ϕ_i will be linked to a set of variables X_i, that are the variables of which the factor is function of. Hence, a factor graph $\mathcal{F}$ embodies the factorization of a global function $\phi(X)$ as [38]:

$$\phi(X) = \prod_{i=1}^{n} \phi_i(X_i) \tag{3.10}$$

Following the definition reported in (3.10) and applying it to the small SLAM example:

$$\begin{aligned} \phi(X) = \phi(l_1, l_2, x_1, x_2, x_3) &= \phi_1(x_1)\phi_2(x_2, x_1)\phi_3(x_3, x_2) \\ &\times \phi_4(l_1)\phi_5(l_2) \\ &\times \phi_6(x_1) \\ &\times \phi_7(x_1, l_1)\phi_8(x_2, l_2)\phi_9(x_3, l_2) \end{aligned} \tag{3.11}$$

that is the unnormalized posterior $p(X|Z)$. Note that there is a direct correspondence between factors in (3.4), describing the Bayes net, in equation (3.9), describing the Bayes net conditioned in Z and in (3.11), describing the factor graph. It holds for example:

$$\phi_6(x_1) = l(x_1; z_1) \propto p(z_1|x_1) \tag{3.12}$$

Finally, given (3.5), inference on factor graphs is performed similarly, being it the set of variables X such that:

$$X^{MAP} = \underset{X}{\operatorname{argmax}}\, p(X|Z) = \underset{X}{\operatorname{argmax}}\, \phi(X) = \underset{X}{\operatorname{argmax}} \prod_{i=1}^{n} \phi_i(X_i) \tag{3.13}$$

It is worth noting that each function $\phi_i(X_i)$ has no restrictions on its properties, it can be linear, non-linear, continuous or discrete, hence every type of measurements or constraint can be added to the graph.

3.2 MAP Inference for Nonlinear Factor Graphs

The MAP, described in (3.13), is the most common estimator used to find an estimate of the variables once a probabilistic distribution is given

To find an estimate of the variables once a probabilistic distribution is given, the most common estimator is the MAP estimate described in (3.13). Since the most frequent involved functions are Gaussian priors and likelihood functions derived from measurements affected by normally-distributed noise, factors will be of the form [37]:

$$\phi_i(X_i) \propto \exp\left(-\frac{1}{2}\|h_i(X_i) - z_i\|^2_{\Sigma_i}\right) \tag{3.14}$$

where $\|.\|_{\Sigma_i}$ is the Mahalanobis norm, involving an $n_i \times n_i$ covariance matrix Σ_i. Performing some mathematical manipulation:

$$\begin{aligned} X^{MAP} &= \underset{X}{\operatorname{argmax}} \prod_{i=1}^{n} \phi_i(X_i) \\ &= \underset{X}{\operatorname{argmax}}\, log\left(\prod_{i=1}^{n} \phi_i(X_i)\right) \\ &= \underset{X}{\operatorname{argmin}} -log\left(\prod_{i=1}^{n} \phi_i(X_i)\right) \\ &= \underset{X}{\operatorname{argmin}} -log\left(\prod_{i=1}^{n} \exp\left(-\frac{1}{2}\|h_i(X_i) - z_i\|^2_{\Sigma_i}\right)\right) \\ &= \underset{X}{\operatorname{argmin}} -log\left(\exp\left(-\frac{1}{2}\sum_{i=1}^{n} \|h_i(X_i) - z_i\|^2_{\Sigma_i}\right)\right) \\ &= \underset{X}{\operatorname{argmin}} \sum_{i=1}^{n} \|h_i(X_i) - z_i\|^2_{\Sigma_i} \end{aligned} \tag{3.15}$$

where (3.14) has been substituted in (3.13) (the proportionality relationship has been substituted with an equality relationship since functions do not need to be normalized in this context), the negative logarithm has been applied and the coefficient 1/2 dropped. After these steps, the MAP estimate X^{MAP} can be formulated as (3.15), that consists in minimizing a sum of non-linear squares. Note that every term involved in this sum is a measurement-derived likelihood function or prior density on some variable, and they are all summed together to create a global cost function to be minimized.

The functions $h_i(X_i)$, introduced in (3.14), that map variables X_i into a proper domain, are in general nonlinear. Hence, the minimization process involved in (3.15) has to be handled with some nonlinear optimization methods (e.g. Gauss-Newton, Levenberg-Marquardt etc.). This topic will be discussed in the next section.

3.2.1 Linearization

Given the non linear functions $h_i(X_i)$, they can be linearized through a Taylor expansion around a linearization point X_i^0, obtaining:

$$h_i(X_i) \simeq h_i(X_i^0) + H_i \Delta X_i \tag{3.16}$$

where $\Delta X_i = X_i - X_i^0$ is the state update vector, while H_i, the Jacobian of the measurement function $h_i(X_i)$ is defined as:

$$H_i \triangleq \left.\frac{\partial h_i(X_i)}{\partial X_i}\right|_{X_i^0} \tag{3.17}$$

Substituting now (3.16) in (3.15), a linearized version of the MAP estimate, in the variable ΔX, is obtained:

$$\begin{aligned}\Delta X^* &= \underset{\Delta X}{\operatorname{argmin}} \sum_{i=1}^{n} \|h_i(X_i^0) + H_i \Delta X_i - z_i\|^2_{\Sigma_i} \\ &= \underset{\Delta X}{\operatorname{argmin}} \sum_{i=1}^{n} \|H_i \Delta X_i - (z_i - h_i(X_i^0)\|^2_{\Sigma_i} \qquad (3.18)\end{aligned}$$

Once ΔX^* is obtained, it holds $X^{MAP} = X^0 + \Delta X^*$. The difference $z_i - h_i(X_i^0)$ is the prediction error.

Note that the Mahalanobis norm can be re-written as an $l_2 - norm$ performing some mathematical steps. Recalling the Cholesky decomposition, given a real, symmetrical positive-definite matrix S, it holds $S = U^T U$, where U is an upper triangular matrix (usually indicated as $S^{1/2}$). Hence, the Mahalanobis norm of a vector a can be re-written as [38]:

$$\|a\|^2_\Sigma \triangleq a^T \Sigma^{-1} a = a^T ((\Sigma^{-1/2})^T \Sigma^{-1/2}) a = (\Sigma^{-1/2} a)^T (\Sigma^{-1/2} a) = \|\Sigma^{-1/2} a\|^2_2 \tag{3.19}$$

where $\Sigma^{-1/2}$ is the Cholesky factor of Σ^{-1} (the inverse of a symmetric matrix S^{-1} is still symmetric).

Given the result in (3.19), the equation (3.18) can be re-written with a change of variables:

$$A_i = \Sigma^{-1/2} H_i \tag{3.20}$$

$$b_i = \Sigma^{-1/2}(z_i - h_i(X_i^0)) \tag{3.21}$$

so that (3.18) becomes:

$$\Delta X^* = \underset{\Delta X}{\operatorname{argmin}} \sum_{i=1}^{n} \|A_i \Delta X_i - b_i\|_2^2 \tag{3.22}$$

Equation (3.22) can be expressed in matrix form compacting the various matrices A_i in a matrix A as well as the vectors b_i in a vector b, obtaining:

$$\Delta X^* = \underset{\Delta X}{\operatorname{argmin}} \|A \Delta X - b\|_2^2 \tag{3.23}$$

Given this equation, there are several efficient algorithms that can be applied to retrieve ΔX^*. However, in the following section, a more general approach will be showed.

The A matrix in (3.23) embodies the structure of the underlying factor graph. Its dimension depends on how many variables x_i are involved and how they are connected but, in most of the applications, it will be a sparse matrix. The sparsity of the matrix, and hence of the factor graph it is related to, comes from inherent properties of the system it is describing. Taking for example the small SLAM problem showed in Figure 3.3, if the object keeps moving, new states $x_4, x_5, ..., x_N$ will be created, obtaining a situation as in Figure 3.4. Assuming them to be described by a Markov-chain (that is a realistic assumption in most of the cases), each of them will be linked by factors only to the adjacent ones. Even considering landmarks' positions to be unknown, the arising edges in the graphs will be far less than the possible ones. The resulting factor graph, and hence the associated matrix A, will be then sparse.

The sparsity property arising in these problems is fundamental for different reasons. First of all, probabilistic graphical models are useful thanks to sparsity (they would not be representative if the factor graph in Figure 3.4 was a fully connected graph). Moreover, computing the solution of problems involving so many variables with a dense (i.e. not sparse) A matrix can lead to computational problems. In fact the algorithms employed to retrieve the solution (i.e. MAP estimate) are designed specifically to exploit sparsity.

Before moving on, assuming that the solution ΔX^* has been found, a Gauss-Newton algorithm can be adopted to retrieve iteratively the solution X^{MAP} [38]. Starting from the linearization point $X_{(1)}^0$, the state update $\Delta X_{(1)}^*$ is found. The

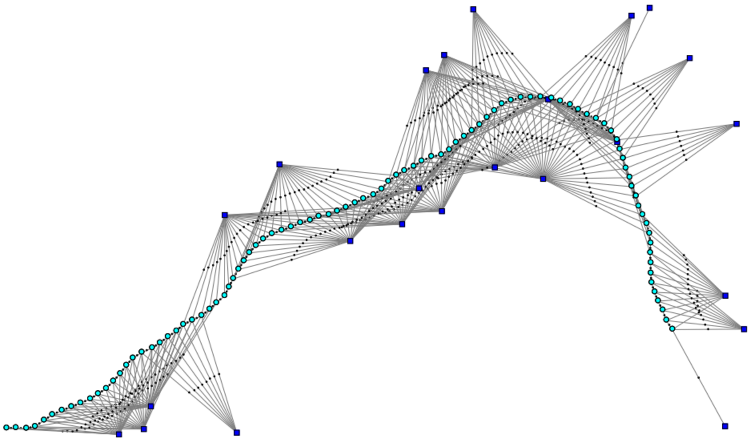


Figure 3.4: Factor graph describing a larger SLAM problem [38]

problem can be re-linearized around $X^0_{(2)} = X^0_{(1)} + \Delta X^*_{(1)}$ so that a new state update $\Delta X^*_{(2)}$ will be obtained, and so on. This process stops when convergence is achieved.

3.2.2 Elimination algorithm

As discussed in the previous section, the computation of ΔX^{MAP} implies repeatedly solving possibly large but sparse linear systems. In fact, under the assumption of priors described by a normal distribution as well as measurements affected by Gaussian-noise, performing the MAP inference can be reformulated in solving multiple linear Least Square problems, for which several efficient algorithms already exist in literature. However, as will be showed, these approaches can be seen as particular cases of a more general algorithm, the Elimination algorithm. This algorithm, which can be applied to any factor graph under any assumption, allows the computation of the corresponding posterior density $p(X|Z)$ factorized so that an easy recovery of the MAP estimate is possible. An intuitive representation over probabilistic graphical models is also possible, thanks to the sparse characteristic of the problems to which factor graphs are usually applied.

As described in the previous section, a factor graph embodies the unnormalized posterior as product of factors, each of which is function of a subset X_i of the variables set X. Hence, it represents a global function $\phi(X)$ for which it holds:

$$p(X|Z) \propto \phi(X) = \prod_{i=1}^{n} \phi_i(X_i) \tag{3.24}$$

Note that, differently from how they were presented, given a problem as in Figure 3.1, it is more intuitive to construct a factor graph as in Figure 3.3 than a Bayes net as in 3.2, since each factor represents given measurements or prior knowledge about some variables X_i. Bayes nets instead are more suitable for computing the MAP estimate, as will be showed afterwards. In fact, the elimination algorithm is a method for retrieving, given a factor graph, the corresponding Bayes net but having just the variables set X as unknown.

The variable elimination algorithm allows to factorize any factor graph:

$$\phi(X) = \phi(x_1, x_2, ..., x_n) \tag{3.25}$$

into a Bayes net of the form:

$$p(X) = p(x1|S_1)p(x2|S_2)...p(x_n) = \prod_{i=1}^{n} p(x_i|S_i) \tag{3.26}$$

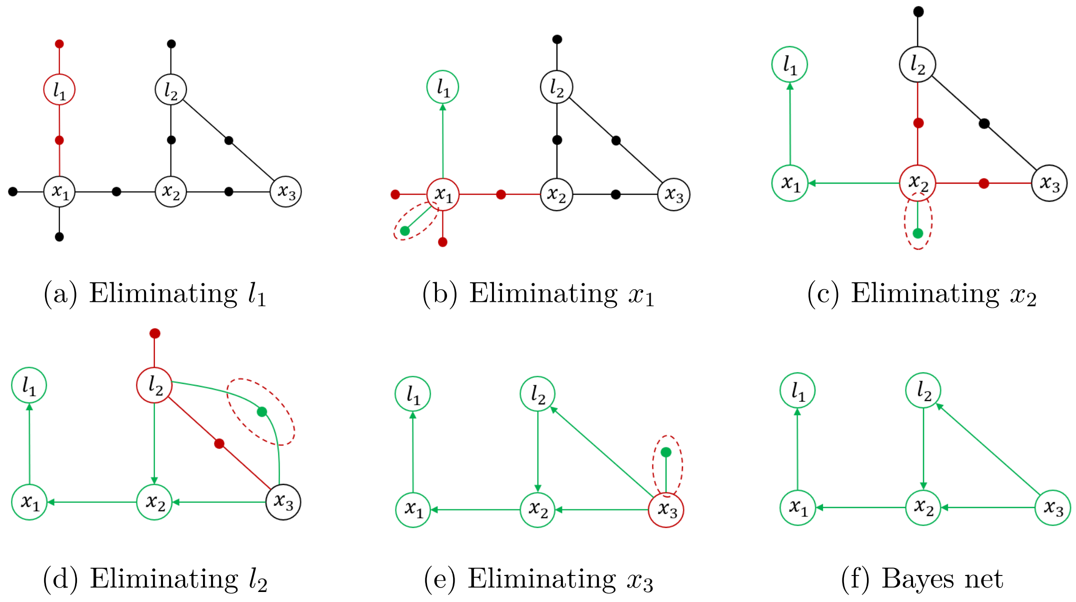
where $x_1, ..., x_n$ are all the variables involved, numerated with a chosen ordering [38]. Given the factor graph $\phi(X)$, eliminating the first variable x_1 means factorizing it as $\phi(X) = p(x_1|S_1)\phi(x_2, ..., x_n)$, where the first term represent the probability distribution of x_1 conditioned on the variables it depends on (i.e. variables connected to x_1 in the factor graph), which are collected into the separator S_1. The second term is a reduced factor graph, since x_1 has been eliminated. The same is done for x_2 and so on, obtaining at the end a Bayes net as in (3.26). This algorithm is a succession of local factorization steps: to eliminate x_i, all remaining factors it is connected to (that only involve the variable x_i and the ones contained in the separator S_i which it depends on) are multiplied together, obtaining the intermediate product $\Psi(x_i, S_i)$. This function is factorized obtaining $\Psi(x_i, S_i) = p(x_i|S_i)\tau(S_i)$, where $\tau(S_i)$ is a new factor that will be added in the resulting partial factor graph $\phi(x_{i+1}, ..., x_n)$. When the last variable x_n is eliminated, the separator S_n will be empty, generating simply a prior $p(x_n)$.

For this algorithm, a convenient graphical representation is possible. Taking again the small SLAM problem as an example, the elimination algorithm applied to it with a chosen elimination ordering ($l_1 \rightarrow x_1 \rightarrow x_2 \rightarrow l_2 \rightarrow x_3$) is showed in Figure 3.5

The obtained Bayes net in figure describe a posterior factorization in only the variables X. Hence, starting from the factor graph $\phi(x_1, x_2, x_3, l_1, l_2)$, the following factorization (Bayes net) is obtained (note that the factorization follows the elimination ordering, a different ordering leads to a different factorization as well as a different Bayes net structure):

$$p(x_1, x_2, x_3, l_1, l_2) = p(l_1|x_1)p(x_1|x_2)p(x_2|l_2, x_3)p(l_2|x_3)p(x_3) \tag{3.27}$$

The algorithm holds in general. If the assumption of linear measurements functions (or linearized ones) affected by Gaussian noise is considered again, the elimination algorithm boils down to a sparse Cholesky or a sparse QR factorization algorithm, as will be showed below.



(a) Eliminating l_1

(b) Eliminating x_1

(c) Eliminating x_2

(d) Eliminating l_2

(e) Eliminating x_3

(f) Bayes net

Figure 3.5: Elimination algorithm applied to the small SLAM example factor graph. The results of a variable elimination are represented in green, while factors and variables involved in an elimination step in red (also the green $\tau(S_i)$ factor obtained from the elimination of x_{i-1} will be involved in the elimination of x_i, hence it is circled in red). Figures 3.5a to 3.5e are elimination steps, while figure 3.5f represents the obtained Bayes net

As analyzed in the previous section, a linearized factor graph (or just a linear one) in which measurements are effected by Gaussian noise and priors are described by a normal distribution can be described by a large but sparse matrix A. This matrix together with the vector b (defined in (3.20) and (3.21) respectively) defines the equation (3.23), that is:

$$\Delta X^* = \underset{\Delta X}{\operatorname{argmin}} \|A\Delta X - b\|_2^2 \tag{3.28}$$

hereinafter the symbol Δ will be dropped for simplicity and because what will be said holds in general for linear problems, regardless of their origin.

Using again the small SLAM problem factor graph as an example (reported here for clarity), the related matrix A will have the following block structure:

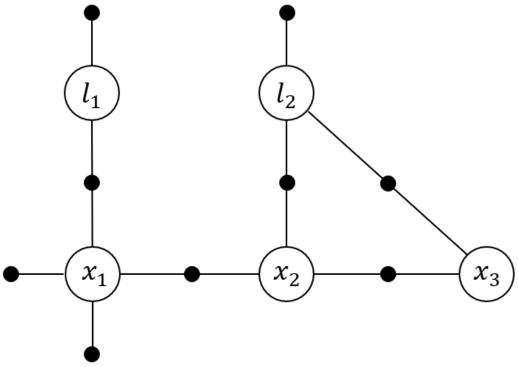


Figure 3.6: Factor graph formulation of the small SLAM example

$$[A|b] = \begin{array}{ccccc|cc} x_1 & x_2 & x_3 & l_1 & l_2 & b & \\ \hline A_{11} & & & & & b_1 & \phi_1 \\ A_{21} & & & & & b_2 & \phi_2 \\ A_{31} & & & A_{34} & & b_3 & \phi_3 \\ & & & A_{44} & & b_4 & \phi_4 \\ A_{51} & A_{52} & & & & b_5 & \phi_5 \\ & A_{62} & & & A_{65} & b_6 & \phi_6 \\ & & & & A_{75} & b_7 & \phi_7 \\ & & A_{83} & & A_{85} & b_8 & \phi_8 \\ & A_{92} & A_{93} & & & b_9 & \phi_9 \end{array} \tag{3.29}$$

it can be seen that every block row in matrix A corresponds to a factor of the factor graph in 3.6, while every block column correspond to a variable (except for the vector b that has been added to it). For example, a unary factor as the one connected only to l_1 is described by the only block matrix in the 4-th block-raw A_{44},that will be referred to as $A_4 \triangleq A_{44}$ with set of variables $X_4 \triangleq x_4$. Similarly, a binary factor as the one between x_1 and l_1 is described by the two block matrices in the 3-th block-row A_{31} and A_{34} and will be referred to as $A_3 \triangleq [A_{31}|A_{34}]$ with $X_3 \triangleq [x_1 \ l_1]^T$. With the assumptions made, each factor involved will be of the form:

$$\phi_i(X_i) = \exp\left(-\frac{1}{2}\|A_iX_i - b_i\|_2^2\right) \tag{3.30}$$

with A_i and b_i being the i-th block-row of $[A|b]$, and X_i the set of involved variables.

Now the elimination algorithm will be run through again in this context. When the variable x_i is being eliminated, the product of the all the adjacent factors $\phi_j(X_j)$ (with $j = 1 : n_f$, n_f = number of adjacent factors) will provide the intermediate product:

$$\Psi(x_i, S_i) = \prod_{j=1}^{n_f} \phi_j(X_j) = \exp\left(-\frac{1}{2}\sum_{j=1}^{n_f} \|A_j X_j - b_j\|_2^2\right) \tag{3.31}$$

where S_i is the separator, that is the set of all the variables, not eliminated yet, which x_i is connected to. This expression can be compacted creating the intermediate matrix $\bar{A}_i$ that stacks all the matrices involved in the considered factors, as well as the vector $\bar{b}_i$, obtaining:

$$\Psi(x_i, S_i) = \exp\left(-\frac{1}{2}\|\bar{A}_i[x_i; S_i] - \bar{b}_i\|_2^2\right) \tag{3.32}$$

For example, considering the elimination step showed in Figure 3.5a, l_1 is being eliminated, involving two factors that can be easily identified in (3.29). Hence it will be:

$$\bar{A}_1 = \begin{bmatrix} A_{11} & \\ A_{31} & A_{34} \end{bmatrix} = \begin{bmatrix} A_1 \\ A_3 \end{bmatrix} \qquad \bar{b}_1 = \begin{bmatrix} b_1 \\ b_3 \end{bmatrix} \tag{3.33}$$

The factorization of $\Psi(x_i, S_i)$ can be done in several ways. Here a QR-factorization approach will be used. Performing a QR-factorization on the augmented matrix $[\hat{A}_i|\hat{b}_i]$ it can be decomposed as [38]:

$$[\bar{A}_i|\bar{b}_i] = Q\left[\begin{array}{cc|c} R_i & T_i & d_i \\ & \tilde{A}_i & \tilde{b}_i \end{array}\right] \tag{3.34}$$

so that equation (3.32) can be re-written as:

$$\begin{aligned} \Psi(x_i, S_i) &= \exp\left(-\frac{1}{2}\|\bar{A}_i[x_i; S_i] - \bar{b}_i\|_2^2\right) \\ &= \exp\left(-\frac{1}{2}\|R_i x_i + T_i S_i - d_i\|_2^2\right)\exp\left(-\frac{1}{2}\|\tilde{A}_i S_i - \tilde{b}_i\|_2^2\right) \\ &= p(x_i|S_i)\tau(S_i) \end{aligned} \tag{3.35}$$

where the orthonormal matrix Q has been dropped since it does not change the values or the involved norms. This can be done variable by variable until all the factor graph is transformed into a Bayes net.

Considering equation (3.35), given the normal-distribution $p(x_i|S_i)$, the Gaussian parameters can be retrieved as:

$$p(x_i|Si) = \exp\left(-\frac{1}{2}\|R_i x_i + T_i S_i - d_i\|_2^2\right) = \exp\left(-\frac{1}{2}\|x_i - \mu_i\|_{\Sigma_i}^2\right) \tag{3.36}$$

hence:

$$\|R_i x_i + T_i S_i - d_i\|_2^2 = \|x_i - \mu_i\|_{\Sigma_i}^2$$

$$\|R_i x_i - (d_i - T_i S_i)\|_2^2 = \|x_i - \mu_i\|_{\Sigma_i}^2$$

$$\|R_i x_i - (d_i - T_i S_i)\|_2^2 = \|\Sigma^{-1/2} x_i - \Sigma^{-1/2} \mu_i\|_2^2 \tag{3.37}$$

obtaining [38]:

$$\Sigma_i = (R_i^T R_i)^{-1} \tag{3.38}$$

$$\mu_i = R_i^{-1}(d_i - T_i S_i) \tag{3.39}$$

Equation (3.38) provides the covariance matrix Σ_i while equation (3.39) the mean μ_i.

Finally, the MAP estimate X^{MAP} (or equivalently the Gauss-Newton step X^*) can be obtained via back-substitution. In fact, at the end of the elimination algorithm, a Bayes net is obtained, that is described by (3.26). Recalling that:

$$X^{MAP} = \underset{X}{\operatorname{argmax}}\, p(X) = \underset{X}{\operatorname{argmax}} \prod_{i=1}^{n} p(x_i|S_i) \tag{3.40}$$

and given that in a normal distribution the maximum happens in correspondence of the mean value μ_i, it holds:

$$x_i^* = R_i^{-1}(d_i - T_i S_i^*) \tag{3.41}$$

For the last eliminated n-variable (x_3 in the small SLAM example) the separator is empty, so it will be just $x_n^* = R_n^{-1} d_n$, then, by construction, the MAP estimate of the separator S_i^* will be known for the other variables.

In the following section factor graphs as a model describing also the temporal evolution of a system will be described.

3.2.3 Fixed-lag smoothing and filtering

In the previous section the analysis of a given factor graph has been performed. However, in most of the localization problems, measurements arrive as a temporal sequence, meaning that the corresponding factor graph is dynamically constructed. In this section, the considered factor graphs will be the typical graphs obtained in a GNSS/INS integration application. This is because a generalization on this topic would divert the focus from the main objective and which is out of the aim of this thesis. In a GNSS/INS context, landmarks considered in the previous discussion will be actually the satellites, for which position and velocity is known. Hence variables as l_1 and l_2 of the SLAM example 3.6 will not be present. There will be then two types of factors: unary factors, connected to just one variable x_i, that describe absolute measurements (GNSS observables) or prior knowledge about that variable; binary factors, connecting two adjacent variables, that describe the dynamic model (it can be assumed or derived from INS measurements).

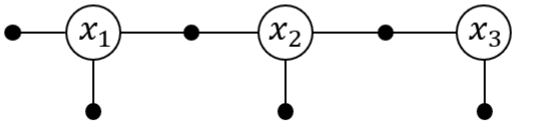


Figure 3.7: A typical factor graph arising in a GNSS/INS application

A graphical representation of this problem will be provided below, recalling that every manipulation done over a graph has its mathematical formulation. A typical factor graph for this application will look like the one in Figure 3.7. Applying the elimination algorithm following the ordering ($x_1 \to x_2 \to x_3$), the Bayes net in Figure 3.8a is obtained.

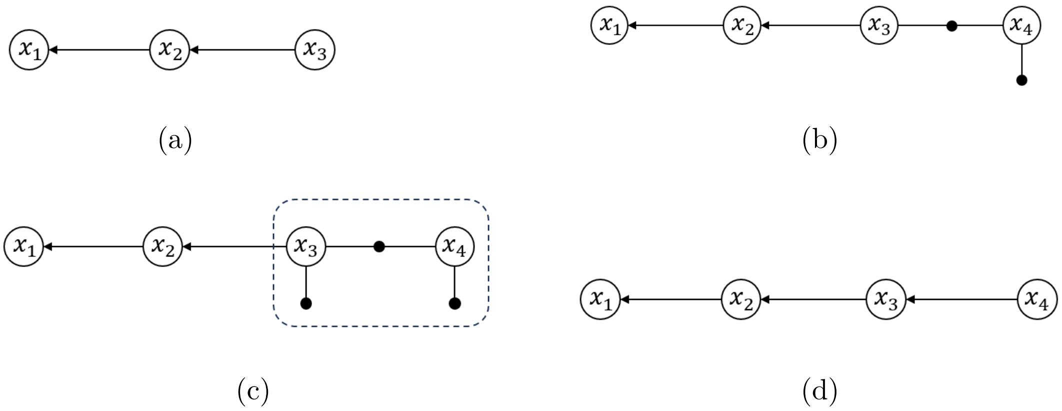


Figure 3.8: Evolution of the graphical model through two successive epochs

The Bayes net in 3.8a can be used to easily retrieve the MAP estimate $X^* = (x_1^*, x_2^*, x_3^*)$ as in section 3.2.2.

As a new set of GNSS observables comes, a new variable x_4 will be added (i.e. the state at the new epoch), together with a relative factor constraining it dynamically to x_3 and an absolute factor describing GNSS-measurements. This is represented in Figure 3.8b. In order to obtain the equivalent Bayes net of Figure 3.8d, from the estimate x_3^*, a prior factor can be retrieve as follows:

$$\phi_{prior}(x_3) = \exp\left(-\frac{1}{2}\|x_3 - x_3^*\|^2_{\Sigma_3}\right) \tag{3.42}$$

where x_3^* has been obtained previously and its covariance Σ_3 describes the uncertainty about this estimate, computed as in (3.38). This factor describes in fact the previous estimation (prior knowledge) of this variable. The graph can then be re-arranged as in Figure 3.8c A small factor graph is then formed involving the last two variables [38]. This can be transformed again into a Bayes net following the ordering $(x_3 \rightarrow x_4)$, obtaining Figure 3.8d. From this new Bayes net, it is possible not only to compute x_4^* but also to re-compute x_1^*, x_2^*, x_3^*, using the additional knowledge coming new measurements.

Note that the Bayes net in Figure 3.8d is described by:

$$p(X) = p(x_1|x_2)p(x_2|x_3)p(x_3|x_4)p(x_4) \tag{3.43}$$

meaning that x_1 is a leaf (i.e. it does not point to any other variable or, equivalently, none of the other variables' conditional density depends on it). Then, excluding x_1 from the Bayes net does not cause any loss of information (differently from removing it from a factor graph). This process of dropping a leaf-variable that has been already eliminated is called marginalization. This is because the intermediate factor $\tau(S_i)$ that appears during the elimination steps as in Section 3.2.2, carries up in the Bayes net the information about previous states.

Moreover it can be noted that if all the previous states are dropped when a new one comes (i.e. the number of nodes kept at each epoch, called window size ws, is $ws = 1$) only the filtering operation is performed, obtaining the equivalent of a Kalman filter. While having a window size $ws > 1$ means that also previous states are estimated at each epoch, performing a smoothing process.

In the next section the actual formulation for GNSS and GNSS/INS applications is given.

3.3 Factor Graphs for GNSS

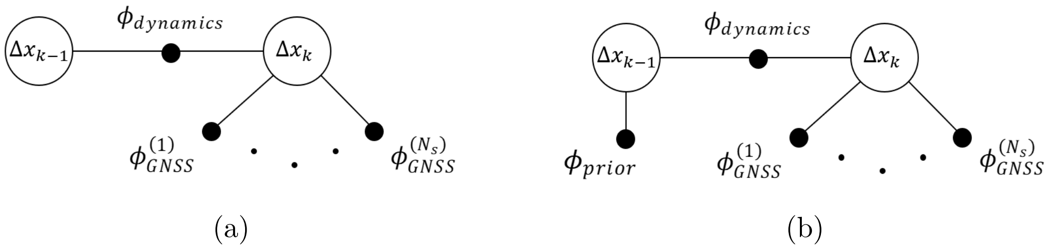
In this section, factor graphs will be used as framework to perform filtering and smoothing for a stand-alone GNSS receiver.

The state to be considered at each k-th epoch in this context is the one involved in the computations in Section 1.3.1 (position and user-clock bias) and in Section 1.3.2 (velocity and user clock-drift). Stacking those variables in a single state vector $\boldsymbol{\Delta x}_k$ for each k-epoch:

$$\boldsymbol{\Delta x}_k = [\Delta x_k \; \Delta y_k \; \Delta z_k \; \Delta \dot{x}_k \; \Delta \dot{y}_k \; \Delta \dot{z}_k \; \; -\Delta b_{u,k} \; \; -\Delta \dot{b}_{u,k}]^T \tag{3.44}$$

The symbol Δ is used since the linearized equations (1.13) and (1.34) will be used. Similarly to what was said in the previous section, at k-th epoch, a new set of GNSS observables will be available, while the state at epoch $k-1$ has been estimated already. Figure 3.9a reports the resulting situation.

The factors involved are: N_s absolute unary factors $\phi_{GNSS}^{(j)}$ (one for each visible satellite) and a relative binary factor $\phi_{dynamics}$ constraining the two states to some



(a)

(b)

Figure 3.9: Factor graph at k-th epoch

dynamics model. To proceed, a prior factor ϕ_{prior} on the already estimated x_{k-1} is retrieved as in Figure 3.9b.

How to solve a factor graph like the one showed here has been analyzed in the previous section. Then what is left is to characterize mathematically the involved factors. This is done writing them in the form $\phi = \exp\left(\|\boldsymbol{A}\boldsymbol{\Delta x} - \boldsymbol{b}\|_2^2\right)$, so that the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ will be used in the elimination algorithm in the QR-factorization form as in 3.2.2.

Prior factor

The prior factor ϕ_{prior} is retrieved from the previous estimate $\hat{\boldsymbol{x}}_{k-1}$. The factor is then described by an estimated covariance matrix $\boldsymbol{\Sigma}_{k-1}$ and the expected value of the new estimate $E\{\boldsymbol{x}_{k-1}\}$ that is $\hat{\boldsymbol{x}}_{k-1}$ itself. This is because without any other information its best estimate is the one already available. Elaborating these two information:

$$\begin{aligned}
\phi_{prior} &= \exp\left(\|\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}\|^2_{\boldsymbol{\Sigma}_{k-1}}\right) \\
&= \exp\left(\|\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{\Delta x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}\|^2_{\boldsymbol{\Sigma}_{k-1}}\right) \\
&= \exp\left(\|\boldsymbol{\Delta x}_{k-1}\|^2_{\boldsymbol{\Sigma}_{k-1}}\right) \\
&= \exp\left(\|\boldsymbol{\Sigma}_{k-1}^{-1/2}\boldsymbol{\Delta x}_{k-1}\|^2_2\right)
\end{aligned} \tag{3.45}$$

leads to:

$$\boldsymbol{A}_{prior} = \boldsymbol{\Sigma}_{k-1}^{-1/2} \qquad \boldsymbol{b}_{prior} = 0 \tag{3.46}$$

GNSS factor

A GNSS factor ϕ^j_{GNSS}, coming from the j-th satellite at the k-th epoch, is described by the measurement $\boldsymbol{\zeta}_{j,k} = [\rho_{j,k}\ \dot{\rho}_{j,k}]^T$ as well as the observation function

$h_k(\cdot)$. The observation model developed in Section 2.3.3, that is based on the linearization process in Sections 1.3.1 and 1.3.2 The only difference is that here the state-components related to INS are not present, so dimensions are lower. Note that the linearization is made around the point $\hat{\boldsymbol{x}}_k$ that is obtained as $f(\boldsymbol{x}_{k-1})$, where $f(\cdot)$ is a dynamic model used to propagate the previous state and perform a prediction. The adopted dynamic model will be described later on.

$$\begin{aligned}\phi^j_{GNSS} &= \exp\left(\|h_{j,k}\left(\boldsymbol{x}_k\right) - \boldsymbol{\zeta}_{j,k}\|^2_{\boldsymbol{R}_{j,k}}\right)\\ &= \exp\left(\|\boldsymbol{H}_{j,k}\boldsymbol{\Delta x}_k - h_{j,k}\left(\hat{\boldsymbol{x}}_k\right) + \boldsymbol{\zeta}_{j,k}\|^2_{\boldsymbol{R}_{j,k}}\right)\\ &= \exp\left(\|\boldsymbol{H}_{j,k}\boldsymbol{\Delta x}_k - (\hat{\boldsymbol{\zeta}}_{j,k} - \boldsymbol{\zeta}_{j,k})\|^2_{\boldsymbol{R}_{j,k}}\right)\\ &= \exp\left(\|\boldsymbol{R}^{-1/2}_{j,k}\boldsymbol{H}_{j,k}\boldsymbol{\Delta x}_k - \boldsymbol{R}^{-1/2}_{j,k}(\hat{\boldsymbol{\zeta}}_{j,k} - \boldsymbol{\zeta}_{j,k})\|^2_2\right)\end{aligned} \tag{3.47}$$

where $\boldsymbol{R}_{j,k}$ is the observation-noise covariance associated to the j-satellite at the k-epoch, computed as [36]. Note that in the second step there is a change of sign: it does not change the value of the norm and the same observation matrix $\boldsymbol{H}_{j,k}$ used in the previous chapters is obtained. Recalling the definition of observation vector $\boldsymbol{z}_{j,k} = \hat{\boldsymbol{\zeta}}_{j,k} - \boldsymbol{\zeta}_{j,k}$, it follows:

$$\boldsymbol{A}^j_{GNSS} = \boldsymbol{R}^{-1/2}_{j,k}\boldsymbol{H}_{j,k} \qquad \boldsymbol{b}^j_{GNSS} = \boldsymbol{R}^{-1/2}_{j,k}\boldsymbol{z}_{j,k} \tag{3.48}$$

For convenience, once every N_s factor ϕ^j_{GNSS} is obtained, they can be compacted in a single factor ϕ^j_{GNSS}, following what was done in Section 2.3.3. Then it is:

$$\phi_{GNSS} = \exp\left(\|\boldsymbol{R}^{-1/2}_k\boldsymbol{H}_k\boldsymbol{\Delta x}_k - \boldsymbol{R}^{-1/2}_k(\hat{\boldsymbol{\zeta}}_k - \boldsymbol{\zeta}_k)\|^2_2\right) \tag{3.49}$$

$$\boldsymbol{A}_{GNSS} = \boldsymbol{R}^{-1/2}_k\boldsymbol{H}_k \qquad \boldsymbol{b}_{GNSS} = \boldsymbol{R}^{-1/2}_k\boldsymbol{z}_k \tag{3.50}$$

for completeness, the measurement vector $\boldsymbol{z}_k$ and the system observation matrix $\boldsymbol{H}_k$ are reported here:

$$\boldsymbol{z}_k = \left[\begin{array}{c}\hat{\rho}_{1,k} - \rho_{1,k}\\ \vdots\\ \hat{\rho}_{N_s,k} - \rho_{N_s,k}\\ \hline \hat{\dot{\rho}}_{1,k} - \dot{\rho}_{1,k}\\ \vdots\\ \hat{\dot{\rho}}_{N_s,k} - \dot{\rho}_{N_s,k}\end{array}\right] \qquad \boldsymbol{H}_k = \left[\begin{array}{cccc}\boldsymbol{a}_{1,k}{}^T & \boldsymbol{0} & 1 & 0\\ \vdots & \vdots & \vdots & \vdots\\ \boldsymbol{a}_{N_s,k}{}^T & \boldsymbol{0} & 1 & 0\\ \hline \boldsymbol{0} & \boldsymbol{a}_{1,k}{}^T & 0 & 1\\ \vdots & \vdots & \vdots & \vdots\\ \boldsymbol{0} & \boldsymbol{a}_{N_s,k}{}^T & 0 & 1\end{array}\right] \tag{3.51}$$

where $\boldsymbol{a}_{j,k} = [(\frac{x_{j,k} - \hat{x}_k}{\hat{r}_{j,k}})\ (\frac{y_{j,k} - \hat{y}_k}{\hat{r}_{j,k}})\ (\frac{z_{j,k} - \hat{z}_k}{\hat{r}_{j,k}})]^T$ is the unit vector, expressed in ECEF-coordinates, pointing to the $j - th$ satellite from the approximation point

$\hat{\boldsymbol{x}}_k$, at the $k-th$ epoch.

Dynamic factor

The general formulation of a dynamic factor $\phi_{dynamics}$ is:

$$\phi_{dynamics} = \exp\left(\|f(\boldsymbol{x}_{k-1}) - \boldsymbol{x}_{\boldsymbol{k}}\|^2_{\boldsymbol{Q}_{k-1}}\right) \tag{3.52}$$

where $f(\cdot)$ is the dynamic model and $\boldsymbol{Q}_{k-1}$ is the process-noise covariance matrix. In a stand-alone GNSS application there is no direct observation of the behaviour of the body (i.e. its dynamics) between two epochs. Hence, the dynamic model has to be chosen a-priori. The most common model used in literature is the constant-velocity model. Given the estimate $\hat{\boldsymbol{x}}_{k-1}$, the prediction $\hat{\boldsymbol{x}}_k$ is obtained as:

$$\hat{\boldsymbol{x}}_k = \boldsymbol{F}_{k-1}\hat{\boldsymbol{x}}_{k-1} \tag{3.53}$$

$$\boldsymbol{F}_{k-1} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.54}$$

where Δt is the time step between two epochs. Instead the process-noise covariance Q has fixed typical values. The dynamic factor $\phi_{dynamics}$ can be then written as:

$$\begin{aligned} \phi_{dynamics} &= \exp\left(\|\boldsymbol{F}_{k-1}\boldsymbol{x}_{k-1} - \boldsymbol{x}_k\|^2_{\boldsymbol{Q}_{k-1}}\right) \\ &= \exp\left(\|\boldsymbol{F}_{k-1}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{F}_{k-1}\boldsymbol{\Delta x}_{k-1} - \hat{\boldsymbol{x}}_k - \boldsymbol{\Delta x}_k\|^2_{\boldsymbol{Q}_{k-1}}\right) \\ &= \exp\left(\|[\boldsymbol{F}_{k-1}\ \ -\boldsymbol{I}][\boldsymbol{\Delta x}_{k-1}\ \boldsymbol{\Delta x}_k]^T - (\hat{\boldsymbol{x}}_k - \boldsymbol{F}_{k-1}\hat{\boldsymbol{x}}_{k-1})\|^2_{\boldsymbol{Q}_{k-1}}\right) \\ &= \exp\left(\|[\boldsymbol{F}_{k-1}\ \ -\boldsymbol{I}][\boldsymbol{\Delta x}_{k-1}\ \boldsymbol{\Delta x}_k]^T\|^2_{\boldsymbol{Q}_{k-1}}\right) \\ &= \exp\left(\|[\boldsymbol{Q}_{k-1}^{-1/2}\boldsymbol{F}_{k-1}\ \ -\boldsymbol{Q}_{k-1}^{-1/2}\boldsymbol{I}][\boldsymbol{\Delta x}_{k-1}\ \boldsymbol{\Delta x}_k]^T\|^2_2\right) \end{aligned} \tag{3.55}$$

defining the two matrices $\boldsymbol{F}_Q = \boldsymbol{Q}_{k-1}^{-1/2}\boldsymbol{F}_{k-1}$ and $\boldsymbol{I}_Q = -\boldsymbol{Q}_{k-1}^{-1/2}\boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix, it holds:

$$\boldsymbol{A}_{dynamics} = [\boldsymbol{F}_Q\ \boldsymbol{I}_Q]; \qquad \boldsymbol{b}_{dynamics} = 0 \tag{3.56}$$

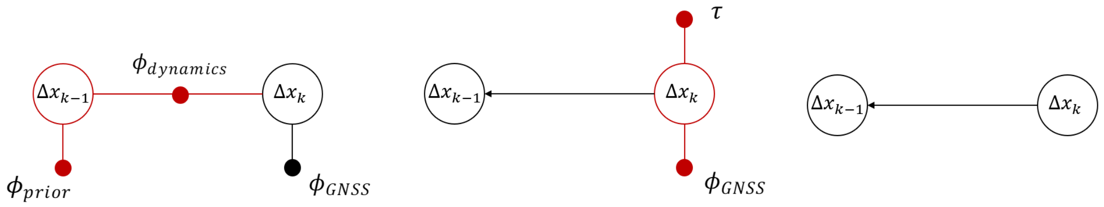
Note that being a binary factor it involves both unknowns $\boldsymbol{\Delta x}_{k-1}$ and $\boldsymbol{\Delta x}_k$.

Elimination algorithm

Now that the involved factors have been mathematically characterized a brief description of the elimination algorithm applied to a typical GNSS factor graph is here reported. Collecting the defined matrices and vectors defining each factor, the matrix $\boldsymbol{A}$ augmented with the vector $\boldsymbol{b}$ describing the factor graph will be:

$$[\boldsymbol{A}|\boldsymbol{b}] = \left[\begin{array}{cc|c} \boldsymbol{A}_{prior} & & \boldsymbol{b}_{prior} \\ \boldsymbol{F}_Q & \boldsymbol{I}_Q & \boldsymbol{b}_{dynamics} \\ & \boldsymbol{A}_{GNSS} & \boldsymbol{b}_{GNSS} \end{array}\right] \tag{3.57}$$

The chosen elimination ordering is $\boldsymbol{\Delta x}_{k-1} \rightarrow \boldsymbol{\Delta x}_k$. In Figure 3.10 the elimination steps are reported. The elimination of the variable $\boldsymbol{\Delta x}_{k-1}$, reported in Figure 3.10a, implies first the creation of the following augmented block matrix (first two block rows of $[\boldsymbol{A}|\boldsymbol{b}]$ corresponding to the two factors):



(a) Elimination of $\boldsymbol{\Delta x_{k-1}}$ (b) Elimination of $\boldsymbol{\Delta x_k}$ (c) Resulting Bayes net

Figure 3.10: Elimination algorithm applied to the factor graph at k-th epoch

$$[\bar{\boldsymbol{A}}_{k-1}|\bar{\boldsymbol{b}}_{k-1}] = \left[\begin{array}{c|c} \boldsymbol{A}_{prior} & \boldsymbol{b}_{prior} \\ \boldsymbol{A}_{dynamics} & \boldsymbol{b}_{dynamics} \end{array}\right] = \left[\begin{array}{cc|c} \boldsymbol{A}_{prior} & & \boldsymbol{b}_{prior} \\ \boldsymbol{F}_Q & \boldsymbol{I}_Q & \boldsymbol{b}_{dynamics} \end{array}\right] \tag{3.58}$$

then, performing the QR-factorization:

$$QR\left([\bar{\boldsymbol{A}}_{k-1}|\bar{\boldsymbol{b}}_{k-1}]\right) \rightarrow \left[\begin{array}{cc|c} \boldsymbol{R}_{k-1} & \boldsymbol{T}_{k-1} & \boldsymbol{d}_{k-1} \\ & \tilde{\boldsymbol{A}} & \tilde{\boldsymbol{b}} \end{array}\right] \tag{3.59}$$

matrix $\tilde{\boldsymbol{A}}$ and vector $\tilde{\boldsymbol{b}}$ are the quantities describing the factor τ in Figure 3.10b. Once $\boldsymbol{R}_{k-1}$, $\boldsymbol{T}_{k-1}$ and $\boldsymbol{d}_{k-1}$ are computed, they are stored and will remain associated to the state at that epoch. Note that this step is equivalent to the propagation step in a Kalman filter (the factor τ represent a prior of the variable $\boldsymbol{\Delta x}_k$ hence, its prediction).

For the next step, elimination of the variable $\boldsymbol{\Delta x}_k$ in Figure 3.10b, another QR-factorization is required (block matrix composed of the remaining factors):

$$[\bar{\boldsymbol{A}}_k|\bar{\boldsymbol{b}}_k] = \left[\begin{array}{c|c} \boldsymbol{A}_{GNSS} & \boldsymbol{b}_{GNSS} \\ \tilde{\boldsymbol{A}} & \tilde{\boldsymbol{b}} \end{array}\right] \tag{3.60}$$

$$QR\left([\bar{\boldsymbol{A}}_k|\bar{\boldsymbol{b}}_k]\right) \rightarrow \left[\boldsymbol{R}_k \mid \boldsymbol{d}_k\right] \tag{3.61}$$

Once the matrix $\boldsymbol{R}_k$ and the vector $\boldsymbol{d}_k$ have been computed, the obtained Bayes net in figure 3.10c can be solved with equation (3.41). The estimate of $\boldsymbol{\Delta x}_k$, with the associated covariance $\boldsymbol{\Sigma_k}$ is then simply:

$$\boldsymbol{\Delta x}_k^* = \boldsymbol{R}_k^{-1}\boldsymbol{d}_k \tag{3.62}$$

$$\boldsymbol{\Sigma}_k = (\boldsymbol{R}_k^T\boldsymbol{R}_k)^{-1} \tag{3.63}$$

This computation is equivalent to the update step of a Kalman filter.

Now the smoothing performed via back substitution is possible. In fact, given a fixed-lag value of n_{lag}, at each epoch k the factor graph will be composed of nodes until the one at epoch $k - n_{lag}$. For all these nodes, the values of $\boldsymbol{R}$, $\boldsymbol{T}$ and $\boldsymbol{d}$ have been stored, and erased when these states become too old. Introducing the variable $lag = 1 : n_{lag}$, the new estimates of these variables for each lag-value are:

$$\boldsymbol{\Delta x}_{k-lag} = \boldsymbol{R}_{k-lag}^{-1}(\boldsymbol{d}_{k-lag} - \boldsymbol{T}_{k-lag}\boldsymbol{\Delta x}_{k-lag+1}^*) \tag{3.64}$$

Once these corrections $\boldsymbol{\Delta x}_i$ are obtained, they are applied to the corresponding state $\boldsymbol{x}_i$

3.4 Factor Graphs for GNSS/INS integration

In order to employ a factor graph in a GNSS/INS integration filter little changes have to be applied to the discussion performed in the previous Section 3.3.

In fact, as explained in Section 2.3, in the integrated navigation filter also IMU-biases $\boldsymbol{\Delta b}_a$ and $\boldsymbol{\Delta b}_g$ are estimated as well as the orientation $\boldsymbol{\Delta \epsilon}$, hence the considered state at the k-epoch, of dimension 17, is:

$$\boldsymbol{\Delta x}_k = [\boldsymbol{\Delta r}_k \; \boldsymbol{\Delta v}_k \; \boldsymbol{\Delta \epsilon}_k \; \boldsymbol{\Delta b}_{a,k} \; \boldsymbol{\Delta b}_{g,k} \; \boldsymbol{\Delta b}_{u,k}]^T \tag{3.65}$$

What has to be changed with respect to the previous discussion are the dynamic factor $\phi_{dynamics}$ and the GNSS factor ϕ_{GNSS}. A schematic representation of the situation is showed in Figure 3.11. The correction $\boldsymbol{\Delta x}_{k-1}^*$ is fed to the INS unit from the integrated navigation filter. Then, as explained in Section 2.3.4, the INS provides updates $\boldsymbol{\Delta x}_{k-1}^{(i)}$ at medium-rate ($10Hz$, 10-times the GNSS-rate), together with the matrices $\boldsymbol{F}_{k-1}^{(i)}$ and $\boldsymbol{Q}_{k-1}^{(i)}$ computed as in 2.3.2. Finally, the last updated state $\boldsymbol{x}_{k-1}^{(10)}$ is used as linearization point, and here is computed $\hat{\boldsymbol{\zeta}}_{\boldsymbol{k}}$. In Figure 3.11 a schematic representation of the situation is shown, together with the relevant quantities that are exchanged with the INS-unit.

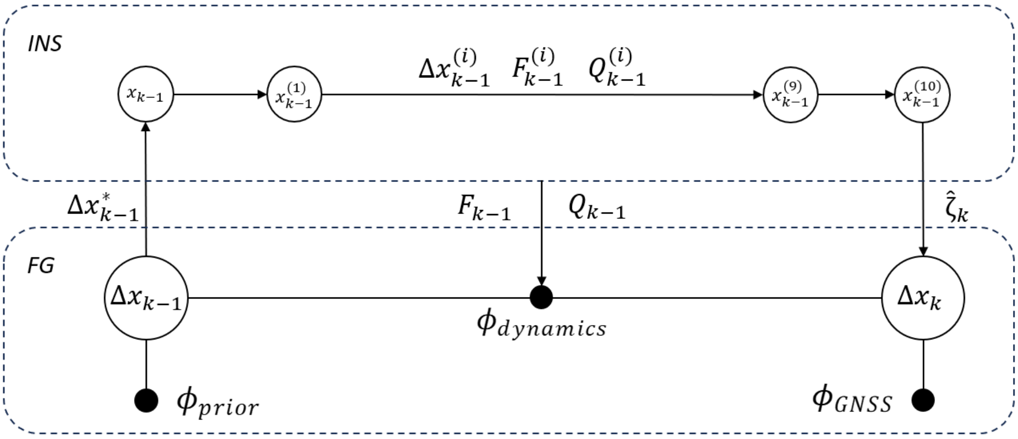


Figure 3.11: Factor graph (FG) with INS unit on top

GNSS factor

The only difference with respect to ϕ_{GNSS} discussed previously is that the system observation matrix has to be accurately filled with zeros, to account for the bigger state vector. The matrix $\boldsymbol{H}_k =$ becomes the one in equation (2.32). Moreover, the computation of $\hat{\boldsymbol{\zeta}}_{j,k} = [\hat{\rho}_{j,k} \;\; \hat{\dot{\rho}}_{j,k}]^T$ is not performed over the prediction done with some dynamic model $f(\boldsymbol{x}_{k-1})$ but is performed over the last INS updated $\boldsymbol{x}_{k-1}^{(10)}$ using the equations (1.7) and (1.33) reported here:

$$\hat{\rho}_{j,k} = \sqrt{(x_{j,k} - x_{k-1}^{(10)})^2 + (y_{j,k} - y_{k-1}^{(10)})^2 + (z_{j,k} - z_{k-1}^{(10)})^2} + b_{u,k-1}^{(10)} \tag{3.66}$$

$$\hat{\dot{\rho}}_{j,k} = (v_{x,j,k} - v_{x,k-1}^{(10)})a_{x,j,k} + (v_{y,j,k} - v_{y,k-1}^{(10)})a_{y,j,k} + (v_{z,j,k} - v_{z,k-1}^{(10)})a_{z,j,k} + \dot{b}_{u,k-1}^{(10)} \tag{3.67}$$

with $\boldsymbol{x}_{j,k}$ the j-satellite position at time k, $\dot{\boldsymbol{x}}_{j,k}$ the velocity and $\boldsymbol{a}_{j,k}$ the norm-vector pointing to the j-satellite at time k.

Dynamic factor

The computed matrices $\boldsymbol{F}_{k-1}^{(i)}$ and $\boldsymbol{Q}_{k-1}^{(i)}$ (as explained in Section 2.3.2) for each i, can be used all together to form a factor $\phi_{dynamics}$ constraining the two states at two consecutive epochs as in a stand-alone GNSS solution. However, now the dynamic model describing the system behaviour between these two epochs is no more assumed but is directly estimated by the INS-unit. In other words, each $\boldsymbol{F}_{k-1}^{(i)}$ and $\boldsymbol{Q}_{k-1}^{(i)}$ will contribute to the construction of the comprehensive state-transition matrix $\boldsymbol{F}_{k-1}$ and process-noise covariance $\boldsymbol{Q}_{k-1}$, both of dimension 17×17, between two consecutive epochs.

Dropping the subscript $k-1$ for simplicity it holds:

$$\boldsymbol{\Delta x}^{(i+1)} = \boldsymbol{F}^{(i)}\boldsymbol{\Delta x}^{(i)} + \boldsymbol{v}^{(i)} \tag{3.68}$$

where $\boldsymbol{v}^i$ is the zero-mean process-noise affecting the system, with covariance $\boldsymbol{Q}^{(i)}$. Similarly $\boldsymbol{x}^{(i+2)}$ can be written as:

$$\boldsymbol{\Delta x}^{(i+2)} = \boldsymbol{F}^{(i+1)}\boldsymbol{\Delta x}^{(i+1)} + \boldsymbol{v}^{(i+1)} = \boldsymbol{F}^{(i+1)}\boldsymbol{F}^{(i)}\boldsymbol{\Delta x}^{(i)} + \boldsymbol{F}^{(i+1)}\boldsymbol{v}^{(i)} + \boldsymbol{v}^{(i+1)} \tag{3.69}$$

Indicating with $E\{\cdot\}$ the expected value linear operator, noting that $E\{\boldsymbol{v}\} = 0$ with $\boldsymbol{v}$ the zero-mean process-noise at any instant, the expected values of $\boldsymbol{\Delta x}_{k-1}^{i+2}$ and $\boldsymbol{\Delta x}_{k-1}^{i+1}$, exploiting equations (3.68) and (3.69) are [37]:

$$\begin{aligned} E\{\boldsymbol{\Delta x}^{(i+1)}\} &= \boldsymbol{F}^{(i)}E\{\boldsymbol{\Delta x}^{(i)}\} + E\{\boldsymbol{v}^{(i)}\} \\ &= \boldsymbol{F}^{(i)}E\{\boldsymbol{\Delta x}^{(i)}\} \end{aligned} \tag{3.70}$$

$$\begin{aligned} E\{\boldsymbol{\Delta x}^{(i+2)}\} &= \boldsymbol{F}^{(i+1)}\boldsymbol{F}^{(i)}E\{\boldsymbol{\Delta x}^{(i)}\} + \boldsymbol{F}^{(i+1)}E\{\boldsymbol{v}^{(i)}\} + E\{\boldsymbol{v}^{(i+1)}\} \\ &= \boldsymbol{F}^{(i+1)}\boldsymbol{F}^{(i)}E\{\boldsymbol{\Delta x}^{(i)}\} \end{aligned} \tag{3.71}$$

Moreover, from the definition [37] of the covariance of a vector $\boldsymbol{\omega}$: $\boldsymbol{\Sigma}_\omega \triangleq E\{(\boldsymbol{\omega} - E\{\boldsymbol{\omega}\})(\boldsymbol{\omega} - E\{\boldsymbol{\omega}\})^T\}$, it follows:

$$\begin{aligned} \boldsymbol{\Sigma}^{(i+1)} &= E\{(\boldsymbol{\Delta x}^{(i+1)} - E\{\boldsymbol{\Delta x}^{(i+1)}\})(\boldsymbol{\Delta x}^{(i+1)} - E\{\boldsymbol{\Delta x}^{(i+1)}\})^T\} \\ &= E\{(\boldsymbol{F}^{(i)}\boldsymbol{\Delta x}^{(i)} + \boldsymbol{v}^{(i)} - \boldsymbol{F}^{(i)}E\{\boldsymbol{\Delta x}^{(i)}\})(\boldsymbol{F}^{(i)}\boldsymbol{\Delta x}^{(i)} + \boldsymbol{v}^{(i)} - \boldsymbol{F}^{(i)}\{(\boldsymbol{\Delta x}^{(i)}\})^T\} \\ &= E\{(\boldsymbol{F}^{(i)}(\boldsymbol{\Delta x}^{(i)} - E\{\boldsymbol{\Delta x}^{(i)}\}) + \boldsymbol{v}^{(i)})(\boldsymbol{F}^{(i)}(\boldsymbol{\Delta x}^{(i)} - E\{\boldsymbol{\Delta x}^{(i)}\}) + \boldsymbol{v}^{(i)})^T\} \\ &= \boldsymbol{F}^{(i)}E\{(\boldsymbol{\Delta x}^{(i)} - E\{\boldsymbol{\Delta x}^{(i)}\})(\boldsymbol{\Delta x}^{(i)} - E\{\boldsymbol{\Delta x}^{(i)}\})^T\}\boldsymbol{F}^{(i)^T} + E\{\boldsymbol{v}^{(i)}\boldsymbol{v}^{(i)^T}\} + \\ &\quad + \boldsymbol{F}^{(i)}E\{(\boldsymbol{\Delta x}^{(i)} - E\{\boldsymbol{\Delta x}^{(i)}\})\boldsymbol{v}^{(i)^T}\} + E\{\boldsymbol{v}^{(i)}(\boldsymbol{\Delta x}^{(i)} - E\{\boldsymbol{\Delta x}^{(i)}\})^T\}\boldsymbol{F}^{(i)^T} \\ &= \left[\boldsymbol{F}^{(i)}\boldsymbol{\Sigma}^{(i)}\boldsymbol{F}^{(i)^T}\right] + \left[\boldsymbol{Q}^{(i)}\right] \end{aligned} \tag{3.72}$$

where: $\boldsymbol{\Delta x}^{(i)}$ and $\boldsymbol{v}^{(i)}$ are supposed uncorrelated, $E\{(\boldsymbol{\Delta x}^{(i)} - E\{\boldsymbol{\Delta x}^{(i)}\})(\boldsymbol{\Delta x}^{(i)} - E\{\boldsymbol{\Delta x}^{(i)}\})^T\} = \boldsymbol{\Sigma}^{(i)}$ by definition as well as $E\{\boldsymbol{v}^{(i)}\boldsymbol{v}^{(i)^T}\} = \boldsymbol{Q}^{(i)}$. Then:

$$\begin{aligned} \boldsymbol{\Sigma}^{(i+2)} &= \boldsymbol{F}^{(i+1)}\boldsymbol{\Sigma}^{(i+1)}\boldsymbol{F}^{(i+1)^T} + \boldsymbol{Q}^{(i+1)} \\ &= \left[\boldsymbol{F}^{(i+1)}\boldsymbol{F}^{(i)}\boldsymbol{\Sigma}^{(i)}\boldsymbol{F}^{(i)^T}\boldsymbol{F}^{(i+1)^T}\right] + \left[\boldsymbol{F}^{(i+1)}\boldsymbol{Q}^{(i)}\boldsymbol{F}^{(i+1)^T} + \boldsymbol{Q}^{(i+1)}\right] \end{aligned} \tag{3.73}$$

Observing the final results of equation (3.72) and (3.73) it can be notice that the covariance describing a given update is given by the sum of the propagation of the initial covariance and of the process-noise covariances affecting the updates. Note that this propagation is performed eliminating the $k-1$ variable, and hence the prior factor ϕ_{prior} (initial covariance) and the dynamic factor $\phi_{dynamics}$ (process-noise), according to some state-transition matrix. Then, $\boldsymbol{F}_{k-1}$ and $\boldsymbol{Q}_{k-1}$ can be computed recursively starting from $\boldsymbol{F}_{k-1} = \boldsymbol{I}$ and $\boldsymbol{Q}_{k-1} = \boldsymbol{0}$. At every INS update i the following computation is performed:

$$\boldsymbol{F}_{k-1} = \boldsymbol{F}_{k-1}^{(i)} \boldsymbol{F}_{k-1} \tag{3.74}$$

$$\boldsymbol{Q}_{k-1} = \boldsymbol{F}_{k-1}^{(i)} \boldsymbol{Q}_{k-1} \boldsymbol{F}_{k-1}^{(i)^T} + \boldsymbol{Q}_{k-1}^{(i)} \tag{3.75}$$

Once the matrices $\boldsymbol{F}_{k-1}$ and $\boldsymbol{Q}_{k-1}$ are obtained, the dynamic factor can be described directly with the matrix $\boldsymbol{A}_{dynamics}$ and the vector $\boldsymbol{b}_{dynamics}$ as in (3.56).

3.5 Factor Graphs iterations and fixed-lag smoothing parameters

One of the advantages of factor graphs employed as filters is that multiple iterations can be performed to achieve the estimate of the state. In fact, as mentioned at the end of Section 3.2.1, a Gauss-Newton algorithm can be exploited. Once the correction $\boldsymbol{\Delta x}_k^*$ has been computed it can be applied to the nominal state $\boldsymbol{x}_k$ and the estimation can be performed again. Eventually this process will converge to the estimate $\boldsymbol{x}_k^{MAP}$. To limit the computations involved in this process it is convenient to start the iterations after the elimination of the variable $\boldsymbol{\Delta x}_{k-1}$, that is the step represented in Figure 3.10b. In fact the intermediate factor τ contains all the information coming from previous estimates. After the elimination of the variable $\boldsymbol{\Delta x}_k$ (step in 3.10c), the estimate $\boldsymbol{\Delta x}_k^{*(1)}$ is performed (together with the computation of its covariance $\boldsymbol{\Sigma}_k^{(1)}$), where the superscript (1) indicates the first iteration. Now assuming $\boldsymbol{x}_k^{(1)} = \hat{\boldsymbol{x}}_k^{(1)} + \boldsymbol{\Delta x}_k^{*(1)}$ as new linearization point $\hat{\boldsymbol{x}}_k^{(2)}$, a prior factor ϕ_k can be computed as (3.46) (and substituting τ), and the factor ϕ_{GNSS} is computed again as (3.50), with $\boldsymbol{H}_k$ and $\hat{\boldsymbol{\zeta}}_k$ computed with respect to the new linearization point $\hat{\boldsymbol{x}}_k^{(2)}$. This process can be repeated until convergence.

The condition of conference has to be defined with respect to a given threshold t. In particular the iteration process stops when:

$$\frac{\|\boldsymbol{\Delta x}_k^{*(n)} - \boldsymbol{\Delta x}_k^{*(n-1)}\|}{\|\boldsymbol{\Delta x}_k^{*(n-1)}\|} < t \tag{3.76}$$

end

$$\boldsymbol{x}_k^{MAP} = \hat{\boldsymbol{x}}_k^{(n)} + \boldsymbol{\Delta x}_k{}^{(n)} = \hat{\boldsymbol{x}}_k{}^{(1)} + \sum_{j=1}^{n} \boldsymbol{\Delta x}_k^{*(j)} = \hat{\boldsymbol{x}}_k{}^{(1)} + \boldsymbol{\Delta x}_k^{MAP} \tag{3.77}$$

where $\boldsymbol{\Delta x}_k^{MAP}$ is a comprehensive single correction to be applied to the initial linearization point $\hat{\boldsymbol{x}}_{\boldsymbol{k}}{}^{(1)}$ to obtain $\boldsymbol{x}_k^{MAP}$. Decreasing the value of the threshold t increases the number of iterations needed to achieve convergence, hence the computation time, but the solution is expected to be improved. In the next chapter the effect of different values of t will be analyzed.

Once the final correction $\boldsymbol{\Delta x}_k^{MAP}$ is obtained, it can be propagated through the Bayes net to perform smoothing. As described in Section 3.2.3, this is done via back substitution using the equation (3.41), reported and adapted to the context here:

$$\boldsymbol{\Delta x}_{k-lag}^{MAP} = \boldsymbol{R}_{k-lag}^{-1}(\boldsymbol{d}_{k-lag} - \boldsymbol{T}_{k-lag}\boldsymbol{\Delta x}_{k-lag+1}^{MAP}) \tag{3.78}$$

where $\boldsymbol{R}_{k-lag}$, $\boldsymbol{T}_{k-lag}$ and $\boldsymbol{d}_{k-lag}$ are obtained for each variable when it is eliminated. The variable lag spans from 1 to a defined n_{lag}. Increasing n_{lag} leads to a larger number of computations as well as a larger number of parameters ($\boldsymbol{R}$, $\boldsymbol{T}$ and $\boldsymbol{d}$) to be stored, but the accuracy level of the solution is expected to increase.

It is clear that a trade-off between performances and computational cost has to be handled. This topic will be discussed in the next chapter, together with a comparison with the Kalman filter.

Chapter 4

Field testing

In the previous chapters an overview of GNSS and INS technologies as well as their mathematical description has been provided. Then the advantages of their integration have been discussed together with an overview of the possible integration strategies that can be adopted. Among them, the tight integration has been chosen for this thesis. This architecture is based on a navigation filter in which the fusion of the measurements coming from the two different technologies is performed. In this thesis a different approach with respect to the most traditional Extended Kalman Filter (EKF) has been adopted: the factor graph framework. Its characterization was discussed in details in Chapter 3, starting from the general mathematical descriptions and then adapting it to the case of GNSS/INS tight integration. The factor graph solution implemented in this thesis has two parameters that can be tuned: a threshold t, defining the convergence of the Gauss-Newton algorithm; and n_{lag}, defining the number of epochs on which the smoothing is performed. In this chapter, a comparison between the different solutions is performed.

The different solutions are obtained using the software MATLAB, in which a GNSS/INS tight integration performed with an EKF was already available. A factor graph framework to provide GNSS stand-alone and GNSS/INS integrated solutions has then been added. The dataset considered in analysis was collected during a car ride in a urban area nearby Politecnico di Torino (Torino, Italia). The employed GNSS-receiver is a low cost NVS NV08C-CSM while inertial measurements (specific force and angular velocity) are collected with a low cost MEMS-technology strapdown IMU sensor: TDK Invensense MPU-9250, composed of two triads of accelerometers and gyroscopes. Even if the discussion performed in this thesis does not put any constraint on the satellite constellations, the obtained dataset involves only GPS satellites. The adopted hardware was mounted on the vehicle platform and the measurements collected synchronously. This setup provided a dataset composed of 1740 GNSS epochs in total. Additionally, also a ground-truth was generated. It is a highly accurate description of the followed trajectory, obtained with a multi-frequency, multi-constellation GNSS-receiver of the Novatel OEM7 family (user manual available at [40]). This receiver uses a combination of a tactical-grade IMU and Real-time kinematic positioning (RTK) technique, supplying real-time localization at sub-centimeter level accuracy. The obtained reference trajectory is showed in Figure 4.1. The availability of a reference trajectory will allow an effective comparison between the different possible trajectory estimations.

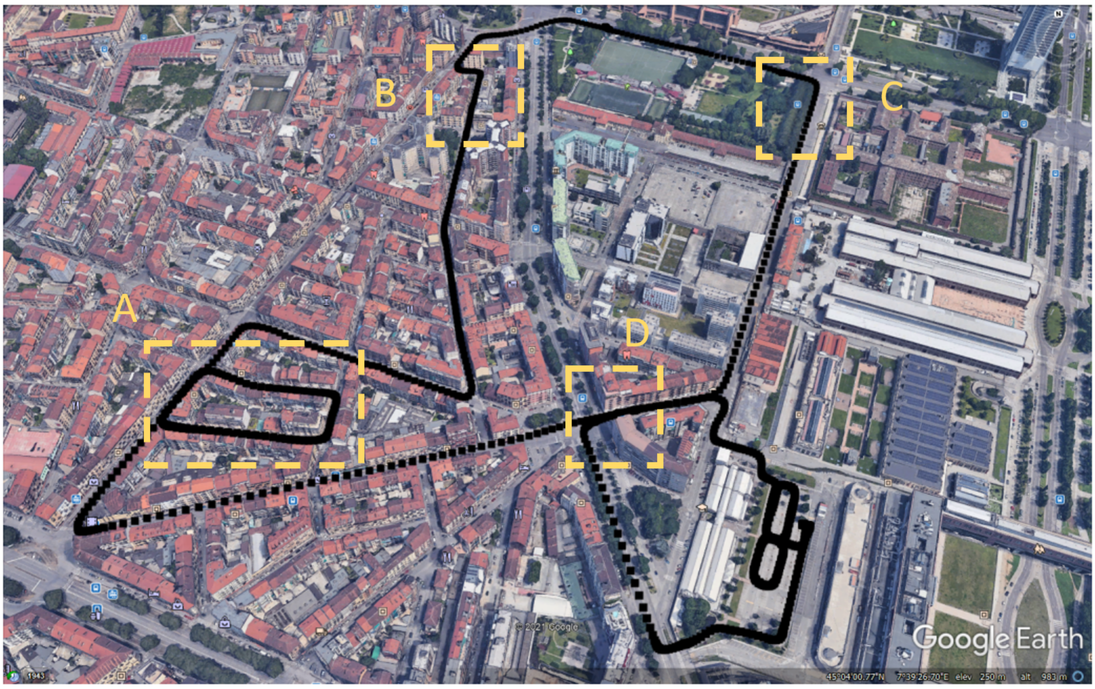
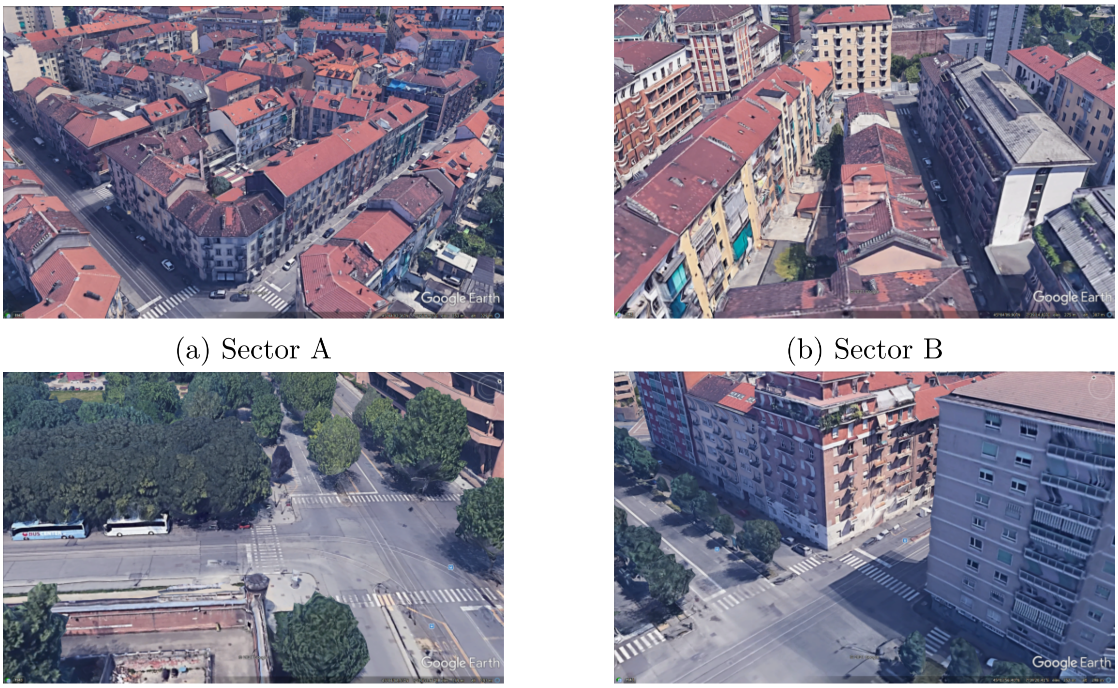


Figure 4.1: round-truth trajectory (Google Earth)

In Figure4.1, representing the reference trajectory, there are four highlighted zones because in this urban scenario there are some segments (sectors) of particular interest due to their challenging characteristics. They are represented in Figure 4.2 and here described:

- Sector A and Sector B: poor satellites' visibility due to the surrounding buildings, expected presence of multipath and shadowing effects (Figures 4.2a and 4.2b).
- Sector C: good visibility condition but presence of dense foliage (Figure 4.2c).
- Sector D: presence of highly reflective and diffractive surfaces (Figure 4.2d).



(a) Sector A

(b) Sector B

(c) Sector C

(d) Sector D

Figure 4.2: Zoomed challenging urban areas (Sectors). (Google Earth images)

The state estimated at each epoch is the one defined in equation (2.1), that is a vector of dimension 17. However, the relevant components for a comparison with the reference trajectory are the first three, regarding the estimate of the position, that is the one used to retrieve the estimated trajectory. In fact the reference trajectory is given in LLA (Latitude(deg), Longitude(deg), and Altitude(m) coordinates, assuming earth's surface WGS84-model), while the position estimate is in the ECEF frame ($x(m)$, $y(m)$, $z(m)$). In order to perform a comparison and analyze the results,it is convenient to express both in a ENU (East(m), North(m), Up(m)) frame, that is defined on the plane tangent to the earth's surface in the local position. The three axis are the vertical axis (pointing externally) defining the plane itself and the other two, laying on the plane, pointing east and north respectively. Then, a position $\boldsymbol{p}_{\alpha}^{(k)}$, computed using some α-filter, and a position $\boldsymbol{p}_{ref}^{(k)}$, obtained from the reference trajectory, will be available at each k-epoch.

$$\boldsymbol{p}^{k} = [p_x^{(k)}\ p_y^{(k)}\ p_z^{(k)}]^T \tag{4.1}$$

The first two components in equation (4.1) are fused together as in (4.2) to obtain the horizontal position $\boldsymbol{p}_{(h)}^{(k)}$ (that is the trajectory seen from above), that will be compared to the horizontal reference trajectory. Instead the vertical positioning (corresponding to the subscript (v)), that is affected by heavier uncertainty (due to geometry arising in the multilateration problem), will be compared separately.

$$p_{(h)}^{(k)} = \sqrt{(p_x^{(k)})^2 + (p_y^{(k)})^2} \tag{4.2}$$

To compare the the obtained solutions to the reference one, it is useful to define the total, horizontal and vertical error affecting the estimated trajectory with respect to the reference one at each epoch one as:

$$\begin{aligned} e^{(k)} &= \sqrt{\left(\boldsymbol{p}_{\alpha}^{(k)} - \boldsymbol{p}_{ref}^{(k)}\right)^T \left(\boldsymbol{p}_{\alpha}^{(k)} - \boldsymbol{p}_{ref}^{(k)}\right)} \\ e_{(h)}^{(k)} &= \sqrt{\left(\boldsymbol{p}_{(h),\alpha}^{(k)} - \boldsymbol{p}_{(h),ref}^{(k)}\right)^T \left(\boldsymbol{p}_{(h),\alpha}^{(k)} - \boldsymbol{p}_{(h),ref}^{(k)}\right)} \\ e_{(v)}^{(k)} &= \sqrt{\left(p_{(v),\alpha}^{(k)} - p_{(v),ref}^{(k)}\right)^2} \end{aligned} \tag{4.3}$$

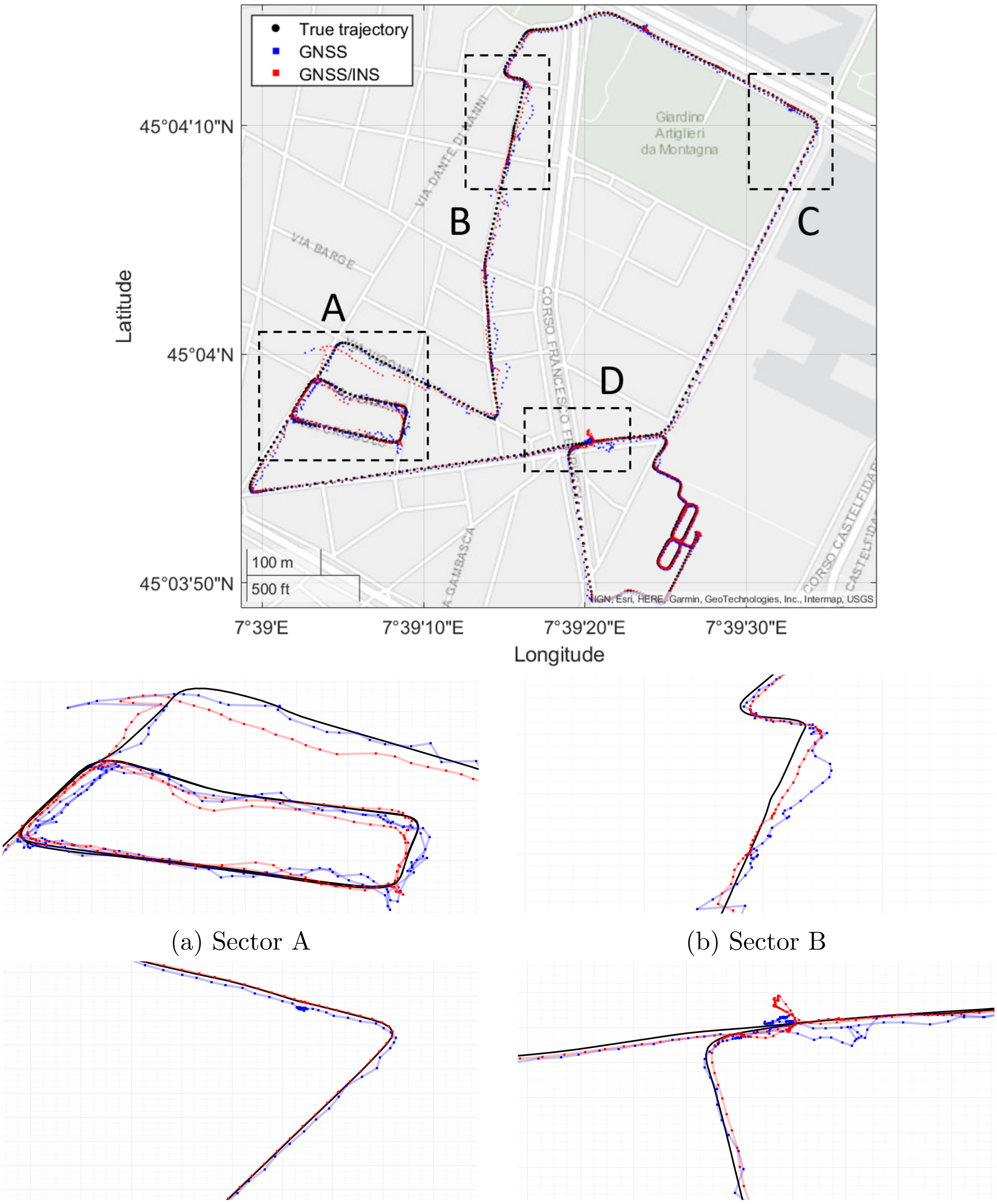
In this chapter first the accuracy of a GNSS stand-alone solution is compared with a GNSS/INS tight integration solution using factor graphs for both, to highlight the advantages coming from fusing different positioning sensors. This will be done in Section 4.1. Then in Section 4.2 the equivalence of a filter implemented through a certain set-up of factor graphs and the EKF is shown, followed by an analysis of the effect of multiple iterations performed at each epoch . Finally in Section 4.2.2 the fixed-lag smoothing of the solution is performed in a factor graph framework and the obtained results will be discussed.

4.1 Stand-alone GNSS and GNSS/INS integrated solution

The comparison between a stand-alone GNSS solution and a GNSS/INS tightly coupled integrated solution is performed in this section. This is done to highlight the advantages of including INS in a positioning system.

In Figure 4.3 the trajectories obtained with the two different solutions is shown together with the reference trajectory. It can be seen how the INS allows the navigation filter to estimate a smoother trajectory. Sectors A,B and D, 4.3a, 4.3b and 4.3d respectively, show in particular how the the INS helps to estimate a trajectory closer to the ground-truth even in challenging environments. This is because a GNSS alone solution is too affected by the external environment in such sectors.

In Figure 4.4 the horizontal and vertical errors over time are represented. Looking at the horizontal component 4.4a, it can be seen how the GNSS stand-alone



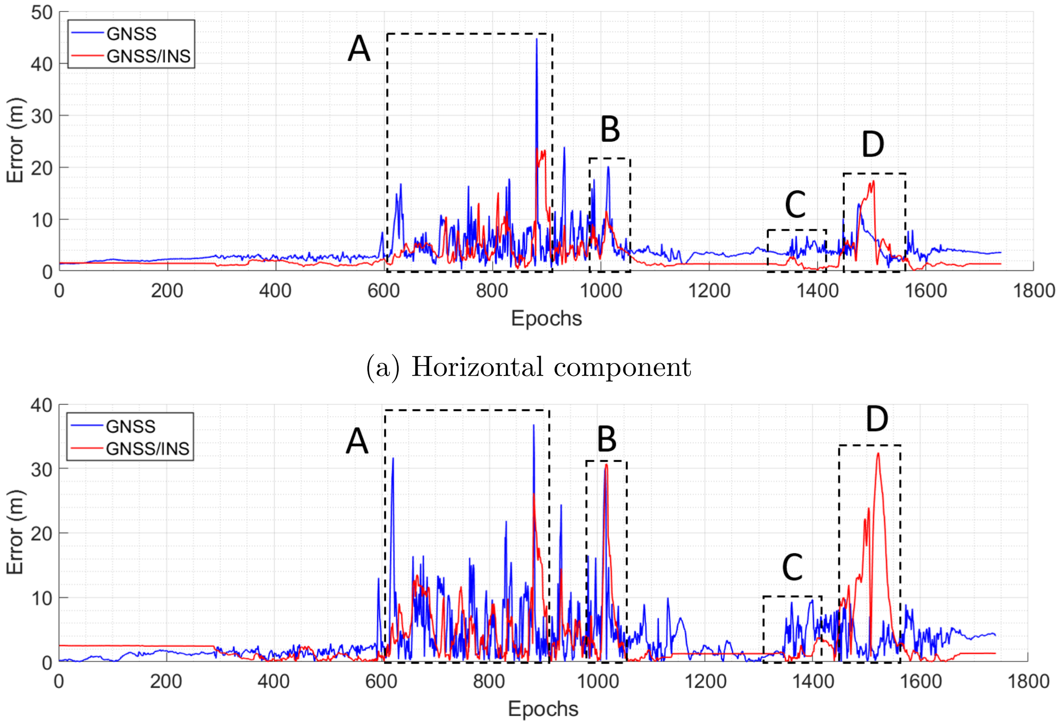
(a) Sector A

(b) Sector B

(c) Sector C

(d) Sector D

Figure 4.3: 2-D trajectories (latitude/longitude) obtained through a stand-alone GNSS and a tight integrated GNSS/INS, w.r.t. ground-truth.



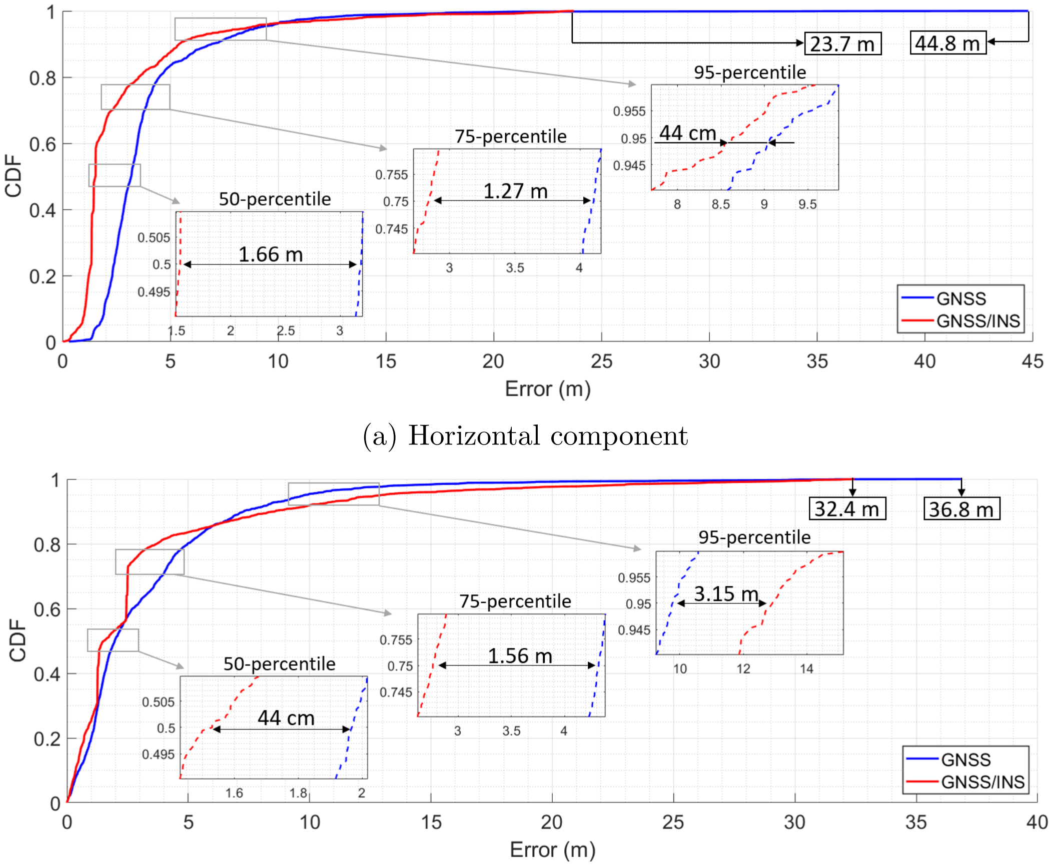
(a) Horizontal component

(b) Vertical component

Figure 4.4: Error on the horizontal and vertical position in ENU-coordinates of GNSS only and tight integrated GNSS/INS solutions, w.t.t. ground-truth.

solution presents several peaks in sectors A and B in both components. This is due to reflection and diffraction effects together with the reduced visibility caused by the the urban architecture. In these sectors the presence of the IMU sensors improves the solution consistently. Sector C still presents some improvement with the GNSS/INS, but of less magnitude with respect to the other sectors. Finally, in sector D both solution are characterized by a similar error magnitude. On the other hand, the vertical component 4.4b shows the vertical component of both solutions. The behaviour of this component is similar to the one described for the horizontal component. The main difference is in sectors B and D: The former shows that the higher peak is shared by both GNSS and GNSS/INS solutions, so the INS is not able to mitigate that particular outlier; in the latter it can be seen how is the GNSS stand-alone solution to perform better.

To better compare the horizontal and vertical solution of the two estimation methods it is useful to visualize the Error Cumulative Density Function (ECDF) computed w.r.t. the ground-truth, describing how the error is distributed along the whole trajectory. Looking at Figure 4.5 it is evident how the addiction of



(a) Horizontal component

(b) Vertical component

Figure 4.5: ECDF of the horizontal and vertical positions in ENU-coordinates of GNSS stand-alone and tight integrated GNSS/INS solutions, computed w.t.t. the reference trajectory.

INS to GNSS leads to a great improvement in the level of accuracy.In fact the former consistently maintains a leftward position relative to the latter, except for a segment in the higher percentiles in the vertical component meaning that given most percentiles, Only at the 95-percentile the GNSS stand-alone vertical solution seems to perform better than the GNSS/INS one, but this is true only for a small fraction of the ECDF curve. It is even more evident looking at the maximum error present in the two solution: with the presence of the INS the maximum error in the horizontal component is reduced of 21.1 m that is a reduction of 47%, while in the vertical one it is reduced of 4.4 m (12%).

Finally, a global characterization of the two solutions is given by the box-plot in Figure 4.6. The box-plot considers the horizontal and vertical components simultaneously. Here the trend of the median and mean of the errors affecting each solution are represented and, for each solution, the 25-th and the 75-th percentiles are represented as upper and lower bound of each light blue box. In particular the median and the mean, starting from 3.84 m and 5.40 m, are reduced of 1.28 m (33%) and 85 cm (16%) respectively. The distribution of the errors is also closer to zero and the outliers are mitigated.

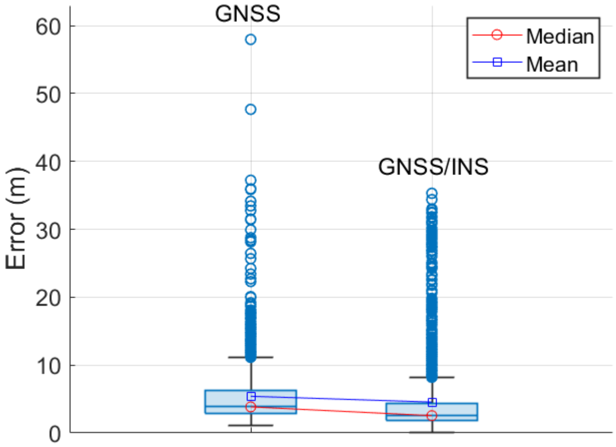


Figure 4.6: Box-plot of the overall error affecting the solutions obtained with different numbers of iterations

4.2 Extended Kalman filter and factor graph comparison

In this section the performances obtained with a GNSS/INS tight integration using a filter based on factor graphs, changing the parameters n_{lag} and t, are compared to the ones characterizing the most common EKF. First, the equivalence between the solution obtained with an EKF and the one obtained with a certain set-up of the factor graph framework is showed. As explained in Section 3.4, at each k epoch, eliminating the node at $(k-1)$ is equivalent to the prediction step, while eliminating the node at k is equivalent to the correction step. Hence, obtaining the MAP estimate of the last node in just one step (no iterations) and without propagating it the older nodes (no smoothing), produces the same solution as the one obtained with an EKF. This factor graph set-up will be indicated as standard factor graph (FG). As can be seen in Figure 4.7, the trajectories obtained with the two different filters are completely overlapped.

Similarly, to compare them with grater accuracy, Figure 4.8 shows their error for each epoch. Again the two curves appear completely overlapped. proving the equivalence of the two considered filters.

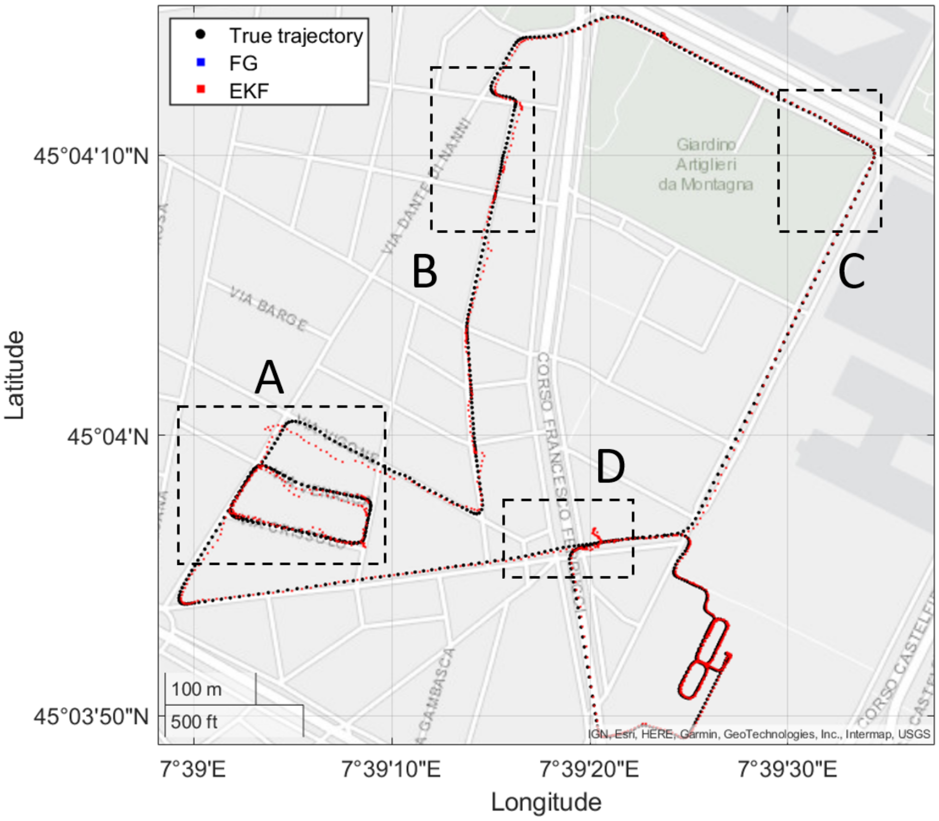


Figure 4.7: 2-D experimental trajectories (latitude/longitude) obtained through an EKF and a FG, w.r.t. ground-truth.)

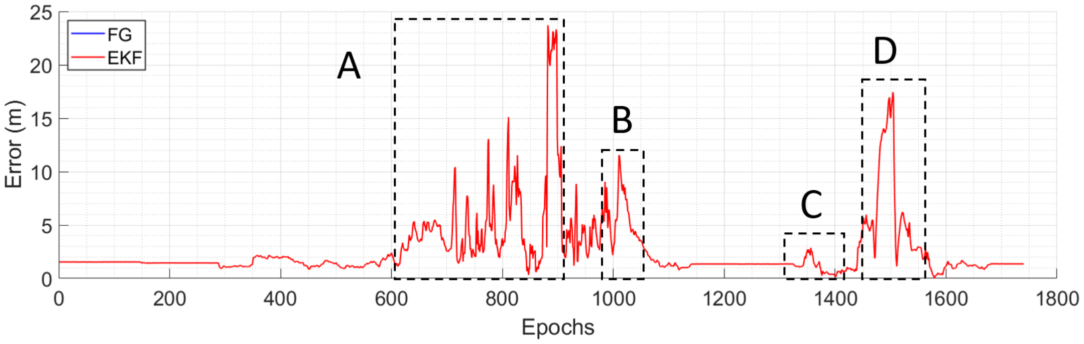


Figure 4.8: Error on the vertical position in ENU-coordinates of EKF and FG solutions, computed w.t.t. the reference trajectory.

4.2.1 Gauss-Newton approach

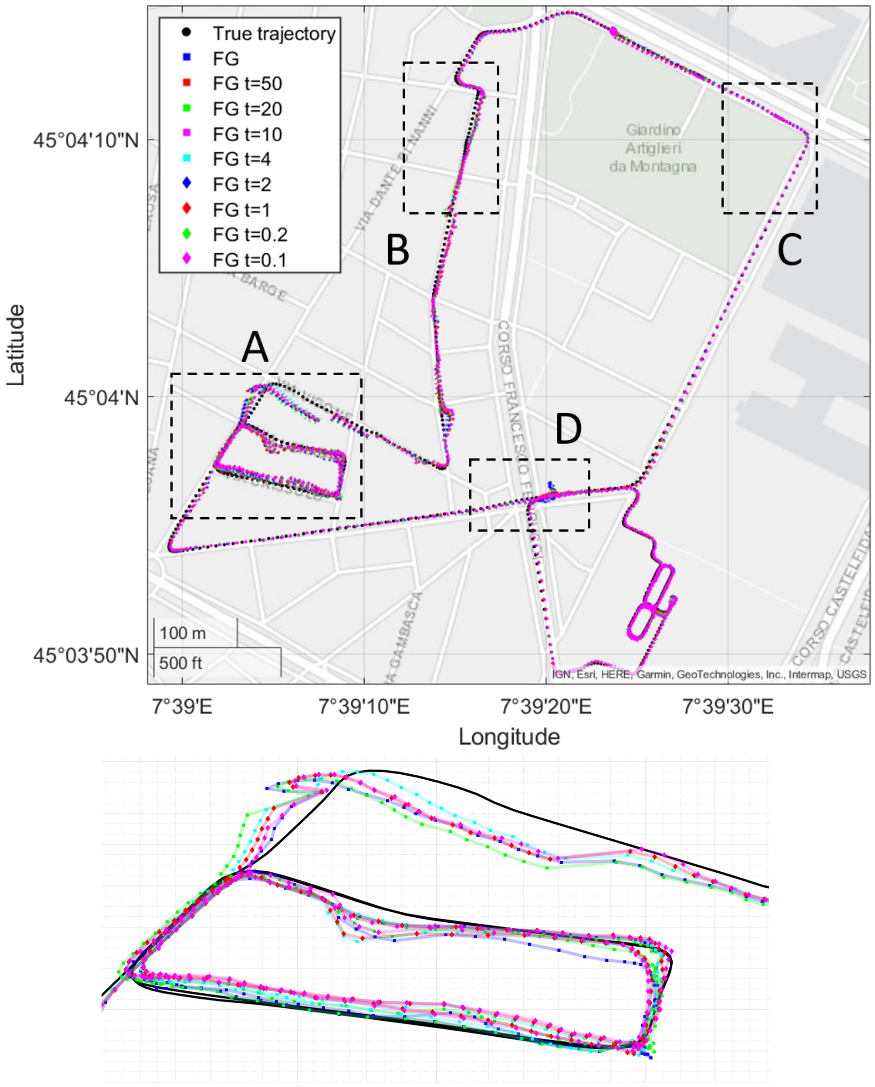
In this section the number of nodes considered at each epoch is fixed to 1 (i.e. $n_{lag} = 0$), meaning that there is no smoothing over the previous epochs. Instead the analysis is focused on the effect of applying the Gauss-Newton algorithm to obtain a refined MAP estimate of the current state, as explained in Section 3.5. That algorithm consists in eliminating the last node multiple times obtaining different corrections to be applied to the full state, until convergence is reached. The convergence itself depends on a predefined threshold t: the lower it is, the more iterations will be performed. The values of the parameter t that has been considered in this work are represented in table 4.1, together with the average number of iterations consequently performed at each epoch. The analysis includes also a set-up of the FG, hereinafter standard FG, equivalent to an EKF.

t(%)	Average iterations per epoch
50	3
20	5
10	11
4	29
2	59
1	101
0.2	324
0.1	558

Table 4.1: Average iterations performed at each epoch for different values of threshold t

In Figure 4.9 the different obtained trajectories are showed, while the zoom in is performed only on sector A and just for a few solutions for clarity. Increasing the number of iterations, the obtained trajectories have a similar behaviour, but they are slightly shifted to the ground-truth solution, as can be seen especially in sector A, Figure 4.9a.

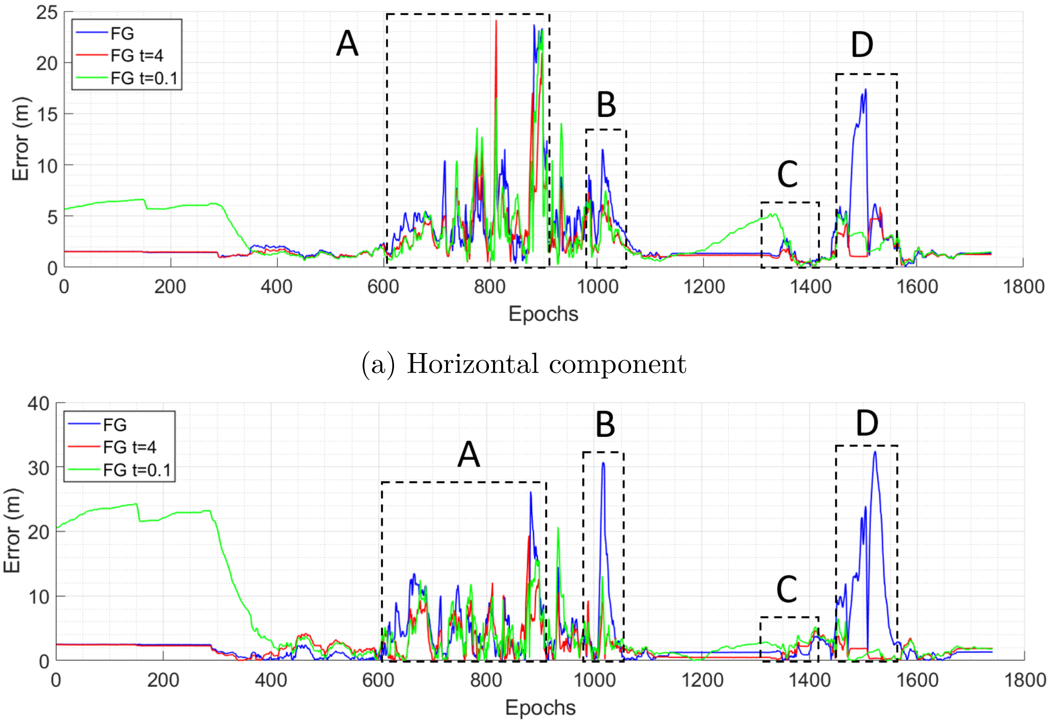
To better analyze the solution, it is convenient to refer to Figures 4.10 showing the horizontal and vertical error over time. Only three solutions are represented for simplicity: FG (no iterations), FG $t = 4$ and FG $t = 0.1$, so that the effect of a medium and the maximum number of iterations can be compared to the standard solution over time. The first thing that can be noticed is that at the beginning of the trajectory and before sector C, the solution with $t = 0.1$ shows the presence of greater errors. This fact suggests that imposing a too low threshold's value leads to a worsening of the accuracy level of the solution in some part of the trajectory,



(a) Sector A

Figure 4.9: 2-D experimental trajectories (latitude/longitude) obtained with different numbers of iterations performed at each epoch, w.r.t. ground-truth.

since it introduce additional errors. This is due to some characteristics of the dataset that makes the Gauss-Newton algorithm to reach a sub-optimal minimum in this regions. In these same segments, however, the solution with $t = 4$ shows a better performance with respect to the standard FG, as the errors have the same behaviour but with a slightly lower magnitude. In sector A the performances are similar among the different solutions. In the horizontal component there is a peak emerging in the FG $t = 4$ solution, that is then mitigated performing a larger number of iterations (i.e. FG $t = 0.1$). Finally, in sectors B and D a significant improvement on the level of accuracy can be noted, for both the vertical and horizontal components.



(a) Horizontal component

(b) Vertical component

Figure 4.10: Error on the horizontal and vertical positions in ENU-coordinates of FG solution obtained with different numbers of iterations performed at each epoch, computed w.t.t. the reference trajectory.

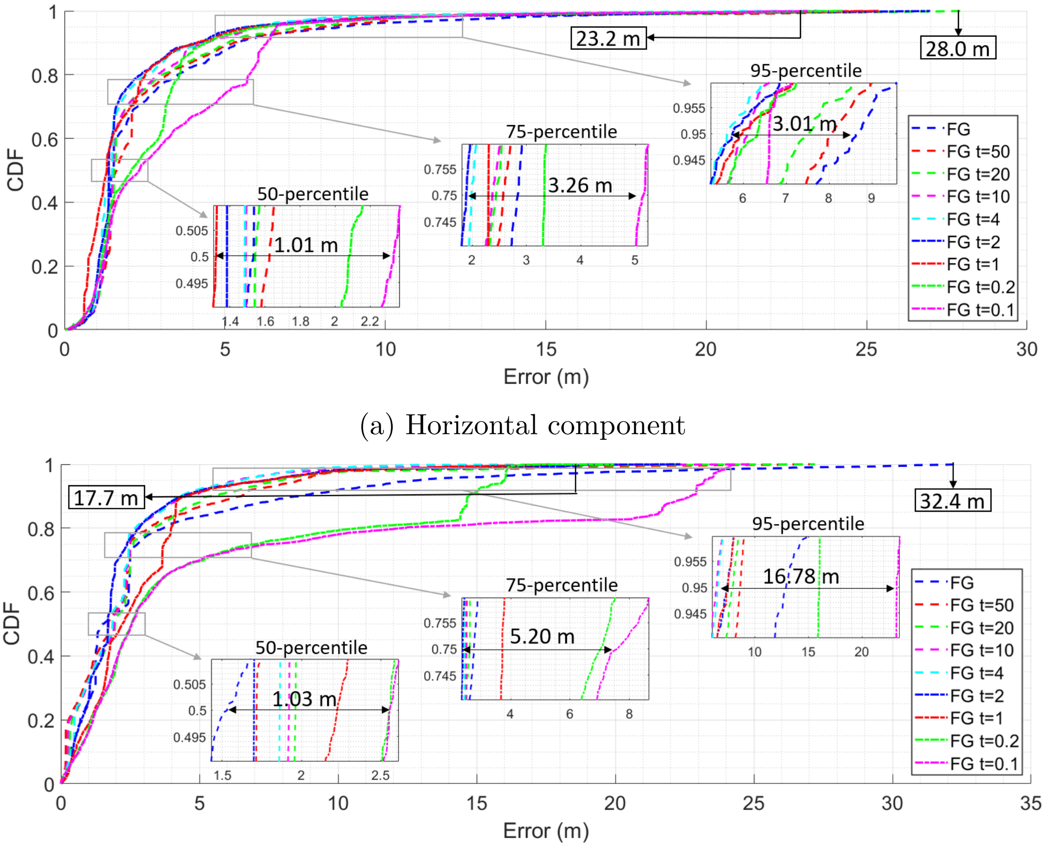
To perform a quantitative comparison among the different solutions it is convenient to analyze the ECDFs in Figure 4.11. Figure 4.11a regards the horizontal component while Figure 4.11b the vertical one. In both figures the 50-th, 75-th and 95-th percentiles are zoomed in, and the absolute variation among the best and worst solutions in each of them is reported.

Starting from the horizontal component, it can be seen that for the lower portion of the curves the solution obtained with FG $t = 1$ stands always to the left side of the other curves, meaning that this solution present a lower error where the trajectory is affected by errors of low intensity. The solutions with a higher number of iterations (FG $t = 0.2$ and FG $t = 0.1$) are, in this portion of the plot till the 90-th percentile almost always to the right side of the other curves, reflecting the fact that in the segments with lower errors these solutions are instead presenting greater ones, as analyzed before. Instead, in the higher portion of the plot, from the 95-th percentile on, they get closer to the other curves. It is in fact FG $t = 0.1$ that has the best performance in terms of maximum horizontal error, with a value of 23.2 m, slightly less then 23.7 m obtained with the standard FG. Instead the greater maximum error of 28 m is given by FG $t = 0.2$. From the 50-th percentile on, the best solution is given by FG $t = 4$, that left-bounds the over curves in this region until the very last percentiles. Overall, the solutions given by the thresholds from $t = 20$ to $t = 1$ stand to the left side of the standard FG, showing an improvement of the solution.

An analogous analysis can be done for the vertical component. For the lower part of the plot, the curve that stands most to the left side is FG $t = 50$, until the 50-th percentile where the best solution is given by the standard FG, but for just a small portion of the plot, followed by FG $t = 2$. However, from this point on, the best solution is given again by FG $t = 4$ and $t = 10$. In the final part, it is the former solution that gives the best performance in terms of maximum error, that is of 20 m, second just to 17.7 m given by $t = 1$. It can be noticed that the greater maximum error of 32.4 m is given by the standard FG solution, hence performing multiple iterations allows to mitigate the highest error peaks present in the vertical component of the trajectory. Except for the solutions with higher number of iterations $t = 1$, $t = 0.2$, $t = 0.1$, that give high errors for most of the ECDF plot and only get better in the last portion, the overall plot shows that performing multiple iterations to compute MAP estimate lead to a better estimation, that translates in shifting the curves to the left side of the ECDF plot.

To better compare the different solutions, their behaviour at the 50-th, 75-th and 95-th percentiles can be plotted in histograms for both the horizontal and vertical components as in Figures 4.12a and 4.12b. Both plots show a similar behaviour, in accordance to the ECDF curves. Decreasing the threshold t, that is incrementing the number of iterations, there is generally a reduction of the error affecting the solution in every percentiles, particularly in the 75-th and 95-th ones. What can also be seen is that the last three solutions, that are $t = 1$, $t = 0.2$ and $t = 0.1$, show instead an increment of the error in these percentiles. This is still related to the degraded performances in the segments at the start and before sector C of the trajectory. Hence, the solutions obtained with a higher number of iterations are characterized by poorer performances related to the trajectory segments affected by lower errors, while succeed in mitigate the errors where they are present in greater intensity, as was showed in the ECDF curves 4.11a and 4.11b.

To summarize the observations reported above, a box-plot describing the errors affecting the different solutions can be exploited, figure 4.13. Starting from the standard FG solution, characterized by a mean of 4.56 m and a median of 2.56 m, these values keep decreasing until a minimum of 3.20 m (-30%) is reach for the



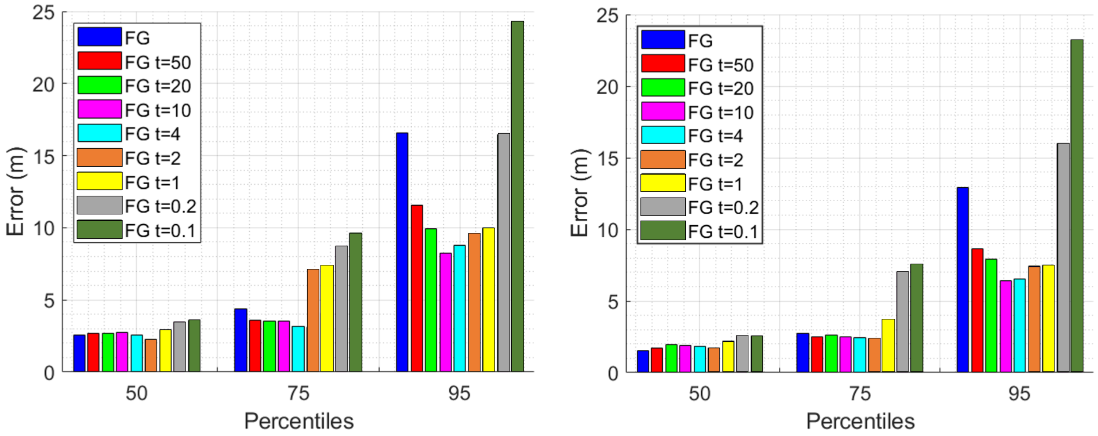
(a) Horizontal component

(b) Vertical component

Figure 4.11: ECDF of the horizontal and vertical positions in ENU-coordinates of FG solution obtained with different numbers of iterations performed at each epoch, computed w.t.t. the reference trajectory.

mean at FG $t = 4$ while the median finds its minimum value of 2.27 m (-11%) at FG $t = 2$. this behaviour is shared also with the percentiles. Instead for the last three solutions these values start growing again. In this plot the maximum values of the error are represented as outliers. It can be seen that increasing the number of iterations allows in general to decrease the maximum error, with the best performance in this sense obtained with $t = 4$, followed by $t = 0.1$.

The performed analysis on the effect of performing multiple iterations at each epoch shows how the level of accuracy is in general increased adapting a Gauss-Newton algorithm. Setting a more stringent threshold leads to a better performance in terms of mean and median of the overall error, as well as a mitigation of the error peaks. In particular, the considered trajectory benefits of this feature



(a) Horizontal component

(b) Vertical component

Figure 4.12: Histogram representing the the horizontal error affecting the solutions obtained with different numbers of iterations at 50-th, 75-th and 95-th percentiles.

mostly where the standard solution is less accurate (i.e. sectors B and D). However, as the threshold defining the convergence of the algorithm gets lower, the solution, in terms of mean and median is worsened, while the maximum error over the whole trajectory is kept relatively low.

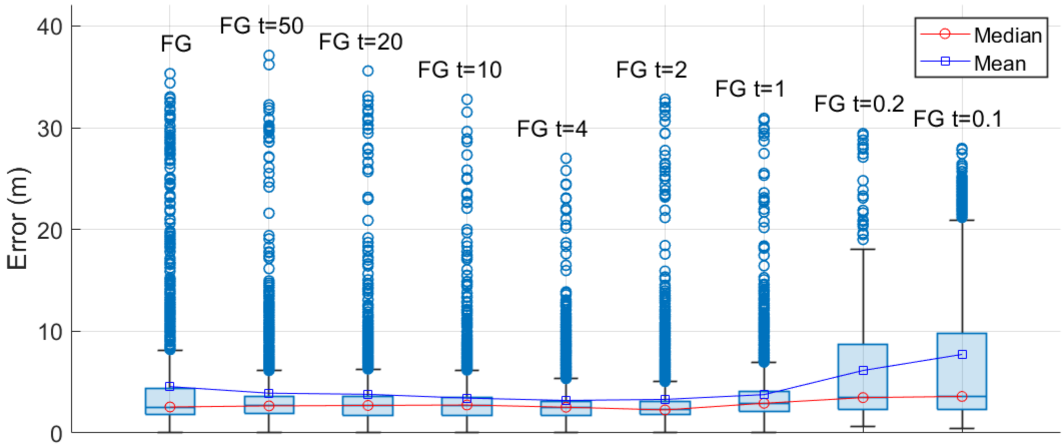


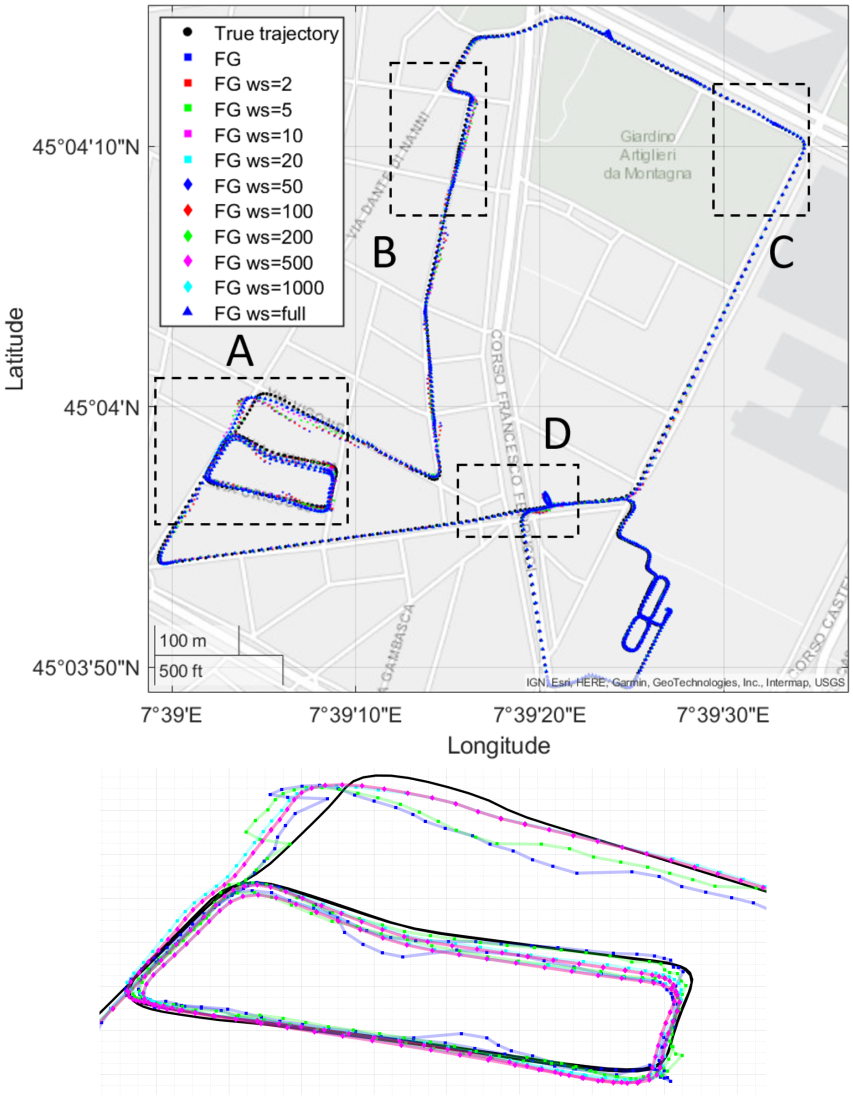
Figure 4.13: Box-plot of the overall error affecting the solutions obtained with different numbers of iterations

4.2.2 Fixed-lag smoothing

In this section the effect of introducing a smoothing operation to obtain the solution is analyzed. Here the number of iterations is fixed to 1, so that the effect can be compared to the standard factor graph filtering process, that is equal to the extended Kalman filter. The parameter involved in this analysis is the already introduced n_{lag}, that represents how old is the last state that is refined through the smoothing process at each epoch. From now on, however, the considered parameter will be the more representative window size ws, describing the number of nodes (states) kept in the evolving Bayes net retrieved from the factor graphs at each epoch, as described in Section 3.2.3. These two parameters are trivially related as $ws = n_{lag} + 1$, since ws include also the new node at each epoch. What will be called standard FG is just the solution obtained with $ws = 1$. Instead, the maximum number of possible nodes to be included is the full trajectory, composed of 1740 nodes. The solution obtained in such configuration will be indicated as $ws = full$.

A first overview on the effect of the smoothing process can be seen looking at Figure 4.14, representing the trajectories obtained for different window sizes. The values of ws considered in this thesis are $ws = [2,\ 5,\ 10,\ 20,\ 50,\ 100,\ 200,\ 500,\ 1000,\ full]$. Differently from the effect of performing multiple iterations, the smoothing process leads to a change in the trajectory shape, making it looking smoother. This can be seen for example in Figures 4.14a, reporting a zoom in sector A. Another thing that can be noticed is that the difference between trajectories obtained with higher values of ws are hardly noticeable.

To analyze the effect of different window sizes in the solution along the trajectory it is useful to refer to Figure 4.15a and 4.15b, where the errors affecting the trajectories, decomposed in horizontal and vertical components respectively, are represented. For simplicity of visualization only the values of $ws = 20,\ 500$ are considered, together with the standard FG filter corresponding to $ws = 1$. Starting from the horizontal component, in the initial portion of the trajectory there is a similar behaviour among the different solutions, with some oscillations around the FG curve introduced by the smoothing in the very first trait. This effect can be seen more accentuated in the portion between sectors B and C. In the critical sectors A and B there is an evident decrease in the magnitude of the errors. In fact here the curve associated with the smoothed solutions are almost always below the one associated with the standard FG solution, and, moreover, in the smoothed solutions the error has a smoother behaviour, without the presence of the evident peaks occurring in the standard solution. Instead, there is not a noticeable decrement in the error magnitude in sectors C and D, but the error trend looks smoothed here too. The vertical component is affected in a similar way by the smoothing: together with sectors A and B, also sector C and D present

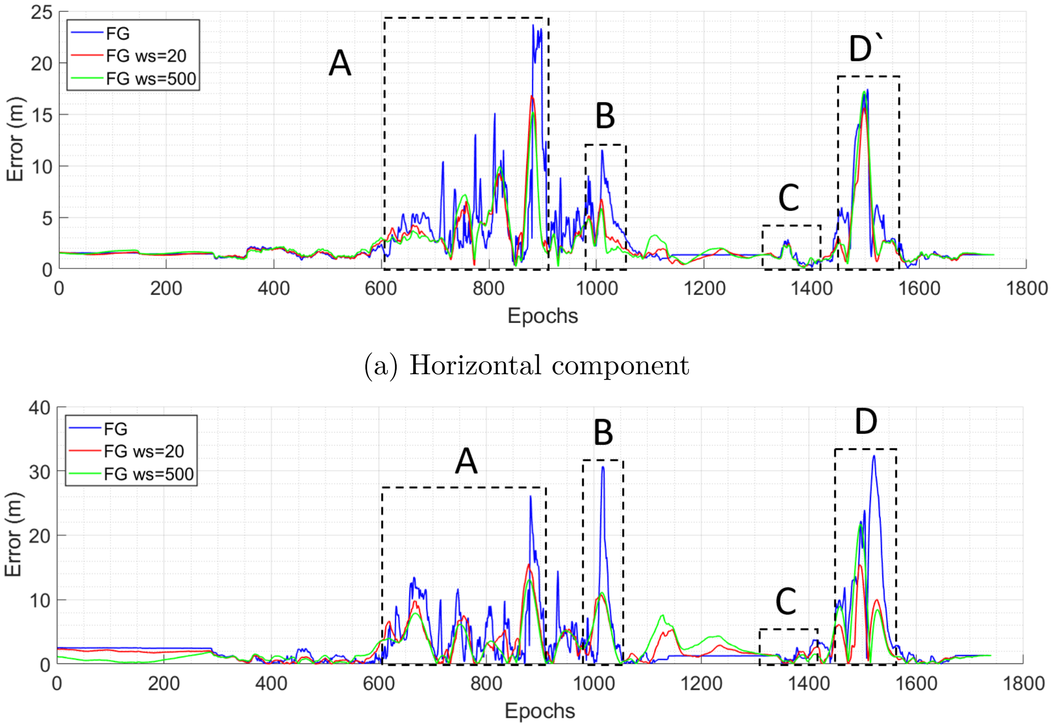


(a) Sector A

Figure 4.14: 2-D experimental trajectories (latitude/longitude) obtained with different numbers of ws, w.r.t. ground-truth.

a visible accuracy increase. However, the portion of the trajectory between B and C presents a degradation of the quality, presenting higher error values.

The average error distribution can now be analyzed referring to Figure 4.16a, ECDF of the horizontal component and Figure 4.16b, ECDF of the vertical one. Both curves have one thing in common: up to 70-th percentile, the curves until $ws = 20$ have an analogous behaviour, that is different from the one shared by



(a) Horizontal component

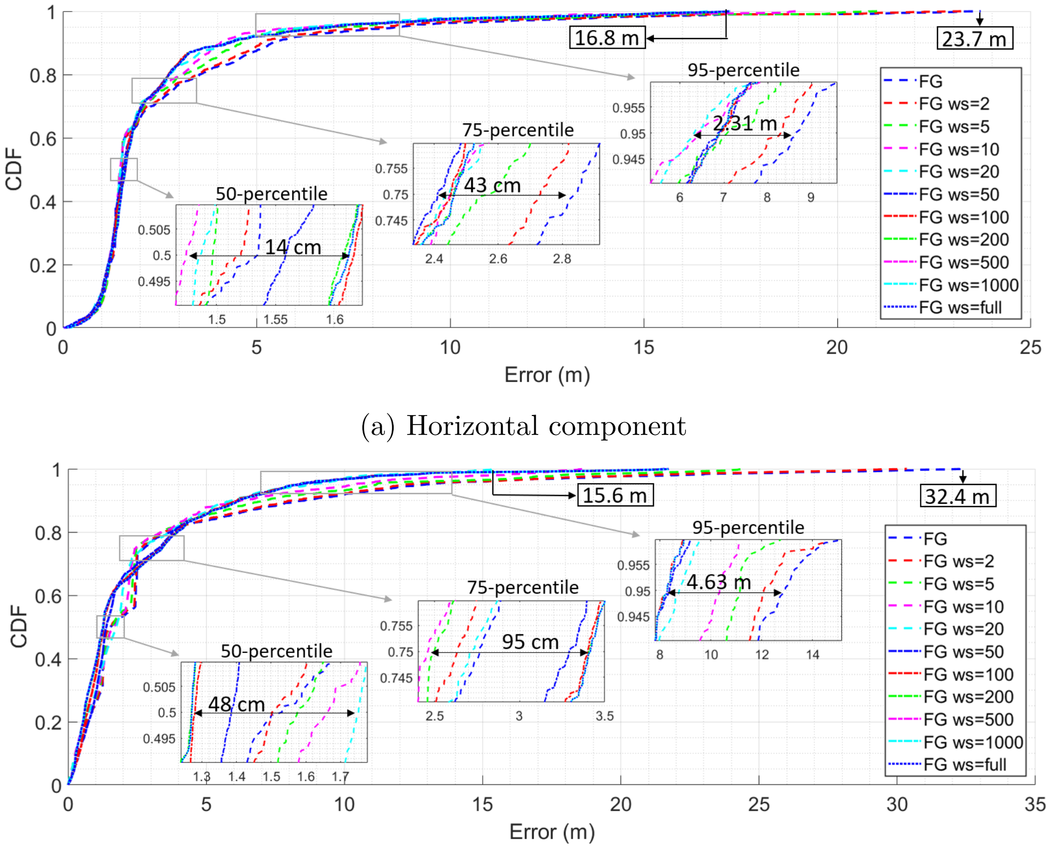
(b) Vertical component

Figure 4.15: Error on the horizontal and vertical positions in ENU-coordinates of solutions obtained obtained with different window sizes, computed w.t.t. the reference trajectory.

the curves from $ws = 200$ to $ws = full$, that are overlapped to millimeter level. Instead, in this region, the curves corresponding to $ws = 50$ and $ws = 100$ have an intermediate behaviour; eventually they will overlap to the ones with grater window sizes. The lower part of the horizontal ECDF is characterized by a small difference in terms of error magnitude (the maximum difference in the 50-th percentile is just of 14 cm). However, around the 50-th percentile it can be noticed how the curves characterized by higher values of window sizes stand to the right side, being described by a slightly higher error, while for the remaining curves, increasing the window size results in a slightly better accuracy level. Looking at the higher part of the plot, from the 70-th percentile on, the differences between the curves start becoming more evident. The standard FG curve, together with $ws = 2$ curve are left-bounded by the others, proving their worst capability to mitigate higher error values. The curves from $ws = 50$ to $ws = full$ can now be considered as a whole. Finally the remaining curves $ws = 5$, $ws = 10$ and $ws = 20$ present similar behaviours, but considering just those three the accuracy increase slightly by increasing the window size. The minimum maximum error of 16.8 m is achieved by $ws = 20$, slightly less of 17.2 m obtained with higher values of window sizes, but fairly better then 23.7 m obtained by the standard FG (the maximum error is decreased by 29%). An analogous trend can be seen for the vertical component in Figure 4.16b. Looking at the zoomed percentiles, at the 50-th the curves with values of $ws = 100$ and higher perform slightly better then the others, while for the other curves there is a reversed trend: higher is the ws, worse is the accuracy level. This situation presents itself completely mirrored in the 75-th percentile, but the difference are sill under the meter-level. In the 95-th percentile, the situation is very similar to the one in the horizontal component: the group of curves with $ws \geq 50$ together with $ws = 20$, which still has a different behaviour with respect to the others, are the ones performing better. Regarding the maximum error, the minimum, 15.6 m, is still achieved by $ws = 20$, while with higher window sizes the maximum error is 21.7 m and the standard FG with a maximum error of 32.4 m is the one performing worse. The obtained reduction with $ws = 20$ is of 52%.

A summary of the analysis carried out above can be visualized in Figures 4.17a and 4.17b, representing in a clearer way what was represented in the zoomed boxes in Figures 4.16a and 4.16b. There is more evident how from $ws = 200$ on, increasing the window size does not improve the level of accuracy of the solution. This is partially caused by the chosen modelling of noise covariance. Assuming the noise to be Gaussian does not correctly model signals reflection and blockage, that are particularly present in urban scenarios. Hence the time correlation will not be accurate, causing the increment of the window size to be noneffective or even leading to worse accuracy [41]. In fact a window size of 20 seems to be the best choice for this dataset, considering also the lighter computational complexity.

In Figure 4.18 the box-plot showing the statistical description of the global error affecting each solution is represented. It can be seen how performing the smoothing lead to an increasing of the accuracy level of the solution. The minimum of the median, of 2.11 m, is reached at $ws = 200$ and remains unchanged with greater window sized. Starting from a mean (RMSE) of 2.56 m there is a improvement of 17.6%. $ws = 20$ is characterized by a mean of 2.36 m, 12% more then the one reach with greater window sizes. Instead the value of the mean, starting from 4.55 m, reaches its minimum at $ws = 20$ with the value of 3.57 m, a difference of 22%. The solution that is able to decrease the values of the maximum errors is obtained with $ws = 20$ as can be seen in the picture.



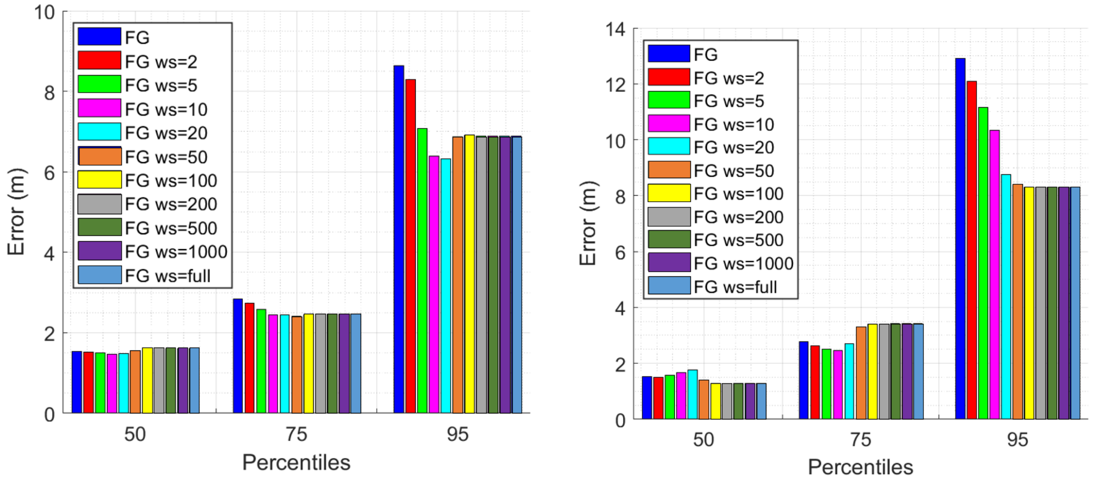
(a) Horizontal component

(b) Vertical component

Figure 4.16: ECDF of the horizontal and vertical components of solutions obtained with different window sizes, computed w.t.t. the reference trajectory.

4.3 Final comparison

In this section, a solution obtained with a Gauss-Newton approach and also performing the smoothing over a fixed window size is obtained and compared to the standard factor graph filtering solution, that is equivalent the the most common extended Kalman filter. Given the results obtained in the previous section, a good trade-off between accuracy level and computational complexity is given by a threshold t set to 4 and a window size ws set to 20. With this parameters the solution is obtained and compared to the standard FG solution. In Figure 4.19 the trajectories obtained with the described architectures are represented. It is evident, especially in sectors A, B and D, Figures 4.19a, 4.19b and 4.19d respectively, how the solution obtained exploiting both Gauss-Newton approach and smoothing process stands closer to the reference trajectory.



(a) Horizontal component

(b) Vertical component

Figure 4.17: Histogram representing the the horizontal error affecting the solutions obtained with different window sizes at 50-th, 75-th and 95-th percentiles.

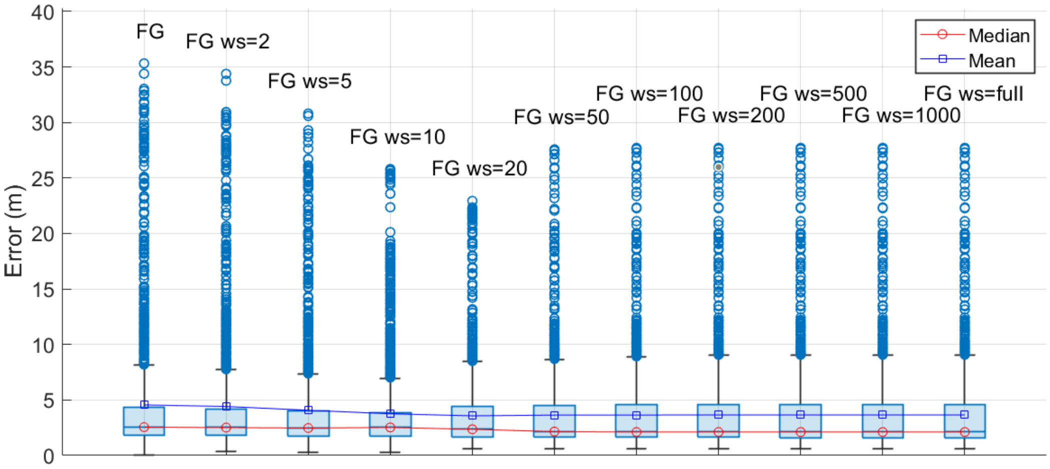
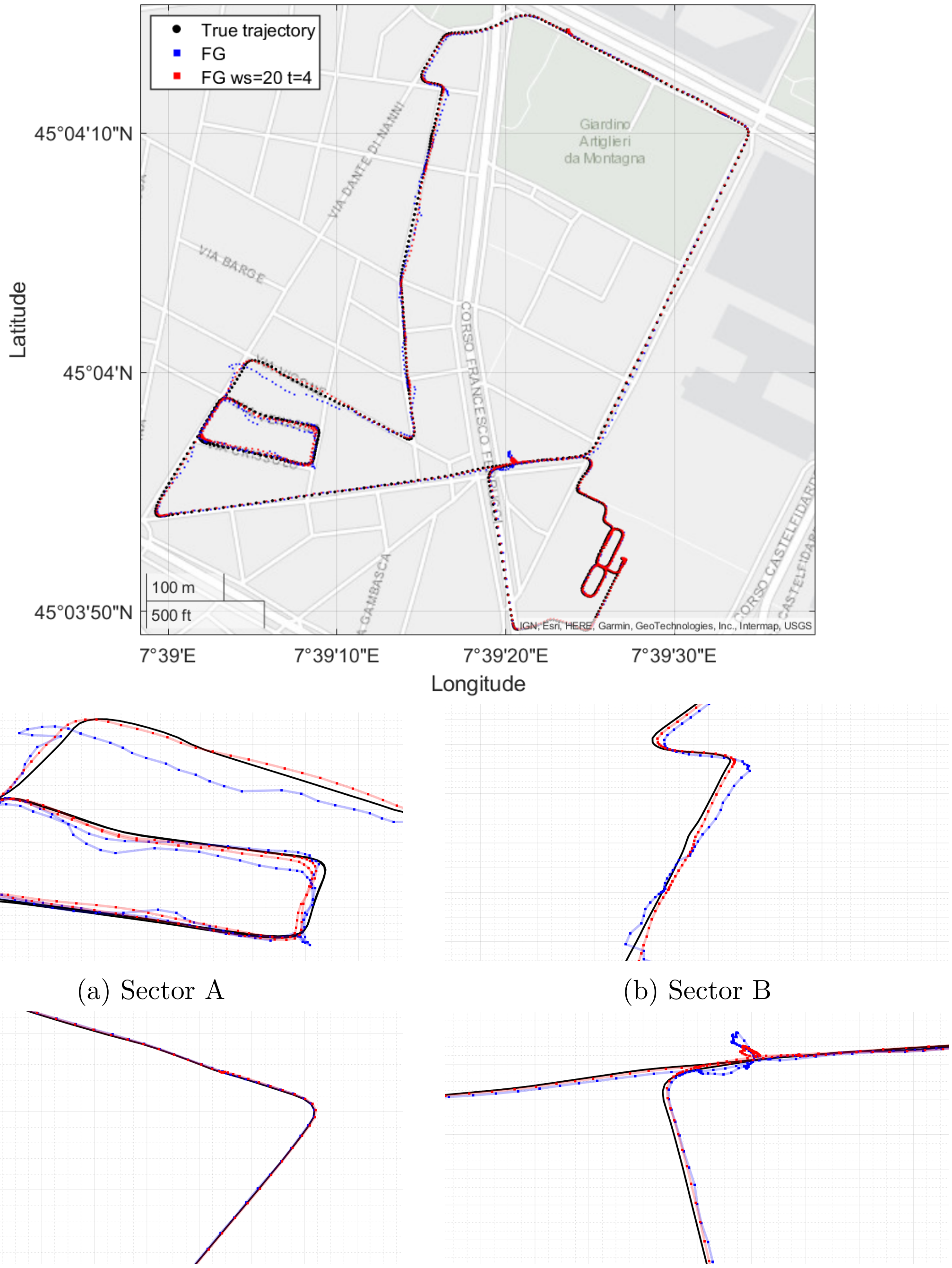


Figure 4.18: Box-plot of the overall error affecting the solutions obtained with different window sizes

The same can be seen in Figures 4.20a and 4.20b, representing the error in the horizontal and vertical components respectively. In both components, it can be seen how the peaks characterizing the standard FG solution are mitigated and the overall error affecting the solution is lowered in almost every part of the plots.

Looking instead at the ECDF describing the average error distribution, it is clear how the solution benefits from having multiple iterations and the smoothing



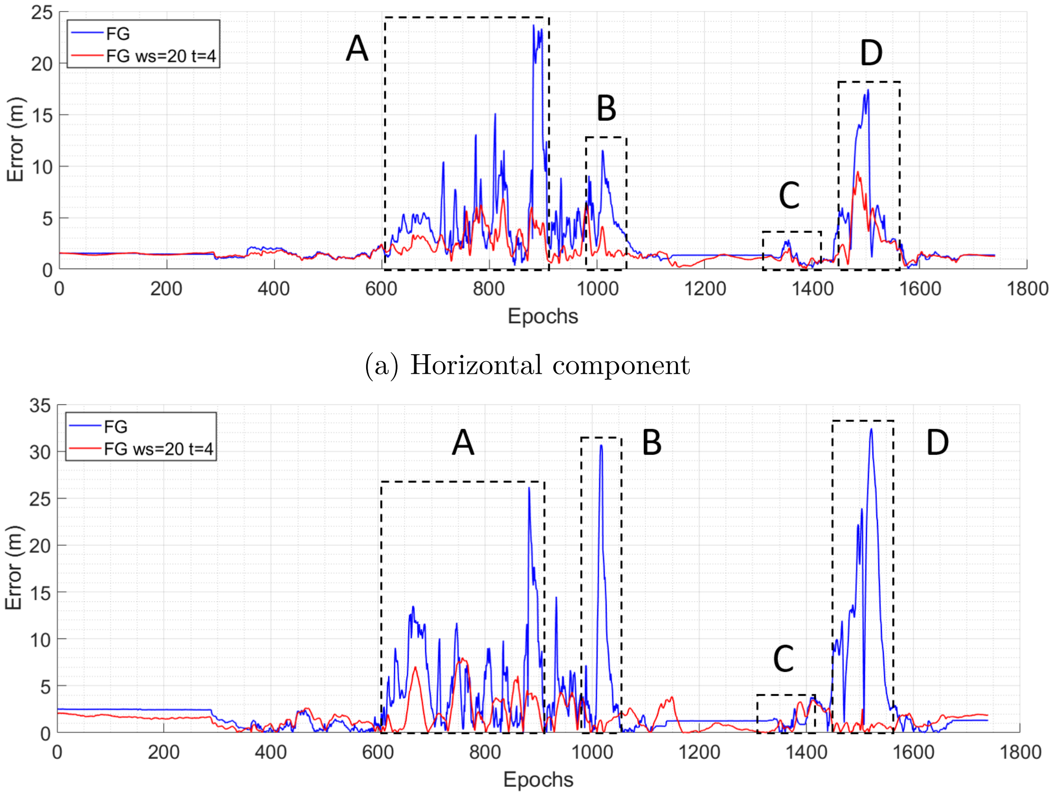
(a) Sector A

(b) Sector B

(c) Sector C

(d) Sector D

Figure 4.19: 2-D experimental trajectories (latitude/longitude), w.r.t. ground-truth.



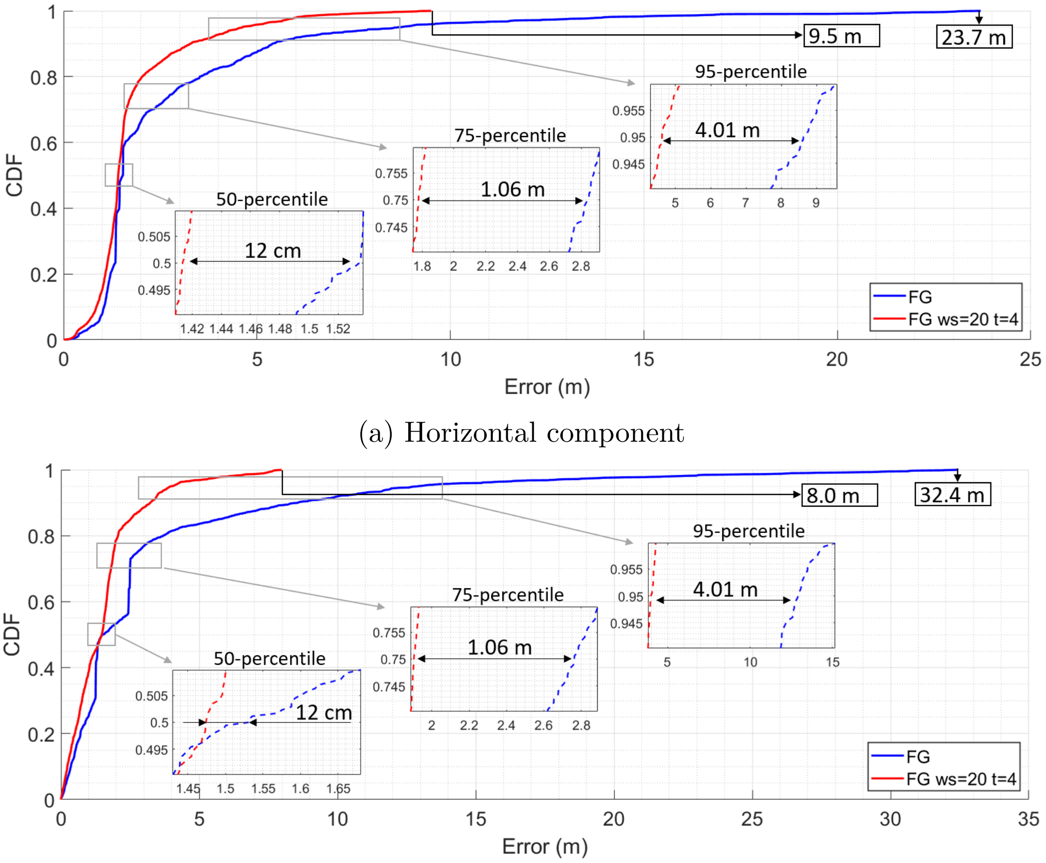
(a) Horizontal component

(b) Vertical component

Figure 4.20: Error on the horizontal and vertical positions in ENU-coordinates, computed w.t.t. the reference trajectory.

process. In fact the curve obtained with these techniques left-bounds completely the curve associated with the standard FG. In particular, for higher percentiles the difference becomes even more evident. For example, at the 95-th percentile the difference is of 4.01 m and 8.94 m for the horizontal and vertical component respectively. Considering the maximum error, the gain in precision level is maximized: for the horizontal component there is a difference of 14.2 m, representing a reduction of the error of the 60%; for the vertical component it is even more important, with a difference of 24.4 m translating in a reduction of the 75%.

Finally, a global statistical description of the errors affecting the two different solutions is reported in Figure 4.22 as box-plot. While the median remains very close, going from 2.56 m to 2.14 m, the mean (RMSE) decreases more, involving a reduction of the 44%, from 4.55 m to 2.56 m. The maximum error is also reduced according to what was said in the analysis of the ECDFs.



(a) Horizontal component

(b) Vertical component

Figure 4.21: ECDF of the horizontal and vertical components of solution, computed w.t.t. the reference trajectory.

4.4 Computational complexity

In this chapter the equivalence of the extended Kalman filter and a standard factor graph (a single iteration performed and a single node kept at each epoch) was shown, followed by an analysis of the effect of performing multiple iterations, as of the benefits of performing the smoothing of the previously estimated states at each epoch. In this section the computational complexity involved in these operations is briefly analyzed.

The code used to formulate a factor graph navigation filter from which the showed results were obtained has been written in MATLAB environment. In the Table 4.2, the times spent to obtain the extended Kalman filter, standard factor graph and multiple iterations plus smoothing (defined with the parameters $t = 4$ and $ws = 20$) factor graph solutions are reported. They are obtained as average

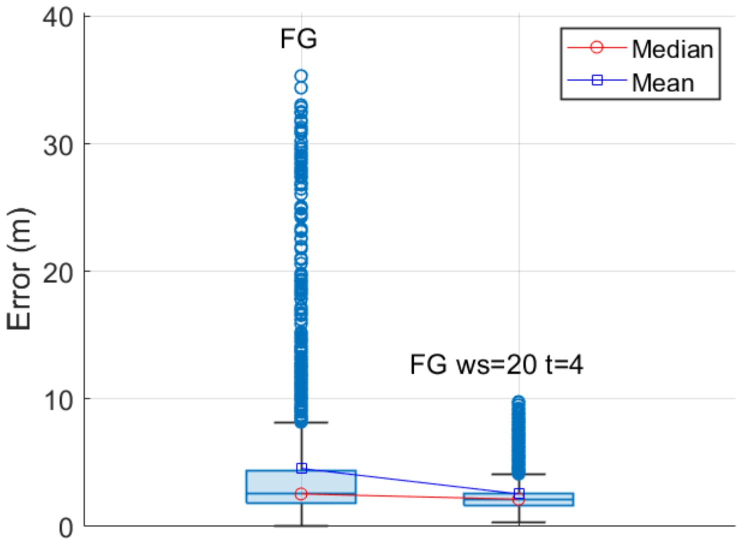


Figure 4.22: Box-plot of the overall error

values of the times spent for computing the whole trajectory (1740 epochs) in several run. Given the time difference of 44 s between the EKF and the standard FG architectures, with the latter presenting an increment of 16% in the taken computational time, it is evident that implementing a factor graph to obtain the equivalent solution of an EKF is not convenient. Instead, when multiple iterations together with the smoothing process is employed, the time difference with an EKF becomes of 73 s, implying an increment of 27%. However, given the superior performances of this architecture, as showed in the previous section, if the accuracy level is more relevant than the complexity of the filter in the considered application, a factor graph filter that exploits the Gauss-Newton approach together with the smoothing process may be the optimal choice.

	EKF	FG	FG (t=4 and ws=20)
Total	270 s	314 s	343 s
Per epoch	155 ms	180 ms	197 ms

Table 4.2: Time spent to estimate the whole trajectory with different navigation filters

Chapter 5

Conclusions

Global Navigation Satellite Systems (GNSS) is a reliable technology that allows accurate and stable localization and navigation in several contexts. However in urban environments there is a limited satellite visibility and presence of multipath due to the presence of tall buildings, causing a stand/alone GNSS application to be less reliable. In order to achieve a more reliable positioning in such challenging scenarios, multi-sensor navigation solutions have attracted a lot of interest. In this thesis, the integration of GNSS with Inertial Navigation Systems (INS) is addressed. The aim is to exploit to the best the complementary nature of these two technologies. This thesis focus was on a probabilistic graphical model approach for the implementation of such integration: factor graphs. They were initially developed to handle Simultaneous Localization And Mapping (SLAM) problems. Formulating the GNSS/INS integration in a factor graph framework leads to the resolution of an optimization problem, for which a vast literature is available. The first advantage of exploiting factor graphs are their flexibility. In fact every factor in the graph represent a probabilistic constraint, which may come from collected measurements or prior knowledge about some variable, which is simply expressed as a cost function, allowing for easy addition of different sensors. Moreover, they can be seen a generalization of an Extended Kalman Filter (EKF) as have been shown in the results. Furthermore, they offer additional features: performing the estimation of the state at the last epoch with multiple iterations applying a Gauss-Newton algorithm to the optimization problem involved and performing the smoothing over the states at previous epochs, refining the previous estimates. The benefits coming from these techniques have been showed.

Further studies can be carried out on this topic. In fact, all the existing literature about non-linear optimization problems, arising in factor graphs applied to multi-sensor navigation, can be exploited: different cost functions can be adopted to enhance robustness (M-estimators); inclusion of Switch Constraints (SC), which define an observation weighting function which is estimated together with the other states (they were developed for robust loop closure detection in SLAM and then extended to GNSS for multipath mitigation); assume a Gaussian Mixture Model (GMM) for pseudorange measurements to better account for the presence of multipath; adoption of more sophisticated algorithm for non-linear optimization such as Levenberg-Marquardt (LM) algorithm or Powell's dogleg (PDL) algorithm. Finally, an analysis of the effect of the addition of other different sensors could be pursued.

Bibliography

[1] J Sanz Subirana, JM Juan Zornoza, and M Hernández-Pajares. Gnss data processing volume i: Fundamentals and algorithms, vol. i. *Noordwijk, Netherlands: ESA Communications*, 2013.

[2] Elliott D Kaplan and Christopher Hegarty. *Understanding GPS/GNSS: principles and applications*. Artech house, 2017.

[3] Evgeny Ochin. Chapter 2 - fundamentals of structural and functional organization of gnss. In George p. Petropoulos and Prashant K. Srivastava, editors, *GPS and GNSS Technology in Geosciences*, pages 21–49. Elsevier, 2021. ISBN 978-0-12-818617-6. doi: https://doi.org/10.1016/B978-0-12-818617-6.00010-X. URL https://www.sciencedirect.com/science/article/pii/B978012818617600010X.

[4] T Mai. Global positioning system history. nasa. 2017. URL https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html.

[5] Gps space segment. *Esa Navipedia*, 2011. URL https://gssc.esa.int/navipedia/index.php/GPS_Space_Segment.

[6] Space segment, 2018. URL https://www.gps.gov/systems/gps/space/.

[7] Unite Nations. Current and planned global and regional navigation satellite systems and satellite-based augmentations systems. In *Proceedings of ICG*, pages 15–40, 2010.

[8] What is galileo? URL https://www.esa.int/Applications/Navigation/Galileo/What_is_Galileo.

[9] Jose Ángel Ávila Rodríguez. Galileo signal plan. *Esa Navipedia*, 2011. URL https://gssc.esa.int/navipedia/index.php/Galileo_Signal_Plan.

[10] Gnss receivers general introduction. *Esa Navipedia*, 2011. URL https://gssc.esa.int/navipedia/index.php/GNSS_Receivers_General_Introduction.

[11] Priyanka Das, Lorenzo Ortega, Jordi Vilà-Valls, François Vincent, Eric Chaumette, and Loïc Davain. Performance limits of gnss code-based precise positioning: Gps, galileo & meta-signals. *Sensors*, 20(8):2196, 2020.

[12] Front end. *Esa Navipedia*, 2011. URL https://gssc.esa.int/navipedia/index.php/Front_End.

[13] Calogero Cristodaro. Advanced integration of gnss and external sensors for autonomous mobility applications. *Prof. F. Dovis, Supervisor. PhD thesis. Turin, Italy: ScuDo, Politecnico di Torino*, 2019.

[14] Rafiullah Khan, Sarmad Ullah Khan, Rifaqat Zaheer, and Shahid Khan. Acquisition strategies of gnss receiver. In *International Conference on Computer Networks and Information Technology*, pages 119–124. IEEE, 2011.

[15] B Motella, L Lo Presti, and MG Petovello. The math of ambiguity what is the acquisition ambiguity function and how is it expressed mathematically. *Inside GNSS*, 5(4):20–28, 2010.

[16] Tracking loops, 2011. URL https://gssc.esa.int/navipedia/index.php?title=Tracking_Loops.

[17] Delay lock loop (dll). *Esa Navipedia*, 2011. URL https://gssc.esa.int/navipedia/index.php/Delay_Lock_Loop_(DLL).

[18] Phase lock loop (pll). *Esa Navipedia*, 2011. URL https://gssc.esa.int/navipedia/index.php/Phase_Lock_Loop_(PLL).

[19] Mark Petovello. How does a gnss receiver estimate velocity? *Inside GNSS*, 14, 2015.

[20] Junchuan Zhou. Low-cost mems-ins/gps integration using nonlinear filtering approaches. 2013.

[21] Mohinder S Grewal, Lawrence R Weill, and PA Andrews. Fundamentals of satellite and inertial navigation. In *Global Positioning Systems, Inertial Navigation, and Integration*, pages 18–52. Wiley, 2007.

[22] Paul D. Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech House, 2008.

[23] Amitava Bose, KN Bhat, and Thomas Kurian. *Fundamentals of navigation and inertial sensors*. PHI Learning Pvt. Ltd., 2014.

[24] Paul G Savage. Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms. *Journal of guidance, control, and dynamics*, 21 (1):19–28, 1998.

[25] Paul G Savage. Strapdown inertial navigation integration algorithm design part 2: Velocity and position algorithms. *Journal of Guidance, Control, and dynamics*, 21(2):208–221, 1998.

[26] Anastasia O Salytcheva. Medium accuracy ins/gps integration in various gps environments. 2004.

[27] Mark G Petovello. Real-time integration of a tactical-grade imu and gps for high-accuracy positioning and navigation. 2003.

[28] Nurlan Boguspayev, Daulet Akhmedov, Almat Raskaliyev, Alexandr Kim, and Anna Sukhenko. A comprehensive review of gnss/ins integration techniques for land and air vehicle applications. *Applied Sciences*, 13(8):4819, 2023.

[29] Christopher Jekeli. *Inertial navigation systems with geodetic applications*. Walter de Gruyter GmbH & Co KG, 2023.

[30] MG Petovello, Cillian O'Driscoll, and Gerard Lachapelle. Weak signal carrier tracking using extended coherent integration with an ultra-tight gnss/imu receiver. In *European Navigation Conference*, pages 23–25, 2008.

[31] Malek Karaim. *Ultra-tight GPS/INS integrated system for land vehicle navigation in challenging environments*. PhD thesis, Queen's University (Canada), 2019.

[32] Gianluca Falco, Marco Pini, and Gianluca Marucco. Loose and tight gnss/ins integrations: Comparison of performance assessed in real urban scenarios. *Sensors*, 17(2):255, 2017.

[33] Isaac Skog and Peter Handel. In-car positioning and navigation technologies—a survey. *IEEE Transactions on Intelligent Transportation Systems*, 10 (1):4–21, 2009.

[34] Antonio Angrisano et al. Gnss/ins integration methods. *Dottorato di ricerca (PhD) in Scienze Geodetiche e Topografiche Thesis, Universita'degli Studi di Napoli PARTHENOPE, Naple*, 21, 2010.

[35] Basilio Bona et al. *Dynamic modelling of mechatronic systems.* Celid, 2013.

[36] Heidi Kuusniemi. *User-level reliability and quality monitoring in satellite-based personal navigation.* Tampere University of Technology, 2005.

[37] Sheldon M Ross. *Introduction to probability models.* Academic press, 2014.

[38] Frank Dellaert, Michael Kaess, et al. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017.

[39] Richard E Neapolitan et al. *Learning bayesian networks*, volume 38. Pearson Prentice Hall Upper Saddle River, 2004.

[40] OEM Commands. Logs reference manual, 2021.

[41] Weisong Wen, Tim Pfeifer, Xiwei Bai, and Li-Ta Hsu. Factor graph optimization for gnss/ins integration: A comparison with the extended kalman filter. *NAVIGATION: Journal of the Institute of Navigation*, 68(2):315–331, 2021.