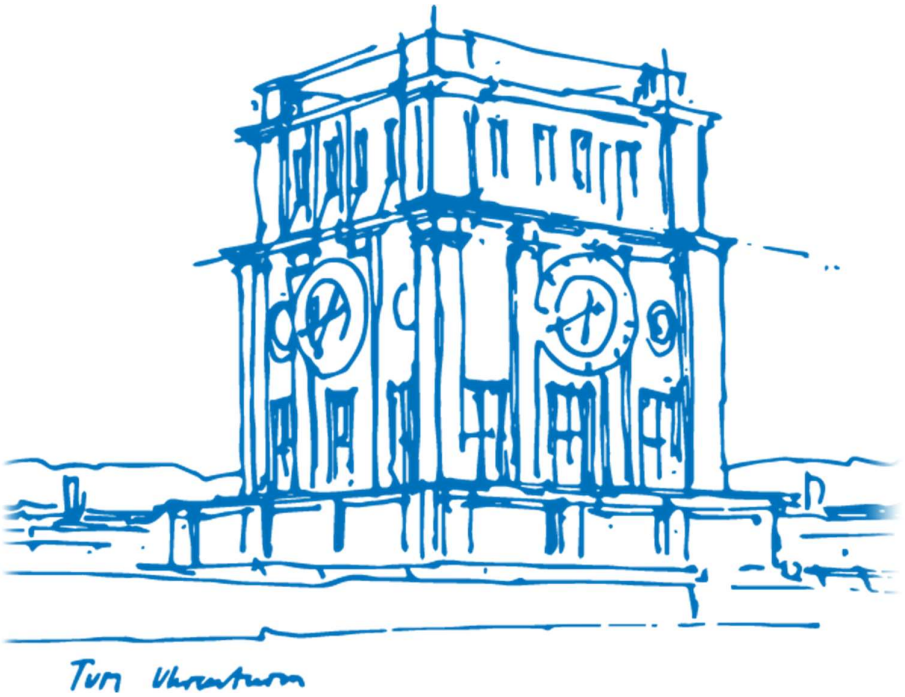
Deutsches Zentrum für Luft- und Raumfahrt

Technische Universität München

# BACHELOR'S THESIS

## Near Real-Time Positioning of Low Earth Orbit Satellites using Galileo High Accuracy Service



Scientific thesis submitted for the degree of
**B.Sc. in Aerospace Engineering** at the School of Engineering and Design at the Technical University of Munich


| | |
|---|---|
| **Submitted by** | Zineb Faiz<br>15.05.2026 in Munich |
| **University Supervisor** | Prof. Dr. Urs Hugentobler<br>Chair of Astronomical and Physical Geodesy,<br>Technical University of Munich (TUM) |
| **External Supervisors** | Cécile Deprez and Christian Trainotti<br>Institute of Communications and Navigation,<br>German Aerospace Center (DLR) |

## Declaration

I hereby declare that the work presented in this thesis is solely my work and that to the best of my knowledge this work is original, except where indicated by references to other authors. No part of this work has been submitted for any other degree or diploma.

Zineb Faiz, 15.05.2026

## Abstract

Low Earth Orbit satellites increasingly rely on precise positioning for navigation and scientific applications. One approach is to equip LEO satellites with GNSS receivers and use Precise Point Positioning to estimate the position from onboard measurements. The Galileo High Accuracy Service (HAS) provides real-time orbit, clock, and bias corrections through the E6-B signal, enabling high-accuracy PPP without relying on post-processed products. Within the GHASP3 software framework, two estimation strategies are available: a Kalman filter for real-time processing and a batch least-squares solver for near-real-time applications. While the filter has been evaluated in previous work, the batch solver required further analysis, particularly under HAS corrections and for a space user.

This thesis evaluates batch least-squares PPP for the Sentinel-6A (S6) LEO satellite using both final precise products and Galileo HAS corrections, and compares its performance against the Kalman filter. The ground station BRUX is used as a validation reference. The batch under HAS is run as a growing window, where the window length equals the time elapsed since the start of the arc, directly reflecting how it would operate in a near-real-time setting. Batch windows ranging from 10 to 180 minutes are tested, and the sensitivity of the solution to sampling interval, correction product, user type, and disturbances is assessed.

The results show that batch becomes reliable from around 60 minutes for both users under FIN, and between 90min and 120min for S6 under HAS. At 114 minutes - one full orbital period - batch gives its best result and outperforms the filter. An unexpected finding is that 30s sampling outperforms 10s for S6 under HAS, the opposite of what was observed under FIN, which may suggest that denser sampling amplifies correction noise under real-time products rather than improving geometry. Under HAS, the filter takes nearly 3 hours to converge for BRUX and over 3.5 hours for S6, meaning batch delivers a reliable solution faster. DIA is found to be essential under disturbed conditions, and batch with DIA enabled is the most resilient configuration tested.

For FIN products, the filter remains appropriate when low latency is the priority. For HAS-based S6 positioning in this setup, batch is the better choice once the minimum useful window is reached.

## Acknowledgements:

I would like to thank my external supervisors Cécile Deprez and Christian Trainotti at DLR's Institute of Communications and Navigation for their guidance, for giving their time and support, and for sharing their passion with me. I learned so much from both of them and honestly could not have done this without their help. A special thank you to Cécile for their mentorship throughout this journey, it means a lot.

I would also like to thank my university supervisor Prof. Dr. Urs Hugentobler for his support and for sharing his knowledge throughout this thesis, and beyond it. His published work was also a valuable resource that helped me build the foundation for this research. As he approaches retirement, I feel very lucky to have had the chance to connect with him and to have been one of his last students.

This thesis would not have been possible without the internship at DLR that preceded it, which gave me the tools and the curiosity to take on this topic in the first place. I would also like to acknowledge Prof. Dr. Urs Hugentobler and TUM for providing access to the computing server used for processing, and European Space Agency for their agreement that made this research possible.

Finally, to my parents Najib and Khadija, and my brothers Hamza and Reda, thank you for always being there and for having faith in me as your default setting.

Satellites have always fascinated me because they watch over the whole Earth from above, helping us understand and protect it. I hope that everything I do, starting with this work, can contribute to a better world for every living being on this planet.

## List of Figures:

## List of Tables:

## List of Abbreviations:

| | |
|---|---|
| DIA | Detection, Identification and Adjustment |
| DOP | Dilution of Precision |
| DOY | Day of Year |
| ENU | East-North-Up |
| FIN | Final precise products |
| GNSS | Global Navigation Satellite System |
| GPS | Global Positioning System |
| GHASP3 | Galileo High Accuracy for Space Precise Point Positioning |
| HAS | High Accuracy Service |
| IGS | International GNSS Service |
| ISB | Inter-System Bias |
| LEO | Low Earth Orbit |
| LOM | Local Overall Model |
| PNT | Positioning, Navigation and Timing |
| POD | Precise Orbit Determination |
| PPP | Precise Point Positioning |
| RMS | Root Mean Square |
| RSW | Radial-Along-track-Cross-track |
| RTK | Real-Time Kinematic |
| S6 | Sentinel-6A |

# Table of Contents

# 1 Introduction

## 1.1 Motivation and Context

Low Earth Orbit satellites increasingly rely on precise, real-time positioning for navigation and scientific applications. One promising approach is to equip LEO satellites with GNSS receivers and apply Precise Point Positioning using corrections broadcast directly from navigation satellites, no ground infrastructure required, no post-processing delay. The Galileo High Accuracy Service makes this possible in near real-time. Within the GHASP3 software framework, two estimation strategies are available for this purpose: a Kalman filter and a batch least-squares solver.

This thesis follows an internship at DLR's Institute of Communications and Navigation in Oberpfaffenhofen from May to August 2025, during which I developed a graphical user interface for a software called GHASP3, from which I got my first hands-on experience with Precise Point Positioning (PPP). Working with this tool and exploring its research context gave me the foundation and the motivation to pursue a thesis on the same topic.

In recent years, the number of Low Earth Orbit (LEO) satellites has dramatically increased, with new constellations being developed for communication and for Positioning, Navigation and Timing (PNT). To provide high-quality services, some of these systems rely on the availability of real-time orbit determination tools onboard the satellite. One way to achieve this is to equip LEO satellites with GNSS receivers and use the navigation signals broadcast by GNSS satellites in Medium Earth Orbit (MEO) to compute the LEO satellite orbit precisely [12].

The Institute of Communications and Navigation at DLR, in collaboration with TU Delft in the context of an ESA[1] project, has been developing a Precise Point Positioning software that uses the High Accuracy Service (HAS) corrections broadcast by Galileo satellites to compute precise LEO satellite orbits. HAS provides real-time orbit, clock, and bias corrections through the Galileo E6-B signal, enabling high-accuracy PPP without relying on post-processed products. This software implements two approaches: an iterative computation based on a Kalman filter for real-time solutions, and a batch computation for non-time-critical applications, which jointly processes data collected over sections of orbits. While the filter has been thoroughly evaluated, the batch computation requires further analysis to verify the quality of the computed solution [11], [12].

This thesis addresses that gap by performing an accuracy campaign for the batch engine using real satellite data and HAS products.

[1] European Space Agency

## 1.2 Scope and Thesis Objectives

This thesis investigates batch least-squares Precise Point Positioning for LEO satellites using Galileo HAS corrections, implemented within the GHASP3 processing framework [12]. The primary space user is Sentinel-6A (S6A), a LEO satellite dedicated to ocean altimetry [13]. The ground station BRUX is used as a validation reference.

The main tasks of this thesis were defined as follows:

- Familiarize with the theory of PPP and the filter and batch estimation strategies
- Configure GHASP3 and generate batch solutions for different scenarios and batch lengths
- Compare batch solutions to filter solutions and real-time results to post-processed reference orbits
- Develop a Python script to automate the processing and ensure consistency
- Evaluate the results and summarize the findings

In preparation for these tasks, a literature review was conducted to build the theoretical background needed. The main concepts and references identified during this review are presented in Chapter 2.

# 2 Background and State of the Art

Global Navigation Satellite Systems (GNSS), such as GPS and Galileo, are based on the reception of radio signals transmitted by satellites. Since the receiver knows the satellite positions from the ephemerids and the signal propagation speed is assumed to be known, the measured signal travel time can be converted into a distance-like measurement. Unlike triangulation in surveying, which uses angles, GNSS positioning is based on multilateration, which determines the receiver position by combining measurements from several satellites. In three-dimensional positioning, at least four satellites are required, because the receiver must solve for its three coordinates and its own clock offset. Figure 1 illustrates this basic geometry, where the receiver uses signals from multiple satellites to estimate its position. In practice, the measurements are affected by clock errors, atmospheric delays, and receiver limitations, which is why more advanced methods such as Precise Point Positioning (PPP) are needed for high-accuracy applications [1].

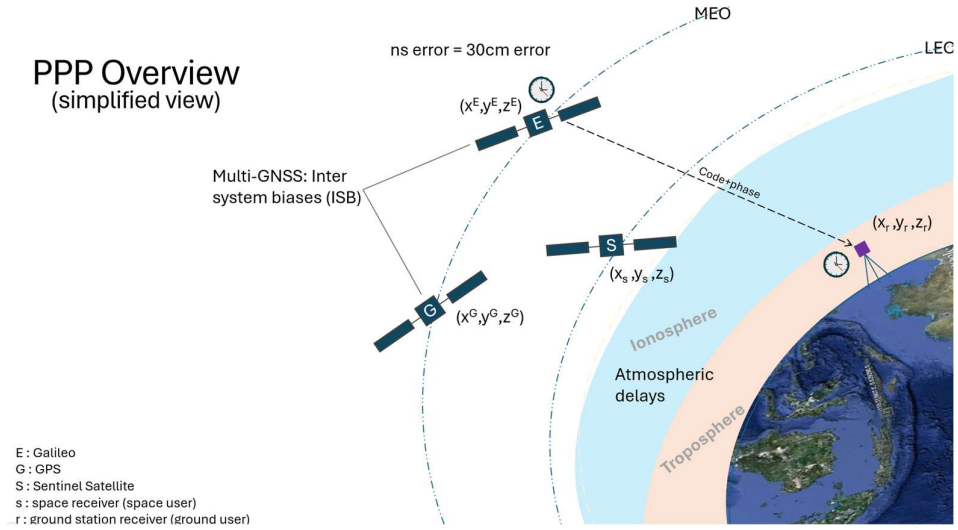



*Figure 1: Basic PPP concept.*

## 2.1 GNSS Observation Model

### 2.1.1 Code and Phase Observation

GNSS positioning is based mainly on two observation types: code observations and carrier phase observations. Both depend on the geometric distance between satellite and receiver, but they are affected by different error sources: clock offsets, atmospheric delays, hardware biases, and measurement noise and provide different levels of precision [2].

In the following equations, the superscript *s* refers to the satellite, the index *r* refers to the receiver, and the index *j* refers to the signal frequency.

The geometric range describes the distance between the satellite position and the receiver position. It can be written as

$$\rho_r^s = \| x^s - x_r \| = \sqrt{(x^s - x_r)^2 + (y^s - y_r)^2 + (z^s - z_r)^2} \quad [m]$$

Where:
$x^s = (x^s, y^s, z^s)$ is the satellite position vector and
$x_r = (x_r, y_r, z_r)$ is the receiver position vector.

The code observation is based on the time delay between the code transmitted by the satellite and a locally generated replica inside the receiver. The receiver shifts its local code until it matches the received signal, and from this shift the signal travel time is estimated. Nevertheless, the code measurements are relatively noisy and therefore much less precise than carrier-phase observations [1], [2].

A code observation equation can be written as

$$P_{r,j}^s = \rho_r^s + c\left(dt_r - dt^s + dt_r^{s,rel}\right) + c\left(d_{r,j} + d_j^s\right) + \zeta_{r,j}^s + T_r^s + I_{r,j}^s + e_{r,j}^s \; [m]$$

Where:
$P_{r,j}^s$ is the code observation expressed in meters,
$dt_r$ and $dt^s$ are the receiver and satellite clock offsets,
$dt_r^{s,rel}$ is the relativistic clock correction,
$\zeta_{r,j}^s$ are the Phase Center Offset (PCO) and Phase Center Variation (PCV) corrections,
$T_r^s$ is the tropospheric delay,
$I_{r,j}^s$ is the ionospheric delay,
$d_{r,j}$ and $d_j^s$ are hardware-related code biases, and
$e_{r,j}^s$ represents noise and remaining unmodeled errors [2].

The carrier phase observation measures the phase of the carrier wave itself rather than the code. Because the carrier wavelength is much shorter, the phase can be measured much more precisely than the code. However, the receiver cannot determine how many complete cycles have accumulated between the satellite and the receiver since tracking began. This unknown integer number of cycles is the phase ambiguity $N_{r,j}^s$, which must be estimated as part of the PPP solution. If signal tracking is interrupted, for example because of a cycle slip, a new ambiguity parameter has to be estimated [1], [3].

A carrier phase observation equation can be written as

$$\Phi_{r,j}^s = \rho_r^s + c\left(dt_r - dt^s + dt_s^{r,rel}\right) + \lambda_j\left(\delta_{r,j} + \delta_j^s\right) + \zeta_{r,j}^s + T_r^s - I_{r,j}^s + \lambda_j(\omega_r^s + N_{r,j}^s) + \epsilon_{r,j}^s \; [m]$$

Where:
$\Phi_{r,j}^s$ is the carrier-phase observation expressed in meters,
$\lambda_j$ is the signal wavelength,
$\omega_r^s$ is the phase wind up,
$N_{r,j}^s$ is the integer phase ambiguity,
$\delta_{r,j}$ and $\delta_j^s$ are receiver and satellite phase biases [3].

Note that the ionosphere appears with opposite sign in the two equations: it delays the code but advances the carrier phase [2].

### 2.1.2 GHASP3 Observation Model

As introduced in Section 2.1.1, the GNSS observation equations contain terms that are provided, modeled, or estimated. Provided terms are values supplied from outside the PPP solution, for example broadcast navigation products, or through final precise products generated by analysis centers such as the International GNSS Service (IGS), or even through Galileo HAS corrections. Modeled terms are the parts computed directly from the observation equations and the signal geometry, while estimated terms are the unknown parameters that the PPP solution has to solve for from the observations [1], [4], [7], [11].

In GHASP3, undifferenced and uncombined code and phase observations are used to compute the user's position. Multi-GNSS (GPS and Galileo) dual-frequency observations are processed, specifically GPS L1/L2 and Galileo E1/E5a. The vector of estimated parameters is:

$$x = \begin{bmatrix} x_r, y_r, z_r & (position) \\ dt_r & (receiver\ clock) \\ ZTD & (troposphere\ wet\ component) \\ ISB & (inter\ system\ biases) \\ I^s_{r,j} & (ionospheric\ delays) \\ N^s_{r,j} & (phase\ ambiguities) \end{bmatrix}$$

The satellite clock and coordinates are either computed from the broadcast navigation message for real-time and near-real-time processing, or obtained from IGS final products for post-processing. Post-processing also includes code and phase bias corrections and PCO/PCV corrections. The relativistic clock error, the dry component of the troposphere, and the phase wind-up effect are modeled.

## 2.2 Point Positioning Methods: SPP/PPP

Single Point Positioning (SPP) is the simplest GNSS positioning method. It uses code observations together with broadcast satellite orbits and clock corrections from the navigation message, and its typical accuracy is at the meter level . Because of this limited accuracy, SPP is usually not the final solution for high precision applications, but it is still useful for providing an initial position estimate before more advanced processing is started [1].

In the PPP implementation used in this thesis, the nonlinear PPP observation equations are linearized around an initial state, and the SPP solution serves as the starting point, as shown in Figure 2.

Precise Point Positioning (PPP) is an absolute positioning technique that uses undifferenced GNSS observations from a single receiver, meaning the raw measurements are processed directly without forming differences between receivers. This is different from differential methods such as Real-Time Kinematic (RTK), which use a nearby reference receiver or reference network to form differential corrections and reduce common errors. PPP instead relies on precise external products for satellite orbit, clock, and bias corrections, and it can reach centimeter level accuracy after convergence [2], [3].

Unlike SPP, PPP additionally makes use of carrier-phase observations, which improves the precision of the solution. The trade-off is that PPP must estimate more unknowns than SPP - in particular, the integer phase ambiguities introduced by the carrier-phase observations - and it typically requires a convergence period before reaching its full accuracy [2], [3].

As previously mentioned, both SPP and PPP are standalone single-receiver techniques, so neither requires a nearby reference station, unlike differential methods such as RTK. This makes PPP particularly attractive for global and space-based applications like LEO orbit determination, where no ground infrastructure is available near the user [2]. In this thesis, PPP is the core positioning method applied to all users.

## 2.3 Precise Orbit Determination (POD)

Precise Orbit Determination (POD) is the process of estimating the trajectory of a satellite with high accuracy. In this thesis, POD refers to the estimation of the orbit of a LEO satellite, such as Sentinel-6A, from GNSS observations collected by an onboard receiver. The kinematic approach estimates the satellite position directly from the observations at each epoch, supporting real-time and near-real-time LEO navigation [6], [12].
Three main approaches exist for LEO POD, as illustrated below:

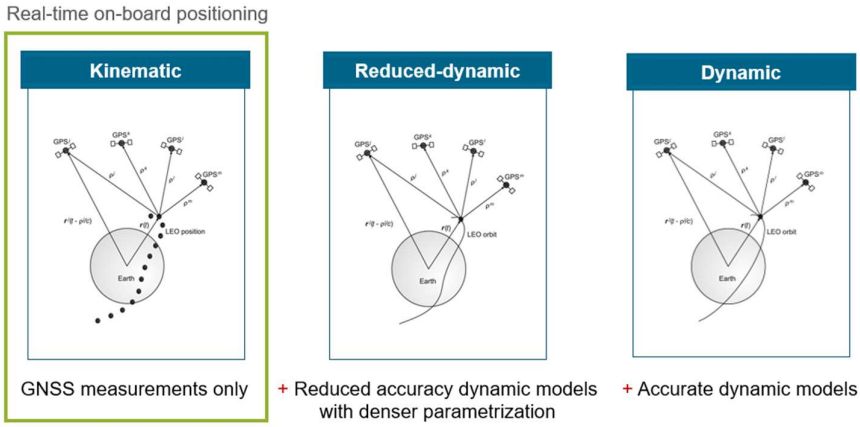



*Figure 2: LEO POD approaches, with the approach selected for GHASP3 real-time on-board positioning highlighted in green. Source: DLR*

In kinematic POD, the satellite position is estimated independently at each epoch from the GNSS observations only, with no assumption about the forces acting on the satellite. This makes it independent of force models, but also more sensitive to the geometry of navigation problem, meaning the spatial distribution of visible GNSS satellites relative to the receiver. If this geometry is weak, the position solution becomes more sensitive to observation noise, data gaps, and cycle slips [6].

In dynamic POD, the orbit is obtained by numerically integrating the satellite equation of motion using physical force models such as Earth gravity, atmospheric drag, and solar radiation pressure. This gives physically smooth orbits solutions, but the result depends strongly on how well these forces are modeled [6].

In reduced-dynamic POD, the physical force model is still used, but additional empirical parameters are introduced to absorb remaining model errors. This makes it a practical compromise between kinematic and dynamic methods, and it is widely used for high precision LEO orbit determination [5].

GHASP3 uses a kinematic POD approach because the project targets real time and near real time LEO orbit determination from GNSS measurements and Galileo HAS corrections.

## 2.4 Estimation Strategies: Filter vs Batch

### 2.4.1 Kalman Filter

The Kalman filter is a sequential estimation method designed for real time processing. Instead of waiting for a full observation arc, it processes new measurements as they become available, one by one, without collecting a period of data first. At each epoch, the current state is first predicted from the previous estimate and then corrected using the newly received observations.
This real time characteristic makes the Kalman filter especially suitable for applications where orbit information is needed quickly and with low latency.
In simplified form, the filter uses a prediction step

$$\hat{x}_{t|t-1} = \Phi_t \hat{x}_{t-1|t-1}$$

followed by an update step

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t \left( y_t - j_t(\hat{x}_{t|t-1}) \right)$$

Where:
$\Phi_t$ is the state transition matrix,
$(y_t - j_t(\hat{x}_{t|t-1}))$ is the residual vector, representing the difference between the actual observations $y_t$and the measurements predicted by the model $j_t(\hat{x}_{t|t-1})$ and
$K_t$ is the Kalman gain and determines how much weight is given to this residual in correcting the predicted state

The idea behind these two steps is straightforward. First, the software predicts where the satellite should be at the next epoch based on the previous estimate and on the model of the satellite's dynamic. Then it compares this prediction with the actual GNSS observations and corrects the solution. In this way, the filter continuously combines measurements predicted by the model and the real obtained values.

A known limitation of the filter in PPP applications is the convergence period at the start of processing, during which the ambiguity estimates are poorly initialized, and the solution has not yet reached its full accuracy [7].

### 2.4.2 Least Squares and the Batch Approach

The batch approach works differently from the Kalman filter. Instead of updating the solution epoch by epoch, it processes all observations from a selected time interval together in one adjustment and estimates all unknown parameters jointly over that full batch interval [8].

The mathematical basis of this approach is usually the least squares method. Its objective is to find the parameter values that minimize the differences between the observations and the model [1]. In matrix form, this leads to the normal equations.
The observation equations can be written in linearized form as:

$$y = Ax + e$$

Where:
$y$ is the vector of observed-minus-computed values,
$A$ is the design matrix,
$x$ contains the corrections to the unknown parameters, and
$e$ is the residual vector.

The design matrix is formed from the linearized observation equations, so that all observations in the selected batch interval contribute together to the estimation of the unknown parameters [1], [8].
In Matrix form, this leads to the normal equations:

$$A^T W A x = A^T W y$$

Where:
$W$ is the weighting matrix.

A possible advantage of batch processing is robustness. Because all observations in the arc are processed together, isolated bad epochs often have a smaller effect than in a purely sequential filter. This makes the batch approach less vulnerable to local data problem.

Its disadvantage is latency. Since the full data arc must be available before the adjustment can be computed, the solution cannot be obtained immediately in the same way as a sequential filter. The choice of batch length and batch shift therefore represents a trade-off between accuracy, robustness, and latency.

This trade-off motivates the batch design choices studied later in Chapter 5, where smaller batch lengths and overlapping batch shifts are tested to balance latency with solution accuracy and stability.

### 2.4.3 Visual Comparison

Figure 3 illustrates the difference in processing flow between the two approaches. The Kalman filter produces a solution at every epoch as new observations arrive, while the batch approach collects all observations first and solves for all parameters at once.

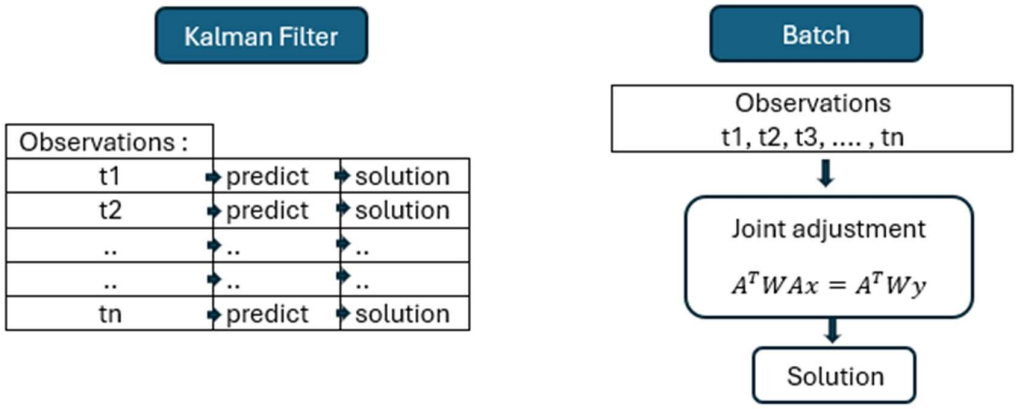



*Figure 3: Comparison of Kalman filter and batch least squares.*

*Table 1: Comparison of Kalman filter and batch least squares.*

| **Method** | **Kalman filter** | **Batch least squares** |
|---|---|---|
| **Processing style** | Sequential, epoch by epoch | Joint adjustment over a full data arc |
| **Main advantage** | Low latency, suitable for real-time use | More robust against local data problems |
| **Main disadvantage** | More sensitive to local disturbances and bad epochs | Higher latency because the full arc must be available first |

### 2.4.4 Relevance for the Thesis

In this thesis, the comparison between filter and batch processing is one of the central research questions, because both methods use the same GNSS observations and the same general PPP framework, but estimate the orbit solution in different ways.
The main objective is to assess the trade-off between low latency sequential processing in the filter and more robust arc wise estimation in the batch approach [7], [8].

The later chapters therefore evaluate how the two strategies differ in terms of positioning and orbit quality, solution stability, and robustness against observation disturbances. Particular attention is given to the effect of degraded data, such as data gaps or cycle slips [8], [10]. The comparison also considers the accuracy and precision of the two solutions, as defined in Section 3.6. The detailed processing settings are introduced in Chapter 3, while the experimental comparison and performance analysis are presented in the later results chapters.

## 2.5 Detection, Identification and Adjustment (DIA)

Detection, Identification and Adjustment (DIA) is a quality-control concept used to handle possible model errors or observation faults in GNSS processing, such as code outliers, cycle slips, or ionospheric disturbances [9], [10]. It consists of three steps : first detect whether the data are inconsistent with the assumed model, then identify which observation is likely faulty, and finally adjust the model so that the estimation can continue correctly [9].
A common detection statistic is based on the residual vector:

$$T_q = \hat{e}^T Q_{\hat{e}}^{-1} \hat{e}$$

where:
$T_q$ is the detection statistic,
$\hat{e}$ is the estimated residual vector,
$Q_{\hat{e}}$ is the covariance matrix of the residual vector and
$Q_{\hat{e}}^{-1}$ is the inverse of the residual covariance matrix.

If this value exceeds a predefined threshold, the null hypothesis is rejected and the observations are considered inconsistent with the assumed model [9]. The null hypothesis means that no fault is present and that the residuals can be explained by the expected observation noise [10].

After detection, the identification step uses statistical tests to determine which observation is most likely responsible for the detected inconsistency. In practice, the adjustment step may exclude a faulty code observation or re-initialize a carrier phase ambiguity after a cycle slip, including all ambiguities on the satellite on which the fault was detected [10]. In GHASP3, two adjustment modes are available: mode 1 adapts the solution by estimating a fault vector, while mode 2 eliminates the faulty measurement and re-estimates the state vector.
GHASP3 also includes a post-processing quality control step based on LOM (Local Overall Model) detection. However, this was not activated in any of the experiments in this thesis, because its implementation for the batch solver has not yet been validated. Evaluating the impact of LOM-based post-processing on batch solutions is left as a direction for future work.

Under clean observation conditions, DIA may not trigger - not because it fails, but simply because there is no fault to detect. Its impact becomes visible only when a sufficiently large observation fault is present, as demonstrated in Chapter 10.

## 2.6 GNSS Products

The quality of PPP and POD results depends strongly on the external GNSS products used in the processing [1]. In the scope of this thesis, the most relevant product types are broadcast navigation products, final precise products, and Galileo High Accuracy Service (HAS) corrections.

Broadcast (NAV) products are transmitted directly by the GNSS satellites and contain standard orbit and clock information for navigation. Their main advantage is that they are available in real time, but their accuracy is limited compared with precise products, which makes them insufficient on their own for high-precision orbit determination [1].

Final precise products, referred to as FIN in this thesis, are post-processed orbit and clock products generated by analysis centers such as the International GNSS Service (IGS). They are computed using global networks of ground stations and typically provide centimeter-level orbit accuracy for scientific and validation purposes. Because they are not available in real time, they are used in this thesis as the reference product for validation, representing the best achievable accuracy against which the other products are compared [1], [6].

Galileo High Accuracy Service (HAS) is a free global service that broadcasts real-time orbit, clock, and code-bias corrections through the Galileo E6-B signal. These corrections are applied on top of the broadcast products to support high-accuracy PPP without relying on post-processed precise products. The service targets real-time positioning performance better than 20 cm horizontally and 40 cm vertically at the 95% level after convergence [7], [11]. This makes HAS the most operationally relevant product in this thesis, because it represents the low-latency scenario in which precise orbit information is required in near real time.

In this thesis, FIN products serve as the post-processed reference, while HAS represents the real-time operational scenario. This allows a direct comparison of positioning performance under different product latency conditions.

# 3 Experimental Setup and Processing Framework

This chapter describes the processing framework used throughout the thesis. It introduces the GHASP3 software, the users and datasets, the GNSS products, the standard processing settings, the process noise configuration, the configuration validation tests, and the evaluation metrics.

All GHASP3 processing runs were executed on a Linux server provided by the Chair of Astronomical and Physical Geodesy at TUM, accessed remotely via SSH, using GNSS observation data and correction products made available through DLR's Institute of Communications and Navigation.
Tables 2 and 3 provide a quick overview of the users and experimental scenarios covered in the next chapters.

*Table 2: Users and datasets used in this thesis.*

| User | Type | Description | Role in this thesis |
|---|---|---|---|
| BRUX | Static ground user | IGS reference station located in Brussels, Belgium. The receiver is fixed but observations are affected by the troposphere and local station environment. | Ground-user validation case. |
| Sentinel-6A (S6) | LEO space user | Low Earth Orbit satellite used for ocean altimetry. operating at an altitude of approximately 1336 km with an orbital period of approximately 113 minutes. It moves rapidly around the Earth and observes a rapidly changing GNSS satellite geometry. | Main space-user orbit determination case and primary subject of this thesis. |

*Table 3: Experimental scenarios used in this thesis, organized by chapter.*

| Chapter | Scenario | Users | Product | Estimator | Key parameters |
|---|---|---|---|---|---|
| **3.7** | Configuration validation | BRUX | FIN | Filter | Correction models, constellation, sampling interval |
| **4** | Reference solution | BRUX, S6 | FIN | Filter | Conv period excluded, 30s/10s sampling for S6 |
| **5** | Batch FIN | BRUX, S6 | FIN | Batch | B10/30/60/90/120/180 min, |
| **7** | Real time reference | BRUX, S6 | HAS | Filter | Conv period excluded, 30s/10s sampling for S6 |
| **8** | Batch HAS | BRUX, S6 | HAS | Batch | B10/30/60/90/120/180 min, |
| **10** | Robustness | S6 | HAS | Filter + Batch | DIA on/off |

## 3.1 GHASP3 Processing Software

The processing in this thesis is carried out using GHASP3, which stands for Galileo High Accuracy for Space Precise Point Positioning. GHASP3 is an ESA project developed by the GHASP3 consortium (DLR and Delft University of Technology ( TU Delft) ) to support precise orbit determination for Low Earth Orbit (LEO) satellites using Galileo High Accuracy Service (HAS) corrections [12]. As discussed in Section 2.3, the thesis uses a kinematic POD framework, which is the approach adopted in GHASP3 for the real-time and near-real-time scenarios considered here.

As introduced in Section 2.4, GHASP3 provides two estimation strategies: GHASP3 Filter for real-time processing and GHASP3 Batch for near-real-time and non-time-critical processing. Both use an undifferenced and uncombined PPP formulation, in which the original GNSS code and carrier-phase observations are processed directly rather than being combined into ionosphere-free observables [4], [12]. This provides a common processing framework for comparing filter and batch solutions under the same observation and correction setup.

Galileo HAS corrections are broadcast through the E6B signal and can therefore be used directly onboard [12]. This makes HAS relevant for low-latency LEO navigation scenarios, where precise orbit information is needed in real time or near real time [12].

To run multiple scenarios efficiently and consistently, a Python automation script was developed to configure and execute GHASP3 for all experiments in this thesis.

## 3.2 Users and Datasets

Two users are considered in this thesis: BRUX as the ground user and Sentinel-6A (S6) as the space user, as summarized in Table 2. All experiments use data from Day of Year (DOY) 182, 2025, corresponding to 1 July 2025, processed with GPS and Galileo dual-frequency observations.
BRUX is a static receiver with precisely known coordinates, making it a reliable validation baseline. Its observations are processed at a 30 second sampling interval and are affected by the troposphere, which is modelled and estimated as part of the PPP solution.
S6 is part of the Copernicus Sentinel-6 mission, dedicated to high-precision sea-level measurements from space [13]. It operates at approximately 1336 km altitude with an orbital period of about 113 minutes. Its observations are processed at a 10 second sampling interval. Operating above the troposphere, S6 is not affected by tropospheric delay. However, ionospheric effects remain relevant and are estimated as a parameter in the uncombined PPP model.

## 3.3 GNSS Products Used

The theoretical background of the GNSS products was introduced in Section 2.6. In the experiments of this thesis, two product types are used: FIN and Galileo HAS.
“FIN” stands for Final products provided by the international GNSS Service (IGS) and used as reference quality products. Since they are post-processed and provide the highest accuracy among the products considered here, they are used to establish the reference filter and batch solutions in Chapters 4 and 5.

Galileo HAS products are used as the real-time operational product. They are used to evaluate how the filter and batch solutions perform when low-latency corrections are used instead of post-processed precise products. The HAS based experiments are analyzed in Chapters 7 and 8.
This product selection allows the thesis to compare not only filter and batch estimation, but also the effect of product latency: post-processed FIN products versus real time HAS corrections.

## 3.4 Processing Settings

The standard processing settings used across all experiments in this thesis are summarized in Table 4. Any deviation from these settings is stated explicitly in the relevant chapter.

*Table 4: Standard processing settings used in all experiments.*

| Parameter | Setting |
|---|---|
| Filter time window | 01:00 – 04:59 (4h) |
| Filter evaluation window | 02:00 – 04:59 (first hour excluded) |
| Batch time window | 02:00 – 04:59 (3h) |
| Constellation | GPS + Galileo |
| Frequencies | GPS: L1/L2 & Galileo: E1/E5a |
| BRUX sampling interval | 30 s |
| S6 sampling interval | 10 s and 30s |
| Elevation mask | 5 deg |
| DIA | DIA is enabled in all experiments. Detected faults are marked with red triangles in plots where the visualization setting is active; their absence does not indicate DIA was inactive. |
| LOMPre threshold | 0.001 |
| Process noise | See Section 3.6 |

Note (1): The first hour of the filter run is excluded from plots and statistics to remove the convergence period, ensuring a fair comparison with the batch solution over the same effective window (02:00–05:00).

Note (2): During initial testing, a boundary artifact was observed at the end of the time window when the end time coincided exactly with a full hour (e.g. 05:00). This was resolved by setting the end time to one sampling interval before the end (e.g. 04:59:30 for 30s sampling, 04:59:50 for 10s sampling).

## 3.5 Python Automation Script

Running GHASP3 for many different scenarios - each with different users, products, solvers, batch lengths, and settings - would be very tedious to do manually. To manage this efficiently, I developed a Python automation script that handles the full processing workflow.

The script starts with an interactive prompt asking for the key parameters: user, day, year, time window, solver and product. Each input triggers automatic lookups that map to the correct configuration values. For example, the year determines which file naming convention to use for the correction products. The user selection retrieves either the precise reference coordinates of the ground station for that day, or the reference SP3 orbit filename and satellite specific constants for S6, including its user code, yaw offset, and solution point in the body frame. The product selection resolves the correct folder paths for orbit, clock, attitude, bias, and SSR correction files, along with their provider, type, and latency identifiers. The solver selection maps to the correct GHASP3 processing mode and triggers the assignment of process noise values for each estimated parameter, as described in next section: Section 3.6.

It then fills in two configuration file templates, one for GHASP3 and one for the Analysis and Validation Tool (AVT), which are saved and used to launch the processing. All outputs are organized into a structured folder by run name.

The workflow is summarized in Figure 4.

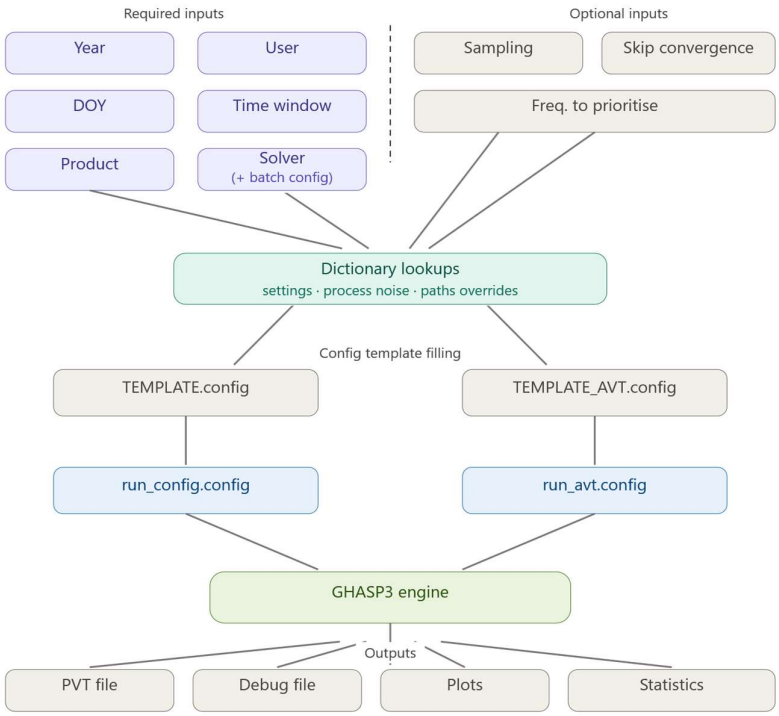



*Figure 4: Overview of the Python automation script workflow.*

## 3.6 Process Noise

As discussed in Section 2.4.1 and shown in the filter-vs-batch figure in Section 2.4.3, the Kalman filter works epoch by epoch, with a prediction step followed by an update step, so the previous epoch still matters. That is why process noise tells the filter how much it should rely on its previous estimate versus the new observation. In the batch approach, all observations are adjusted together over the full arc, so the same process noise values act more like a constraint on the whole batch than as a step by step prediction term [16].

Low process noise means the filter thinks the value will stay almost the same from one epoch to the next. So it does not let the predicted state move very much, and it relies more on the previous estimate than on a big change from the new observation.

In GHASP3, the process noise values differ between the filter and the batch. For the batch, the ambiguity process noise is set to 0 in all cases, while for the filter a value of 0.001 m/$\sqrt{s}$ is used for the ambiguities when processing HAS data. For the ionosphere, the space user values are reduced in the batch compared to the filter, from 1 m/$\sqrt{s}$ in the filter to values ranging between 0.001 and 0.05 m/$\sqrt{s}$ in the batch depending on the scenario.

## 3.7 Configuration Validation

### 3.7.1 Correction Model Sensitivity

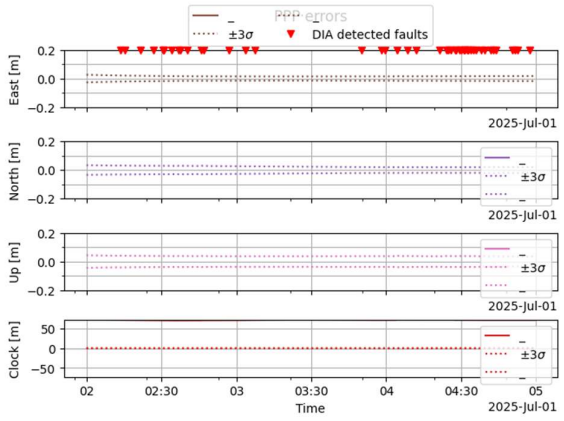
A first sensitivity analysis was realized with GHASP3 to familiarize myself with the error models used in GNSS processing. To assess which modeled effects tend to have the greatest impact on the solution, two sensitivity tests were performed on BRUX: one with troposphere correction disabled and one with solid Earth tides correction disabled. These tests isolate how much each modeled effect contributes to the final solution.

Tropospheric delay is a major correction because the GNSS signal passes through the full neutral atmosphere on its way to the ground receiver, where it experiences a substantial path delay. This delay is strongly correlated with station height and receiver clock, so leaving it unmodeled can seriously bias the solution [17], [1].

Solid Earth tides are the periodic elastic deformation of the Earth's crust caused mainly by the gravitational pull of the Moon and the Sun. They physically shift the station position by small but measurable amounts, typically at the centimeter level. Omitting this correction leaves a periodic position error unmodeled, even though the effect is much smaller than the troposphere [18].
The results are compared to the full model reference in Table 5.

*Table 5: RMS statistics for BRUX filter FIN with individual corrections disabled.*

| Configuration | 3D RMS [m] | 2D RMS [m] |
|---|---|---|
| Full model (reference) | 0.019 | 0.011 |
| No troposphere corrections | 9.028 | 2.388 |
| No  Earth tides corrections | 0.069 | 0.044 |




The troposphere test has by far the largest effect. The 3D RMS increases from 1.9 cm to 9.0 m, confirming that unmodeled tropospheric delay severely degrades the BRUX solution.

*Figure 5: BRUX filter FIN position error plot with troposphere correction disabled*

Disabling the solid Earth tides correction has a much smaller but still noticeable effect, with the 3D RMS increasing to 6.9 cm.

### 3.7.2 Multi GNSS and Frequency Configuration

All experiments in this thesis use GPS and Galileo dual-frequency observations, specifically GPS L1/L2 and Galileo E1/E5a. This configuration is kept fixed across all scenarios so that the results reflect the influence of the processing strategy, rather than changes in the observation model. Dual-frequency carrier-phase observations are preferred because they improve ionospheric observability and make ambiguity handling more robust in the uncombined PPP formulation [4], [12], [17].

To justify the use of both constellations, two additional sensitivity runs were performed using GPS only and Galileo only. The results are summarized in Table 6.

*Table 6: RMS statistics for BRUX filter FIN with single constellation processing.*

| **Configuration** | **3D RMS [m]** | **2D RMS [m]** |
|---|---|---|
| GPS + Galileo (reference) | 0.019 | 0.011 |
| Galileo only | 0.032 | 0.017 |
| GPS only | 0.038 | 0.018 |

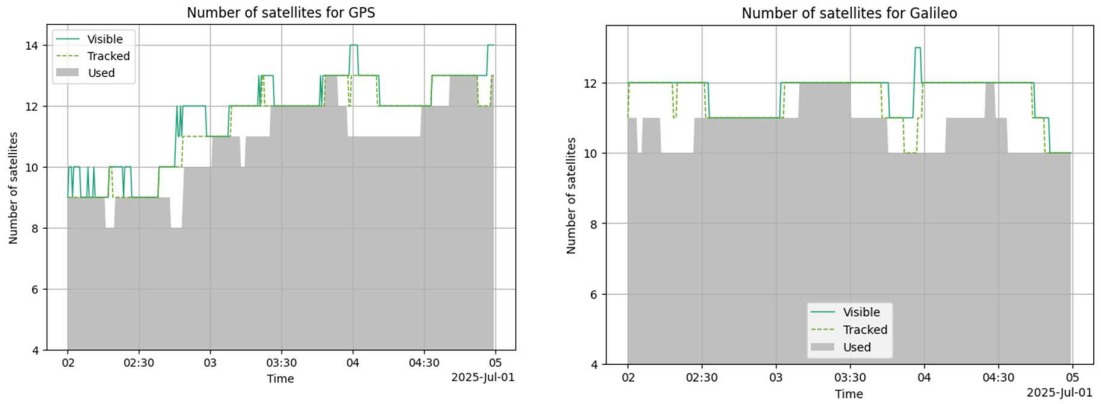
Both single constellation solutions perform worse than the combined GPS + Galileo reference, which shows that the improvement does not come from one constellation alone, but from the larger and more diverse observation set obtained when both are used together [18], [19].

The main reason is satellite geometry. In PPP, the quality of the position estimate depends not only on the number of available satellites, but also on how well those satellites are distributed in the sky. This is captured by DOP, or dilution of precision, which is a measure of geometry quality, when the satellites are spread well across the sky : DOP is low and the position estimate is better conditioned, but when they are clustered, DOP is high and the same measurement errors have a larger effect on the final position.

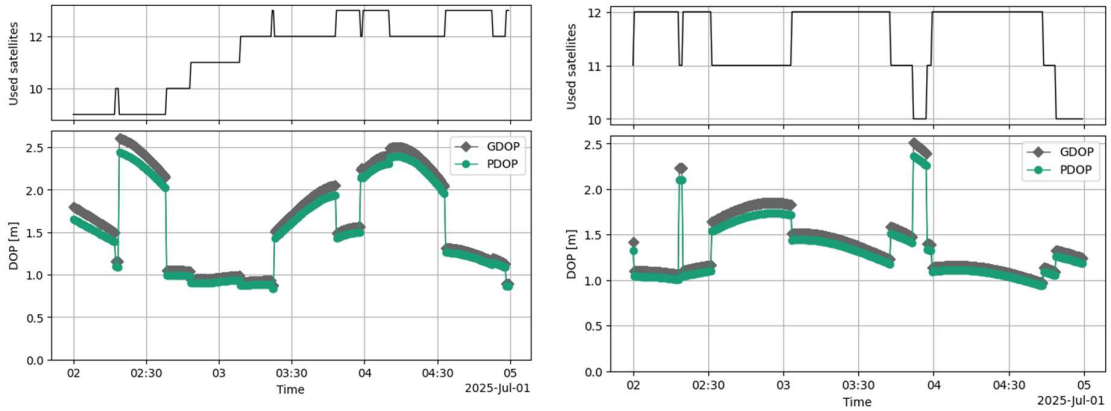
*Table 7: number of satellites and DOP for GPS only ,Galileo only and Multi-GNSS runs.*

**GPS only** **Galileo only**

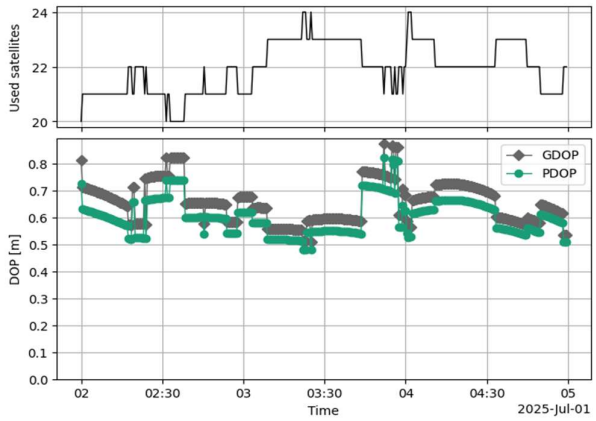
Number of Satellites Plot




Dilution of Precision (DOP) Plot




**Multi GNSS**




A combined multi-GNSS setup generally gives better spatial coverage and better observability than a single constellation, especially when one constellation is temporarily less favorable [18]. For this reason, the Multi GNSS and dual-frequency setup is used as the standard configuration for the rest of the thesis.

## 3.8 Evaluation Metrics

This section introduces the metrics used to evaluate the filter and batch solutions throughout the analysis chapters (4 - 8) in this thesis. The same metrics are applied consistently across all experiments to allow direct comparison.

### 3.8.1 Reference Frames and Component Interpretation

Position errors are expressed in a local reference frame. For BRUX – ground user -, errors are given in the East-North-Up (ENU) frame in metres, relative to the known reference coordinates of the station. For S6 – space user -, errors are given in the Radial-Along-track-Cross-track (RSW) frame in metres, relative to a precise reference orbit. The thesis focuses only on position estimation comparison, not on clock errors (even though this option is also available in GHASP3).

**Reading the plots:**

*Table 8: Reference frames and typical behavior of position error components for BRUX (ENU) and S6 (RSW).*

| Component | Frame | Physical meaning | Typical behavior |
|---|---|---|---|
| East | ENU | Horizontal, pointing East | Clean and stable |
| North | ENU | Horizontal, pointing North | Clean and stable |
| Up | ENU | Vertical, away from Earth | Often noisier, weaker vertical geometry, sensitive to troposphere |
| Radial | RSW | Orbit height, outward from Earth's center | Often the most sensitive, absorbs clock, orbit, and calibration errors |
| Along-track | RSW | Direction of satellite motion | Often smoother, may show slow drift |
| Cross-track | RSW | Perpendicular to orbital plane | Often comparatively stable, can reflect orbit-plane constraint quality |

For BRUX, the Up component is weaker because all GNSS satellites are above the station, giving poor vertical geometry, and residual tropospheric delay projects directly onto it. Additional variability comes from local site effects such as multipath - this happens when GNSS signals bounce off nearby surfaces such as buildings or the ground before reaching the antenna, causing the receiver to pick up both direct and reflected signals.

For S6, the Radial component is the most sensitive and typically shows the largest variability. Since GHASP3 uses a kinematic approach with no force model, all

components rely entirely on the GNSS observations and the quality of the correction products. Among them, the Radial direction tends to be the hardest to constrain [6], [14].

GHASP3 reduces these error sources through modelling, estimation, and external corrections: tropospheric delay is modelled and estimated for ground users; ionospheric delay is estimated as a parameter using the undifferenced uncombined observation model; clock, orbit, and bias errors are reduced through precise external products such as FIN or HAS; phase center calibration is accounted for, and outliers and cycle slips are handled by DIA. But multipath for both users remain as a residual error [15]. These modelling choices are described in detail in Sections 2.1.2.

Throughout Chapters 4 to 8, only deviations from the expected behaviour described above are discussed in detail.

### 3.8.2 RMS Error

The Root Mean Square (RMS) error is the main accuracy metric used in this thesis, expressed in metres. Two values are reported:

- **3D RMS**: combines all three spatial components (East, North, Up for BRUX or Radial, Along-track, Cross-track for S6)
- **2D RMS**: combines only the horizontal components (E, N for BRUX; Along-track and Cross-track for S6, excluding Radial)

Lower RMS means better accuracy.

### 3.8.3 Availability of Accuracy

The availability of accuracy is the fraction of epochs where the position error stays below a given threshold, expressed as a percentage. For example:

*Table 9: Example of Availability of Accuracy.*

| | |
|---|---|
| Availability 3D <=1.0m | 1 |
| Availability 2D <=0.01m | 0.47 |

A value of 1.0 at the 1m threshold means the solution stays within 1m at every single epoch. A value of 0.47 at the 1cm threshold means that 47% of epochs have a horizontal error below 1cm. The full statistics tables are presented in the analysis chapters.

### 3.8.4 Accuracy, Precision and Formal Uncertainty

Accuracy refers to how close the solution is to the true position, measured in metres by the RMS error.

Precision refers to how consistent and stable the solution is from epoch to epoch, reflected by the ±3σ formal uncertainty bounds in the plots.

These bounds, the dotted lines above and below, are derived from the covariance matrix of the estimated parameters: a narrow band means the estimator is confident, a wide band means it is uncertain.

A solution can be precise but inaccurate if it is consistently biased, or accurate on average but imprecise if it varies a lot from epoch to epoch. Both are reported throughout Chapters 4 to 8.

### 3.8.5 Convergence and Evaluation Window

The filter requires some time at the start of processing before the solution stabilizes. This convergence period depends on the scenario, but in all cases tested in this thesis one hour appears to be sufficient. In GHASP3, convergence is defined as the point at which the 3D RSW positioning error enters and remains within ±10 cm. Exact convergence times by scenario and correction product are reported in Section 11.1.

Figure 6 shows a zoomed view of the first 30 minutes of the BRUX filter FIN run. At the very start, the ±3σ bounds are noticeably wide, the estimator is uncertain and the solution has not yet settled. As more observations accumulate, the bounds progressively narrow and the errors reduce, which is the convergence happening in real time. By around 02:00 the solution is stable and the bounds are tight.

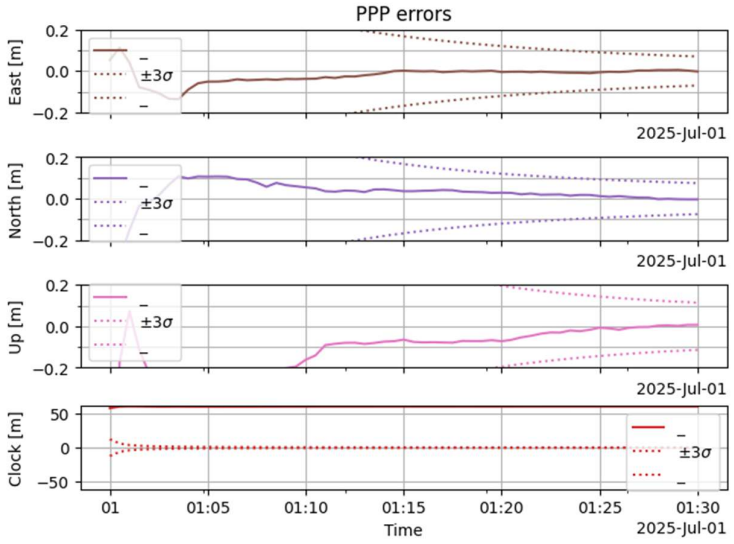



*Figure 6: Zoomed view of BRUX filter FIN solution during the convergence period (01:00–01:30)*

For this reason, the first hour of each filter run is excluded from all plots and statistics. Figure 7 shows the full processing window from 01:00 to 04:59, where the contrast between the wide bounds during convergence and the tight bounds after 02:00 is clearly visible. All statistics are computed over the evaluation window 02:00–04:59, which is the same for both filter and batch runs, to ensure a fair comparison.

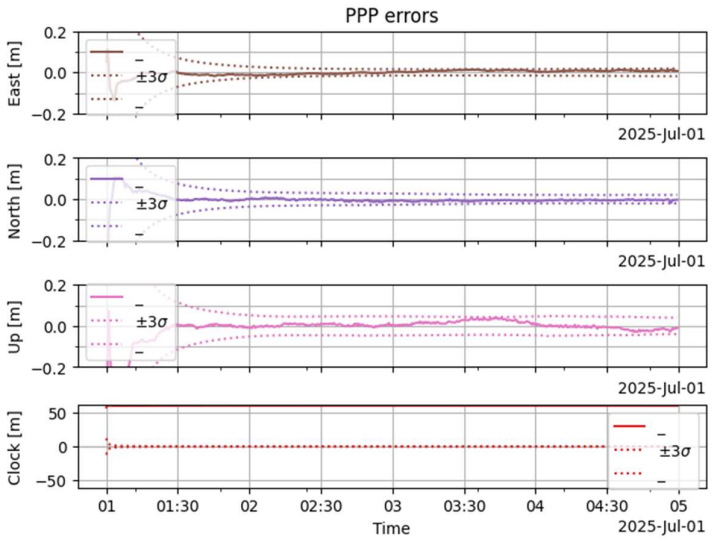



*Figure 7: BRUX filter FIN position error plot over the full processing window*

# 4 Reference Solution - Filter with FIN Products

This chapter presents the filter solution for BRUX and S6 using FIN products as the reference quality scenario. The filter is run over a 4hour window (01:00–04:59), with the first hour excluded from plots and statistics to remove the convergence period. The results serve as the baseline against which all batch solutions in Chapter 5 are compared.

## 4.1 Filter Solution - BRUX

Figure 8 shows the position error plot for BRUX using the Kalman filter with FIN products over the evaluation window 02:00–04:59.

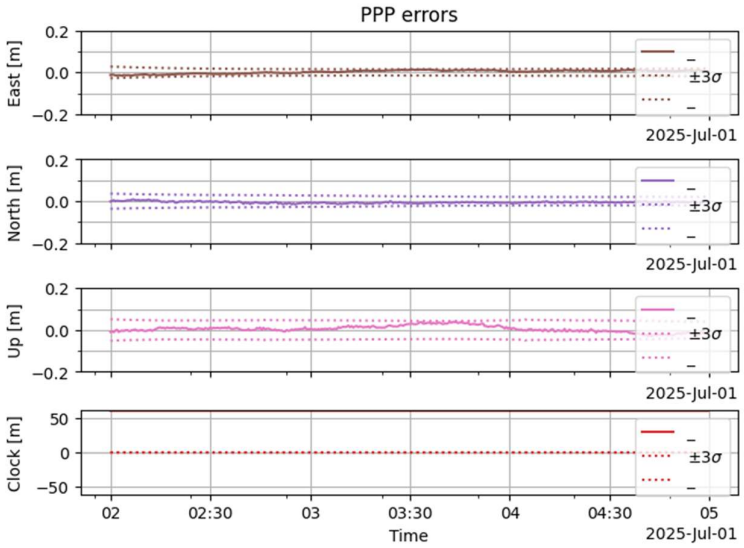



*Figure 8: Filter BRUX FIN position error plot.*

The solution is stable and well converged throughout the evaluation window. The East and North errors remain close to zero with very narrow ±3σ bounds, indicating high precision in the horizontal components. The Up component shows slightly larger variations, reaching up to approximately 5cm, consistent with expectations discussed in Section 3.8.1. The statistics are summarized in Table 10.

*Table 10: Statistics for BRUX filter FIN. Evaluation window 02:00–05:00, DOY 182 2025.*

| | |
|---|---|
| ENU_3DErrorRMS | 0.019 m |
| ENU_2DErrorRMS | 0.011 m |
| | |
| Availability 3D <=0.01m | 17.8 % |
| Availability 3D <=0.02m | 71.1 % |
| Availability 3D <=0.5m | 100 % |
| Availability 2D <=0.01m | 46.9 % |
| Availability 2D <=0.02m | 99.4 % |

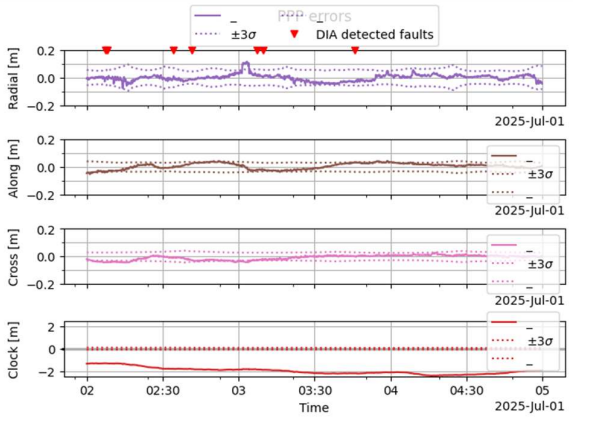
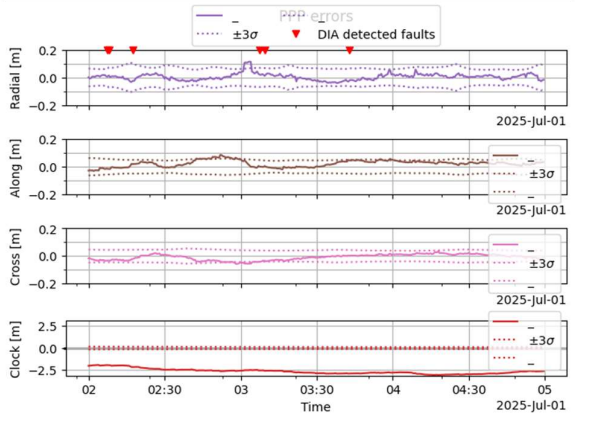
The filter achieves a 3D RMS of 1.9cm and a 2D RMS of 1.1cm, confirming that cm level accuracy is achieved for a static ground user with post-processed products. The solution is available within 5cm at every epoch, and within 2cm at 71% of epochs in 3D. This result establishes the reference performance level for BRUX with FIN products, against which all batch solutions in Chapter 5 are compared.

## 4.2 Filter Solution - S6

Table 11 shows the RSW position error plots for S6 using the Kalman filter with FIN products at 10s and 30s sampling, over the evaluation window 02:00–04:59.

The filter was run for S6 using both 10-second and 30-second sampling intervals to assess the impact on positioning accuracy.

*Table 11: RSW position error RMS for S6 filter with final products at 10s and 30s sampling*

| S6 FIN 10s sampling | S6 FIN 30s sampling |
|---|---|
|  |  |
| **RSW 3D Error RMS [m]** | |
| 0.038 | 0.046 |
| **RSW 2D Error RMS [m]** | |
| 0.029 | 0.040 |

The Radial component shows the largest variability, consistent with the interpretation given in Section 3.8.1. Visually the two sampling rates appear very similar, the 10 second solution looks bolder simply because more epochs are plotted. The difference only becomes apparent in the statistics.

*Table 12: Availability statistics for S6 filter FIN at 10s and 30s sampling.*

| | S6 FIN 10s | S6 FIN 30s |
|---|---|---|
| Availability 3D <=0.01m | 0.8% | 3.8% |
| Availability 3D <=0.02m | 8% | 21.5% |
| Availability 3D <=0.05m | 79.2% | 90.7% |
| Availability 3D <=0.1m | 98.3% | 98.6% |
| Availability 3D <=0.2m | 100% | 100% |

At 10 second sampling, a 3D RSW RMS of 3.8 cm was achieved, compared to 4.6 cm at 30 second sampling. At the 5 cm threshold, availability increases from 79.2% to 90.7% from 30s to 10 second sampling rate. The higher sampling rate provides more observations per processing window. This is particularly relevant for a fast-moving LEO satellite such as Sentinel-6, where the geometry changes rapidly and denser observations better capture the satellite's trajectory.

## 4.3 Convergence Analysis

As introduced in Section 3.8.5, the filter requires a convergence period at the start of processing during which the solution is not yet reliable. This section examines how convergence appears in practice for both users.

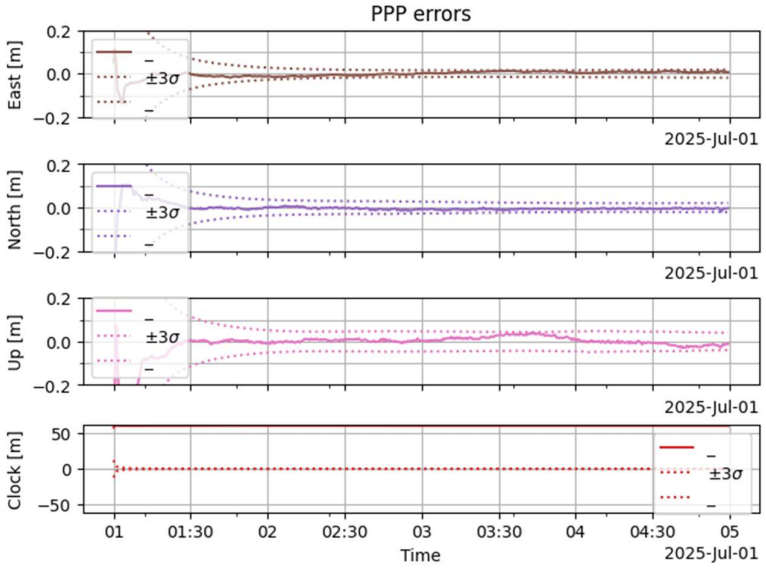



*Figure 9: Filter BRUX FIN position error plot including convergence period.*

For BRUX, Figure 9 shows the full filter run from 01:00 to 05:00. During the first hour, the ±3σ bounds are wide in all three components, and the errors are larger and less stable than after 02:00. This is consistent with the filter needing time to initialize the phase ambiguities and estimate the tropospheric wet delay. By 02:00 the bounds narrow significantly and the solution stabilises, which is why the first hour is excluded from all statistics.

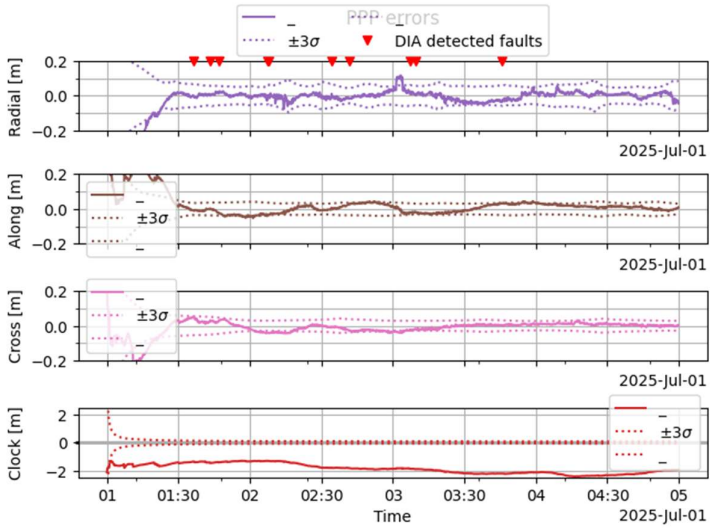



*Figure 10: Filter S6 FIN position error plot including convergence period.*

For S6 with 10s sampling interval: Figure 10= shows a different convergence pattern. The Along-track and Cross-track components settle relatively quickly, within the first 30 to 45 minutes. The Radial component however still varies and its ±3σ bounds remain wider than the other components even after convergence. This is consistent with the Radial component being the hardest to constrain for a space user, as discussed in Section 3.8.1, since it absorbs clock, orbit, and calibration errors.

In both cases, the solution is visually stable well within the first hour, which justifies the evaluation window starting at 02:00 for all experiments. It is also worth noting that convergence behavior varies significantly depending on the correction product used. Under HAS corrections, the filter takes considerably longer to stabilize than under FIN, particularly for S6, where convergence becomes a critical limitation rather than a minor inconvenience. A formal convergence analysis with exact convergence times across all scenarios and correction products is provided in Section 11.1.

## 4.4 Ground vs Space User Analysis

The filter results for BRUX and S6 using FIN products are summarized in Table 13. The two users show clearly different accuracy levels, which reflect the fundamental differences between ground and space PPP rather than a limitation of the processing.

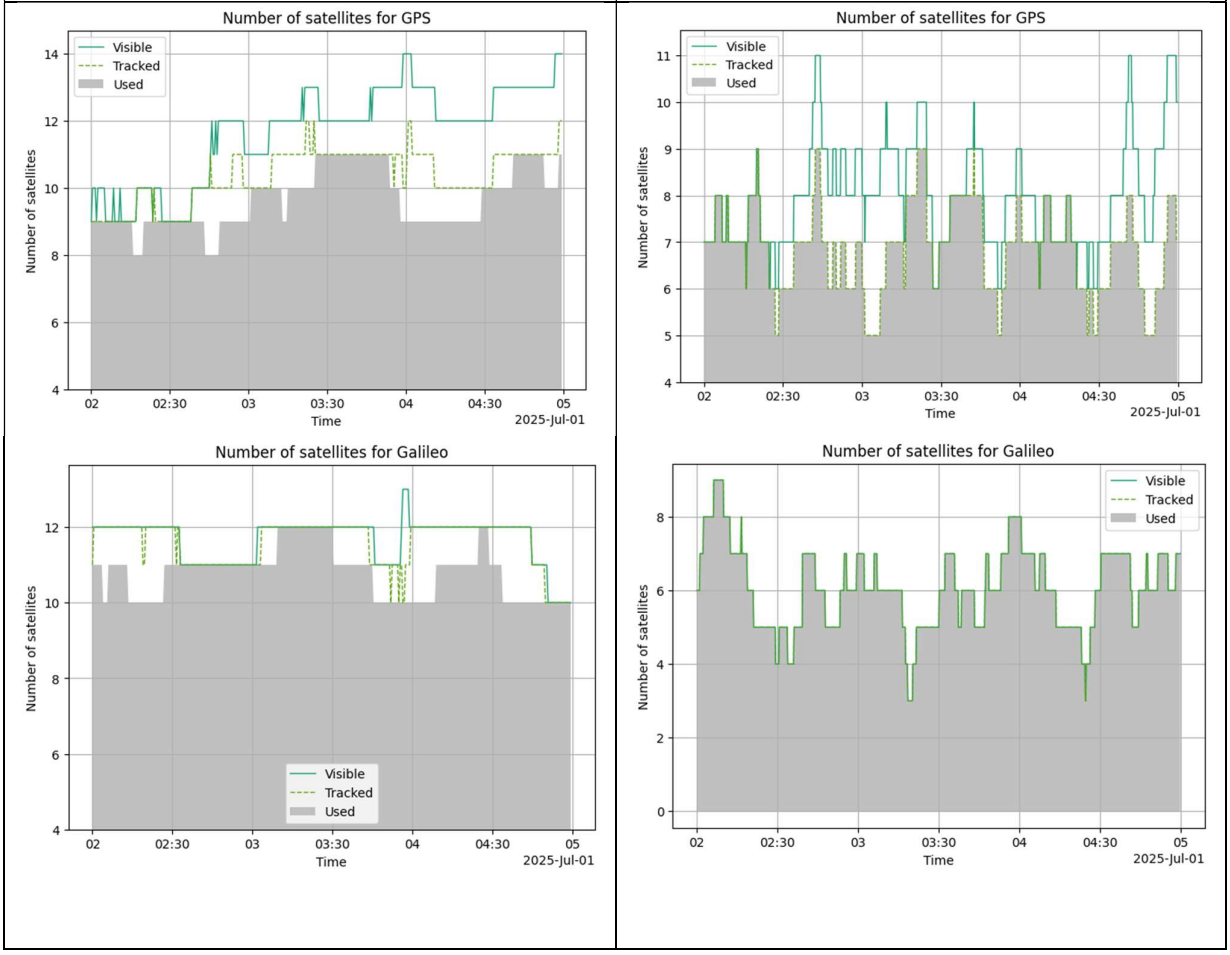
*Table 13: 3D and 2D RMS comparison between BRUX and S6 filter FIN.*

| User | 3D RMS [m] | 2D RMS [m] |
|---|---|---|
| BRUX (ENU) | 0.0191 | 0.0107 |
| S6 (RSW) | 0.0376 | 0.0292 |

BRUX achieves a 3D RMS of 1.9 cm, while S6 achieves 3.76 cm. The difference is expected and can be explained by several factors.

*Table 14: Number of Multi-GNSS satellites for BRUX and S6.*

| Number of Multi GNSS Satellites for BRUX | Number of Multi GNSS Satellites for S6 |
|---|---|




The most visible difference is in satellite tracking and the nature of the estimation problem. For BRUX, the satellite count is stable throughout: GPS maintains 8 to 11 satellites and Galileo 10 to 12. For S6, the GPS count fluctuates between 5 and 9 and Galileo drops as low as 3, with frequent rises and falls throughout the window. This is a direct consequence of S6's orbital motion at approximately 7.5 km/s, as S6 moves rapidly around the Earth, GNSS satellites continuously rise and set, producing a rapidly changing and sometimes sparse observation geometry. Each time a satellite is lost a new phase ambiguity must be initialized, reducing redundancy.

Furthermore, unlike BRUX which is static and benefits from accumulated observations refining the same fixed point, S6 must estimate a new position at every epoch independently, making the estimation problem fundamentally more dynamic. BRUX also uses a geodetic-grade receiver, which may contribute to its higher positioning quality compared to S6. More details in Section 5.1 Design Matrix Analysis.

The Radial component adds another layer of difficulty. As discussed in Section 3.8.1, for S6 the Radial direction absorbs errors from clock, orbit, and phase center calibration, which directly inflates the 3D RMS compared to BRUX where all three ENU components are well constrained. Small remaining model errors may also contribute to the observed differences between the two users, though their effect is at the centimeter level and therefore limited.

Despite these differences, both users achieve centimeter level accuracy with FIN products, confirming that the filter provides reliable solutions for both ground and space users. The S6 results serve as the reference space user baseline against which all batch solutions in Chapter 5 are compared.

# 5 Batch Processing with FIN Products

This chapter investigates the batch least squares solutions for BRUX and S6 using FIN products. The same 3-hour window (02:00–04:59) and correction products used in the filter reference solution in Chapter 4 are applied here, allowing a direct and fair comparison between the two estimation strategies. Before presenting the results, Section 5.1 analyses the structure of the batch design matrix, as the observation-to-unknown ratio and geometric diversity directly determine how reliably the batch can estimate all parameters. Two experiments are then presented: different batch lengths over the full 3-hour window (Section 5.2) and batch length equal to the computation window, for different interval lengths (Section 5.3).

## 5.1 Design Matrix Analysis

In the batch least squares approach, all observations over the selected arc are stacked into one adjustment problem and the unknown parameters are estimated jointly. As introduced in Section 2.4.2, this is the fundamental difference from the Kalman filter, which processes observations one epoch at a time [8]. The structure of the resulting design matrix directly determines how well the solution can be estimated, and understanding it helps explain the batch behaviour observed in Sections 5.2 and 5.3 for FIN and in Sections 8.1 and 8.2.

The design matrix A is formed from the linearized observation equations. Each row corresponds to one observation and each column corresponds to one unknown parameter. Each entry of the matrix is a partial derivative - it describes how sensitive a given measurement is to a small change in one specific parameter. If the matrix is well conditioned, meaning the observations constrain the unknowns from diverse geometric directions, the solution is reliable and accurate. If the geometry is weak or the observations do not constrain certain parameters strongly enough, the solution becomes less stable and the position estimate degrades [1][8].

To give an idea of the scale of the system, the number of observations and unknowns can be approximated as:

- Observations : $4 \times N_{sat} \times N_{epochs}$ (2obs per freq = 4 per sat)
- Unknowns : $3 \times N_{epochs} + N_{epochs} + 1 + N_{sat} \times N_{epochs} + N_{sat} \times N_{freq}$ (=2)
  (position) (clock) (ISB) ( iono ) ( amibguities)
  Multi GNSS

*Figure 11: Simplified parameter count for the batch design matrix, S6 case*

These are simplified counts used to illustrate the structure of the batch design matrix for the S6 case where the position is estimated at every epoch since the satellite is continuously moving. For BRUX, the receiver position is estimated once as a static 3D state over the full arc. In practice, the exact number of observations depends on satellite visibility and data gaps, and the number of unknowns depends on which satellites are present and which ambiguities are introduced during the arc [8].

With this many parameters spread across the full arc, the batch design matrix becomes large but crucially, it is also sparse.

The source of this sparsity becomes clearer when looking at how the batch matrix is built. Because all epochs across the full arc are processed together from the start, the matrix must be constructed upfront with columns for every satellite that will appear at any point during the window. This means that at the very first epoch, columns already exist for satellites that have not yet risen. Those columns are filled with zeros for all epochs before the satellite appears, and only start receiving non-zero entries once the satellite joins the arc. Similarly, a cycle slip mid-arc forces a new ambiguity column to be introduced, which is again only partially filled from that point onwards [1][8][12].

Figure 12 illustrates this clearly for a representative 5-minute arc of S6 with 30-second sampling, giving 10 epochs and 57 unknowns. G19 joins at t6 and E33 joins at t8, so their ambiguity columns are empty for the first several epochs. Their parameters are therefore weakly constrained compared to satellites tracked from the beginning, and they need enough subsequent observations before they become well estimated.


*Figure 12: Batch design matrix structure, S6, DOY 182, 5-minute arc, 30s sampling*

The Kalman filter handles this very differently. Because it processes observations one epoch at a time, each update uses only the satellites currently visible. When a new satellite appears, it is added to the state update at that moment, and when one disappears, it is no longer used in that epoch's update. At each epoch, the filter therefore works with a smaller and less sparse system than the batch matrix. The batch approach pays a price for its global estimation, because it must pre-allocate space for all satellites that appear anywhere in the arc, which produces the sparse partially filled structure described above [8].

This structural sensitivity is why batch performance depends strongly on window length, which is examined in the following sections.

## 5.2 Repeated Batch Analysis - BRUX and S6

In this experiment, the full 3-hour window is divided into consecutive batches of a fixed length, repeated until the end of the window. Six batch lengths are tested: 10, 30, 60, 90, 120, and 180 minutes. The goal is to understand how the batch length affects the accuracy and stability of the solution when the same total amount of data is available.

The position error plots for B10, B30, B60, B90, B120 and B180 are shown for BRUX in Figures 13 - 18. The full statistics for all batch lengths are summarized in Table 15, with the filter FIN result from Chapter 4 included as reference.

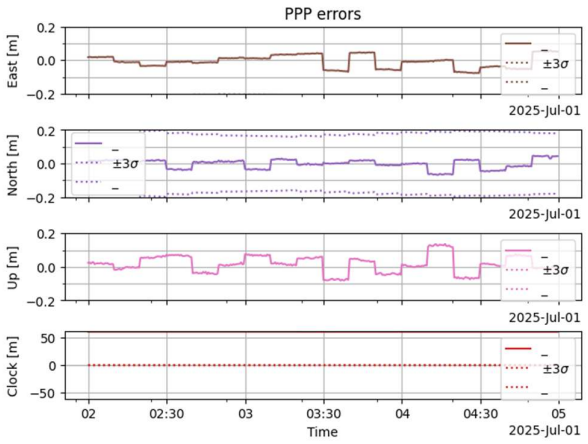



*Figure 13: BRUX FIN B10 plot*

### B10 : too short to be useful

At 10 minutes the 3D RMS is 7.2 cm. Boundary jumps appear at every batch transition in all three components, each new batch starts fresh with no memory of the previous one, with only 10 minutes of data the ambiguities and other parameters cannot be estimated reliably before the batch ends. The ±3σ bounds are very wide throughout, even out of the frame, confirming that the estimator itself is not confident in the solution. This is a direct illustration of what happens when the batch arc is too short to provide sufficient geometric diversity.

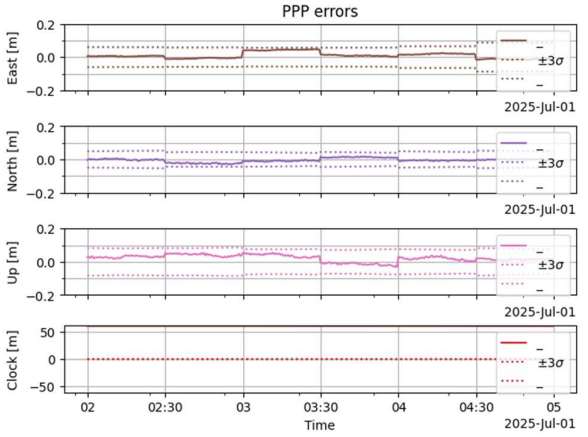



*Figure 14: BRUX FIN B30 plot*

### B30 : improving but still unstable

With 30 minute batches the 3D RMS improves to 3.8cm, but boundary jumps are still visible in the East component and the ±3σ bounds remain wider than the filter but this time present for all components. The solution is not yet stable enough to be considered reliable.

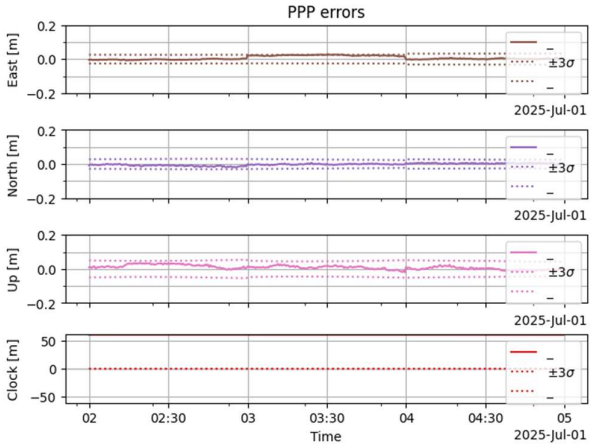



*Figure 15: BRUX FIN B60 plot*

### B60 : the turning point

At 60 minutes something interesting happens. The 3D RMS drops to 2.1cm, very close to the filter at 1.9cm. The boundary jumps essentially disappear and the ±3σ bounds tighten significantly. This suggests that 60 minutes is close to the minimum useful batch length for BRUX with FIN products. At this point the batch has enough observations and enough satellite geometry variation to estimate all parameters reliably.

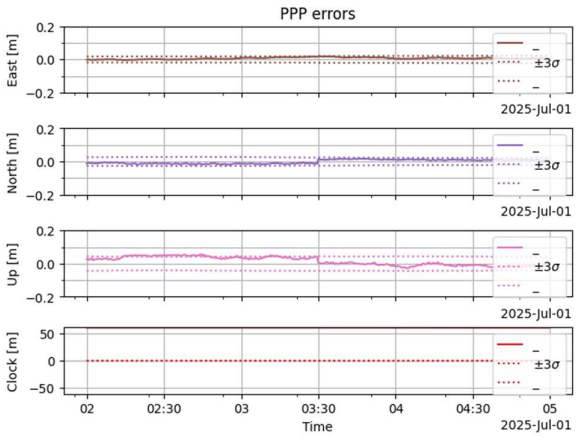



*Figure 16: BRUX FIN B90 plot*

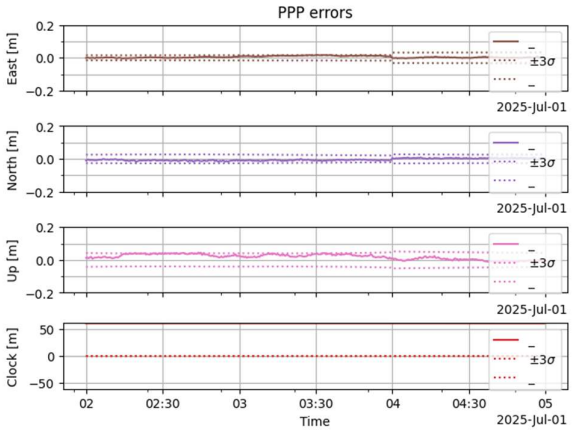



*Figure 17: BRUX FIN B120 plot*

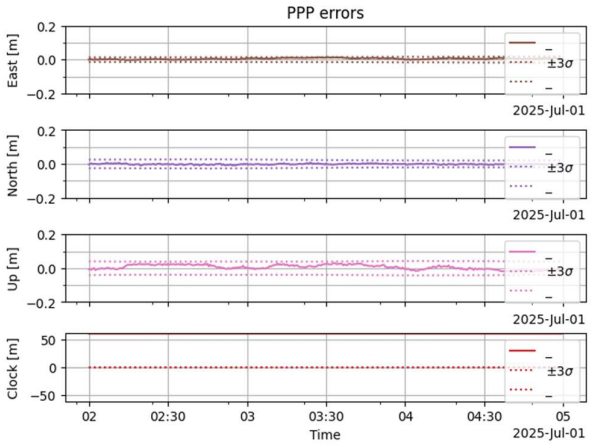



*Figure 18: BRUX FIN B180 plot*

**B90 : an unexpected result**

B90 is surprisingly worse than B60 at 3.3cm despite using more data per batch. Looking at the plot the solution appears visually clean, with East and North errors close to zero, yet the 3D RMS is higher than B60. The 2D RMS however remains comparable at 1.5 cm, meaning the degradation is mainly in the vertical component. One interpretation that is consistent with the design matrix analysis in Section 5.1: longer windows introduce more satellite arrivals and departures, creating additional partially constrained ambiguity columns that can degrade the vertical solution without strongly affecting the horizontal components.

**B120, B180 : matching and beating the filter**

B120 achieves 2.8cm, better than B90 and approaching the filter.
B180 achieves the best result of all at 1.7cm 3D RMS - slightly outperforming the filter reference at 1.9cm.
The B180 solution is remarkably clean: East and North errors stay very close to zero throughout the full window, and the ±3σ bounds are the tightest of all configurations. The 2D RMS of 0.76cm is also the best of all methods tested.

This is an interesting findings: a single 3 hour batch with FIN products is marginally more accurate than the Kalman filter over the same window. This makes intuitive sense: the batch uses all observations simultaneously in one global adjustment, so every observation contributes to every parameter estimate. The filter by contrast processes observations sequentially and its earlier estimates are not updated by later observations.

*Table 15: RMS statistics for BRUX FIN*

| Configuration | 3D RMS [m] | 2D RMS [m] |
|---|---|---|
| Filter FIN (reference) | 0.019 | 0.011 |
| Batch B10 | 0.072 | 0.047 |
| Batch B30 | 0.038 | 0.024 |
| Batch B60 | 0.021 | 0.015 |
| Batch B90 | 0.033 | 0.015 |
| Batch B120 | 0.028 | 0.011 |
| Batch B180 | 0.017 | 0.008 |

**Summary**

For a static ground user with FIN products, batch processing matches or exceeds filter accuracy for batch lengths of 60 minutes or longer. Below 30 minutes the solution becomes unreliable. Performance does not improve monotonically, B90 shows an unexpected degradation despite using more data. The full 3 hour batch achieves the best accuracy, slightly outperforming the filter. A minimum batch length of approximately 60 minutes appears to be needed for BRUX to achieve filter comparable performance.

For S6, the same six batch lengths are tested over the full 3-hour window. The position error plots are shown in Figures 19 - 24. The full statistics are summarized in Table 16, with the filter FIN result from Chapter 4 included as reference.

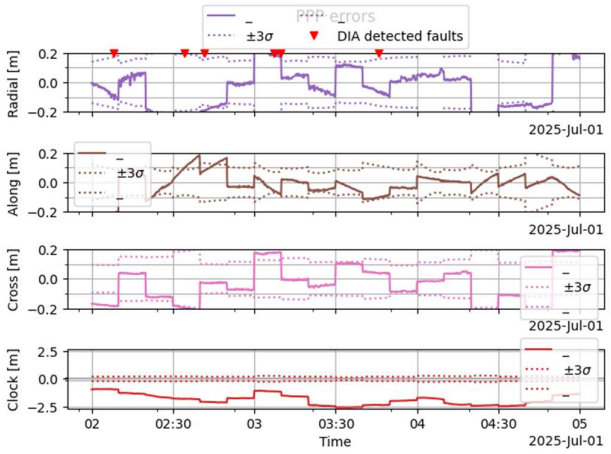



*Figure 19: S6 FIN B10 plot*

**B10 : too short for LEO**

The 10-minute batch produces the worst results by far, with a 3D RMS of 20.62 cm. The plots show large step-like jumps at every batch boundary across all three RSW components, even more pronounced than for BRUX. Each new batch starts fresh with no memory of the previous solution, and with only 10 minutes of data the fast-changing LEO geometry cannot provide sufficient satellite diversity for reliable parameter estimation.

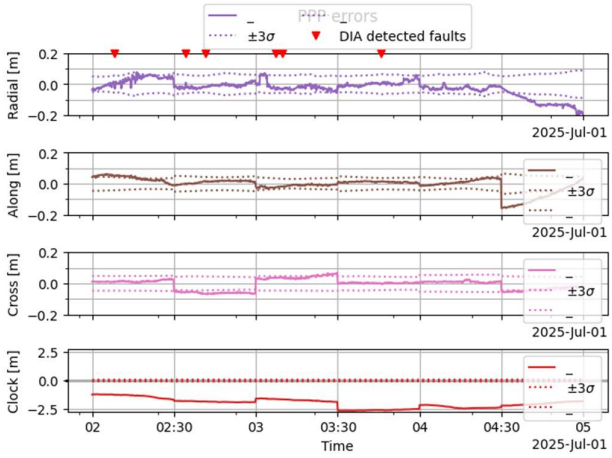



*Figure 20: S6 FIN B30 plot*

**B30 : improving but still unreliable**

At 30 minutes the 3D RMS improves to 7.99 cm, but this remains more than twice the filter reference. All components still show visible jumps at batch boundaries and radial component is noisier, the solution cannot yet be considered reliable.

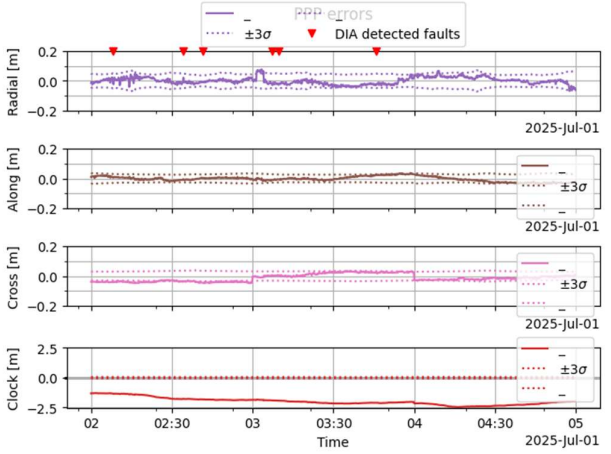



*Figure 21: S6 FIN B60 plot*

**B60 : the turning point**

At 60 minutes the 3D RMS drops to 3.91 cm, approaching the filter at 3.76 cm. The along-track and cross components are stable, though the radial still shows occasional spikes. As with BRUX, 60 minutes appears to be the minimum useful batch length for S6.

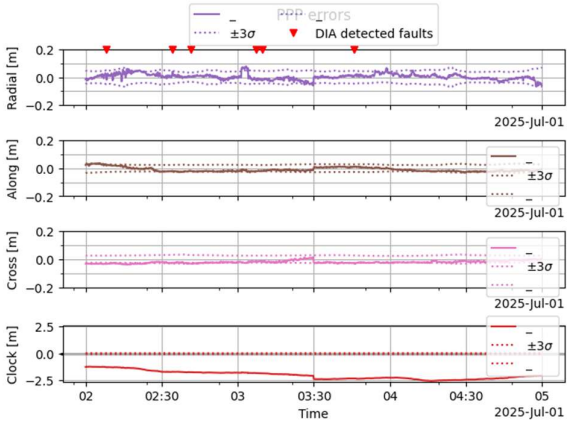



*Figure 22: S6 FIN B90 plot*

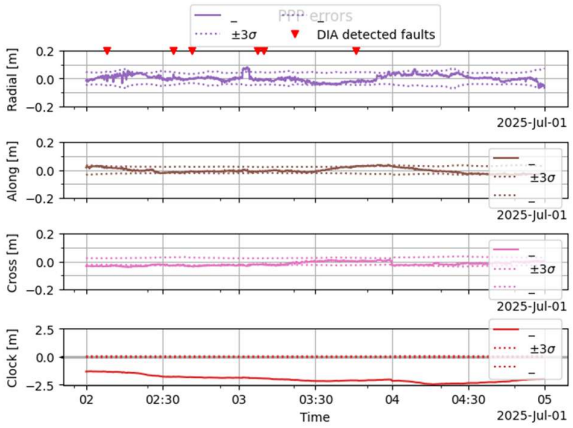



*Figure 23: S6 FIN B120 plot*

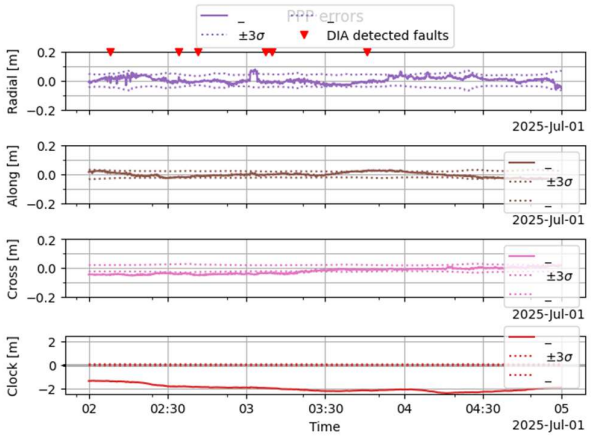



*Figure 24: S6 FIN B180 plot*

**B90 : better but radial still active**

At 90 minutes the 3D RMS improves to 3.60 cm and the 2D RMS to 2.93 cm, both better than B60. The along-track and cross-track components are clean and stable throughout the window. The radial however remains the most active component with visible variability, consistent with what was discussed in Section 5.1: the radial direction is the hardest to constrain for a space user, and at 90 minutes the batch is still accumulating enough satellite geometry to fully stabilise it.

**B120 : continuing to improve**

At 120 minutes the 3D RMS is 3.70 cm and the 2D RMS 2.88 cm, broadly comparable to B90. The ±3σ bounds are tighter than at shorter batch lengths, suggesting the estimator is becoming more confident as more data accumulates.

**B180: best horizontal, radial still dominant**

The single 3-hour batch achieves a 3D RMS of 3.57 cm and a 2D RMS of 3.08 cm. The horizontal components are the cleanest of all batch lengths, with very tight ±3σ bounds throughout the full window. The radial however remains the most variable component. Unlike BRUX where B180 clearly outperformed the filter, for S6 the gap is much smaller. But the difference is small enough that the two approaches can be considered equivalent at this window length.

*Table 16: RMS statistics for S6 FIN*

| Configuration | 3D RMS [m] | 2D RMS [m] |
|---|---|---|
| Filter FIN (reference) | 0.038 | 0.029 |
| Batch B10 | 0.206 | 0.148 |
| Batch B30 | 0.080 | 0.057 |
| Batch B60 | 0.039 | 0.032 |
| Batch B90 | 0.036 | 0.029 |
| Batch B120 | 0.037 | 0.029 |
| Batch B180 | 0.036 | 0.031 |

**Summary**

For S6, the results follow a similar trend to BRUX in that short batch lengths are clearly insufficient. B10 and B30 produce large errors and cannot be considered reliable. From B60 onwards the solution stabilizes and approaches the filter reference. Unlike BRUX however, performance for S6 improves more consistently with window length, without the pronounced dip at B90 observed for the ground user. This may be because the rapidly changing LEO geometry provides enough satellite diversity at each batch length to avoid the design matrix conditioning issues seen for BRUX. The best result is achieved at B180, though the improvement over the filter is marginal compared to the clearer advantage seen for BRUX. Overall, a minimum batch length of around 60 minutes also applies for S6, but longer windows do not guarantee proportionally better results given the sensitivity of the radial component.

## 5.3 Growing Window Analysis - BRUX and S6

In this experiment, a single batch is run over a window that matches the batch length. Unlike Section 5.2 where the full 3-hour window was always processed with repeated batches, here the computation window grows with the batch length: a single B10 batch processes only 10 minutes of data, a single B60 processes only 60 minutes, and so on. This allows us to evaluate how much data is actually needed to produce a reliable solution, independent of what happens at batch boundaries.

The position error plots for BRUX are shown in Figures 25 – 30. The full statistics are summarized in Table 17.

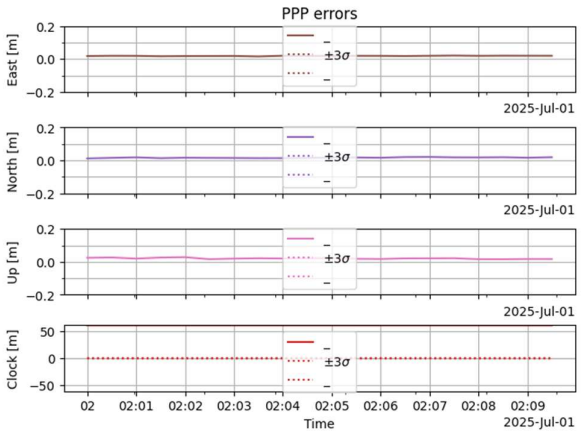



*Figure 25: BRUX FIN B10 plot*

**B10 : too short**

A single 10-minute batch yields a 3D RMS of 3.28 cm. Without boundary jumps the solution looks cleaner than the 5.2 B10 result, but the ±3σ bounds are very wide and the estimator cannot reliably solve all parameters in such a short arc.

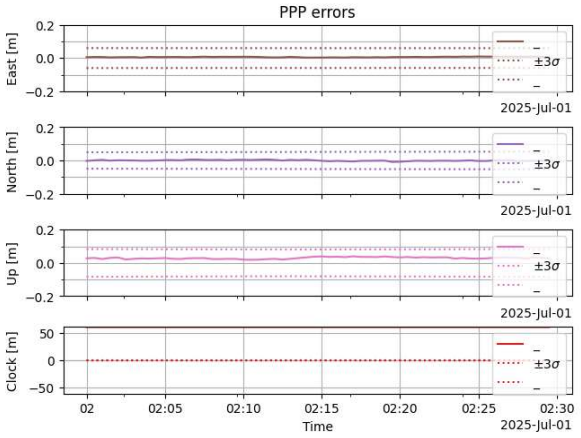



*Figure 26: BRUX FIN B30 plot*

**B30 : improving**

At 30 minutes the 3D RMS drops to 3.06 cm and the 2D RMS reaches a surprisingly good 0.72 cm. The error time series is noticeably cleaner than B10, though some minor variability remains in the Up component. And the ±3σ bounds are now visible.

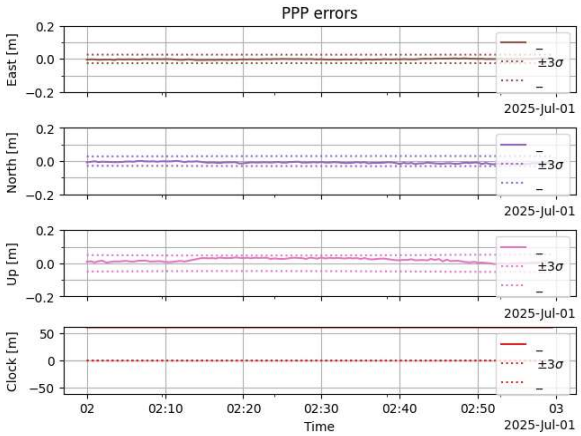



*Figure 27: BRUX FIN B60 plot*

**B60 : approaching the filter**

At 60 minutes the 3D RMS is 2.45 cm, close to the filter at 1.91 cm, and the 2D RMS of 1.07 cm matches the filter exactly. The solution looks clean and stable across East and North components, with tight ±3σ bounds. UP component is not there yet.

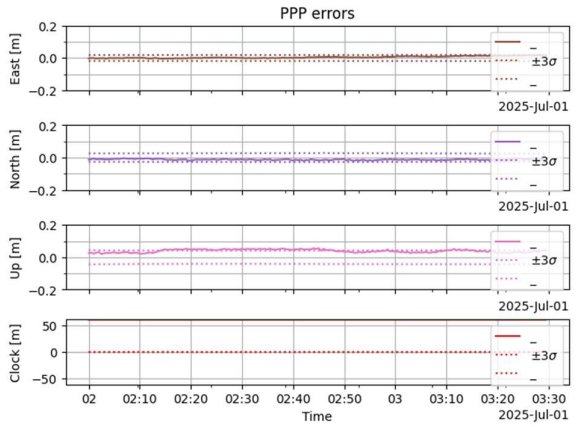



*Figure 28: BRUX FIN B90 plot*

**B90 : unexpected degradation**

B90 shows a 3D RMS of 4.29 cm, worse than B60 despite more data, and the worst result of all batch lengths in this experiment. The 2D RMS remains reasonable at 1.49 cm, suggesting again that the degradation is mainly in the vertical component, which also could be seen in Figure 29. As in Section 5.2, the cause is not immediately clear from the plots alone.

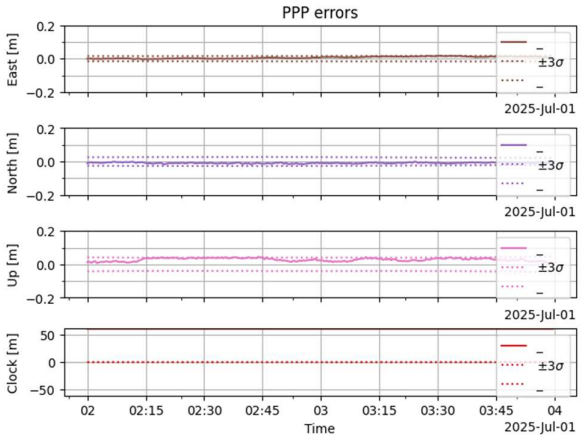



*Figure 29: BRUX FIN B120 plot*

**B120 : recovering**

B120 recovers to 3.25 cm 3D and 1.30 cm 2D, better than B90 but not yet matching B60. The solution is stable and clean across North and East components. Up component still fluctuates.

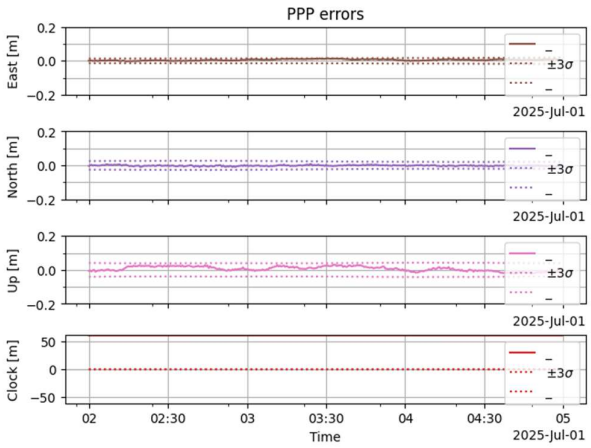



*Figure 30: BRUX FIN B180 plot*

**B180 : best result**

B180 achieves the best result at 1.70 cm 3D and 0.76 cm 2D, identical to the 5.2 BRUX B180 result since one batch covering the full 3-hour window is the same in both experiments.

*Table 17: RMS statistics for BRUX FIN*

| Configuration | 3D RMS [m] | 2D RMS [m] |
|---|---|---|
| Filter FIN | 0.019 | 0.011 |
| B10 | 0.033 | 0.025 |
| B30 | 0.031 | 0.007 |
| B60 | 0.024 | 0.011 |
| B90 | 0.043 | 0.015 |
| B120 | 0.032 | 0.013 |
| B180 | 0.017 | 0.008 |

**Summary**

For BRUX in the growing window experiment, longer windows generally perform better, with B180 achieving the best accuracy. Compared to Section 5.2, the absence of batch boundaries improves performance for short windows: B10 drops from 7.17 cm to 3.28 cm, confirming that boundary jumps are a significant source of error in the many-batch experiment. The non-monotonic dip at B90 appears in both experiments, suggesting it is not related to boundary effects but rather to the design matrix structure for that specific window length. A minimum window of around 60 minutes remains the practical threshold for reliable performance.

For S6, the same six batch lengths are tested, each covering only its own window length. The position error plots are shown in Figures 31 - 36. The full statistics are summarized in Table 18, with the filter FIN result from Chapter 4 included as reference.

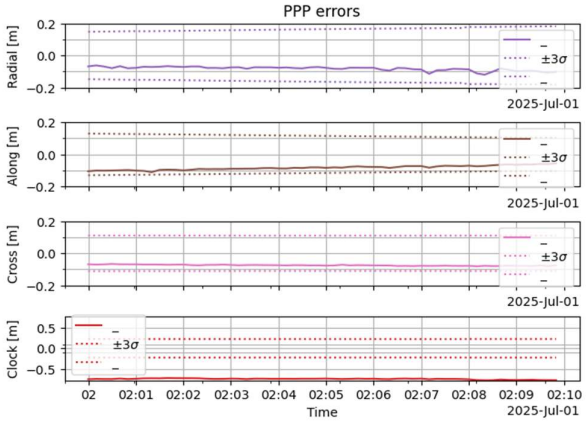



*Figure 31: S6 FIN B10 plot*

**B10 : too short for LEO**

The 10-minute batch produces a 3D RMS of 13.95 cm. Same reasoning as in the previous B10 results. The ±3σ bounds are very wide throughout, confirming low estimator confidence.

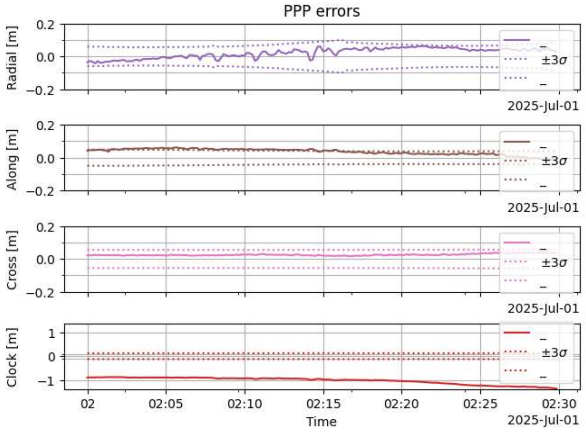



*Figure 32: S6 FIN B30 plot*

**B30 : improving but still unreliable**

At 30 minutes the 3D RMS improves to 8.06 cm. The solution is not yet stable enough to be considered reliable. The radial component behavior follows the same pattern observed in Section 5.2 for S6.

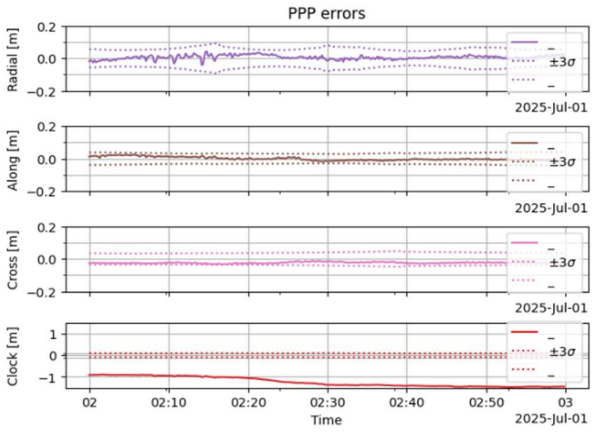



*Figure 33: S6 FIN B60 plot*

**B60 : the turning point**

At 60 minutes the 3D RMS drops to 2.95 cm, marginally better than the filter at 3.76 cm. The error time series is clean and the ±3σ bounds tighten. As with BRUX, 60 minutes appears to be the minimum useful batch length for S6. Radial component still oscillating.

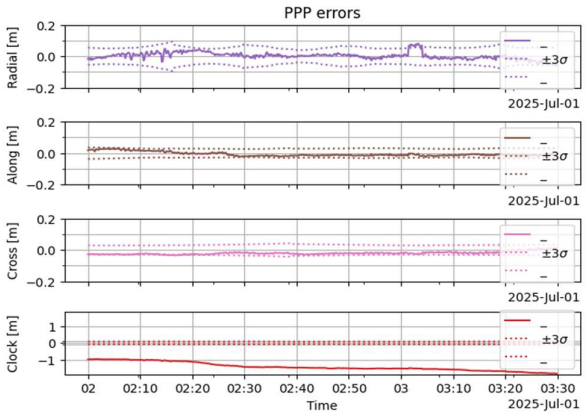



*Figure 34: S6 FIN B90 plot*

### B90 : an unexpected result

B90 is worse than B60 at 3.39 cm despite using more data. The 2D RMS remains comparable at 2.56 cm, confirming again that the degradation is mainly in the vertical component.

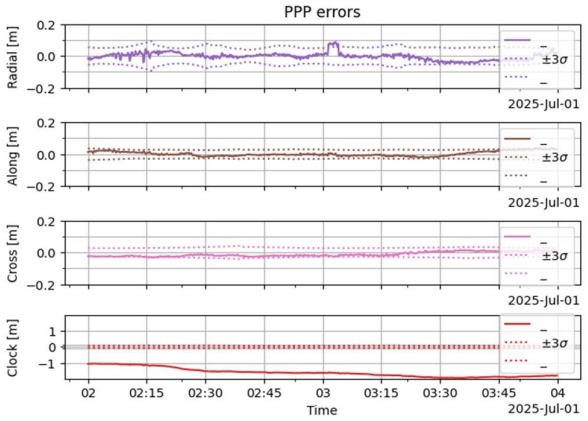



*Figure 35: S6 FIN B120 plot*

### B120 : recovering and matching the filter

B120 achieves 3.24 cm in 3D RMS, better than B90 and close to the filter reference. The 2D RMS of 2.34 cm essentially matches the filter level, though the vertical component still shows some disturbance.

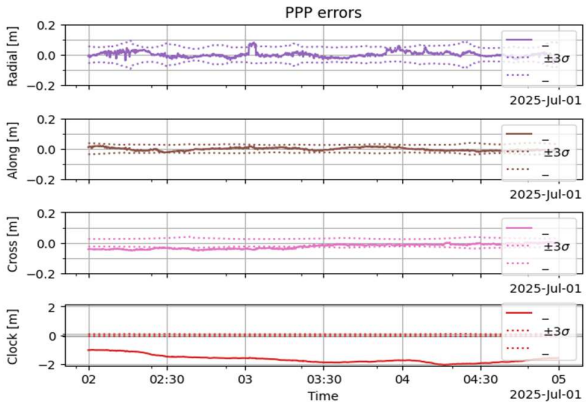



*Figure 36: S6 FIN B180 plot*

### B180 : continued non-monotonic behaviour

B180 shows a slight degradation to 3.57 cm 3D and 3.08 cm 2D compared to B120, despite using the most data. This confirms that for S6, a longer window does not reliably translate to better performance.

*Table 18: RMS statistics for S6 FIN*

| Configuration | 3D RMS [m] | 2D RMS [m] |
|---|---|---|
| Filter FIN (reference) | 0.038 | 0.023 |
| Batch B10 | 0.139 | 0.118 |
| Batch B30 | 0.081 | 0.048 |
| Batch B60 | 0.029 | 0.025 |
| Batch B90 | 0.034 | 0.026 |
| Batch B120 | 0.032 | 0.023 |
| Batch B180 | 0.036 | 0.031 |

**Summary**

For S6 in the growing window experiment, the results are broadly consistent with Section 5.2. Short windows below 60 minutes remain unreliable, and B60 again appears as the practical minimum threshold. Unlike BRUX however, performance for S6 does not improve monotonically beyond 60 minutes. B90 shows a slight degradation compared to B60, and B180 is not the best result, with B60 actually achieving the lowest 3D RMS of all batch lengths at 2.95 cm, marginally better than the filter at 3.76 cm. This non-monotonic behavior is more pronounced for S6 than for BRUX and is consistent with the rapidly changing LEO geometry introducing additional design matrix sensitivity as the window grows and a persistent disturbance in the Radial component appears throughout the S6 growing window results. Overall, for S6 the sweet spot appears to be around 60 minutes, beyond which longer windows do not reliably improve the solution.

## 5.4 Effect of Sampling Interval - S6 10s vs 30s

As established in Section 4.2, a 10-second sampling interval produces better filter results for S6 compared to 30 seconds, and 10s was adopted as the standard for all subsequent S6 analyses. This section examines whether the same tendency holds for the batch solution. Since the growing window experiment in Section 5.3 showed that batch length equal to computation window gives the cleanest results without boundary jump effects, the 30s comparison here follows the same setup, each batch covers only its own window length.

*Table 19: RMS statistics for S6 batch at 10s and 30s sampling, all batch lengths*

| Configuration | 3D RMS 10s [m] | 3D RMS 30s [m] | 2D RMS 10s [m] | 2D RMS 30s [m] |
|---|---|---|---|---|
| Filter FIN (ref) | 0.038 | 0.046 | 0.029 | 0.040 |
| Batch B10 | 0.139 | 0.284 | 0.118 | 0.282 |
| Batch B30 | 0.081 | 0.047 | 0.048 | 0.033 |
| Batch B60 | 0.029 | 0.047 | 0.025 | 0.045 |
| Batch B90 | 0.034 | 0.037 | 0.026 | 0.032 |
| Batch B120 | 0.032 | 0.036 | 0.030 | 0.030 |
| Batch B180 | 0.036 | 0.039 | 0.031 | 0.033 |

Comparing the two sampling intervals, 10s consistently outperforms 30s across all batch lengths, with the exception of B30 where 30s achieves a slightly better 3D RMS of 4.72 cm compared to 8.06 cm at 10s. This may be a result of the specific satellite geometry at that window length rather than a general tendency, and given the unreliability of both solutions at 30 minutes it is not considered significant. From B60 onwards, 10s is consistently better. At B60, the 3D RMS increases from 2.95 cm at 10s to 4.67 cm at 30s. At B180, the difference is smaller but still visible, with 3D RMS going from 3.57 cm to 3.88 cm. This is consistent with the filter result in Section 4.2: for a fast-moving LEO satellite like S6, denser sampling provides more observations per window, better capturing the rapidly changing geometry and improving the redundancy of the least-squares solution.

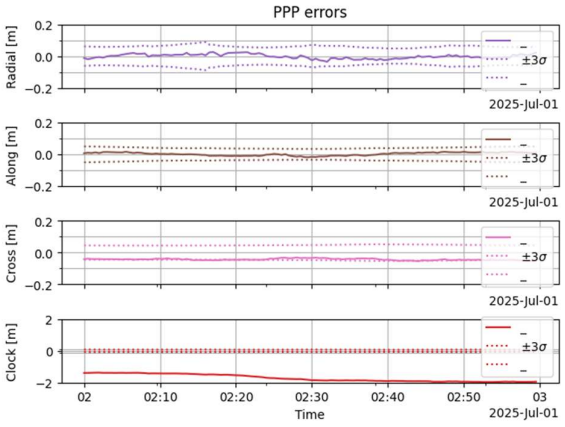



*Figure 37: S6 batch B60 position error plot, 30s sampling*

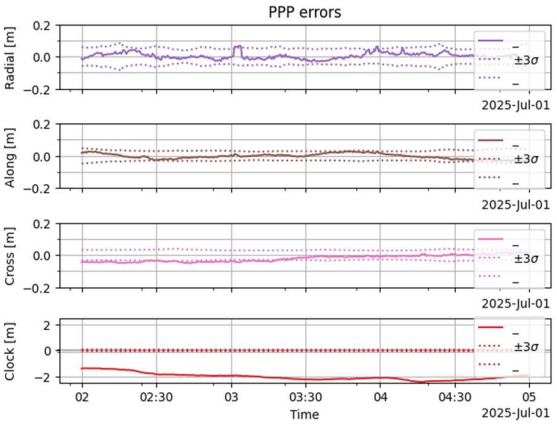



*Figure 38: S6 batch B180 position error plot, 30s sampling*

The 60-minute minimum threshold identified in Sections 5.2 and 5.3 also applies at 30s sampling, confirming that it is driven by the batch window length rather than the sampling interval.

## 5.5 Minimum Dataset Recommendation for FIN

Based on the results of Sections 5.2 and 5.3, a minimum batch window of 60 minutes is recommended for reliable PPP with FIN products, for both BRUX and S6. Below this threshold the solution is unstable and significantly worse than the filter, which serves as the reference baseline throughout this chapter. At 60 minutes, both users reach filter-comparable accuracy regardless of whether the repeated batch or growing window setup is used.

In operational terms, at least 60 minutes of continuous GNSS observations are needed before a reliable batch PPP solution can be produced. Below that, the filter remains the more appropriate choice.

# 6 Conclusion Filter vs Batch – BRUX / S6 – FIN

This chapter brings together the theoretical background from Section 2.4 and the experimental results from Chapters 4 and 5 to address the central question for FIN products: when does batch processing appear to be a better choice than the Kalman filter, and how does that depend on user type and batch window length?

**Accuracy and Precision**

The first thing that stands out is that batch performance depends heavily on the window length. The filter reference is always an already converged solution, so for short batch windows the comparison is not really fair - the batch simply does not have enough data yet. From B60 onwards however the batch becomes competitive for both users. For BRUX, B180 marginally outperforms the filter at 1.70 cm vs 1.91 cm, with the tightest ±3σ bounds of all configurations, suggesting the solution is both accurate and precise. For S6 the advantage is less consistent, B60 reaches filter-comparable accuracy in the growing window experiment but falls slightly short in the repeated batch experiment, and the radial component remains the hardest to constrain throughout.

**Minimum Batch Length**

The 60-minute minimum threshold appears robustly across both users and both experiments, which gives some confidence that it reflects something real about the batch design matrix needing sufficient geometry diversity to estimate all parameters reliably. The unexpected degradation at B90 for both users is noted but not fully explained, and is left for future investigation.

**When to Use Batch vs Filter**

Overall, batch processing appears to be a valid alternative to the filter for FIN products when the window is at least 60 minutes. For BRUX, longer is generally better. For S6, the results are less conclusive but 60 minutes appears to be a reliable minimum. Below that, the filter remains the more appropriate choice.

**Repeated Batch vs Growing Window**

The two batch experiments also serve different purposes. The repeated batch experiment is mainly diagnostic, showing how solutions evolve over an orbital period. The growing window experiment is the operationally relevant one, it reflects how batch processing would actually be used in near-real-time, processing whatever data is available at that moment. This motivates the use of batch equal to window length in the HAS experiments of Chapter 8.

# 7 Reference Solution - Filter with HAS Products

The previous chapters established the FIN filter and batch solutions as the reference baseline for BRUX and S6. As introduced in Section 2.6, Galileo HAS broadcasts real-time orbit, clock, and code-bias corrections through the E6-B signal, targeting approximately 20 cm horizontal and 40 cm vertical accuracy at the 95% level after convergence. Compared to FIN, HAS is expected to be noisier and less stable, not only because the corrections are available in real time rather than post-processed, but also because FIN products offer significantly higher accuracy, with orbit errors at the centimeter level and clock errors at approximately 0.1 ns. The same 3-hour evaluation window and processing setup from Chapter 4 are used here, so the results remain directly comparable.

## 7.1 BRUX Filter HAS

The BRUX filter solution using HAS products is evaluated over the same 3-hour window used in Chapter 4, allowing a direct comparison with the FIN reference. Figure 39 shows the position error time series and Table 20 summarises the key statistics.

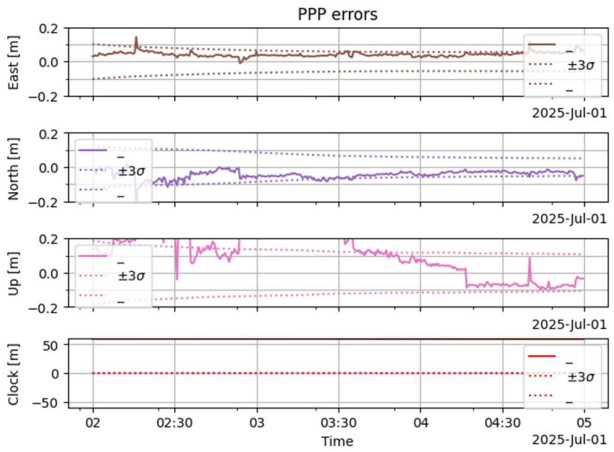



*Figure 39: BRUX filter HAS position error plot*

*Table 20: BRUX filter HAS performance statistics*

| | |
|---|---|
| 3D RMS [m] | 0.196 |
| 2D RMS [m] | 0.067 |
| Availability 3D ≤ 0.1m | 33.3% |
| Availability 3D ≤ 0.2m | 65.8% |
| Availability 2D ≤ 0.05m | 20.3% |
| Availability 2D ≤ 0.1m | 94.4% |

**Accuracy**

The solution achieves a 3D RMS of 19.62 cm and a 2D RMS of 6.67 cm. The horizontal component is relatively strong at 6.67 cm, while the vertical component appears to be the main source of degradation, driving the 3D RMS to nearly 20 cm. This gap between 2D and 3D RMS is likely related to the geometric weakness of the vertical direction in GNSS, since all satellites are observed from above, the Up component is inherently less well constrained than East and North. Under noisier real-time HAS corrections, this weakness is likely more visible than in the FIN case.

**Convergence and settling behaviour**

Some residual settling is visible in the North component in the early part of the window. This may be because the filter is still absorbing the HAS correction noise into a stable estimate at the start of the evaluation window, even though the main convergence transient has already passed. The Up component remains noisy throughout, with excursions frequently exceeding ±20 cm. This persistent vertical noise is likely related to the higher uncertainty of real-time HAS orbit and clock corrections compared to post-processed FIN products; any remaining correction error tends to project most strongly into the vertical direction.

**Stability and availability**

The ±3σ bounds are wider than in the FIN case throughout the window, which is consistent with the higher noise level of HAS corrections. The solution stays within 0.5 m 3D for 100% of the time, within 0.2 m for 65.8%, and within 0.1 m for 33.3%. The limited availability at tighter thresholds is likely driven primarily by the vertical component; when the Up error exceeds the threshold, it tends to pull the 3D error above it even when the horizontal solution is relatively clean.

**Summary**

The BRUX HAS filter solution operates at the decimeter level, with the vertical component as the main limitation. The horizontal accuracy appears consistent with the HAS service target after convergence, while the vertical remains more variable. This is likely a known characteristic of real-time PPP rather than something specific to HAS, it reflects the geometric limitation of GNSS in the vertical direction, which tends to become more apparent under noisier corrections.

## 7.2 S6 Filter HAS: 10s vs 30s

The S6 filter solution using HAS products is evaluated over the same 3-hour window, with results shown for both 10s and 30s sampling intervals. Figure 40 and Figure 41 show the position error time series, and Table 21 summarises the post-convergence statistics. The same processing setup is used as in the previous chapters, so the comparison with the FIN reference remains direct.

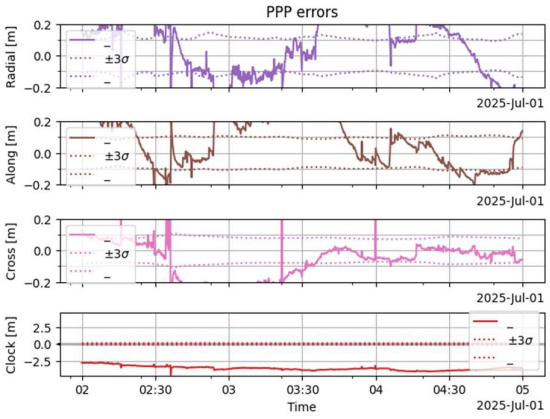



*Figure 40: S6 filter HAS 10s position error plot*

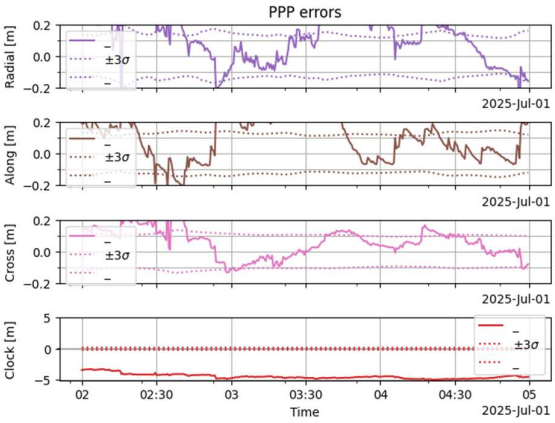



*Figure 41: S6 filter HAS 30s position error plot*

*Table 21: S6 filter HAS performance statistics at 10 s and 30 s sampling*

| Metric | S6 Filter HAS 10s | S6 Filter HAS 30s |
|---|---|---|
| 3D RMS [m] | 0.345 | 0.297 |
| 2D RMS [m] | 0.261 | 0.213 |
| Avail. 3D ≤ 0.2m | 20.7% | 33.3% |
| Avail. 3D ≤ 0.5m | 98.2% | 97.5% |
| Avail. 2D ≤ 0.1m | 25.9% | 28.1% |
| Avail. 2D ≤ 0.2m | 53.1% | 56.4% |

**Accuracy**

The 10s solution achieves a 3D RMS of 34.55 cm and a 2D RMS of 26.15 cm. The 30s solution performs slightly better at 29.75 cm 3D and 21.33 cm 2D. As discussed in

Section 4.4, S6 is a LEO space user with rapidly changing geometry, so a larger error level is expected compared to BRUX. The radial component appears to be the most sensitive for S6, which is consistent with what was discussed in Section 3.8.1, the radial direction tends to absorb orbit, clock, and phase center calibration errors. Under HAS corrections, which carry more noise than FIN, this sensitivity may be amplified, any residual error in the real-time orbit and clock products is likely to project most strongly into the radial direction. The along-track and cross-track components appear somewhat more stable, though still considerably noisier than in the FIN case.

**Convergence and settling behaviour**

The position error plots show that the solution is less stable than in the FIN case throughout the window. Unlike BRUX where the horizontal components settle relatively quickly, for S6 the rapidly changing geometry may prevent the filter from reaching the same level of stability, it must continuously adapt to new satellite configurations, which could explain the persistent variability observed in all three RSW components.

**Effect of sampling interval**

The 30s sampling outperforms 10s across all metrics in this experiment. This is the opposite of what was observed for FIN in Section 4.2, where denser 10s sampling was beneficial. A possible explanation is that under HAS, the correction noise per observation is high enough that adding more observations amplifies the noise rather than improving the geometry. In other words, the benefit of denser sampling may be outweighed by the cost of processing noisier corrections more frequently. This is an interesting observation, though a deeper investigation is beyond the scope of this thesis.

**Summary**

The S6 HAS filter solution operates at the decimeter level for both sampling intervals, with the radial component likely being the dominant error source. The 30s sampling performs slightly better than 10s in this experiment, which is the opposite tendency compared to the FIN case. This may suggest that under noisier real-time corrections, denser sampling amplifies rather than reduces the error, though this would need further investigation to confirm. These results serve as the baseline for the batch HAS comparison in Chapter 8. It is also worth noting that the filter takes considerably longer to converge under HAS than under FIN, particularly for S6. The implications of this for near real-time applications are discussed in Section 11.1.

## 7.3 FIN vs HAS Filter Comparison

This section brings together the filter results from Chapters 4 and 7 to directly compare FIN and HAS performance for both users. Table 22 summarizes the key statistics side by side. Convergence period of 1-hour was removed from filter results to ensure fair comparison.

*Table 22: Key statistics of FIN and HAS for BRUX and S6*

| | 3D RMS [m] | 2D RMS [m] |
|---|---|---|
| BRUX FIN | 0.019 | 0.011 |
| S6 FIN 10s | 0.038 | 0.029 |
| S6 FIN 30s | 0.046 | 0.040 |
| BRUX HAS | 0.196 | 0.067 |
| S6 HAS 10s | 0.345 | 0.261 |
| S6 HAS 30s | 0.297 | 0.213 |

The most immediate observation is that HAS produces considerably larger errors than FIN for both users and both sampling intervals. For BRUX the 3D RMS increases from 1.91 cm to 19.62 cm, and for S6 from 3.76 cm to 34.55 cm at 10s and from 4.63 cm to 29.75 cm at 30s.
This is expected, because FIN is the post-processed reference product, while HAS is the real-time operational product. The accuracy difference between the two products is therefore not surprising, but it is useful to quantify it for this specific setup.

An interesting pattern is that the degradation is not evenly distributed across components. For BRUX, the 2D RMS increases by a factor of around six while the 3D RMS increases by a factor of ten, suggesting that the vertical component absorbs a disproportionate share of the HAS correction noise. This is likely related to the geometric weakness of the vertical direction in GNSS, as discussed in Section 7.1. For S6, the degradation affects all three RSW components, with the radial being the most affected. This may be because the radial direction is particularly sensitive to orbit and clock errors, as discussed in Sections 3.8.1 and 7.2, and HAS corrections carry more residual error in these quantities than FIN.

For S6, it is also worth noting that 30s sampling consistently outperforms 10s under HAS, while the opposite tendency was observed for FIN in Section 4.2. This is consistent with the discussion in Section 7.2: under noisier HAS corrections, denser sampling may amplify rather than reduce the error, though this would need further investigation to confirm.

Another observation worth noting is that both users show a similar relative degradation, roughly a factor of nine to ten in 3D RMS. This consistency is somewhat unexpected given how different the two users are. A possible explanation is that the limiting factor in both cases is the HAS correction quality itself rather than the user geometry. If the noise floor imposed by HAS dominates the error budget for both users, it would explain why the

relative degradation is similar regardless of whether the receiver is static or moving. This is however an assumption based on the available results.

For BRUX, the horizontal result of 6.67 cm 2D is consistent with the published HAS service target of approximately 20 cm horizontal and 40 cm vertical at the 95% level after convergence [11][19]. The solution is therefore within the expected range for a static ground user. For S6, the results remain above that reference, which is not unexpected given the more challenging LEO environment. It is worth noting that the HAS target is defined for ground users, so applying it directly to a LEO space user is not straightforward.

The broader objective of using HAS is its real-time availability, unlike FIN products which require post-processing, HAS corrections are broadcast directly through the Galileo E6-B signal and can be used onboard without any external data link. This makes HAS the operationally relevant product for near-real-time LEO applications, which is the core motivation for evaluating it in this thesis.

Beyond accuracy, the convergence behavior of the filter under HAS corrections raises an additional concern. While under FIN the filter stabilizes within minutes, the much noisier HAS corrections slow down ambiguity resolution significantly, and the filter takes considerably longer to reach a stable solution. This has direct implications for near real-time applications: if the filter needs several hours to converge under HAS, its latency advantage over batch is no longer guaranteed. The full convergence analysis with exact times across all scenarios is provided in Section 11.1, once the batch HAS results are available for comparison.

Overall, replacing FIN with HAS results in a clear accuracy loss for both users, driven mainly by the higher noise of real-time corrections. The vertical component for BRUX and the radial component for S6 appear to be the most affected. These findings set the context for Chapter 8, where the same comparison is made for the batch solutions.

The key question being whether batch processing behaves differently from the filter under HAS corrections, and whether the batch window length sensitivity observed in Chapter 5 carries over to the HAS case.

# 8 Batch Processing with HAS Products

Chapter 7 showed how the filter performs with HAS corrections for both BRUX and S6. This chapter evaluates the batch solver under the same conditions, following the growing window setup from Section 5.3. The choice to use only the growing window here is deliberate and comes from the core motivation of this chapter: in a near-real-time scenario, a LEO satellite collects observations from a fixed start time and at some point needs to produce a position estimate from whatever data it has accumulated.

The question is therefore not how repeated short batches behave over a full orbital period, that was already explored in Section 5.2 as a diagnostic tool to assess batch robustness and consistency. The question here is more direct: how much data does the batch need before it can produce a reliable position, and can it compete with the filter from that point onwards? Under HAS corrections, which are noisier than FIN, this question becomes even more interesting: does the minimum window length change, and does the batch handle correction noise differently than the filter?

For S6, both 10s and 30s sampling intervals are tested, to check whether the sampling effect observed in Section 5.4 carries over to the HAS case. The same batch lengths as in Chapter 5 are used: 10, 30, 60, 90, 120 and 180 minutes, each covering only its own window starting from 02:00. BRUX is used as the ground user validation case and S6 as the primary space user. The FIN batch results from Chapter 5 and the filter HAS results from Chapter 7 both serve as references throughout.

## 8.1 BRUX Batch HAS

The BRUX batch HAS results are evaluated using the growing window setup, with batch lengths from 10 to 180 minutes all starting at 02:00. The filter HAS result from Section 7.1 serves as the real-time reference, the FIN batch results from Section 5.3 serve as the high-accuracy reference. The full statistics are summarized in Table 23, and the position error plots for all batch lengths are shown in Figures 42 to 47.

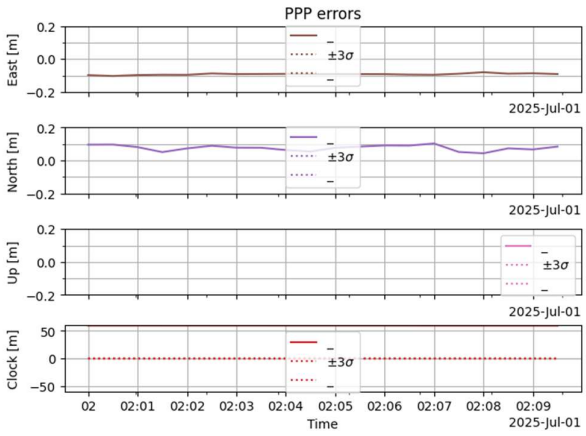



*Figure 42: BRUX HAS B10 Plot*

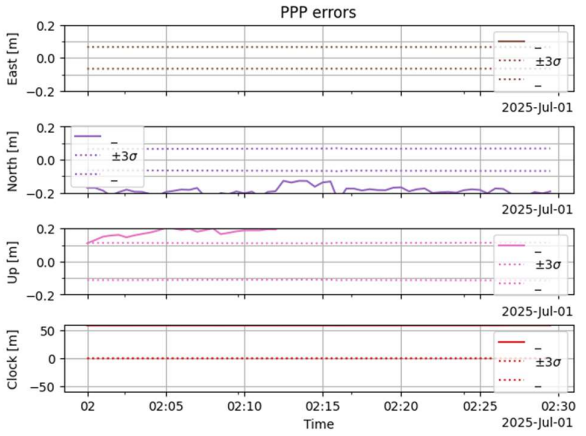



*Figure 43: BRUX HAS B30 Plot*

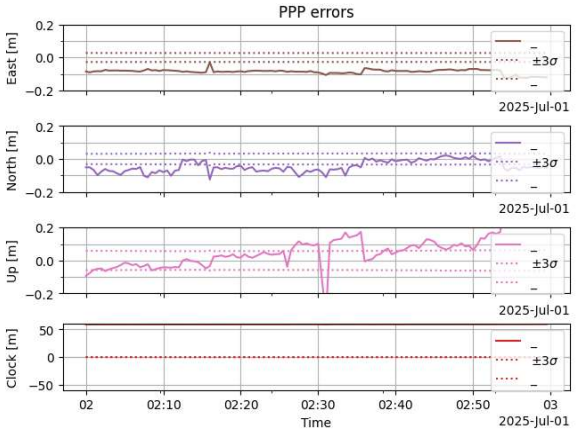



*Figure 44: BRUX HAS B60 Plot*

**B10 : neither accurate nor precise**

At 10 minutes the 3D RMS is 52.1 cm. No ±3σ bounds are visible and the Up component goes beyond the plot frame; the estimator has no confidence in the solution at this window length.

**B30 : becoming more precise, but not yet accurate**

Adding 20 more minutes brings the 3D RMS to 42.6 cm. The ±3σ bounds now appear in all three components, precision is improving. Interestingly, while the bounds appear for the first time, the East solution line itself is no longer visible within the frame, suggesting the error has shifted further from zero compared to B10 despite the estimator gaining some formal confidence. The Up component, completely out of frame at B10, is now partially visible but still drifting beyond the axis in the first few minutes. The solution is not yet accurate enough to be useful.

**B60 : approaching the filter**

At 60 minutes the 3D RMS drops to 17.4 cm, just below the filter reference of 19.6 cm. All three components are now visible within the frame and the bounds tighten noticeably. The Up component eventually goes out of frame, with a spike around 02:30. Despite this, the solution is becoming both more accurate and more precise compared to shorter windows.

This spike appears consistently across all batch lengths and is likely a feature of the HAS correction quality at that specific time on this day rather than a batch-specific artifact. It is noted here and not discussed further in subsequent plots.

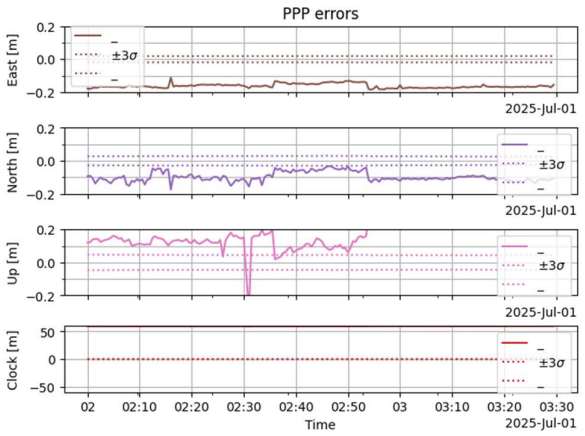



*Figure 45: BRUX HAS B90 Plot*

**B90 : unexpected degradation**

Despite more data, the 3D RMS jumps to 34.4 cm and the 2D RMS also degrades. The Up component shows similar behavior to B60. The degradation is therefore more visible in the statistics than in the plot, and affects both vertical and horizontal components, suggesting HAS noise is amplifying the design matrix sensitivity discussed in Section 5.1.

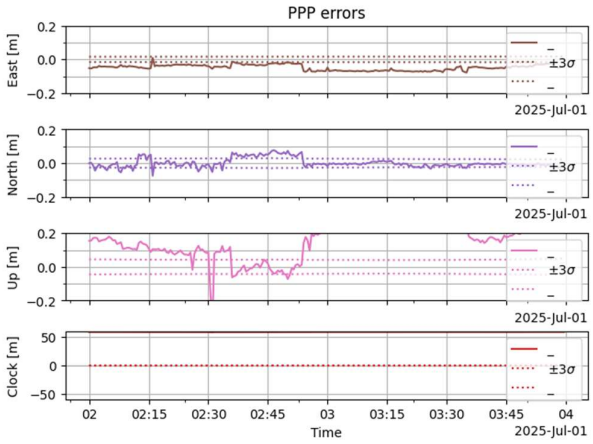



*Figure 46: BRUX HAS B120 Plot*

**B120 : recovering**

The 3D RMS returns to 19.6 cm, matching the filter, and the 2D RMS of 5.75 cm is better than the filter HAS 2D reference of 6.67 cm. Visually however the components still shows similar behavior to B90. The improvement is therefore more visible in the statistics than in the plot.

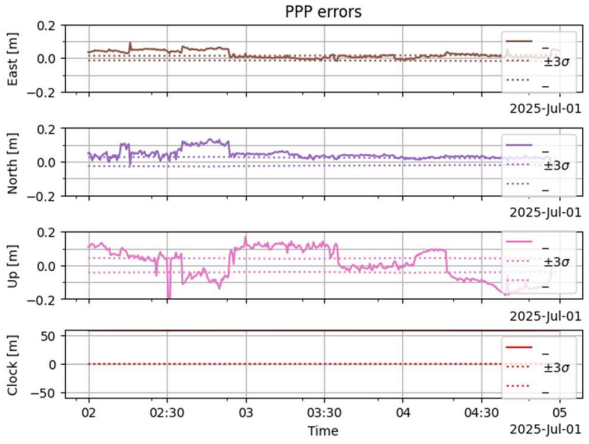



*Figure 47: BRUX HAS B180 Plot*

**B180 : best result**

The full 3-hour batch achieves 11.0 cm 3D and 5.9 cm 2D, well below the filter reference. For the first time, the Up component stays within the plot frame throughout the entire window. the spike at 02:30 is still visible but it no longer goes out of frame. This is the clearest visual sign that the solution has stabilized, and combined with the statistics it confirms that B180 is both the most accurate and most precise configuration tested.

*Table 23: RMS statistics for BRUX*

| Configuration | 3D RMS [m] | 2D RMS [m] |
|---|---|---|
| Filter HAS (reference) | 0.1962 | 0.0667 |
| Batch B10 | 0.5213 | 0.1208 |
| Batch B30 | 0.4259 | 0.3233 |
| Batch B60 | 0.1737 | 0.1036 |
| Batch B90 | 0.3439 | 0.1917 |
| Batch B120 | 0.1963 | 0.0575 |
| Batch B180 | 0.1105 | 0.0586 |

**Summary**

For BRUX under HAS corrections, 60 minutes appears as the minimum reliable batch window, consistent with the FIN results in Chapter 5. The progression from B10 to B180 tells a clear story, as more data accumulates, the batch design matrix becomes better conditioned, more parameters get reliably constrained, and both accuracy and precision improve. This is visible not only in the statistics but also in the plots: the Up component which goes completely out of frame at B10 is eventually contained within the frame for the full window at B180, with the persistent spike at 02:30 still present but no longer dominant.

The fact that B180 clearly outperforms the filter suggests that the global adjustment advantage of the batch carries over to the noisier HAS case, and may actually be more beneficial here since more observations help average out the higher correction noise. The B90 degradation affects both 3D and 2D RMS, unlike the FIN case where it was mainly vertical, suggesting HAS noise interacts with the design matrix structure in a way that amplifies sensitivity at that specific window length. This remains tentative and a deeper investigation is left for future work.

## 8.2 S6 Batch HAS – 10s vs 30s

The S6 batch HAS results are evaluated using the growing window setup for both 10s and 30s sampling intervals. For 10s sampling, batch lengths B60, B90 and B120 are available. B10 and B30 are excluded from the detailed analysis as they consistently produce unreliable results, as already established in Chapters 5 and 8.1. B180 at 10s could not be run due to the computational cost of inverting the large design matrix at that sampling density. For 30s sampling, batch lengths B60, B90, and B120 are available. Direct comparison between sampling intervals is made at B60, B90 and B120.

The filter HAS results from Section 7.2 and the FIN batch results from Section 5.3 serve as references throughout. The full statistics are summarized in Table 24.

The plots for all batch lengths and both sampling intervals are shown in Figures 48 to 53. Rather than describing each plot individually, a few patterns seem to appear consistently across all of them and are worth keeping in mind while going through the figures.

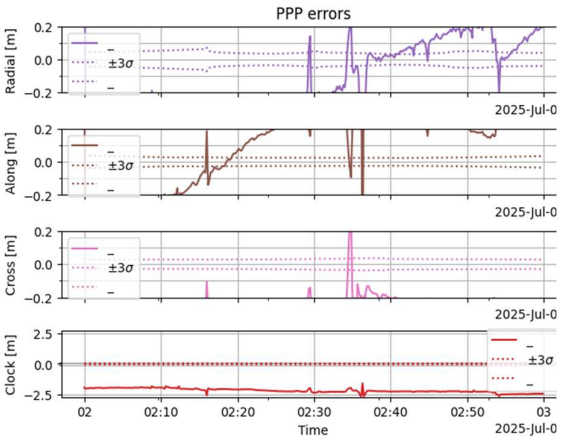



*Figure 48: S6 HAS 10S B60 plot*

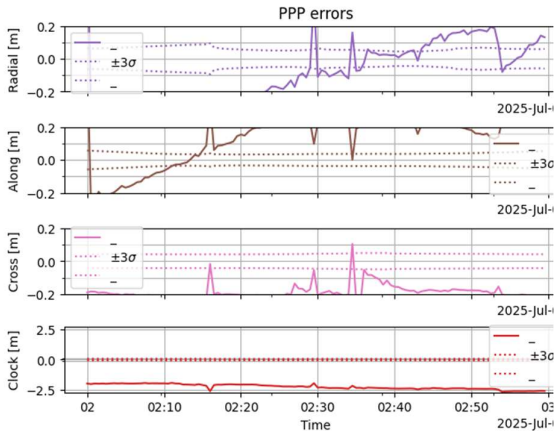



*Figure 49: S6 HAS 30S B60 plot*

The first thing to notice is how the solution changes as the batch window grows. At B60 (Figures 48 and 49), the radial is very active and the along-track rises and goes out of the plot frame for both sampling intervals. The cross-track component is barely visible at this stage, suggesting the batch does not yet have enough data to constrain it reliably.

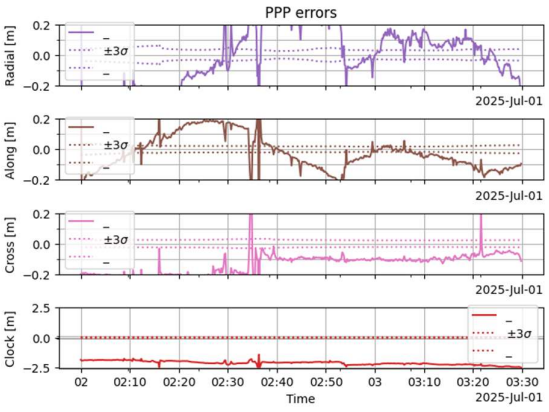



*Figure 50: S6 HAS 10S B90 plot*

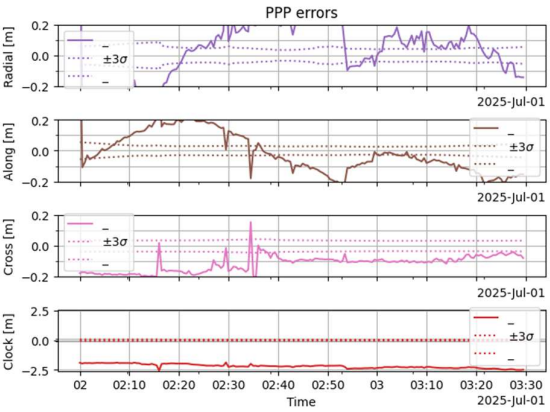



*Figure 51: S6 HAS 30S B90 plot*

Moving to B90 (Figures 50 and 51), things start to look a bit different. The along-track is still active but appears more contained, and the cross-track starts to enter the frame. This seems to indicate that with more data, the batch is beginning to constrain all three components rather than just the radial and along-track. By B120 (Figures 52 and 53), all three components are visible within the frame, though the radial remains the most variable throughout, which is consistent with what was discussed in Section 3.8.1.

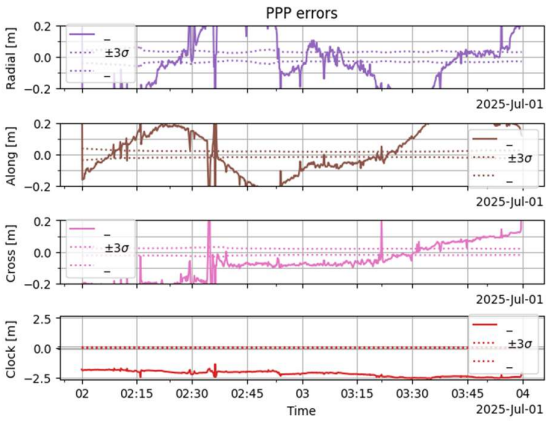



*Figure 52: S6 HAS 10S B120 plot*

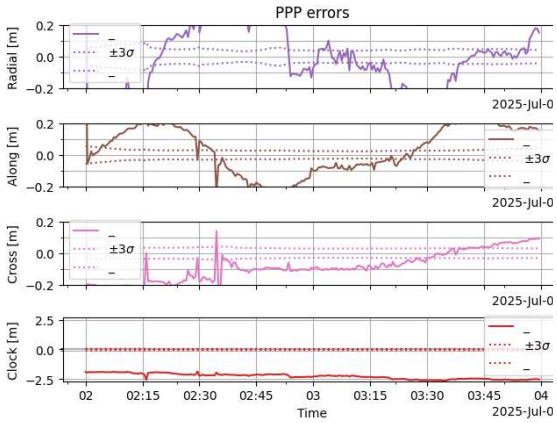



*Figure 53: S6 HAS 30S B120 plot*

The second pattern worth pointing out is the effect of sampling interval. Comparing 10s and 30s at the same batch length, the plots follow a very similar trajectory, same general shape, same timing of features. The main visible difference seems to be the magnitude of the spikes, which appear roughly halved in the 30s case. The ±3σ bounds however tell a slightly different story, they stay broadly similar within the same sampling interval across batch lengths, but are noticeably wider for 30s than for 10s. So while 30s appears to give better accuracy, the estimator seems less certain about its own solution. This tension between accuracy and precision under noisy HAS corrections was not observed under FIN, which is an interesting observation that may be worth investigating further.

*Table 24: RMS statistics for S6 batch HAS*

| Configuration | 3D RMS 10s [m] | 3D RMS 30s [m] | 2D RMS 10s [m] | 2D RMS 30s [m] |
|---|---|---|---|---|
| Filter HAS (reference) | 0.3455 | 0.297 | 0.261 | 0.213 |
| Batch B60 | 0.5284 | 0.399 | 0.401 | 0.280 |
| Batch B90 | 0.3227 | 0.294 | 0.238 | 0.191 |
| Batch B120 | 0.3234 | 0.299 | 0.240 | 0.207 |
| Batch B180 | n/a | 0.327 | n/a | 0.240 |
| Batch B114[2] | n/a | 0.256 | n/a | 0.157 |

**Summary**

The S6 HAS batch results do not follow as clear a pattern as what was observed with FIN products. Looking at the statistics, B60 at 10s actually performs worse than expected at

[2] supplementary run, one full orbital period

52.8 cm 3D, while B90 and B120 give the best results at around 32.3 cm and 29.4 cm respectively for both sampling intervals. Longer windows do not consistently lead to better accuracy here, which is different from what was observed under FIN corrections.

One observation that does hold consistently is that the batch retains its core advantage, with more observations covering past and future epochs jointly, the solution becomes smoother as the window grows. This effect is further enhanced by the 30s sampling interval, which plays a noticeable role in reducing spike magnitudes across all batch lengths, even when the overall accuracy improvement in the statistics is modest.

## 8.3 Minimum Dataset Recommendation for HAS

Based on the results of Sections 8.1 and 8.2, the minimum batch window needed for reliable results with HAS products seems to differ between the two users. For BRUX, 60 minutes appears to be sufficient to reach filter-comparable accuracy, consistent with what was found for FIN in Chapter 5. For S6, the picture is less clear, but the results suggest that around 90 minutes may be needed before the solution becomes competitive with the filter. Below these thresholds, the solution appears unstable and significantly worse than the filter reference for both users.

Unlike the FIN case, longer windows do not seem to reliably improve the solution further under HAS corrections. Beyond the minimum threshold, performance appears to plateau, possibly because the correction noise becomes the dominant limitation rather than the amount of data. This is something worth keeping in mind for near-real-time applications, though a more systematic investigation would be needed to confirm it.

This accuracy-based recommendation is further supported by the convergence analysis in Section 11.1, which shows that under HAS corrections the filter takes nearly 3 hours for BRUX and over 3.5 hours for S6 to stabilize, well beyond the minimum batch window required for a reliable result. This means batch not only matches the filter in accuracy, but delivers its solution faster.

# 9 Conclusion Filter vs Batch – BRUX / S6 – HAS

This chapter brings together the results from Chapters 7 and 8 to answer the central question for HAS products: when does batch processing appear to be a better choice than the Kalman filter, and how does that depend on the user type and the batch window length?

**Was batch expected to work under HAS?**

Short-arc batch processing with HAS has already been shown to achieve decimeter-level accuracy for S6 in the literature [7], so the expectation was that batch could work, the question was under what conditions. The key argument is that by processing all observations jointly, the batch can average out correction noise across the full arc in a way the sequential filter cannot. Whether that advantage is enough depends on how long the window is, and that is what the results here try to answer.

**For BRUX, batch works, and works well**

The standard 30s sampling interval applies for BRUX throughout this thesis. The batch becomes competitive from B60 onwards and at B180 clearly outperforms the filter, reaching 11.0 cm 3D compared to the filter reference of 19.6 cm, with the tightest ±3σ bounds of all configurations, meaning the solution is both more accurate and more precise as defined in Section 3.8.4. For a static ground user, longer batch windows appear particularly beneficial under HAS.

**For S6, it is more complicated**

For S6 the batch does not become competitive until around B90, where the solution first drops below the filter reference for both sampling intervals, reaching 32.3 cm 3D at 10s and 29.4 cm at 30s. B120 gives very similar results, suggesting the sweet spot is somewhere between 90 and 120 minutes, close to one full orbital period. Beyond that the solution plateaus, and a supplementary test at 114 minutes confirms this, as discussed below.

**A preliminary test at one full orbital period**

Given that the sweet spot for S6 appears to fall between B90 and B120, close to one full orbital period, a supplementary run was performed at exactly 114 minutes. This gave a 3D RMS of 25.6 cm and a 2D RMS of 15.7 cm at 30s sampling, the best result of all batch configurations tested, and better than the filter reference of 29.75 cm 3D. This suggests that one full orbital period may be a natural and physically meaningful threshold for S6 HAS batch processing, as it allows the satellite to complete a full revolution and the batch to observe a complete set of satellite geometries. Whether this generalizes to other LEO missions at different altitudes is an open question, discussed further in Section 11.2.

**The sampling interval**

For S6, 30s sampling consistently outperforms 10s in accuracy across all batch lengths, while the ±3σ bounds are wider for 30s, meaning the estimator is formally less certain

about the 30s solution as discussed in Section 3.8.4. This tension between accuracy and precision under noisy corrections was not observed under FIN, and may suggest that denser sampling amplifies HAS correction noise for a fast-moving LEO satellite rather than improving the geometry. This is consistent with what was observed for the filter in Section 7.2, and would be worth investigating further.

**What does this mean for near-real-time LEO applications?**

The practical question behind this chapter was whether a small batch could give a reliable position in near-real-time for a LEO satellite like S6. The results suggest that around 90 minutes is the minimum, with one full orbital period giving the best result. Given that the filter takes over 3.5 hours to converge under HAS for S6, even low-latency requirements are better served by batch once one orbital period of data is available.

**Overall**

Batch processing appears viable and in some cases clearly better than the filter for HAS products, but depends strongly on user type and window length. For BRUX longer windows are beneficial and the batch is a strong choice. For S6 roughly one orbital period is needed, and at 114 minutes the batch clearly outperforms the filter. These findings are brought together with the FIN results in Section 11.1.

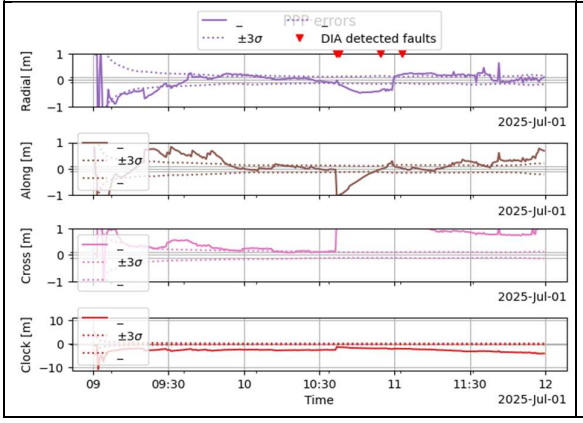
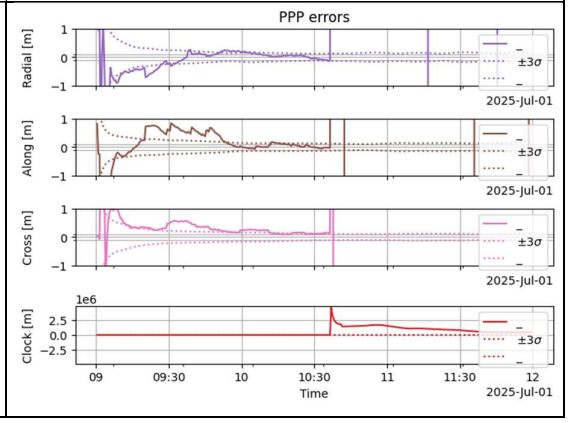
# 10 Resilience to Disturbances

This chapter investigates how the filter and batch solutions respond to real observation disturbances in the S6 HAS data on DOY 182 (1 July 2025) and DOY 183 (2 July 2025). Two questions are addressed in sequence: what happens when DIA is disabled, and how do the two estimators compare when DIA is active.

## 10.1 DIA on/off

As introduced in Section 2.5, DIA detects, identifies, and adjusts for observation faults during processing. The filter runs start one hour earlier than the batch windows to account for the convergence period established in Section 3.8.5. The convergence behavior looks broadly similar whether DIA is enabled or disabled, which suggests that no fault large enough to trigger detection occurs during the initialization phase.
The difference becomes visible after convergence. Table 25 shows the DOY 182 filter results with and without DIA.

*Table 25: S6 HAS filter position error plots for DOY 182*

| S6 HAS filter DOY 182 with DIA on | S6 HAS filter DOY 182 with DIA off |
|---|---|
|  |  |



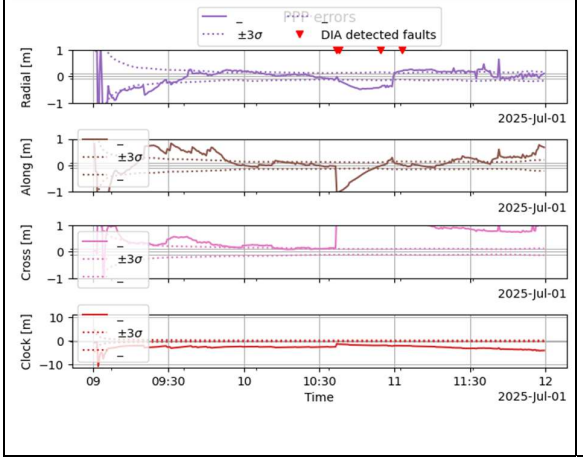
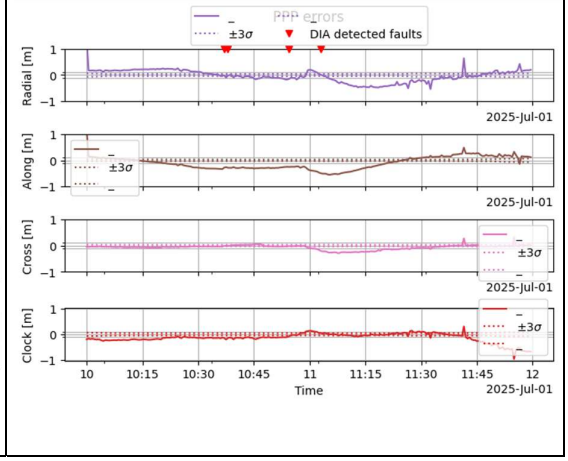
With DIA disabled, all components grow rapidly after convergence and the solution becomes unusable. With DIA enabled, the same disturbances are detected and handled - for example by excluding faulty observations or reinitializing affected ambiguities as described in Section 2.5 - and the solution recovers within a few epochs each time. The effectiveness of DIA is clear: the same data that causes a complete breakdown without it produces a stable and usable solution with it.

For the batch, running without DIA does not produce a solution for either day, it crashes. Without the quality-control step, the unmodeled faults appear too large for the batch to handle in these cases.

## 10.2 Batch vs Filter under Disturbance

With DIA enabled, both estimators produce usable solutions. Table 26 shows the DOY 182 results side by side.

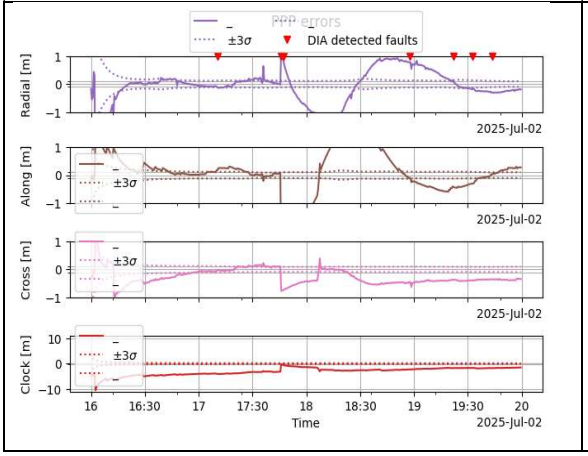
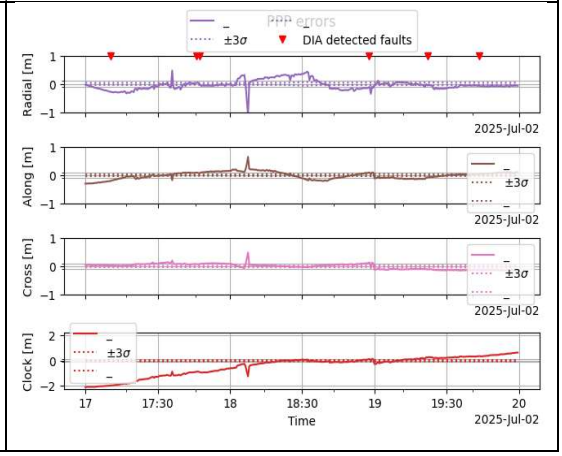
*Table 26: S6 HAS position error plots for DOY 182 with DIA enabled*

| S6 HAS filter DOY 182 (09-12) | S6 HAS Batch DOY 182 (10-12) |
|---|---|
|  |  |

The filter converges in the first hour and then handles the disturbances reasonably well. The batch responds to the same disturbances with smaller and shorter spikes while the along-track and cross-track components remain more or less stable throughout. Beyond simply recovering from faults, the batch appears to smooth out the residual effect of disturbances across the arc, which could be a meaningful advantage for space users like S6 where observation quality is harder to control than for a static ground receiver.

The DOY 183 results confirm the same pattern. Table 27 shows the filter and batch with DIA for that day.

*Table 27:S6 HAS position error plots for DOY 183 with DIA enabled*

| S6 HAS filter DOY 183 (09-12) | S6 HAS Batch DOY 183 (10-12) |
|---|---|
|  |  |



On DOY 183 the disturbances are more severe and the filter takes longer to recover, while the batch again absorbs them with brief contained spikes and a solution that remains in control. This is consistent with the structural difference discussed in Section 2.4: the batch processes the full arc jointly, so observations from before and after a disturbance both constrain the affected epochs, which may help limit how far the error spreads.

For these disturbed S6 HAS cases, both estimators require DIA to produce usable results. With DIA enabled, the batch appears less affected by the disturbances than the filter. The combination of DIA and batch processing appears to be the most resilient configuration tested here.

# 11 Conclusion

## 11.1 Summary of Findings

This thesis investigated whether batch least-squares PPP can deliver reliable positioning for LEO satellites using Galileo HAS corrections faster than the Kalman filter, and how much data it needs to do so. The idea is simple: if a short batch is enough to get a good position, batch becomes a practical choice for near real-time applications. FIN corrections served as the baseline reference, and all processing was done within GHASP3. For HAS corrections, the batch was run as a growing window: the batch window equals the time elapsed since the start of the arc, meaning the latency of the solution is directly determined by the minimum window required for reliable positioning. The goal is therefore to find the shortest window that still delivers a good position, the shorter the window the faster the result. For FIN corrections, both fixed and growing windows were used for analysis, validation, and visualization purposes.

The batch needs a minimum amount of data to work reliably. For both users under FIN, and for BRUX under HAS, 60 minutes is enough, and at 180 minutes batch clearly outperforms the filter for BRUX under both products. For S6 under HAS the threshold is higher, at 90 minutes, after which batch produces reliable results. At 114 minutes, corresponding to one full orbital period, batch delivers its best result and outperforms the filter. The difference between ground and space users comes down to geometry: the faster motion of S6 means more data is needed before the batch can constrain all parameters.

*Table 28: Summary of minimum batch window and best configuration by user and product*

| User | Product | Min batch window | Best config |
|---|---|---|---|
| BRUX | FIN | 60 min | B180 |
| S6 (30s) | FIN | 60 min | B180 |
| BRUX | HAS | 60 min | B180 |
| S6 (30s) | HAS | 90 min | B114 |

One unexpected result is that 30s sampling outperforms 10s for S6 under HAS, which is the opposite of what was seen under FIN. This may mean that denser sampling amplifies correction noise for a fast-moving LEO receiver rather than improving geometry, but this needs further investigation to confirm.

Once the minimum window is reached, batch gets to a reliable solution faster than the filter. Under HAS, the filter takes nearly 3 hours to converge for BRUX and over 3.5 hours for S6 at 30s. At 10s, S6 never converges within the observation window. A 114-minute batch delivers a good position before the filter has even stabilized. For the HAS scenarios tested here, batch reaches a useful solution before the filter has stabilized.

*Table 29: Filter convergence times (error remaining within ±10 cm)*

| User | Product | Sampling | Convergence time |
|---|---|---|---|
| BRUX | FIN | 30s | 11 min 30s |
| S6 | FIN | 10s | 25 min 50s |
| S6 | FIN | 30s | 15 min 30s |
| BRUX | HAS | 30s | 2h 58min |
| S6 | HAS | 10s | never |
| S6 | HAS | 30s | 3h 40min 30s |

The resilience tests showed that DIA is not optional. Without it, neither estimator gave a usable result in the disturbed cases tested. With DIA enabled, batch was the most resilient configuration - global estimation spreads the effect of faults across the full arc rather than carrying them forward.

For FIN products, the filter is competitive and simpler when low latency matters. For HAS-based LEO positioning, the results are clear: once the batch window reaches one orbital period, batch is both faster and more reliable than the filter. In the S6 HAS cases tested here, the filter does not stabilize quickly enough to remain competitive with batch for near-real-time use. At 30s sampling, batch delivers a good position well before the filter converges. Batch with DIA enabled is the recommended configuration.

## 11.2 Recommendations

For FIN products, the filter remains the appropriate choice when low latency is the priority and less than 60 minutes of data are available. For HAS products, the filter's slow convergence means batch is recommended as soon as sufficient data is available.

For S6 under HAS, 30s sampling and a batch window of approximately one full orbital period (~114 minutes) with DIA enabled is the recommended configuration. A preliminary test at 114 minutes confirmed this, giving the best result of all configurations tested at 25.6 cm 3D and 15.7 cm 2D, outperforming the filter. Whether one orbital period is the natural threshold for all LEO missions or specific to S6 remains to be confirmed.

The growing window setup, where the batch window equals the computation window, is recommended as the operationally relevant configuration for near-real-time scenarios.

*Table 30: Recommended GHASP3 configuration for near-real-time batch PPP of S6 using HAS products, based on the results of this thesis*

| Parameter | Recommended setting |
|---|---|
| Estimation strategy | Batch least-squares |
| Correction product | Galileo HAS |
| Sampling interval | 30s |
| Batch window length | ~114 min (one full orbital period) |
| Computation window | Equal to batch window |
| DIA | Enabled |

## 11.3 Future Work

Several questions raised by this thesis could not be fully answered within its scope and are worth investigating further.

The B90 anomaly observed in several configurations remains unexplained. It could be related to the satellite geometry on the specific days tested, or it could reflect something more fundamental about how the batch design matrix is conditioned at that window length. A systematic test campaign across multiple days and users would help clarify whether this is a recurring pattern or a day-specific artifact.

The results suggest that one full orbital period (113-114 minutes) may be a natural threshold for S6 HAS batch processing. Whether this scales with the orbital period or is mainly driven by observation geometry and design matrix conditioning is still an open question. Testing with LEO satellites at different altitudes would help determine whether the threshold is mission-specific or more general.
A possible explanation is the relatively low effective observations-to-unknowns ratio in the batch design matrix for LEO users, as discussed in Chapter 5. Future work should test whether the minimum window is mainly driven by this design matrix structure, by the orbital period, or by both.

The reversal between 10s and 30s performance from FIN to HAS is an interesting finding that deserves further investigation. It is unclear whether the optimal sampling interval depends on the noise level of the correction product, the satellite dynamics, or both. A dedicated study across multiple days and orbital conditions would help determine whether this is a general characteristic or specific to this dataset.

The batch shift parameter in GHASP3 allows overlapping consecutive batches, which could theoretically improve solution continuity at batch boundaries. Initial testing was carried out but did not produce improvements with the current strategy. Achieving smooth transitions would require averaging solutions at the overlap, which would need modifications to the GHASP3 software. Since GHASP3 is ESA property, such changes are outside the scope of this thesis and left for future work.

The LOM post-processing quality control available in GHASP3 was not activated in this thesis as its implementation for the batch solver has not yet been validated. Its impact on batch solutions under disturbed conditions has yet to be evaluated and could reveal further improvements in robustness.

In the early HAS test data reported in the literature, no fractional phase biases were available, so processing was limited to float ambiguities. In the fully operational HAS, phase biases are expected to support integer ambiguity resolution. Evaluating how ambiguity fixing would affect batch PPP performance for LEO users is therefore a relevant direction for future work, especially for short batch windows where improved ambiguity handling may have the greatest benefit [7].

This thesis contributes to the evaluation of batch PPP for near-real-time LEO positioning using Galileo HAS, and the results provide a concrete foundation for future operational implementations.

# 12 Bibliography

[1] O. Montenbruck and P. J. G. Teunissen, eds., *Springer Handbook of Global Navigation Satellite Systems*. Cham: Springer, 2017. doi: 10.1007/978-3-319-42928-1.

[2] X. Li, X. Zhang, X. Ren, M. Fritsche, J. Wickert, and H. Schuh, "Precise positioning with current multi-constellation Global Navigation Satellite Systems: GPS, GLONASS, Galileo and BeiDou," *Scientific Reports*, vol. 5, Art. no. 8328, 2015. doi: 10.1038/srep08328.

[3] M. Ge, G. Gendt, M. Rothacher, C. Shi, and J. Liu, "Resolution of GPS carrier-phase ambiguities in Precise Point Positioning (PPP) with daily observations," *Journal of Geodesy*, vol. 82, no. 7, pp. 389-399, 2008. doi: 10.1007/s00190-007-0208-3.

[4] B. Zhang, P. Hou, J. Zha, and T. Liu, "PPP-RTK functional models formulated with undifferenced and uncombined GNSS observations," *Satellite Navigation*, vol. 3, Art. no. 7, 2022. doi: 10.1186/s43020-022-00064-4.

[5] A. Jäggi, U. Hugentobler, and G. Beutler, "Pseudo-Stochastic Orbit Modeling Techniques for Low-Earth Orbiters," *Journal of Geodesy*, vol. 80, pp. 47-60, 2006. doi: 10.1007/s00190-006-0029-9.

[6] O. Montenbruck, S. Hackel, M. Wermuth, and F. Zangerl, "Sentinel-6A precise orbit determination using a combined GPS/Galileo receiver," *Journal of Geodesy*, vol. 95, Art. no. 129, 2021. doi: 10.1007/s00190-021-01563-z.

[7] A. Hauschild, O. Montenbruck, P. Steigenberger, I. Martini, and I. Fernández-Hernández, "Orbit determination of Sentinel-6A using the Galileo high accuracy service test signal," *GPS Solutions*, vol. 26, Art. no. 120, 2022. doi: 10.1007/s10291-022-01312-5.

[8] K. Wang, J. Liu, H. Su, A. El-Mowafy, and X. Yang, "Real-Time LEO Satellite Orbits Based on Batch Least-Squares Orbit Determination with Short-Term Orbit Prediction," *Remote Sensing*, vol. 15, no. 1, Art. no. 133, 2023. doi: 10.3390/rs15010133.

[9] P. J. G. Teunissen, "Distributional theory for the DIA method," *Journal of Geodesy*, vol. 91, pp. 713-727, 2017. doi: 10.1007/s00190-017-1045-7.

[10] P. J. G. Teunissen and P. F. de Bakker, "Single-receiver single-channel multi-frequency GNSS integrity: outliers, slips, and ionospheric disturbances," *Journal of Geodesy*, vol. 87, pp. 161-177, 2013. doi: 10.1007/s00190-012-0588-x.

[11] N. Naciri, D. Yi, S. Bisnath, F. J. de Blas, and R. Capua, "Assessment of Galileo High Accuracy Service (HAS) test signals and preliminary positioning performance," *GPS Solutions*, vol. 27, Art. no. 73, 2023. doi: 10.1007/s10291-023-01410-y.

[12] L. Massarweh et al., “Kinematic Precise Orbit Determination for real- and near-real-time using Galileo High Accuracy Service, GHASP3 project,” in *Proceedings of the 9th International Colloquium on Scientific and Fundamental Aspects of GNSS*, Wroclaw, Poland, 2024. Available: https://elib.dlr.de/206666/.

[13] C. J. Donlon et al., “The Copernicus Sentinel-6 mission: Enhanced continuity of satellite sea level measurements from space,” *Remote Sensing of Environment*, vol. 258, Art. no. 112395, 2021. doi: 10.1016/j.rse.2021.112395.

[14] N. Zehentner and T. Mayer-Gürr, “Precise orbit determination based on raw GPS measurements,” *Journal of Geodesy*, vol. 89, no. 12, pp. 1165-1176, 2015. doi: 10.1007/s00190-015-0872-7.

[15] G. Colosimo, M. Crespi, and A. Mazzoni, “Real-time GPS seismology with a stand-alone receiver: A preliminary feasibility demonstration,” *Journal of Geophysical Research*, vol. 116, B11302, 2011. doi: 10.1029/2010JB007941.

[16] L. Carlin, A. Hauschild, and O. Montenbruck, “Precise point positioning with GPS and Galileo broadcast ephemerides,” *GPS Solutions*, vol. 25, Art. no. 41, 2021. doi: 10.1007/s10291-021-01111-4.

[17] T. Hadaś, F. N. Teferle, K. Kaźmierski, P. Hordyniec, and J. Bosy, “Optimum stochastic modeling for GNSS tropospheric delay estimation in real-time,” *GPS Solutions*, vol. 21, pp. 1069-1081, 2017. doi: 10.1007/s10291-016-0595-0.

[18] L. Yuan, B. F. Chao, X. Ding, and P. Zhong, “The tidal displacement field at Earth's surface determined using global GPS observations,” *Journal of Geophysical Research: Solid Earth*, 2013. doi: 10.1002/jgrb.50159.

[19] L. Cucchi, S. Damy, C. Gioia, B. Motella, and M. Paonni, “Galileo High Accuracy Service: Tests in Different Operational Conditions,” *NAVIGATION: Journal of the Institute of Navigation*, 2024. doi: 10.33012/navi.665.