

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Master thesis for the International Master's Degree in Geomatics

Implementation of PPP as new GNSS Observation Type in the Geomonitoring System GOCA

Author: Raquel Luján

Supervisor: Prof. Dr.-Ing. Reiner Jäger

Co-Supervisor: Dipl.-Ing. Lyudmila Gorokhova

Co-Supervisor: Prof. Dr.-Ing. Ángel Esteban Martín Furones

November 2019

Abstract

Early detection of significant movements in both natural and artificial structures is crucial to prevent human, environmental and economic losses. For this reason, Geomonitoring in an active field. GNSS technics are also a field in which lot of research and improvement have been made in recent years. Some studies have indicated the potential of GNSS technics in the field of Geomonitoring. The aim of this master thesis is developing a software that allows processing GNSS data with Precise Point Positioning technic in the context of the geomonitoring project GOCA. With this implementation, potential of PPP with low cost receiver (U-Blox ZED-F9P) using different products and settings is evaluated in this document.

Based on a literature review, that includes the study of GOCA project and a summary of main PPP approaches, a C++ dialog-based software was design and developed, using RTKLIB and WaPPP as software engines. Besides that, two different observations were made (one 12 hours to post-processing and one real time) in order to test the developed software and evaluate the obtained results using different parameters or products.

The obtained results reaffirm the potential of the PPP technique, even using low cost receiver. Even some differences between different software engines or IGS products were found, the results allow us to conclude that PPP is a technique with many advantages in the field of geomonitoring, since it avoids the use of several receivers and good accuracies are obtained. However, some aspects need further research in this context, as there is no common criterion for establishing convergence time and new methodologies and algorithms are being developed in the field of PPP processing.

Keywords: Precise Point Positioning (PPP), Geomonitoring, Global Navigation Satellite System (GNSS), GOCA.

Contents

List of figures	vi
List of Abbreviations	vii
Introduction	1
1 Motivation	2
2 Objectives	4
Theoretical background	7
3 GOCA - GNSS/LPS/LS-based online Control and Alarm System .	8
3.1 Mathematical models used in GOCA	9
3.1.1 Gauss-Markov model	9
3.1.2 Kalman filtering	12
3.2 GOCA components	13
3.3 GOCA computation steps	14
3.3.1 Step 3. Deformation Analysis	17
3.4 Further developments: Integrated Deformation Analysis	18
3.4.1 Stuttgart TV Tower: Reference Object for SHM	19
3.4.2 SHM-Implementation on Stuttgart TV Tower	21
4 GNSS positioning	23
4.1 Terminology	23
4.2 Absolute Positioning: Benefits and Limitations	24
4.3 Fundamentals of Point Positioning	26
4.3.1 Ionosphere-free linear combination	28
4.4 Approaches to Precise Point Positioning	30
4.4.1 GNSS Error Mitigation for PPP	30
4.4.2 Ambiguity resolution	35
4.4.3 Quasi Ionosphere Free (QIF) Algorithm	36
4.4.4 PPP-RTK	38
4.5 GNSS processing software engines	39
4.6 GNSS Low Cost Receiver: u-blox ZED-F9P	42
Development	43
5 Wa Software updating study	44
5.1 Theoretical comparison	44
5.2 Example comparison	46
6 GKA files	52
6.1 GKA 13 for baselines	53

6.2	GKA 14 for PPP	55
7	PPP software implementation	57
7.1	Study of processing parameters for PPP	57
7.1.1	Processing parameters in WaPPP	57
7.1.2	Processing parameters in RTKLIB	59
7.1.3	Comparison between WaPPP and RTKLIB for PPP	61
7.2	Graphical user interface	63
7.2.1	Configuration error management	69
7.3	Software architecture - General aspects	71
7.3.1	Function files	71
7.3.2	Dialog files	77
7.3.3	Other files	79
7.4	Software architecture - Post-processing	80
7.4.1	Dialog files	80
7.4.2	Other files	83
7.4.3	Software architecture diagram	86
7.5	Software architecture - Real-time	88
7.5.1	Dialog files	88
7.5.2	Other files	89
7.5.3	Software architecture diagram	91
7.6	GKA information	93
7.6.1	Cofactor matrix calculation	95
8	Integration proposal of PPP software with GOCA GNSS Control	97
8.1	Add dialogs and source code files	97
8.2	Create new PPP mode in GOCA	99
8.3	Create code for PPP mode	99
8.4	Re-write run PPP function	101
9	Field tests	102
9.1	Test 1: 12-hour observation for post-processing	102
9.1.1	Data collection	103
9.1.2	Processing parameters	106
9.2	Test 2: Real-time PPP processing	108
9.2.1	Processing parameters	109
	Results	113
10	Test 1: 12-hour observation.	115
10.1	RTKLIB as software engine	115
10.1.1	Ultra-rapid products	115
10.1.2	Final products	121
10.1.3	RTKLIB results comparison	123
10.2	WaPPP as software engine	125

10.2.1 Ultra-rapid products	125
10.2.2 Final products	127
11 Test 2: real-time processing.	129
12 Result comparison	131
Discussion	135
Conclusion	139
13 Conclusions	140
14 Future work	142
References	143
Annexes	147
Annex A	148
Annex B	158

List of Figures

1	GOCA Components	8
2	computational steps in GOCA	14
3	Deformation network	15
4	Integrated deformation analysis methods	19
5	Structure for SHM with GNSS and MEMS sensors in Stuttgart TV Tower	20
6	Electromagnetic wave representation	27
7	GNSS error mitigation	35
8	SSR corrections	39
9	Comparative between Wa1 and Wa2	46
10	General information Wa1 and Wa2	47
11	Parameters Wa1 and Wa2	48
12	Station information in Wa1 and Wa2	48
13	Baseline information in Wa1 and Wa2	49
14	DGNSS solution in Wa1 and Wa2	50
15	Ionospheric model in Wa1 and Wa2	51
16	Comparison between WaPPP and RTKLIB for PPP	62
17	PPP software main window	63
18	PPP processing mode	64
19	PPP Settings GUI	64
20	WaPPP orbit and clock information	65
21	RTKLIB Settings GUI	66
22	WaPPP Settings GUI	67
23	Real-time settings GUI	68
24	NTRIP connection for SSR corrections GUI	68
25	Configuration error example	70
26	Open file from directory function	72
27	Open file from directory function call	72
28	Create file in directory function detail	72
29	Choose directory function	73
30	Extract information of line	73
31	Read RINEX header information example	74
32	Write GKA file example	74
33	Get property and write it on configuration file	75
34	RTKLIB command	76
35	Example of WaPPP parameter in command	76
36	WaPPP command	76

37	RTKLIB real-time command	77
38	Open dialog example	77
39	Parameters to create process	78
40	Global variables	80
41	Errors and messages example	81
42	Check box code example	81
43	Choose file or folder example	82
44	Combo box code example	82
45	Get function example	83
46	Set function example	84
47	RTKLIB settings object parameters	85
48	WaPPP settings object parameters	86
49	Software architecture diagram post-processing	87
50	Real-time settings object	90
51	Software architecture diagram real-time	92
52	Rinex header example	93
53	RTKLIB solution header example	94
54	RTKLIB solution format example	94
55	WaPPP solution format example	95
56	Class assistant Visual Studio	98
57	Code to add PPP mode	100
58	Code to edit PPP mode	100
59	If example to execute mode functions	101
60	GOCA messages example	101
61	Antenna at pillar 300	102
62	12 hour observation information	102
63	STRSVR configuration	103
64	Input for STRSVR	103
65	Output for STRSVR	104
66	RINEX header	105
67	Files to PPP processing	106
68	PPP settings example	107
69	RTKLIB settings example	108
70	Real-time observation information	108
71	Real time observation input parameters	109
72	Real time observation parameters	110
73	Real time observation files	111
74	GKA output file example	114
75	Coordinates and errors RTKLIB with ultra-rapid products	115
76	Coordinates along time RTKLIB with ultra-rapid products	116

77	Coordinates along time RTKLIB with ultra-rapid products (detail)	117
78	Errors along time RTKLIB with ultra-rapid products	118
79	Errors along time RTKLIB with ultra-rapid products (detail)	119
80	Coordinates and errors RTKLIB with ultra-rapid products - 30 seconds	119
81	Coordinates along time RTKLIB with ultra-rapid products - 30 seconds (detail)	120
82	Errors along time RTKLIB with ultra-rapid products - 30 seconds (detail)	121
83	Coordinates and errors RTKLIB with final products	121
84	Coordinates along time RTKLIB with final products (detail)	122
85	Errors along time RTKLIB with final products (detail)	123
86	Cartesian coordinates pillar 300 with RTKLIB	124
87	Errors in coordinates pillar 300 with RTKLIB	125
88	Coordinates and errors WaPPP with ultra-rapid products	125
89	Coordinates along time WaPPP with ultra-rapid products	126
90	Errors along time WaPPP with ultra-rapid products	127
91	Coordinates and errors WaPPP with ultra-rapid products - 30 seconds	127
92	Coordinates and errors WaPPP with final products	128
93	Real time processing result	129
94	Real time processing result graph	129
95	Real time processing result errors graph	130
96	Cartesian coordinates pillar 300	131
97	Cartesian coordinates errors pillar 300	133
98	Geodetic coordinates pillar 300	133
99	Geodetic coordinates errors pillar 300	134
100	PPP error graph example	136

List of Abbreviations

DCB	Differential Clock Bias
DGNSS	Differential Global Navigation Satellite System
EOP	Earth Orientation Parameters
ERP	Earth Rotation Parameters
FEM	Finite Element Model
GMM	Gauss Markov Model
GNSS	Global Navigation Satellite System
GOCA	GNSS/LPS/LS-based Online Control and Alarm System
GPS	Global Positioning System
HSKA	Karlsruhe University of Applied Sciences
IERS	International Earth Rotation and Reference Systems
IFB	Integer Ionosphere-free Biases
IGS	International GNSS Service
ITRF	International Terrestrial Reference Frame
LPS	Local Positioning System
LS	Local Sensors
NTRIP	Networked Transport of RTCM via Internet Protocol
OSR	Observation Space Representation
OTL	Ocean Tide Loading
PPP	Precise Point Positioning
QIF	Quasi Ionosphere Free
RMS	Root Mean Square
RTK	Real Time Kinematic
RTS	Real Time Service
SBAS	Satellite Based Augmentation System
SHM	Structural Health Monitoring
SSR	State Space Representation
TEC	Total Electron Content
TVEC	Total Vertical Electron Content
VRS	Virtual Reference Station

Introduction

1. Motivation

Geomonitoring is an active research and application field, as structural failures or natural hazards are worldwide problems that often lead to significant economic, environmental and personal loss. Monitoring this kind of issues in order to early detection of significant movements and a good alarming system are crucial to prevent and minimize these problems.

PPP (Precise Point Positioning) is also a field in which great advances are being made in recent years. The main utility of this technique is getting precise absolute position, based on GNSS signal and only one receiver, what makes it an interesting methodology to use when no GNSS network is available or if only one receiver wants to be used in order to decrease costs.

Between different applications of PPP, geomonitoring has been pointed out as a possible application of PPP technique ([23], [5]).

In the context of geomonitoring, GOCA software has been developed by the Institute of Applied Research (IAF) of Karlsruhe University of Applied Sciences (HSKA). This project uses baselines in order to detect changes of some points with respect to others that are considered fixed. This implies that there must necessarily exist fixed points and several receivers have to be used. As a possible alternative or complement to this, PPP can be considered.

This idea, the use of PPP in the field of geomonitoring under GOCA project, has motivated this master thesis, named *Implementation of PPP as new GNSS observation type in the Geomonitoring System GOCA*.

The main aim of this thesis is the implementation of PPP (Precise Point Positioning) as further observation component into the C/C++ based geomonitoring system GOCA.

In this development, first implementation in C++ language has to be done to PPP processing. Here the RTKLIB-software and the WaPPP software will be used as GNSS processing engines for PPP. After that, the PPP software should be integrated in already existing GOCA_GNSS Control as first part of the geomonitoring chain. At the same time the GOCA data interface GKA version 4.0 has been extended to version 4.1 respectively, namely by the declaration of the new observation type PPP in terms of absolute GNSS positions and covariance matrices.

The present document is divided as follows. First of all, a theoretical background about the topic is provided. Then the development that has been done is presented.

After that, results are presented, followed by a discussion of them and finally some conclusions are shown.

The theoretical background has been mainly divided in two different parts. The first one provides a general overview of GOCA project, including its components and mathematical base. The second one is focused on GNSS positioning. This includes general GNSS technics, PPP fundamentals and approaches and also information about software and receiver that is going to be used.

The development part includes firstly theoretical information about the specific aspects that are going to be used in the implementation (as for example GKA files structure or PPP processing parameters available in both software engines). Then, the software implementation is described in detail, including all functions and variables involved in the code and finally, different field tests are described.

By last, obtained results and their corresponding discussion are presented and final conclusions of the whole thesis are exposed.

2. Objectives

The objectives that are expected to be covered in the context of this master thesis named *"Implementation of PPP as new GNSS Observation Type in the Geomonitoring System GOCA"* can be grouped in four groups: the first one includes the theoretical aspects, the second one the software development to PPP processing, the third objective block is the realisation of measurements with the U-blox ZED-F9P GNSS receiver in order to get real data to process and the last one includes the analysis and comparison of the obtained results.

The objectives to be covered in relation to theoretical aspects are:

- Provide a theoretical overview of the Geomonitoring System GOCA, including how does it work and the developments that are currently been done under this project.
- Provide a theoretical introduction to PPP processing, including the principles and algorithms and different approaches that can be applied.
- Provide a comparison of version 1 and version 2 of WaSoft, in order to provide an overview of the new version of that software that is currently used in GOCA software.
- Provide a theoretical comparison of RTKLIB and WaPPP as software engines to PPP processing. This theoretical background of both software engines would be used as support to the software development.

The objectives related with the PPP software implementation can be summarized as:

- Design, according to the previous theoretical study, of the user interface of the software for PPP processing.
- Development of the software in C++ language, as additional module of GOCA GNSS Control. This new module will be developed with a similar structure that the one in the GOCA GNSS Control module to facilitate the later integration of both softwares.
- Elaboration of technical documentation of the software, in order to ensure its comprehension.
- Integration proposal of PPP software in GOCA GNSS Control. Detail information of how the integration should be done needs to be presented.
- Design of new GKA format, to hold absolute PPP position as new observable. The results of PPP processing will be saved in this new GKA definition.

The practical measurements to be realized in the context of this master thesis are:

- Performance of 12-hour measurement with the U-blox ZED-F9P GNSS receiver and the NavXperience antenna, in the pillar number 300 placed in the B-building of the Hochschule Karlsruhe Technik and Wirtschaft University of Applied Sciences. This data will be used as prove of concept of the developed software.
- Performance of another measurement in the same pillar using the same equipment. This second measurement will be processed on real time to test the potential of this technique. This measurements and processing will be made using RTKLIB NAVI module.

Finally, the analysis objectives that are pretended to be covered are:

- Compare the obtained results using both software engines, RTKLIB and WaPPP.
- Compare the results using RTKLIB in real time and in post-processing, to check the potential of both techniques.
- Check the differences in the obtained results depending different products used (ultra-rapid or final products), to analyse their importance to achieve good accuracy and decrease convergence time using PPP.

Theoretical background

3. GOCA - GNSS/LPS/LS-based online Control and Alarm System

GOCA (GNSS/LPS/LS based Online Control and Alarm System) (see www.goca.info) is a geodetic geomonitoring system, developed at the Institute of Applied Research (IAF) of Karlsruhe University of Applied Sciences (HSKA). Its aim is the protection from natural hazards (landslides, dislocations zones, volcanoes), deformation monitoring of geotechnical installations (mining, tunnelling, etc.) as well as monitoring of deformations and changes in the physical parameters of object in structural and civil engineering (dams, towers, bridges, buildings) [19].

The geomonitoring system architecture is based on different components: data acquisition (sensor network operation and data communication), modelling (network adjustment and deformation analysis, detection of process changes), the reporting (protocolling, web visualization and virtual sensor computation) and reaction (alerting management) (see Figure 1).

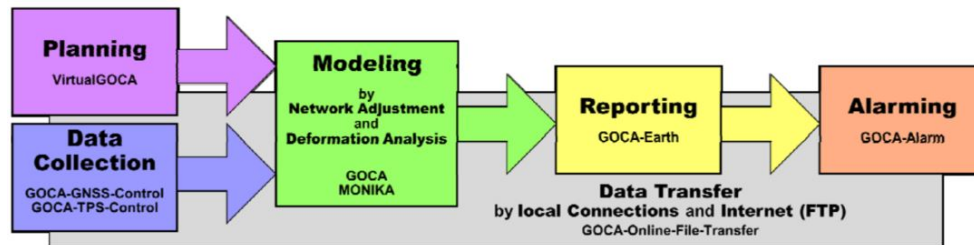


Figure 1: GOCA Components [19]

The real-time multi-sensor system GOCA applies GNSS/GPS, terrestrial sensors (LPS) such as total stations, levelling and hydrostatic levelling instruments and local sensors (LS) for a deformation monitoring and analysis [27].

The deformation analysis concept implemented in the GOCA software is due to a classical deformation analysis. It is based on a strict network adjustment and is realized in three subsequent adjustment steps (1st, 2nd and 3rd step). The monitoring network is physically realized by an array of GPS/GNSS and LPS sensors, while the respective deformation network design has to be specified in the initialization step (1st step) [18].

In this section, the different components and steps are introduced and the mathematical models involved are presented.

3.1 Mathematical models used in GOCA

In this section the main mathematical models used in GOCA system are described.

3.1.1 Gauss-Markov model

The parameter estimation in GOCA is based on the general concept of an M-estimation in a so-called Gauss-Markov model. This Gauss-Markov model is constituted by the two components of the functional and the stochastic model. According to [21], this reads as (3.1), (3.2). The functional model of the GMM generally represents observations l as nonlinear functions of the unknown.

$$\tilde{l} = l - \varepsilon = l(\tilde{x}) \quad (3.1)$$

and

$$C_{ll} \quad (3.2)$$

where l are sensor observations, C_{ll} is the covariance matrix, x are the parameters and ε represents the observation errors. The symbol $\tilde{}$ means expectation values or true values.

For M-estimation of the parameters x from the sensor observations l , the model in (3.1), that is not linear, has to be linearized. That linearization is based on approximate values x_0 for the parameters. With the simultaneous introduction of correction v and estimated values $d\hat{x}$ for the additions to the linearization, we obtain the following model:

$$l + v = A(x_0)d\hat{x} + l(x_0), \text{ with } a_{i,j} =: \left[\frac{\partial l_i(\hat{x})}{\partial x_j} \right]_{x_0} \quad (3.3)$$

In this linearized model, residuals (observation errors) are introduced as v and the estimated parameters are also introduced with $\hat{x} = x_0 + d\hat{x}$.

After this linearization, another step has to be applied, called homogenization. Homogenization is carried out by the left multiplication of the functional model with the matrix $C_{\varepsilon\varepsilon}^{-\frac{1}{2}}$:

$$\bar{A} = C_{\varepsilon\varepsilon}^{-\frac{1}{2}} A, \quad \bar{l} = C_{\varepsilon\varepsilon}^{-\frac{1}{2}} l \text{ and } \bar{\varepsilon} = C_{\varepsilon\varepsilon}^{-\frac{1}{2}} \varepsilon \quad (3.4)$$

The above transformation matrix $C_{\varepsilon\varepsilon}$ can be calculated with the diagonal matrix Λ of the eigenvalues and the associated modal matrix M of the eigenvectors, as follows:

$$C_{\varepsilon\varepsilon} = M \Lambda M^T \quad \text{and} \quad C_{\varepsilon\varepsilon}^{-\frac{1}{2}} = M \Lambda^{-\frac{1}{2}} M^T \quad (3.5)$$

With the above transformations the homogenized GMM invariant with respect to the parameter \tilde{x} .

After carrying out the parameter estimation \hat{x} in the homogenized model, corrections \bar{v} can be compared with the inverse transformation:

$$v = C_{\varepsilon\varepsilon}^{\frac{1}{2}} \bar{v} \quad (3.6)$$

Maximum Likelihood Estimation (M-Estimation)

Maximum likelihood estimation is a method of estimating the parameters of a statistical model so the observed data is most probable. Specifically, this is done by finding the value of the parameter (or parameter vector) that maximizes the likelihood function $L(\hat{x}, \bar{l})$, which is the joint probability of the observed data, over a parameter space.

That M-estimation is done by the minimization of the estimation function $\rho(\bar{v})$ of the homogenized observation errors \bar{v} with respect the linearized parameter part $d\hat{x}$. The M-estimation of parameters in a GMM is:

$$\sum_{k=1}^n \rho(\bar{v}_k) = \text{Min}|_{d\hat{x}} \quad \text{with} \quad (3.7)$$

$$\sum_{k=1}^n \rho((C_{\varepsilon\varepsilon}^{-\frac{1}{2}} A)_k d\hat{x} - (C_{\varepsilon\varepsilon}^{-\frac{1}{2}} l)_k) \quad (3.8)$$

Depending on the type of M-estimation function, the parameters are due to one or another kind of estimation. In GOCA system both least squares and robust estimation are used. The availability of a robust parameter in parallel to ordinary least squares increases the reliability of the online geomonitoring system and avoids wrong alarm due to gross errors [27].

- Least squares estimation.

In case of least squares estimation (L2-norm), the estimation function is given by:

$$\rho(\bar{v}_i) = \frac{1}{2} \hat{v}_i^2 \quad (3.9)$$

that means the assumption of a normal distribution of residuals, and the solution minimizes the sum of the squares of the residuals.

A M-estimation is robust, if the derivative, the so-called influence function $\Psi(\bar{v}_i)$ (3.10) is bounded, so it is not sensitive to gross errors.

$$|\Psi(\bar{v}_i)| = \left[\frac{d\rho(\bar{v})}{d\bar{v}} \right] \quad (3.10)$$

So the L2-norm (least squares) is not robust, as this influence function is not bounded, and must be robustified by data snooping.

- Robust estimation.

In order to avoid gross-errors in the sensor observations, another estimation function can be used. L1-estimation is used in GOCA, where:

$$\rho(\bar{v}_i) = |\bar{v}_i| \quad (3.11)$$

In this case, the result of the derivative (3.10) is bounded, so the estimation is robust.

The numerical solution of M-estimation for the GMM can be led back to an iterative least square estimation using the homogenized GMM and the diagonal matrix (3.12) as weight-matrix in the iteration procedure (see [21]).

$$W(\bar{v}_i) = \text{diag} \left[\frac{\partial \Psi(\bar{v}_i)}{\bar{v}_i} \right] \quad (3.12)$$

M-estimation provides unique estimates for the parameters \hat{x} at any time t . This M-estimation also allows the computation of the covariance matrix $C_{\hat{x}}$ of the estimated parameters as shown in (3.13).

$$C_{\hat{x}} = (A^T W A)^{-1} (A^T W^2 A) (A^T W A)^{-1} \quad (3.13)$$

With the covariance matrix we have also an statistical foundation with respect to the statistical analysis and assessment of estimated parameters.

3.1.2 Kalman filtering

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modelled system is unknown [36]. In the 3rd GOCA step, Kalman filtering is used, related to the transition equation $T(t)$ for the parametric state vector $x(t)$.

In the GOCA project, for the transition from the past $(t - \Delta t)$ to the present time t we have [27]:

$$u(t) = T(t) u(t - \Delta t), \quad \text{with} \quad (3.14)$$

$$u(t) = X_o(t) - X_o(t_0) \quad \text{and} \quad (3.15)$$

$$\begin{bmatrix} u(t) \\ \dot{u}(t) \\ \ddot{u}(t) \end{bmatrix} = \begin{bmatrix} I & [\Delta t] & [\frac{1}{2}\Delta t] \\ 0 & I & [\Delta t] \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} u(t - \Delta t) \\ \dot{u}(t - \Delta t) \\ \ddot{u}(t - \Delta t) \end{bmatrix} \quad \text{and} \quad (3.16)$$

$$x(t) = [u(t), \dot{u}(t), \ddot{u}(t)]^T \quad (3.17)$$

The state vector of the parameters $x(t)$ comprises the individual 3D displacements $u(t)$, the velocities $\dot{u}(t)$ and the accelerations $\ddot{u}(t)$ of the object-points between two subsequent time intervals.

The state transition model shown in (3.14) - (3.17) corresponds to the Taylor series of the unknown displacement function, truncated after the second term (considering constant acceleration within the interval).

The covariance-matrix C_x of the prediction $x(t)$ can be computed according to the law of error propagation from the covariance-matrix of the preceding state vector $x(t - \Delta t)$ and the preceding Kalman-filtering respectively.

The observations $l(t)$ are the difference between the present object-point positions (out of GOCA step 2 (FIN-files) in the time interval) and the initial position derived for the start of the Kalman-Filtering (3.18).

$$l(t) = l(x(t)) =: u(t) = x_o(t) - x_o(t_0) \quad (3.18)$$

Their covariance matrices C_l referenced to the time t and read:

$$C_l = C_{x_o}(t) + C_{x_o}(t_0) \quad (3.19)$$

The Kalman-Filtering is equivalent to the common adjustment of the predicted state vector and the state vector related observations. The generalized M-estimation (see section 3.1.1) of the parameters $\hat{x}(t)$ for these two observation components and their stochastic models can be computed iteratively as:

$$\hat{x}(t)^{(j)} = x(t) + K^{(j)} (\bar{I}(t) - \bar{I}(x(t))), \quad \text{with} \quad (3.20)$$

$$K^{(j)} = (C_x^{-\frac{1}{2}} W_x^{(j)} C_x^{-\frac{1}{2}} + \bar{A} W_l^{(j)} \bar{A})^{-1} \bar{A} W_l^{(j)} \quad (3.21)$$

The Kalman matrix $K^{(j)}$ is recomputed in each step. In each step diagonal weight matrices are set up according to (3.12). These weighting functions are related to the argument of homogenized corrections $\bar{v}_{x,i}$ and $\bar{v}_{l,i}$ of the parameter and the observation component.

3.2 GOCA components

GOCA is divided in different components (see Figure 1). This components are [19]:

- Component 1 - Data acquisition: sensor network operation and data communication.

This component uses an open data interface (GKA format), that has been defined for GOCA. In that way, any GNSS and terrestrial sensors (LPS) can be connected to the GOCA modelling and deformation analysis software. As components of it, several packages GNSS-Control and TPS-Control have been developed in GOCA.

- Component 2 - Modelling: network adjustment and deformation analysis, statistically founded evaluation of the state parameters, detection of process changes.

This component is responsible of the mathematically and statistically rigorous geodetic network adjustment in a so-called observation-related deformation-analysis. This includes the three different computational steps in the network adjustment. These steps are defined in section 3.3 and the mathematical models behind them are discussed in section 3.1.

- Component 3 - Reporting: web-visualization and virtual sensor computation.

GOCA- Earth, developed for this component, visualizes, based on Google Earth, the deformation states and also provides the respective numerical values.

- Component 4 - Reaction: alerting management.

The software component GOCA-Alarm sets alarm when critical values for the state parameter estimations or predictions and/or statistical significance levels are exceeded. GOCA- Alarm then transmits a respective short information out of the stored results files.

3.3 GOCA computation steps

GOCA network adjustment and deformation analysis is made in three steps, as mentioned before. Step 1, the initialization step, is responsible of determining coordinates of the reference frame. These coordinates are used in step 2 to obtain the object coordinates and their covariance matrices. This information is used in the step 3 for the estimation of 3D displacements of the object points, applying different approaches. Steps 2 and 3 are running online parallel.

A schema of these processing steps is shown in Figure 2.

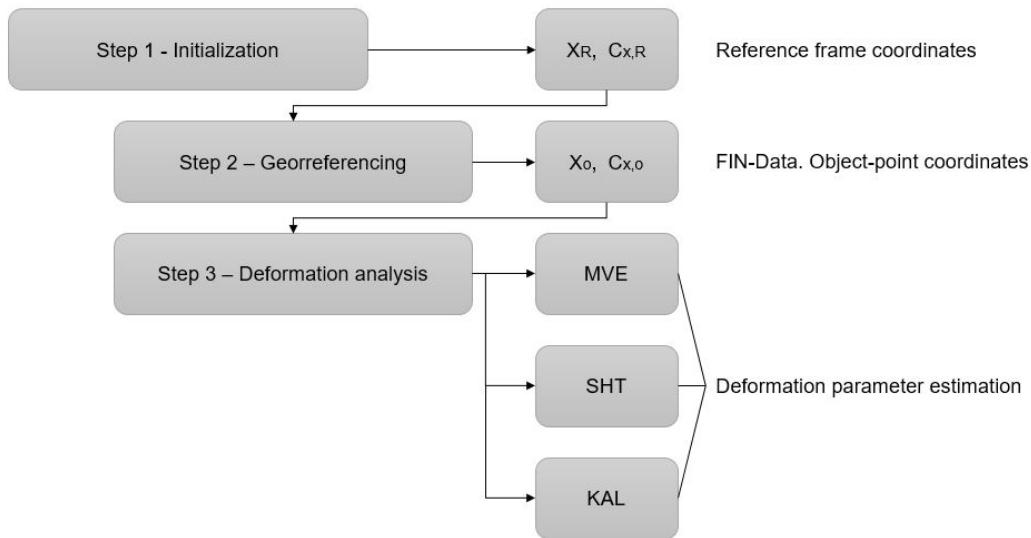


Figure 2: computational steps in GOCA

In this section, how these steps work and their relationship with the mathematical models (see section 3.1) are exposed.

Steps 1 and 2. Initialization and Georeferencing

The GOCA sensor array is divided into a stable reference point frame x_R and a deformable object area x_O (Figure 3). The sensor points x_R set up the unique 3D coordinate reference frame for the network adjustment-based computation and further modelling of the object point positions x_O in the different areas. For this, the GNSS and LPS sensor data are transmitted online or near-online and processed in a three steps network adjustment and deformation state $x(t)$ parameter estimation concept.

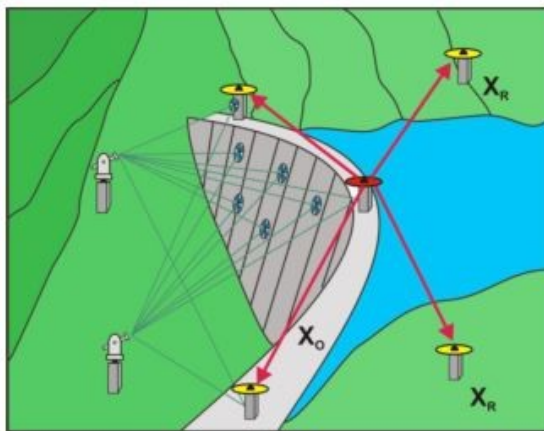


Figure 3: Deformation network [27]

In the GOCA network adjustment step 1, also called initialization, the functional adjustment model is based on observation equations $l = l(t, x)$. The adjustment step 1 objective is the determination of the 3D reference point x_R frame and the respective covariance matrix $C_{x,R}$, previously to the permanent monitoring. The GOCA step 2 covers then the continuous 3D georeferencing of the object points x_O in the reference point datum [19].

This first adjustment step is based on a least squares (L2-norm) free network adjustment of the GNSS- and LPS-based related to the Gauss- Markov model (see section 3.1). This adjustment is related to the observation equations ((3.22) - (3.26)).

GNSS horizontal baselines (2D):

$$\begin{bmatrix} \Delta x_{ij} \\ \Delta y_{ij} \end{bmatrix}_{GNSS} + \begin{bmatrix} v_{\Delta x,ij} \\ v_{\Delta y,ij} \end{bmatrix}_{GNSS} = \begin{bmatrix} \Delta \hat{x}_{ij} \\ \Delta \hat{y}_{ij} \end{bmatrix} \quad (3.22)$$

Horizontal distances (2D):

$$s_{ij} + v_{s,ij} = \sqrt{\Delta \hat{x}_{ij}^2 + \Delta \hat{y}_{ij}^2} \quad (3.23)$$

Directions (2D):

$$r_{ij} + v_{r,ij} = \arctan \frac{\Delta \hat{y}_{ij}}{\Delta \hat{x}_{ij}} - \hat{\delta}_i \quad (3.24)$$

GNSS vertical baselines (1D):

$$\Delta h_{GNSS,ij} + v_{\Delta h,ij} = \Delta \hat{h}_{ij} \quad (3.25)$$

Terrestrial height difference (1D):

$$\Delta H_{terr,ij} + v_{\Delta H,ij} = s_h^m \Delta \hat{h}_{ij} + (\hat{a}_{00} + \hat{a}_{10} x_j + \hat{a}_{01} y_j)^m - (\hat{a}_{00} + \hat{a}_{10} x_i + \hat{a}_{01} y_i)^m \quad (3.26)$$

where m is the area index, Δs_h^m is the scale difference and a_{ik}^m the polynomial coefficients for the modelling of height reference surface in the different local area parts m (\hat{g}^m).

The parameters of the initialization step depend on the applied sensor types. The initialization adjustment of the sensor data is done for a user-defined starting epoch and initialization time interval [27].

In the GOCA-adjustment step 2, the 3D permanent georeferencing of the object-point positions is done, making again use of the observation equations $l = l(t, x)$ in the 2D/1D concept given by ((3.22) - (3.26)). The reference frame coordinates and the additional and observation specific parameters s , s_h and \hat{g}^m are kept as fix parameters in this second step according to the results of the first one. With that, influence of observation errors on the reference frame are avoided, and the risk of a wrong alarm is reduced [19].

The step 2 step is running online and the resulting estimated object-point positions $x_O(t)$ and their covariance matrix $C_O(t)$ are stored in daily object-point FIN-files [19].

In order to avoid possible false alarms due to gross sensor data and other systematic errors, the parameter estimations $y(t)$ in the GOCA deformation analysis in the real-time step 2 (and also in step 3, see section 3.3.1) is based on the concept of robust M-estimation.

Both adjustment steps 1 and 2 are based on the linearized GMM (section 3.1).

3.3.1 Step 3. Deformation Analysis

The GOCA adjustment step 3, named deformation analysis, deals with the estimation of the parameters of different deformation functions, and it is again based either on least squares (L2 norm) or on robust M estimations. The parameter estimation runs online and parallel to GOCA step 2 and its objective is to set u an alarm in case of significant critical parameters.

The observation input for the different deformation state estimations is the observation data in terms of the object-point position time series $x_O(t)$ and $C_O(t)$, which have been stored in the simultaneous running GOCA adjustment step 2 as FIN files.

Three different deformation functions are used: moving average estimation, online displacement estimation and Kalman filtering prediction.

Moving average estimation.

The moving average estimation (MVE) is a simple deformation function used in GOCA software where $x_O(t)$ and $C_O(t)$ are direct observations with respect the MVE position $y(t)^T = (x(t), y(t), h(t))$ in each adjustment interval. Local sensor data and their covariance matrix information can be also used in this function.

This function includes an alarm setting due to critical displacements compared to the current displacements $u(t) = y(t) - x_O(t_0)$, derived with respect to the initial object point positions $x_O(t_0)$ [27].

Online displacement estimation.

Another deformation function is the online displacement estimation (SHT) between different extended epochs t_0 and t_i . That means that the two epochs starts at individual times and have interval lengths. The functional model of the object point displacement is:

$$\begin{bmatrix} l_{t_0} \\ l_{t_i} \end{bmatrix} + \begin{bmatrix} v_{t_0} \\ v_{t_i} \end{bmatrix} = \begin{bmatrix} E_1 & 0 \\ E_2 & E_2 \end{bmatrix} \begin{bmatrix} \hat{x}_0 \\ \hat{u}(t) \end{bmatrix} = A \hat{y} \quad (3.27)$$

$$\hat{y} = [\hat{x}_0(t_0), \hat{u}(t_0, t_i)]^T \quad (3.28)$$

where v are the observation residuals. The deformation parameters $\hat{y}(t)$ are for each object point the 3D adjusted epoch state position $\hat{x}_0(t_0) = [\hat{x}, \hat{y}, \hat{h}]^T$ and the 3D displacements $\hat{u}(t_0, t_i) = [y_x, u_y | u_h]_{t_0, t_i}^T$ between t_0 and t_i . Design matrices E_1 and E_2 are column matrices composed of (3x3) unit matrices for each 3D point observation

in the respective epoch intervals.

The observations l_{t_0} and l_{t_i} and their covariance matrices are taken of the object-point time series ($x_O(t)$ and $C_O(t)$).

The displacement estimation can be done in three different modes:

- Epoch 1 = Static initialization (Step 1 of GOCA). Epoch 2 repeated periodically.
- Epoch 1 = Static and fixed by the user. Epoch 2 repeated periodically.
- Epoch 1 = Repeated also periodically such as the epoch 2.

Here also local sensor data can be added to the displacement estimation.

Kalman-Filtering.

The third component of the deformation parameter is the Kalman filtering (see section 3.1.2).

The GOCA Kalman-filtering can be performed by setting either a least squares or a robust L1-norm estimation. The results ($\hat{x}(t), C_{\hat{x}}(t)$) are stored as Kalman-Files (KAL). Local sensor data can be used as observation input of the Kalman-Filtering. The equation (3.16) allows to predict the time Δt , which is left until given critical values for the state vector are reached. This enables to use GOCA as a early-warning system.

3.4 Further developments: Integrated Deformation Analysis

In order to improve the deformation analysis in geodetic monitoring, new developments are being made. In this context, named integrated deformation analysis, aims to integrate the classic geodetic geomonitoring (based on geometry) together with additional physical observations.

The geometric parameters $y(t)$ can be used commonly with physical system parameters p in an integrated deformation analysis, which then allows to detect changes Δp in the physical parameters p of structures, meaning Structural Health Monitoring (SHM) [19].

Advanced evaluation models for deformation analysis do not only consider the change of the geometry of an object in space and time. They rather investigate and incorporate also the influencing factors (causative forces, internal and external loads) causing the deformation. They regard in addition the object’s physical properties which are characteristic and responsible for the response of the object to the acting forces [37].

In the general classification of integrated deformation analysis in black, grey and white box system descriptions (Figure 4) .

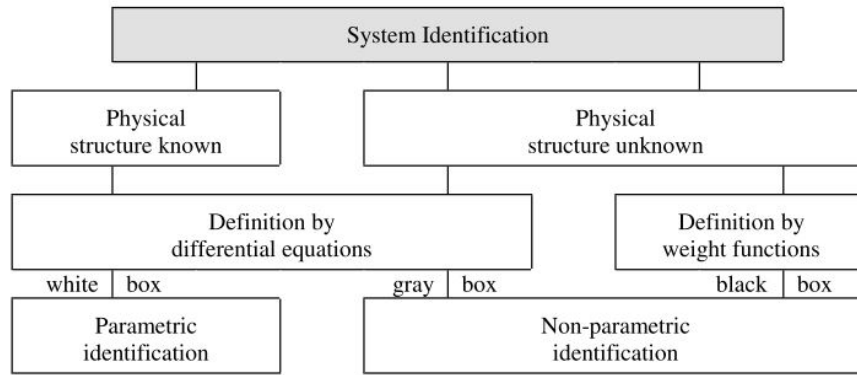


Figure 4: Integrated deformation analysis methods [37]

If the physical relationship between input and output signals, is supposed to be known and can be described by differential equations, then the model is called a parametric model (structural model). The system identification is carried out in a so-called white box model [37]. The class of finite-element models (FEM) belongs to the white-box category.

There exist two different approaches for FEM-based structural health monitoring, the static and the dynamic cases. In the static case, only changes in the parametrized stiffness matrix $K(p_K)$ can be identified and in the dynamic FEM approach is enabled additionally the identification of parameter changes in the damping and the mass matrix $C(p_C)$ and $M(p_M)$, respectively of a structure.

3.4.1 Stuttgart TV Tower: Reference Object for SHM

The PPP developments made in the context of this Master thesis in the GOCA project will be tested in this Stuttgart as a proof of concept by a TCP/IP Internet-based addressing and data reception of the ublox ZED-F9P GNSS receiver as one sensor part of the new SHM (Structural Health Monitoring) GNSS/MEMS box used

as a so-called passive sensornetwork node.

Stuttgart TV Tower is used as a reference for innovative methods for the early detection of potential hazards of structures (SHM) by new algorithms, sensor systems and information technologies presented. The latter includes a general Internet-based server client for integrated geomonitoring of objects (Figure 5) [20].

Stuttgart TV Tower is used in the project "The Stuttgart TV tower as reference object for testing of innovative sensor systems, mathematical models, algorithms, software and IT for the early detection of damage and potential risks of structural systems". This is carried out in the GOCA and NAVKA projects of the GNSS and Navigation Laboratory of the University of Karlsruhe and also by the engineering office for Applied Geodesy, Photogrammetry and Geoinformatics E. Messmer, Schwaikheim and other partners (as the owner of the TV tower and the State Office for Geobasis Information and Land Development (LGL) Karlsruhe).

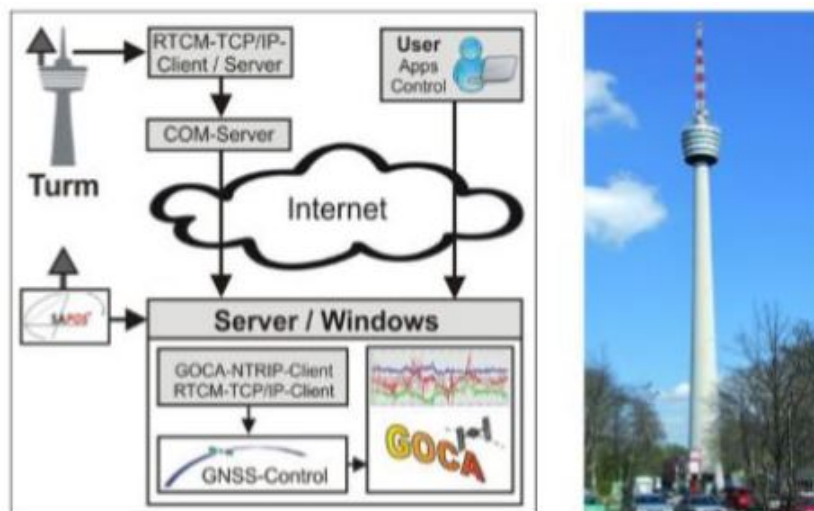


Figure 5: Structure for SHM with GNSS and MEMS sensors in Stuttgart TV Tower [20]

In this project, GNSS data and MEMS (accelerometers, gyroscopes, cameras) are integrated, so the state vector $y(t)$ is composed of a total of 15 state parameters (equation (3.29)).

$$y(t) = [x^e \ y^e \ z^e \ | \ \dot{x}^e \ \dot{y}^e \ \dot{z}^e \ | \ \ddot{x}^e \ \ddot{y}^e \ \ddot{z}^e \ | \ r \ \rho \ \gamma \ | \ s]^T \quad (3.29)$$

In contrast to navigation, in this case the data fusion can be essentially stabilized via the introduction of inequalities to the state space with a robust SIMPLEX-based L1

standard parameter estimation [20]. It is based on the dynamic state form of FEM-based vibration equation (see 3.4). The equations of this FEM-based oscillation equations are the following:

$$K(p_K)\Delta u(t) + C(p_c)\Delta \dot{u}(t) + M(p_M)\Delta \ddot{u}(t) = f(t) \quad \text{and}, \quad (3.30)$$

$$K(p_K)\Delta u(t) + C(p_c)\Delta \dot{u}(t) + M(p_M)\Delta \ddot{u}(t) = 0 \quad (3.31)$$

where equation (3.30) is the case of damped oscillation with excitation $f(t)$ and equation (3.31) corresponds to damped free natural oscillation. In the static case of SHM, only the stiffness matrix K is accessible for parametrization.

The state transition of the FEM-based reads as follows (note the difference with equation (3.16) of the pure geometric case):

$$\begin{bmatrix} u(t) \\ \dot{u}(t) \\ \ddot{u}(t) \end{bmatrix} = \begin{bmatrix} I & [\Delta t] & [\frac{1}{2}\Delta t] \\ 0 & I & [\Delta t] \\ 0 & [-M(p_M)^{-1} K(p_M) \Delta t] & [I - M(p_M)^{-1} C(p_c) \Delta t] \end{bmatrix} \begin{bmatrix} u(t - \Delta t) \\ \dot{u}(t - \Delta t) \\ \ddot{u}(t - \Delta t) \end{bmatrix} \quad (3.32)$$

In the dynamic case, the parameter estimation is based agains on a Kalman-Filtering approach, based on equation (3.32) and the integration of different kind of suitable sensors.

The parametrization of changes in the physical parameters $\Delta p_k, \Delta p_M$ as a function of changes (ω_i refers to vibration of frequencies and u_i to vibration modes) in the spectral and modal matrix are given by:

$$\Delta \omega_i^2(\Delta p_k, \Delta p_M) = u_i^T [dK(\Delta p_K) - \omega_i^2 dM(\Delta p_M)] u_i \quad (3.33)$$

$$\Delta u(\Delta p_K, \Delta p_M) = \frac{u_i^T dM(\Delta p_M) u_i}{2} u_i + \sum_{j=1, j \neq i}^n \left[\frac{1}{\omega_i^2 - \omega_j^2} u_i^T [dK(\Delta p_K) - dM(\Delta p_M)] u_j \right] u_j \quad (3.34)$$

This approach implies a so-called inverse eigenvalue-eigenvector problem, meaning the task to conclude from changes in the respective spectral characteristics to changes in the parametrization of the general eigenvalue problem itself [19].

3.4.2 SHM-Implementation on Stuttgart TV Tower

As exposed in section 3.4.1, the TV Tower of Stuttgart is used as a reference object for Structural Health Monitoring (SHM). In this section, two different approaches are going to be explained: the first one, that uses only GNSS sensors as the current project and the second one, under development, that also includes other kind of sensors.

GOCA GNSS Control and pure GNSS Box

Currently, the TV Tower of Stuttgart has installed a GNSS receiver (Leica receiver), in a fixed position of the TV tower. Low cost receiver will be installed and tested on it also.

It is known that it rotates (approximately one rotation each 5 seconds) and the aim of having this tower monitored is to see how it rotates.

The receiver sends raw data to a server, where GOCA GNSS Control is installed and the data in GKA format is collected to compute it using afterwards the main module of GOCA software.

In GNSS Control module, software engines RTKLIB and WaSoft are used to process data. In the existing GOCA GNSS Control, information about base lines is storage in GKA format. The PPP information, developed under the context of this master thesis, will be also a part of this process, taking PPP position and storing it in another GKA file.

GONA GNSS Control and Multisensor SHM Box

As explained in the previous point, currently just pure GNSS data is used to the monitoring of TV Tower of Stuttgart, but new multi-sensor box is being developed and it is going to be placed in the TV Tower also in order to have, not only the GNSS information, but also additional information of different sensors.

This box will have LAN/WLAN, internet access and a TCP/IP addressing, so information will be able to stream down via Internet and used in GNSS Control. It also will contain different sensors. The GNSS sensor in this box will be the ZED F9 low cost receiver. This box is under design jet.

4. GNSS positioning

GNSS positioning has been long time used in geomonitoring projects. In this chapter, the main aspects of it are exposed, focussing on the PPP technic as principal objective of study of the master thesis regarding this document.

4.1 Terminology

In this section, some basic concepts regarding GNSS positioning are exposed.

Code Pseudoranges and Phase Pseudoranges

The satellite navigation observables are ranges which are deduced from measured time or phase differences based on a comparison between received signals and receiver-generated signals [13].

Code pseudoranges are obtained by calculating the travel time of the signal, correlating the received signal from the satellite and the replicated receiver signal, until the correlation is maximum. To achieve a good accuracy, time synchronization and other factors has to be taken into account. Formulation is shown in Equation(4.3).

Phase pseudoranges are obtained from the difference of phase of the carrier wave of satellite and the one generated by the oscillator in receiver. It is a distance related with the integer number of wave lengths and its phase. Mathematic equation for phase is shown in Equation(4.11).

These carrier phase measurements are much more precise than the code measurements (typically two orders of magnitude more precise), but they are ambiguous by an unknown integer number of wavelengths. Indeed, this ambiguity changes arbitrarily every time the receiver loss the lock on the signal [8].

Static and Kinematic Positioning

Static denotes a stationary observation location, while kinematic implies motion. So, static positioning is used when the receiver is fixed in one place, such as in case of base of control establishing and kinematic when the receiver is moving, for example in vehicle navigation.

Kinematic and dynamic are usually used as synonymous, but they are slightly different. The term kinematic describes the pure geometry of a motion, whereas dynamic considers the forces causing the motion.

Absolute and Relative Positioning

Absolute positioning is when the coordinates of a single point are determined by using a single receiver which measures pseudoranges to four or more satellites. The terms point positioning, single-point positioning, and the term absolute point positioning are synonymously used.

Relative positioning refers to simultaneous measurements for two different receivers and the same satellites. The measurements taken at both sites are directly combined. Normally, the coordinates of one site are known and the position of the other site is to be determined relatively to the known site.

Relative positioning and differential positioning are often used as synonymous but they are different. Differential positioning is a technique based on applying corrections to pseudoranges measured at an unknown site. The technique provides instantaneous solutions where improved accuracies with respect to a reference station are achieved.

Advantages and disadvantages of both approaches are discussed in section 4.2

Real-time Processing and Post-Processing

For real-time GNSS, the results must be available in the field immediately. The results are denoted as instantaneous if the observables of a single epoch are used for the position computation and the processing time is negligible. A different and less stringent definition is quasi (or near) real-time which includes computing results with a slight delay.

Post-processing refers to applications when data are processed after the fact.

4.2 Absolute Positioning: Benefits and Limitations

The powerful of GNSS in geomonitoring applications has been proved in different studies [23, 5, 11, 31]. In section 4.1 different terms of GNSS positioning has been introduced. In case of geomonitoring both code and phase can be used; normally static technique is applied as the points are supposed to be fixed; with regard absolute or relative, both can be again used, in case of GOCA project, relative positioning is used but with the implementation of PPP also absolute positioning could be used; real-time or near real-time is important for geomonitoring to achieve early alarms.

Benefits and limitation of using absolute positioning instead of relative positioning are discussed in this section.

Absolute positioning or Precise Point Positioning, presents some advantages. The main ones are [38]:

- PPP involves only a single GNSS receiver and, therefore, removes the need for GNSS users to establish local base stations. As a result, it eliminates the spatial operating range limit as well as the constraint of simultaneous observations on both rover and base receivers.
- PPP can be regarded as a global positioning approach because its position solutions are referred to a global reference frame. As a result, PPP provides much greater positioning consistency than the differential approach in which position solutions are relative to the local base station or stations.
- PPP can bring to applications is that it reduces labor and equipment cost and simplifies operational logistics to field work since it eliminates the dependency on base station(s).
- In case of relative positioning, the reference station must typically be placed in a stable area. Such a place could be difficult to find: in the case of strong earthquakes, for example, the reference station may also be displaced. PPP technique overcomes this limitation as it does not require reference stations [5].

Despite these benefits of PPP technique, there exist also several limitations in its use:

- PPP requires a long initialization period for phase ambiguities to converge to near constant values and for the solution to reach its optimal precision. This convergence time depends on the required accuracy and the chosen approach to achieve PPP [3].
- The primary factors that limit the accuracy of PPP are the limited precision of current precise orbit and clock products and the effects of unmodeled error sources [3].
- Single frequency PPP users are requested to obtain external information on the ionospheric delay. This ionospheric delay can be obtained in real-time from models delivered for example by the IGS, SBAS services or regional service providers. This uncertainty maps into range errors in the order of 30 cm up to 1 m [15].
- If two frequency receiver is used to mitigate ionospheric error, the integer characteristic of ambiguities is lost (see section 4.4.2).

4.3 Fundamentals of Point Positioning

The basic undifferentiated observation equations for code and carrier phase measurements are related to the user position, clock, troposphere, ionosphere, and ambiguity parameters [23, 13].

Denoting $t^s(sat)$ the signal emission time referred to the satellite clock and $t_r(rec)$ the signal reception time referred to the receiver clock; δ_s the satellite clock bias and δ_r the receiver clock bias, understanding bias as the difference with respect a common time system, the difference between the clock readings of the receiver and the satellite can be expressed as:

$$t_r(rec) - t^s(sat) = [t_r + \delta_r] - [t^s + \delta^s] = \Delta t + (\delta_r - \delta^s) \quad (4.1)$$

When multiplying the time interval by the speed of lighth, we obtain the code pseudorange:

$$R = c\Delta t + c(\delta_r - \delta^s) \quad (4.2)$$

The first term in the obtained equation (4.2) is the range calculated from the true signal travel time, expressed as ρ . Considering this and the other biases that affect the measurement, the general equation for code observations is obtained:

$$R = \rho + c(\delta_r - \delta^s) + d_{TROP} + d_{ION} + \varepsilon_p \quad (4.3)$$

where R is the measured pseudo-range; ρ is the geometric range; c represents the speed of the light; δ^s corresponds to the satellite clock bias; δ_r is the receiver clock bias; d_{TROP} is the tropospheric delay; d_{ION} represents the ionospheric delay and ε_p represents all remaining biases including the measurement noise and multipath effect in code pseudo-range.

The corresponding observation equation for phase is based on the basic principles of waves (Figure 6).

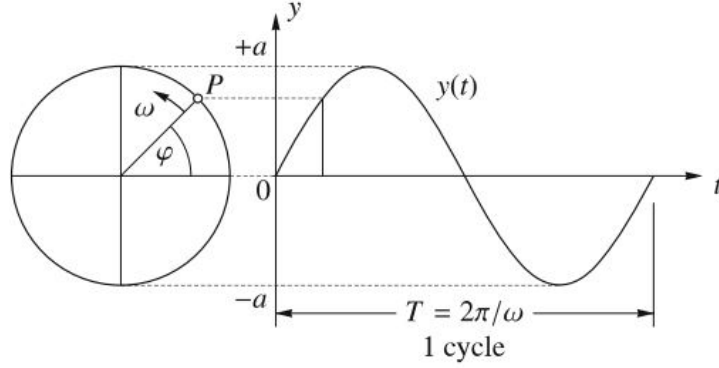


Figure 6: Electromagnetic wave representation [13]

The phase of a wave can be given in radians or in cycles:

$$\varphi = \omega t = ft \quad (4.4)$$

where ω is the angular velocity; f is the frequency and t represents the time.

Considering a single epoch, the phase also varies with increasing distance between emitter and receiver. In this spatial context, the phase is expressed by range(ρ) and wavelength(λ). The complete phase equation combines both variations, so phase can be expressed as:

$$\varphi = ft - \frac{\rho}{\lambda} \quad (4.5)$$

Applying this to both the phase of the received and reconstructed carrier($\varphi^s(t)$) with frequency f^s , and the phase of a reference carrier generated in the receiver($\varphi_r(t)$) with frequency f_r , taking also into consideration the initial phases caused by clock errors ($\varphi_{0s}^s(t) = -f^s\delta_s, \varphi_{0r}(t) = -f_r\delta_r$), and using the relation $c = f\lambda$, we obtain:

$$\varphi^s(t) = f^s t - f^s \frac{\rho}{c} - \phi_0^s(t) \quad (4.6)$$

$$\varphi_r(t) = f_r t - \varphi_{0r}(t) \quad (4.7)$$

So, the phase difference is given by:

$$\varphi_r^s(t) = \varphi^s - \varphi_r = -f^s \frac{\rho}{c} + f^s \delta_s - f_r \delta_r + (f^s - f_r)t \quad (4.8)$$

The deviation of the frequencies from the nominal frequency is in the order of only some fractional parts of hertz. So a frequency error may be neglected ($f^s = f_r = f$). Eq. (4.8) may be written in the simplified form:

$$\varphi_r^s(t) = -f\frac{\rho}{c} - f(\delta_r - \delta^s) \quad (4.9)$$

The phase is formed by a fractional phase and a number of integer cycles, that means $\varphi_r^s(t) = \Delta\varphi_r^s + N$. Rewriting Eq. (4.9) with this, and considering $\phi_c = -\Delta\varphi_r^s$ we obtain the expression of phase in cycles:

$$\phi_c = f\frac{\rho}{c} + f(\delta_r - \delta^s) + N \quad (4.10)$$

Taking into account again the relationship $c = f\lambda$, and multiplying the above expression by λ , we obtain the phase expressed in range ϕ (in meters). Additionally, as in case of code, we have to consider other biases as atmospheric effects. So finally, we obtain the expression:

$$\phi = \rho + c(\delta_r - \delta^s) + d_{TROP} - d_{ION} + \lambda N + \varepsilon_\phi \quad (4.11)$$

where ϕ is the measured carrier phase; λ is the carrier wavelength; N corresponds to the integer phase ambiguity and ε_ϕ represents all remaining biases including the measurement noise and multipath effect in phase pseudo-range.

In both (4.3) and (4.11), the geometric range is given by:

$$\rho(t_i) = \sqrt{\left(X_s\left(t_i - \frac{\rho(t_i)}{c}\right) - X(t_i)\right)^2 + \left(Y_s\left(t_i - \frac{\rho(t_i)}{c}\right) - Y(t_i)\right)^2 + \left(Z_s\left(t_i - \frac{\rho(t_i)}{c}\right) - Z(t_i)\right)^2} \quad (4.12)$$

where (X_s, Y_s, Z_s) is the satellite position and (X, Y, Z) represents the receiver position.

4.3.1 Ionosphere-free linear combination

The most used mathematical model in PPP is the dual frequency, that means that is based on an ionosphere-free combination of code pseudoranges and carrier phases, that eliminates the effect of the ionosphere by using two different frequencies.

The ionosphere free combination is based on a linear combination of two different frequencies, in which the ionosphere effect is eliminated, or more precisely, the reduced. The ionosphere effect can be represented as:

$$d_{ION} = \pm \frac{1}{\cos z'} \frac{40.3}{f^2} TVEC \quad (4.13)$$

where z' is the zenith angle of the satellite, f is the frequency and $TVEC$ represents

the total vertical electron content. In equation (4.13), positive correction (+) is applied in code measurements and the negative one (-) in case of phase measurements.

In case of code pseudoranges, the ionosphere free combination is obtained from applying (4.3) (without taking into account d_{TROP} and ε_p) to both frequencies f_1 and f_2 :

$$R_1 = \rho + c(\delta_r - \delta^s) + d_{ION1} \quad (4.14)$$

$$R_2 = \rho + c(\delta_r - \delta^s) + d_{ION2} \quad (4.15)$$

Multiplying both (4.14) and (4.15) by f_1^2 and f_2^2 respectively, and taking into account the relationship shown in (4.13), we obtain the difference as:

$$R_1 f_1^2 - R_2 f_2^2 = (f_1^2 - f_2^2)(\rho + c(\delta_r - \delta^s)) \quad (4.16)$$

Dividing the previous equation by $(f_1^2 - f_2^2)$ and adding the other errors that affect the signal, we obtain the iono-free equation for code:

$$\left[R_1 - R_2 \frac{f_2^2}{f_1^2} \right] \frac{f_1^2}{f_1^2 - f_2^2} = \rho + c(\delta_r - \delta^s) + d_{TROP} + \varepsilon_p \quad (4.17)$$

In case of phase, ionosphere free combination is obtained following a similar development. In this case we start from simplified version of (4.11), dividing it by λ and then rewriting it using the relationship $c = f\lambda$. So we obtain, for each frequency:

$$\phi_1 = \frac{f_1}{c} \rho + f_1(\delta_r - \delta^s) + N_1 - \frac{f_1}{c} d_{ION1} \quad (4.18)$$

$$\phi_2 = \frac{f_2}{c} \rho + f_2(\delta_r - \delta^s) + N_2 - \frac{f_2}{c} d_{ION2} \quad (4.19)$$

Can be rewritten as:

$$\phi_1 = a f_1 + N_1 - \frac{b}{f_1} \quad (4.20)$$

$$\phi_2 = a f_2 + N_2 - \frac{b}{f_2} \quad (4.21)$$

where a is the geometry term and b the ionospheric term:

$$a = \frac{\rho}{c} + (\delta_r - \delta^s) \quad (4.22)$$

$$b = \frac{f_i^2}{c} d_{ION} \quad (4.23)$$

In (4.23), d_{ION} refers to (4.13). So, the linear combination is, in this case:

$$\phi_1 f_1 - \phi_2 f_2 = a(f_1^2 - f_2^2) + N_1 f_1 - N_2 f_2 \quad (4.24)$$

Multiplying by $f_1/(f_1^2 - f_2^2)$, reordering it and adding the additional biases, we finally obtain the iono-free equation for phase:

$$\left[\phi_1 - \frac{f_2}{f_1} \phi_2 \right] \frac{f_1^2}{f_1^2 - f_2^2} = \frac{f_1}{c} \rho + f_1 (\delta_r - \delta^s) + d_{TROP} + \varepsilon_\phi \quad (4.25)$$

The unknown parameters to be determined are the point position contained in ρ , the biases of receiver's clock δ_r , the tropospheric delay d_{TROP} , and the ambiguities N [15]. It should be noted that the ionosphere-free ambiguity term is no longer an integer number.

As an alternative for this linear combination, were the ambiguity term is not more an integer number, Quasi Ionosphere Free can be used. This algorithm is shown in section 4.4.3.

4.4 Approaches to Precise Point Positioning

The traditional PPP model, presented in section 4.3, uses the ionosphere-free code and phase observations. The main limiting factors with respect to the achievable accuracy are the orbit errors, the clock errors, and the atmospheric influences.

In order to go from Point Positioning to Precise Point Positioning corrections have to be applied and different approaches has been studied to increase the accuracy and reduce the convergence time. In this section, different corrections and approaches are presented.

4.4.1 GNSS Error Mitigation for PPP

Various positioning accuracies are obtainable depending on the employed GNSS error mitigation methods. With only one single receiver, PPP has to separately consider all GPS errors in order to obtain comparable positioning accuracy as DGNSS [29].

GNSS errors and their respective mitigations are presented in three groups: errors related to satellites, errors in the propagation of the signals and errors related to the receiver.

Satellite related errors

The satellite-related errors include the GNSS satellite orbit and clock errors, the relativistic effect, the satellite antenna phase centre offset and variation, and the phase wind-up effect [29].

- Satellite orbit and clock errors

As mentioned before, errors in clocks and satellite orbits are the most limiting accuracy factors in a single receiver positioning. To achieve PPP, good corrections need to be applied.

There are two types of widely used GPS satellite orbit and clock corrections: GNSS broadcast ephemeris and IGS precise products. GNSS broadcast ephemeris has not enough accuracy, as they are based on orbit predictions.

IGS releases three satellite orbit and clock products: IGS final, IGS rapid, and IGS ultra-rapid products [29]. However, these products are made available to the user with significant lag times or latencies, ranging from 3 hours for Ultra Rapid, 17 hours for Rapid, and days for Final products [23], so the use of them in real-time applications is not possible. In recent years, the accuracy of IGS orbit and clock products has improved drastically: the orbit accuracy of the predicted IGU products (IGS ultra-rapid products, which are the basis for the real-time products) is better than 5 cm, and the real-time estimated and short-term predicted clock accuracies are approximately 0.1 ns [23].

IGS had developed real-time infrastructure and processes, launched of the IGS Real-Time Service (IGS-RTS) in the second half of 2012 [4]. The Real-time Service (RTS), the IGS extends its capability to support applications requiring real-time access to IGS products. RTS is a GNSS orbit and clock correction service that enables precise point positioning and related applications. RTS is based on the IGS global infrastructure of network stations, data centres and analysis centres that provide world standard high-precision GNSS data products. The RTS is currently offered as a GPS-only operational service [17].

- Relativistic error

The relativistic effect occurs for two objects with different motions and gravity potential. The frequency drift due to relativistic effects which comprises of a constant

offset and a periodical variation.

The constant frequency offset can be calibrated by satellite clock manufacturers before the satellite launch. The periodical variation is caused by the satellite orbit eccentricity, that is the no completely circular orbit of satellites. This effect can be modelled by the expression:

$$d_{rel} = -\frac{2v_{sat} x_{sat}}{c^2} \quad (4.26)$$

where v_{sat} is the satellite velocity, x_{sat} is the satellite position and c is the speed of light in vacuum.

The correction in Equation (4.26) is directly applied in the satellite's clock term.

- Satellite antenna phase centre offset and variation

IGS orbit products refers to the satellite mass centre, while GNSS measurements are made to the satellite antenna phase centre, that are not exactly the same point. Therefore, satellite antenna phase centre corrections must be taken into account. IGS provides satellite antenna phase centre corrections files. The most used are called `igs08.atx`, but there are other newer files (such as `igs14_www.atx`) [16].

- Phase wind-up effect

For a receiver with fixed coordinates, the wind-up is due to the satellite orbital motion. As the satellite moves along its orbital path it must perform a rotation to keep its solar panels pointing to the sun direction in order to obtain the maximum energy while the satellite antenna keeps pointing to the earth's centre. This rotation causes a phase variation that the receiver misunderstands as a range variation [8].

The phase wind-up effect can be eliminated in double-difference applications, or absorbed into the receiver clock for kinematic applications. However, this effect cannot be removed and must be calibrated for un-differenced static point positioning applications.

The correction to this effect is given by [25]:

$$\Delta\varphi = \text{sign}(\zeta) \cos^{-1} \left(\frac{\vec{D}' \cdot \vec{D}}{|\vec{D}'| |\vec{D}|} \right) \quad (4.27)$$

where $\zeta = e^{loc} \left(\vec{D}' \cdot \vec{D} \right)$, e^{loc} is the satellite to receiver line-of-sight unit vector.

The \vec{D}' , \vec{D} are the effective dipole vectors of the satellite and receiver computed based on the unit vectors of the satellite's body-fixed b-frame (e_x^b, e_y^b, e_z^b) and the receiver's local level or local geodetic ll-frame East, North, Up (ENU) as follows:

$$\vec{D}' = e_x^b - e^{loc}(e^{loc} e_x^b) - e^{loc} x e_y^b \quad (4.28)$$

$$\vec{D} = e_N^{ll} - e^{loc}(e^{loc} e_N^{ll}) - e^{loc} x e_E^{ll} \quad (4.29)$$

Propagation errors

- Tropospheric effect

Troposphere, or neutral atmosphere, contains atomic and molecular constituents which can delay and bend radio signals. When compared to the delay effect, the signal bending effect is very small.

There exist different formulations to model the tropospheric delay. Usually the effect is divided into a wet and a dry component. The tropospheric hydrostatic delay can be calculated by, for example, the Saastamoinen model, based on the surface pressure, the latitude and the geodetic height.

The wet delay can be estimated also with the Saastamoinen model, from the temperature and water vapour pressure at the user location, but it is difficult to obtain precisely a model, so it is usually estimated in PPP with the other unknown parameters.

The precise a-priori troposphere hydrostatic delay is only available with precise meteorological data. In order to overcome this limitation, an empirical global pressure and temperature model was proposed and recommended as the current IGS and IERS standard, so it can be used as a model without meteorological data [29].

- Ionospheric effect

The ionosphere error is the biggest error related to the propagation of the signal.

Usually, for PPP applications the combination iono-free (see section 4.3.1) is used, so the majority of the effect is eliminated, there still some effect.

Also there is the possibility to use the original equations, so the ionosphere effect has to be corrected. To correct this effect, there exist also models that allow this

estimation. The content of electrons is needed in these corrections, so again a good model of the atmosphere is needed.

Receiver related errors

The receiver-related errors include the receiver antenna phase centre offset and variation, the sagnac effect, the receiver clock error, the multipath and the observation noise. All these errors can be classified as observation-dependent.

- Receiver antenna phase centre offset and variation

The receiver's antennas, as it occurs in case of satellite's antennas, needs to be corrected, as the phase centre is not the same as the mass centre. IGS ANTEX files for receiver's antennas are also available. In some high accuracy applications, the receiver's antennas are individually calibrated to obtain specific corrections.

- Sagnac effect

In general, GNSS satellite orbit is determined in an Earth-centre inertial reference frame, while the user location is required on the Earth-centre Earth-fixed reference frame [29]. Due to that, the earth rotation during the signal transmitting period prevents the clock synchronization for the satellite and receiver. This time synchronization issue, called the sagnac effect, must be calibrated for high precision applications.

According to [2] the correction to this effect can be done using the expression:

$$d_{sag} = \frac{v dr}{c^2} \quad (4.30)$$

where v is the receiver velocity in Earth-centre inertial reference frame, dr is the vector increment of path in the direction of signal propagation and c is the speed of light.

- Receiver clock error

The receiver clock error is defined as the difference between the receiver nominal time and the reference GNSS time. For PPP applications, the receiver clock error should be estimated along with other unknown parameters at every epoch.

- Receiver site displacement

The obtained coordinate solution by GNSS of the receiver is not consistent with the commonly used reference frames such as IGS08 or ITRF2008. This is because some periodic receiver site displacements, including the solid earth tides, the polar tides, the ocean loading and the effect of earth rotation parameters, affect receiver coordinates.

For the mentioned effects, there exists corrections that need to be applied in precise GNSS positioning applications. For each effect different corrections are applied, based on several factors such as gravitational parameters, mean pole coordinates, etc.

To apply these corrections, there exist different models, such as global ocean tide models or gravitational models, that allow the obtention of the implicated values in each correction for the user's approximate position.

Figure 7 shows the above mentioned errors and their corresponding mitigations as a summary of this section.

	Error source	Mitigation methods
Satellite - related	Satellite and orbit clock	Orbit/clock products (IGS)
	Relativistic effect	Model correction
	Satellite antenna phase center	ANTEX files
	Phase wind-up	Model correction
Propagation - related	Troposphere	Troposphere modeling
	Ionosphere	Ionosphere modeling Lineal combinations (iono-free)
Receiver - related	Receiver antenna phase center	ANTEX files
	Signal effect	Model correction
	Receiver clock error	Estimation for each epoch
	Receiver displacements	Model corrections

Figure 7: GNSS error mitigation (adapted from [29])

4.4.2 Ambiguity resolution

In the traditional PPP model, based on ionosphere-free combination, the ambiguity parameter estimated cannot be resolved to the integer value. In fact, the estimated ambiguity parameter is a combination of the integer ambiguity, the receiver biases and the satellite biases, so the integer property of the ambiguity parameter is lost [30].

The fact that losing the integer property causes two negative effects [29]. First, it takes a long time to achieve the convergence of ambiguity parameters. If the ambiguities could be isolated and estimated as integer values then, in principle, their integer nature represents more information that can be exploited to accelerate convergence [7].

The second effect is that, the fractional ambiguity part, defined as the difference between real-valued and integer ambiguities, still remains even after the ambiguity convergence. Furthermore, the fractional ambiguity part degrades accuracies of other parameters such as the coordinates and the troposphere delay.

For these reasons, several PPP integer ambiguity resolution methods have been developed and implemented in recent years, such as "Uncalibrated Hardware Delays" method and the "Integer-Recovery-Clocks" or "Decoupled Clock Model". The above-mentioned PPP-AR (Ambiguity Resolution) methods vary in terms of the strategies used to separate the hardware delays from integer ambiguities. [6] but these methods provide equivalent positioning solution and precision once the phase ambiguities are correctly resolved to their integer values [30].

Isolating the phase ambiguities as integer values in PPP does not by itself permit rapid ambiguity resolution. Even fixing the ambiguities as an integer, the convergence period of standard float-PPP processing remains. What is even more problematic is that this convergence process has to be repeatedly applied whenever satellite tracking loss occurs [6], so this approach need to continue in development to achieve real-time PPP applications.

4.4.3 Quasi Ionosphere Free (QIF) Algorithm

Alternatively to the Ionosphere-free linear combination, Quasi Ionosphere Free (QIF) Algorithm of Bernese software package (Bernese) can be used. The main characteristic of this algorithm is to add ionospheric pseudo-observables (artificial observation) to the observation model [14].

The quasi-ionosphere-free (QIF) strategy is used to solve ambiguities of baselines over several hundred kilometers long. The criterion used in the QIF strategy is to minimize the difference between the real-valued and integer ionosphere-free biases (IFBs).

In the QIF algorithm, the L1 and L2 ambiguities are fixed as pair. Firstly, the IFB standard deviations of all pairs of the original float ambiguities and the newly formed float ambiguities with changed reference satellite are computed based on the variance-covariance matrix of float ambiguities estimates. Then the ambiguity fixing process

starts from the ambiguity pair with the smallest IFB standard deviation. Its integer candidates are defined by the integer candidates of L1 and widelane ambiguities. The pair of integer candidates with the smallest difference between the real-valued and integer IFBs is accepted as the optimal solution, as long as such difference is smaller than the user- defined maximum value. Once such a pair of integer candidates is accepted, the entire sequential LS adjustment and the procedure described above are repeated so that all or some of the ambiguity pairs are fixed [22]

The algorithm definition can be read in Bernese manual [28]. It begins with the ionosphere-free linear combination. The initial least-squares adjustment using both frequencies L_1 and L_2 gives real-valued ambiguity estimates b_1 and b_2 and we may compute the corresponding ionosphere-free bias \tilde{B}_3 as:

$$\tilde{B}_3 = \frac{c}{f_1^2 - f_2^2} (f_1 b_1 - f_2 b_2) \quad (4.31)$$

This bias may be expressed in narrow-lane cycles (one cycle corresponding to a wavelength of $\lambda_3 = c/(f_1 + f_2)$).

$$\tilde{b}_3 = \frac{\tilde{B}_3}{\lambda_3} = \frac{f_1}{f_1 - f_2} / b_1 - \frac{f_2}{f_1 - f_2} / b_2 = \beta_1 b_1 + \beta_2 b_2 \quad (4.32)$$

Denoting the resolved integer ambiguity values by n_{1p} and n_{2q} and introducing the associated L_3 bias:

$$b_{3pq} = \beta_1 n_{1p} + \beta_2 n_{2q} \quad (4.33)$$

we may use the difference between the real valued and integer L_3 bias as a criterion for the selection of the “best” pair of integers n_{1p}, n_{2q} . However, many pairs give differences of the same (small) order of magnitude. These pairs lie on a narrow band in the space. The band width is essentially given by the RMS of the bias \tilde{b}_3 . A unique solution only results if it is possible to limit the search range.

For baselines longer than about 10 km separate processing of the two frequencies L1 and L2 does not result in sufficiently good initial real valued estimates b_1 and b_2 due to the influence of the ionospheric refraction. To solve that, two kind of models can be used to reduce the ionospheric biases.

- **Satellite and Epoch Specific Ionosphere Estimation.** Estimating these parameters without any a priori constraints would be equivalent to processing the ionosphere-free linear combination. If we want to resolve the integer ambiguities it is necessary to constrain these parameters to within a few decimeters. This constraining may be achieved by introducing an artificial observation for each epoch with a non-zero a priori weight (see more at [28]).

- **Deterministic Model.** Single-layer model developing the electron content within a layer of infinitesimal thickness at a height of about 450 km above the surface of the Earth into a series of harmonical coefficients in latitude and hour angle of the Sun.

4.4.4 PPP-RTK

Real Time Kinematics (RTK) is a differential GNSS technique which provides high positioning accuracy. The technique is based on the use of carrier measurements and the transmission of SSR corrections to the rover, so that the main errors that drive the stand-alone positioning cancel out [8]. RTK can be done with a reference station set up by the user or with a virtual reference station when connected to a GNSS network.

On conventional RTK, the errors for all sources are observed and provided together to the rover as range corrections, for each combination of station, satellite, frequency and signal.

As seen in previous sections, real-time PPP is limited by the convergence period and the need of corrections. To solve this, the method known as PPP-RTK has been developed in recent years. PPP-RTK is a synthesis of the positive characteristics of PPP and network [6], so it takes advantage of the GNSS network to have corrections to apply to the PPP observations.

Network- RTK solutions can be generalised in two ways. The mostly commonly used technique is the use of Observation Space Representation (OSR) such as the Virtual Reference Station (VRS) techniques (DGNSS) and the other is State Space Representation (SSR- RTK).

In State-Space-Modelling, all relevant physical effects are represented by a mathematical model with parameters that are estimated in real time using the network observations. In SSR corrections, the different error source (satellite clocks, satellite orbits, satellite signal biases, ionospheric delay, and tropospheric delay) are modelled and distributed separately (Figure 8).

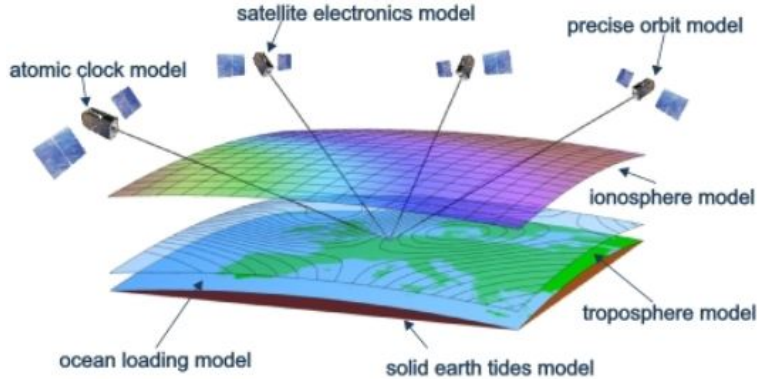


Figure 8: SSR corrections [12]

Improved modelling of the spatial behaviour of the physical effects enables better error estimation between reference stations. This allows for an increased reference station distance up to 200 km [12].

PPP is a positioning technique that can truly offer global solutions without the requirements of local or regional reference networks; whereas RTK will continue to dominate regional positioning especially when a dense regional GNSS infrastructure has already been established. Integration of these two techniques would lead to improved position accuracy and convergence time but the performance is now dependent on the extent and density of the reference networks, which is critical for the provision of accurate atmospheric information to aid rapid ambiguity fixing [6].

4.5 GNSS processing software engines

In order to process GNSS data, there are different software available, with different possibilities and characteristics. Depending on the need of the user, one or another software would be most adequate.

To standard applications, commercial software is often used, as they are easy to use. For example, LEICA Geo Office. In works that requires a bigger control of the processing parameters, scientific software is usually applied. An example of this kind of software could be BERNESE, developed by the Astronomical Institute of the University of Bern (AIUB).

Apart from these two kinds of software, there also exist software engines, that allow using GNSS processing algorithms from user's developed code. That means that user can develop his own application and use that software engines inside of it. In GOCA project, these software engines are required, so they can be applied directly inside the

code of the project. GOCA project uses both RTKLIB and WA software as processing engines, so both are introduced in this section.

RTKLIB

RTKLIB is an open source program package for standard and precise positioning with GNSS. RTKLIB consists of a portable program library and several APs (application programs) utilizing the library. The main features of RTKLIB are [1]:

- It supports standard and precise positioning algorithms with GPS, GLONASS, Galileo, QZSS, BeiDou and SBAS.
- It supports various positioning modes with GNSS for both real-time and post-processing (Single, DGNSS, kinematic, static...).
- It supports many standard formats and protocols for GNSS (RINEX 2.X, RINEX 3.X, NTRIP, RTCM...).
- It supports several GNSS receivers' proprietary messages (NovAtel, u-blox...).
- It supports external communication (NTRIP, FTP/HTTP...).
- It provides many library functions and APIs (application program interfaces), such as Satellite and navigation system functions and matrix and vector functions.
- It includes the following GUI (graphical user interface), that allows using it by itself and CUI (command-line user interface), that is the option used in GOCA because it can be integrated by code.
- All of the executable binary APs for Windows are included in the package as well as whole source programs of the library and the Aps.

The RTKLIB software package is distributed under the following BSD 2-clause license and additional two exclusive clauses. Users are permitted to develop, produce or sell their own non-commercial or commercial products utilizing, linking or including RTKLIB as long as they comply with the license.

WaSoft

WaSoft is a software engine that have several modules to different GNSS processing developed by Lambert Wanninger. These modules are [33]:

- WaRINo+WaRINn: Editing and manipulating of GNSS data (GPS, GLONASS, Galileo, BeiDou...).

- Wa2: baseline processing engine.
- WaV2: virtual GNSS reference stations.
- Wa2Ant+CCANTEX: antenna calibration.
- WaPPP: PPP processing engine.
- WaPNet: Highly precise GNSS network processing.
- WaPNet+Mul: GNSS carrier phase multipath detection and localization

The modules that concern GOCA project are Wa2 and WaPPP.

This software engine works by command line, where different parameters are defined and then a .exe file is executed, so it can be integrated in other code as in case of GOCA project.

Wa2

The baseline processing engine Wa2 is designed for adding precise post-processing GNSS positioning capabilities to application software. Wa2 computes baseline coordinates based on the observations in RINEX-format of two GNSS receivers.

Wa2 includes static and kinematic positioning based on sophisticated ambiguity search algorithms. Fast static processing is applicable to baselines with lengths of up to several kilometres or Virtual Reference Station (VRS) observations. It supports single and dual frequency processing. Antenna phase centre corrections can be applied to the observations. Wa2 processes precise ephemerides and estimates ionospheric and tropospheric models in the case of long baseline (up to a few 100 km). Then, it also uses ionosphere-resistant ambiguity resolution algorithms to obtain a high ambiguity fixing rate.

WaPPP

The Precise Point Positioning processing engine WaPPP is designed for adding precise GNSS single station positioning capabilities to application software. WaPPP computes coordinates based on GNSS observations in RINEX-format. It is useful for obtaining sub-metre, decimetre, or centimetre level accuracies. Highest accuracies require long-term of continuous dual-frequency carrier-phase observations. Precise satellite orbits and clock corrections are needed as well. They are available for free from several data centres via internet connection. The geodetic datum of the position solution is determined by the satellite orbit data.

WaPPP expects corrections for satellite antennas and the user antenna. In general PPP is based on dual-frequency observations, but WaPPP is also able to process single-frequency observations. WaPPP weights the different observations types (code and carrier phase, various GNSS) according to a variance component estimation. The position results are obtained using a robust estimation algorithm.

4.6 GNSS Low Cost Receiver: u-blox ZED-F9P

There exists a high number of GNSS receivers in the market, whose characteristics and specifications differ from each other so they are adequate to different applications. In the context of the Master Thesis presented at this document, the low cost receiver u-blox ZED-F9P will be used, so in this section its main characteristics are presented.

The ZED-F9P positioning module features the u-blox F9 receiver platform, which provides multi- band GNSS to high volume industrial applications. The ZED-F9P has integrated u-blox multi-band RTK technology for centimetre level accuracy [32].

Its main characteristics are:

- Accuracy on time pulse signal: RMS 30ns.
- Frequency of time pulse signal: from 0.25 Hz to 10 MHz.
- Velocity accuracy: 0.05 m/s.
- Dynamic heading accuracy: 0.3 deg.

With regard to supported GNSS constellations, ZED-F9P can receive all four major GNSS constellations (GPS, GLONASS, Galileo and BeiDou) plus QZSS satellites. It supports also different protocols as UBX, NMEA and RTCM 3.3.

Development

5. Wa Software updating study

GOCA software, in the module GNSS control, uses Wa1 as software engine to process baselines. There is already available the second version of this software, called Wa2. As a previous step from the implementation of PPP, this new version is going to be studied, in order to analyse what changes should be done in GOCA GNSS control if the new version wants to be used.

So, a comparison between both versions has been done in order to detect changes between both versions. This comparison is shown in this section, and it is organised as follows: first of all, a theoretical comparison with regard both manuals has been done in order to detect relevant changes; secondly, a simple example has been executed using both versions with the command line to analyse if the obtained result is the same or not in both versions.

5.1 Theoretical comparison

The first comparison that has been made is regarding both manuals ([34],[35]). Generally, the software works in the same way: the software can be controlled by command line options, by options written to the option file wa2.ini, or in mixed mode. In case of GOCA GNSS control, the configuration is done using the .ini file.

The changes in the configuration options of both versions that can be seen in both manuals are presented here.

Output

There exist some differences between the options of the output between the first and in the second version.

- Solution format. In the second version (wa2) more options have been added to the output file format, as EMDS (Einheitliche Messdatenschnittstelle) and Carlson (this format is a JSON (JavaScript Object Notation) format. This output is for Carlson's SurveyGNSS).
- Keep file. With this option some information can be stored in a file. In case of version 1 only the processed observations can be stored, while in the version 2 also ionospheric corrections can be stored in a file.

Processing parameters

In processing parameters options there are also some differences between versions.

- In version 2, there is an option to read a table of a priori GLONASS inter-frequency biases from file. This option is not included in the version 1.
- Approximate value identical for L1 and L2 or two different values for L1 and L2 of GLONASS inter-frequency bias differences between adjacent channels. This option has little differences between both versions. In the first one, this has three different options: *GloLinFreqCorr*, that works for both stations; *GloLinFreqCorr1*, that is only for station 1 and *GloLinFreqCorr2* that is for the second station. In case of version 2, only the option that includes both stations are available and the name of parameter has changed into *GloFreqCorr*.
- Delete observation to satellite. In this option there are some differences. In the version 1, with the option *DeleteSV* only the number of satellite has to be included while in the version 2 also the constellation of the satellite has to be specified. In version 2 additional parameters to control the exclusion of satellites have been included: *DeleteSVO* to delete observations to satellites with orbit type, delete GLONASS observations according to their frequency number and also the opposite option (use observations to satellite) is included in second version (both to GNSS (*OnlySV*) and to orbit satellites (*OnlySVO*)).
- Phase single frequency. In version 1, you can force single-frequency (code and carrier-phase L1) processing. In version 2, you can do it also but a new option is included: you can force single-frequency processing using ionosphere-free phase-code combination.
- Code single frequency. As in the previous point, you can force single-frequency code only (C1) processing. In the version 2 there is also the option of activate it only for some data.

Kinematic processing parameters

In this section, a new parameter has been included in version 2. This option is Stop and go processing (*StopAndGo*). In case of Stop & Go baseline processing, the coordinates of several static stations are output to *wa2sol.txt*, while in case of kinematic baseline processing, coordinates are written to the position file *wa2pos.txt*.

Other options

Few additional parameters are included in version 2 with respect version 1:

- Read further options from file (with the option @).

- The software internal standard table of GLONASS inter-frequency biases can be written to file.

Figure 9 shows these changes, as a resume of the above described.

	Parameter	Wa1	Wa2
Output	File format	txt, XML, bal	Wa1 + EMDS and Carlson
	Keep file	Processed observations	Wa1 + ionospheric corrections
Processing parameters	Read GLONASS bias from table	-	Implemented
	GLONASS bias	Options: both stations, station 1, station 2	Only both stations option. Changed command
	Delete satellite observation	Indicate satellite's number	Indicate satellite's constellation and number. Also possible with orbit satellites.
	Phase single frequency	Force single-frequency (phase)	Wa1 + ionosphere-free
	Code single frequency	Force single-frequency (code)	Can be applied for some data
Kinematic processing parameters	Processing options	-	New processing option: Stop & Go
Other	Read further options from file	-	Implemented
	Write GLONASS bias to file	-	Implemented

Figure 9: Comparative between Wa1 and Wa2

5.2 Example comparison

In order to compare the obtained results with both versions and also check if there are any change in the result file, a simple test with same data and same options has been made using both versions of Wa software.

In this example, two GNSS stations of the Spanish Permanent GNSS Stations Network has been used (called UTIE and VCIA), with observations for one day (01/02/2016).

The options that have been defined are:

- Input data: file 1 corresponding to the station UTIE and file 2 corresponding to the station VCIA.
- Use of antenna corrections (igs08.atx file).
- Use of precise ephemerids (.sp3 file).

- Write extended log file to see possible differences between versions.

The rest of options has been set as default.

Both the resulting log file and solution file are going to be analysed in order to detect differences between versions.

Solution file

The first part of the solution file, the one that includes basic information is the same in both results with the exception of standard deviation of unit weight and the standard deviation of the single observation [m], that is slightly different (Figure 10).

Wa1		Wa2	
Reference_station:	utie UTIE	Reference_station:	utie UTIE
Rover_station:	vcia VCIA	Rover_station:	vcia VCIA
DOY+Session:	032 0	DOY+Session:	032 0
SOLUTION QUALITY		SOLUTION QUALITY	
Solution_type:	FixedL0	Solution_type:	FixedL0
Solution_quality:	high	Solution_quality:	high
S0:	0.6	S0:	0.5
S0 m:	0.0055	S0 m:	0.0053
Ambfix_‰:	100.0	Ambfix_‰:	100.0
Re-weighted_obs_‰:	0.0	Re-weighted_obs_‰:	0.0
SV_min/ave/max:	5 7.7	SV_min/ave/max:	5 7.7
PDOP_min/ave/max:	0.9	PDOP_min/ave/max:	0.9

Figure 10: General information Wa1 and Wa2

In the observations parameter section, we find as a little difference, the default value of the parameter elevation minimum in degrees, that is 0 in case of Wa1 and 10 in Wa2. We can also see a new parameter in the version 2, that is selected frequencies (Figure 11).

Wa1		Wa2	
OBSERVATION AND PROCESSING PARAMETER		OBSERVATION AND PROCESSING PARAMETER	
Processing type:	static	Processing type:	static
Start_time:	1882 864	Start_time:	1882 864
End_time:	1882 1719	End_time:	1882 1719
Duration_minutes:	1425.0	Duration_minutes:	1424.50
Duration_hours:	23.7	Duration_hours:	23.7
Interval_s:	30.00	Interval_s:	30.00
Distance_km:	76	Distance_km:	76
Elevation_mask_deg:	10.0	Elevation mask deg:	10.0
Elevation_min_deg:	0.0	Elevation min deg:	10.0
PreciseEph:	Yes	PreciseEph:	Yes
APCorrection:	Yes	APCorrection:	Yes
APCfileRef:	igs08.atx	APCfileRef:	igs08.atx
APCfileRov:	igs08.atx	APCfileRov:	igs08.atx
Tropo_parameters:	Abs	Tropo_parameters:	Abs
Number_frequencies:	2	Number_frequencies:	2
Satellite_systems:	GPS	Satellite systems:	GPS
		Selected_frequencies:	GPS 1-

Figure 11: Parameters Wa1 and Wa2

The part containing information of the stations is completely equal in both versions (Figure 12).

```

REFERENCE STATION
Ref_Name_short:      utie
Ref_Name_long:       UTIE
Ref_Name_file:       utie0320.16o
Ref_Receiver_type:   TRIMBLE 5700
Ref_Receiver_number: 0220312532
Ref_Receiver_version: NP 1.30 / SP 1.22
Ref_Antenna_type:    TRM41249.00    TZGD
Ref_Antenna_number:
Ref_Antenna_height_m: 0.0750
Ref_Coordinates_XYZ: 4922873.2490  -103857.7308  4041693.7280
Ref_Coordinates_LLH: 39.568678616  -1.208588307  799.6968
Ref_Coordinates_UTM: 30653876.8971  4381419.1725  799.6968

ROVER STATION
Rov_Name_short:      vcia
Rov_Name_long:       VCIA
Rov_Name_file:       vcia0320.16o
Rov_Receiver_type:   LEICA GRX1200GGPRO
Rov_Receiver_number: 356632
Rov_Receiver_version: 9.20/3.823
Rov_Antenna_type:    LEIAX1202GG    NONE
Rov_Antenna_number:
Rov_Antenna_height_m: 0.0800
Rov_Coordinates_XYZ: 4932702.9563  -29607.7824  4029833.0448
Rov_Coordinates_LLH: 39.435707050  -0.343904876  62.9641
Rov_Coordinates_UTM: 30728594.6697  4368496.3107  62.9641

```

Figure 12: Station information in Wa1 and Wa2

Finally, in the result of the baseline, small differences in the result can be pointed out. Additionally, version 2 includes the baseline azimuth, that was not shown in the result of version 1 (Figure 13).

Wa1			
BASELINE			
Baseline_Coo_XYZ:	9829.7073	74249.9484	-11860.6832
Baseline_length_m:	75831.0873		
Baseline_Dheight_m:	-736.7327		
Stddev_XYZ_m:	0.0005	0.0002	0.0004
Corrcoeff_XYZ:	-0.0665	-0.0907	0.9105
Stddev_NEH_m:	0.0001	0.0002	0.0006
Corrcoeff_NEH:	-0.0558	-0.0607	-0.0594
Stddev_UTM_m:	0.0002	0.0001	0.0006
Corrcoeff_UTM:	-0.0558	-0.0594	-0.0607
Wa2			
BASELINE			
Baseline_Coo_XYZ:	9829.7034	74249.9472	-11860.6866
Baseline_length_m:	75831.0860		
Baseline_azimuth_deg:	100.9523		
Baseline_Dheight_m:	-736.7378		
Stddev_XYZ_m:		0.0001	0.0004
Corrcoeff_XYZ:	-0.0669	-0.0919	0.9107
Stddev_NEH_m:	0.0001	0.0001	0.0006
Corrcoeff_NEH:	-0.0577	-0.0614	-0.0593
Stddev_UTM_m:	0.0001	0.0001	0.0006
Corrcoeff_UTM:	-0.0577	-0.0593	-0.0614

Figure 13: Baseline information in Wa1 and Wa2

So, the result file is almost the same in both versions. Few parameters has been added in the version 2. Little differences in the result must be due to changes and adjustments in the internal algorithms of the software.

Log file

A log file is also generated, where little differences can be also detected. In this case there are more differences than in the solution file, but the majority of them are not relevant to the functionality, and are mainly related to the way that information is presented.

First, there are a difference regarding the observations, as this is shown with more detail in version 2. In case of broadcast ephemerids, version 2 only shown the ones corresponding to the used constellation, while in the version 1 all constellations were shown even if only one is selected (then we have 0, as in this case that we have 0-GLONASS ephemerids).

Additional information is included in version two, as in case of the absolute position of the second station, where, apart from que information that appeared in version 1, also information about the number of epochs in which solution is based is shown.

In baseline pre-processing section of the log file, we can appreciate in version 1 a section including all the observations deleted, while in version 2 this section is divided into observations deleted due to not having broadcast ephemerids and the same with precise ephemerids, so is more detailed than in version 1.

In DGNSS solution, little differences in the result are obtained and also additional information is shown (Figure 14).

Wa1				
DGNSS (code) solution (iteration 2/2)				
obs/ell/unk/dfre	21878	2851	3	19024
obs GPS/GLONASS/SBAS	21878	0	0	
min/average/max # of SV	5	7.7	11	
min/average/max PDOP	0.9	1.3	4.2	
sigma0 (code) [m]	0.18			
selected solution: C1				
old coordinates	4932702.8910	-29607.8285	4029833.0416	
new coordinates	4932702.8909	-29607.8285	4029833.0416	
coo. difference	-0.0001	0.0000	-0.0001	
code-carrier ratio set to 100				
Wa2				
DGNSS (code) solution (iteration 2/2)				
obs/ell/unk/dfre	21870	2850	3	19017
obs G/R/E/C/S/J/I	21870	-	-	-
min/average/max # of SV	5	7.7	11	
min/average/max PDOP	0.9	1.3	4.2	
code std. G/R/E/C/S/J/I	1.0	-	-	-
sigma0 (code) [m]	0.18			
selected solution C1st				
old coordinates	4932702.8486	-29607.8292	4029833.0075	
new coordinates	4932702.8485	-29607.8292	4029833.0075	
coo. difference	0.0000	0.0000	0.0000	
code-carrier ratio set to 100				

Figure 14: DGNSS solution in Wa1 and Wa2

Another difference is that in the version 1 more cycle-slip were detected and there are also some differences in cycle-slip fixing. That kind of difference could be also due to changes in internal algorithms.

The part regarding ionospheric model is different in both versions. Both the way of presenting it and even the parameters differs from one version to another (Figure 15). Tropospheric delays also present differences in both versions.

Wa1								
Parameters of single layer ionospheric models								
#	VEC	LAT	HRS	LAT2	HRS2	L/H	DFre	s0
1	5.012	-0.239	-0.161	0.000	0.267	0.000	3000	0.648
2	4.006	-0.118	-1.075	0.000	0.408	0.000	2993	0.748
3	7.272	-0.228	3.305	0.000	0.284	0.000	2570	1.249
4	18.283	-0.141	2.716	0.000	-0.797	0.000	2480	1.238
5	15.146	-0.557	-1.624	0.000	0.517	0.000	2767	1.008
6	12.148	-0.276	-1.255	0.000	-0.490	0.000	3274	0.894
7	5.661	-0.291	-0.771	0.000	0.328	0.000	2560	0.607
8	3.625	-0.168	0.423	0.000	0.152	0.000	2081	0.460

Wa2								
Parameters of single layer ionospheric models								
used amb. blocks, obs: 47 21813								
unused amb. blocks, obs: 12 57 (= 0.26%)								
	Lt0	VEC	LAT	HRS	LAT2	HRS2	L/H	
	DFre	s0						
IonoVEC	32.040	4.92	-0.242	-0.305	0.000	0.340	0.000	
2729	0.61							
IonoVEC	32.170	4.44	-0.107	-1.201	0.000	0.299	0.000	
3076	0.79							
IonoVEC	32.301	6.59	-0.223	3.148	0.000	0.651	0.000	
2617	1.25							
IonoVEC	32.431	17.88	-0.146	2.839	0.000	-0.600	0.000	
2516	1.26							
IonoVEC	32.562	15.24	-0.552	-1.660	0.000	0.488	0.000	
2815	1.02							
IonoVEC	32.692	12.13	-0.277	-1.261	0.000	-0.480	0.000	
3327	0.90							
IonoVEC	32.823	5.52	-0.293	-0.754	0.000	0.293	0.000	
2596	0.62							
IonoVEC	32.953	3.73	-0.166	0.447	0.000	0.124	0.000	
1998	0.43							

Figure 15: Ionospheric model in Wa1 and Wa2

As in version 2 several internal algorithms could have been changed, the obtained final result (that is the one presented in the solution file) is slightly different in both versions even using the same data and input parameters.

6. GKA files

The output format of GOCA GNSS control is GKA files. These files are the ones containing the information obtained by the module GOCA GNSS Control that will be use in the main module of GOCA afterwards.

Currently, version 4.0 is used. It includes different GKA blocks depending on the data that is contained on it. In GOCA GNSS Control, block GOKA13 is used, which contains GPS-Session (including uncorrelated Baseline). A new block (GOKA14) has to be designed under the context of this master thesis to contain the PPP information in order to use it as further observation type in GOCA main module.

General characteristics for GKA version 4.0 files are:

- GKA files are saved daily. From the beginning of the data recording a file for every day is expected.
- The file name is built as: *yymmdd.gka*, where *yy* refers to the year, *mm* to the month and *dd* to the day.
- The end of observations on a file is represented by the string "Ende" or "End".
- A comment starts with ";" and ends with a line end.
- Blank lines are accepted.

The different block that are possible currently in GKA are:

- GOKA10: Approximate coordinates for the GOCA network adjustment.
- GOKA11: LPS-Total station data.
- GOKA12: LPS-Height differences (level, barometric level...).
- GOKA 13: GPS-Session (including uncorrelated Baseline).

In this section, both the already existing GKA 13 and the proposed new GKA 14 format are exposed.

6.1 GKA 13 for baselines

In GOCA GNSS Control, GKA13 is used. It contains GPS sessions, including the baselines. The information that it contains is the following one:

```
# GOKA13; GPS session (special case baseline included)

BASE, Descr, X_BASE, Y_BASE, Z_BASE, h_BASE, N_Rover, N_Sess_Type, I_EX_BASE,
EX_BASE, EY_BASE, EZ_BASE ; N_Rover lines
ROV, Descr, GPS_wv, GPS_d, GPS_ss, X_ROV, Y_ROV, Z_ROV, h_ROV, I_Status,
I_Stat_Typ, Stat_String, I_ex_ROV, EX_ROV, EY_ROV, EZ_ROV
; 1 line
Sigma, QXX_MAT_WERTE

# END13
```

Where the different values are:

- BASE = Point name of the GPS reference / base station.
- Descr = Description.
- X_BASE = ITRF-based geocentric Cartesian X coordinate [m].
- Y_BASE = ITRF-based geocentric Cartesian Y-coordinate [m].
- Z_BASE = ITRF-based geocentric Cartesian Z-coordinate [m] .
- h_BASE = Antenna height of the base receiver .
- N_Rover = Number of following GPS Rover .
- N_Sess_Typ = Correlation or not of baselines (0: mutually uncorrelated baselines, 1: fully correlated session).

Optional parameters: the following eccentric data are only transferred if present (otherwise empty string)

- I_EX_BASE = Eccentricity (1: eccentricity in Cartesian geocentric ITRF coordinates, 2: eccentricity in geographic ITRF coordinates, 3: eccentricity in topocentric Cartesian coordinates (u, v, w)) .
- EX_BASE = Eccentric value 1 (X or B or u) .
- EY_BASE = Eccentric value 2 (Y or L or v) .

- EZ_BASE = Eccentric value 3 (Z or h or w) .

Single Rover line:

- ROV = Point number of the GPS reference / base station .
- Descr = Description .
- GPS_ww = GPS week of the observation time.
- GPS_d = GPS day of the week .
- GPS_ss = GPS second .
- X_ROV = ITRF-based geocentric Cartesian X coordinate [m] .
- Y_ROV = ITRF-based geocentric Cartesian Y-coordinate [m].
- Z_ROV = ITRF-based geocentric Cartesian Z-coordinate [m].
- h_ROV = Standard antenna height of the receiver.
- I_Status = 0: No further information on the evaluation of the baseline , N: Number of data fields in the Stat_String.
- I_Stat_Typ = evaluation mode (0: I_Status = 0 , 1: Trimble_RTK specifications, 2: Leica_RTK information, 3: Topcon_RTK information, 4: other types or evaluation modes).
- Stat_String = Status_String with N information (at the moment just accept value 0).

Optional parameters: the following eccentric data are only transferred if present (otherwise empty string)

- Lex_ROV = Coordinate system (1: Cartesian geocentric ITRF coordinates, 2: geographic ITRF coordinates, 3: topocentric Cartesian system).
- EX_ROV = Eccentric value 1 (X or B or u).
- EY_ROV = Eccentric value 2 (Y or L or v).
- EZ_ROV = Eccentric value 3 (Z or h or w).

Variance factor and cofactor matrix:

- Sigma = Variance factor for subsequent cofactor matrix.

- QXX_MAT= (6 * N_ROV Q_{xx} matrix values for NSESS_TYP=0, 4.5 * N_ROV * N_ROV + 1.5 * N_ROV) Q_{xx} matrix values for NSESS_TYP=1. The QXX_MAT values represent always the upper trip matrix of the cofactor matrix.

For the covariance matrix C_{xx} : $C_{xx} = \text{Sigma} * \text{Sigma} * \text{QXX_MAT}$.

6.2 GKA 14 for PPP

The proposed new GKA block, corresponding to PPP observations, is exposed in this section.

GOKA14; GNSS PPP session

*NAME, Rec, Descr, GPS_ww, GPS_d, GPS_ss, X, Y, Z, h , I_Status, I_Stat_Typ, Stat_String, I_EX, EX, EY, EZ
; 1 line
Sigma, QXX_MAT*

END14

Where the different values are:

- NAME = Point name of the point.
- Rec = Receiver used in the observation.
- Descr = Description.
- GPS_ww = GPS week.
- GPS_d = GPS weekday.
- GPS_ss = GPS seconds.
- X = ITRF-based geocentric Cartesian X coordinate [m].
- Y = ITRF-based geocentric Cartesian Y-coordinate [m].
- Z = ITRF-based geocentric Cartesian Z-coordinate [m] .
- h = Antenna height of the receiver .
- I_Status = 0: No further information , N: Number of data fields in the Stat_String.

- I_Stat_Typ = evaluation mode (0: I_Status = 0 , 1: Trimble_RTK specifications, 2: Leica_RTK information, 3: Topcon_RTK information, 4: other types or evaluation modes).
- Stat_String = Status_String with N information (at the moment just accept value 0).

Optional parameters: the following eccentric data are only transferred if present (otherwise empty string)

- I_EX = Eccentricity (1: eccentricity in Cartesian geocentric ITRF coordinates, 2: eccentricity in geographic ITRF coordinates, 3: eccentricity in topocentric Cartesian coordinates (u, v, w)) .
- EX = Eccentric value 1 (X or B or u) .
- EY = Eccentric value 2 (Y or L or v) .
- EZ = Eccentric value 3 (Z or h or w) .

Variance factor and cofactor matrix:

- Sigma = Variance factor for subsequent cofactor matrix.
- QXX_MAT = Cofactor matrix. The QXX_MAT values represent always the upper trip matrix of the cofactor matrix.

For the covariance matrix Cxx : $Cxx = Sigma * Sigma * QXX_MAT$.

7. PPP software implementation

In the context of this master thesis, a PPP dialog-based software in C++ language has been developed in order to perform PPP processing. This software has been developed in two different phases.

The first one, was focussed on post-processing options, using both software engines RTKLIB and WaPPP. After that, also the real-time option was added, in order to connect with a sensor using TPC/IP connection and do real-time processing. Both versions of the software, the one with only post-processing and the one with both real-time and post-processing are available under the context of this master thesis, as the post-processing part has been tested deeper.

In this section, all aspects related with its implementation including the previous study of PPP parameters, the architecture of the developed PPP software and the use from a point of view if the user are explained. The explanation has been done according to the version including both processing modes (real-time and post-processing).

7.1 Study of processing parameters for PPP

A short introduction about both software engines WaPPP and RTKLIB was provided in section 4.5 as theoretical background. Now, in the context of PPP implementation both software engines have to be studied in more detail, in order to analyse the different options they provide for PPP processing.

In this section, main options for both processing engines are described, focussing on that parameters that are important for PPP processing, such as the corrections that can be applied.

7.1.1 Processing parameters in WaPPP

This software engine expects, by default, finding the following files in the working directory: antenna correction in ANTEX format, precise orbits in SP3 format, satellite clock corrections in CLK format and RINEX broadcast ephemeris for GLONASS if these observations has to be processed.

There exist different options for the PPP processing.

Orbits and clock corrections

Both orbit information and clock information can be provided, using only one file to both information (.sp3) or two different files (.sp3 and .clk). The main formats that can be provided are:

- SP3 format. The first Standard Product 3 format (SP3-a) was proposed in 1989, with the main purpose of exchanging satellite related data (orbit and clock information). The basic format of an SP3 file is a header, followed by a series of records containing the position and clock records for each satellite listed in the header. A second, optional, record contains the satellite velocity and clock correction rate-of-change. After this first format, SP3 version b, proposed in 1998 was defined to allow the combination of GPS orbits and GLONASS orbits. The current SP3 version is c, proposed in 2000 [8].
- RINEX clock format CLK (only for clocks). RINEX CLOCK format is based on RINEX format and it was developed with the purpose of having a common standard to exchange satellite and receiver clock offsets data [8].
- Broadcast ephemerids (mixed files, GPS, GLONASS, Galileo. . .).

Application of corrections

Antenna corrections can be applied from the same file to both receiver and satellites antennas or specify different files for each case. Correction tolerance can be also adjusted and antenna files can even not be used, but this will decrease the achieved accuracy.

P1C1 and P1P2 code biases corrections can be applied from files. These are Differential Code Bias (DCB). They are the systematic errors, or biases, between two GNSS code observations at the same or different frequencies. DCBs are required for code-based positioning of GNSS receivers, extracting ionosphere Total Electron Content (TEC), and other applications [24]. Manual 2 of this software indicates that the file used is `gnssbias.dcb`, from Bernese GNSS group, but is said in the manual that it is not more available at their webpage.

About the ionospheric corrections also different options can be chosen:

- Ionospheric corrections from a model in IONEX format can be applied to code and phase observations.
- Parameters of single-layer ionosphere model from dual-frequency observation data can be estimated and applied as corrections.

By default, tropospheric zenith delay and horizontal gradients are estimated, but these options can be disabled.

Other processing parameters

Elevation mask angle can be changed. The default value is 10 degrees.

Certain observations can be removed, entry constellations or specific satellites if needed.

According to the manual, WaPPP is able to read RINEX observation files of versions 2.X only. Observation files in RINEX 3.X must be converted to RINEX 2.X before being processed by WaPPP. This conversion can be done using WaRINo. If BeiDou or QZSS observations are to be processed, WaRINo must be used. Thus, a non-standard WaSoft-specific RINEX 2.X file including BeiDou and QZSS observations is produced. So, in this case, only RINEX 2.X version can be used and constellations allowed are GPS, GLONASS, Galileo and SBAS, because WaRINo is no available in the context of this master thesis.

Output options

Result can be obtained in different formats and also containing different information. This is not essential to PPP processing, so is not going to be detailed, but it has to be taken into account to analyse the obtained results.

7.1.2 Processing parameters in RTKLIB

RTKLIB has no specific module for PPP, so the PPP processing is done depending on the chosen options. In this document, only the options affecting PPP are going to be exposed.

RTKLIB allows both real time and post processing, using different executables for that purpose: RTKLIB NAVI for real-time and RTKLIB POST to post-processing

The main configuration options that affect PPP are shown in this section.

Orbits and clock corrections

This software engine also need to be provided with orbit and clock information. Formats that are allow for that are:

- SP3 format, as in case of WaPPP software.

- RINEX clock format CLK.
- RTSM SSR corrections. These corrections are explained in section 4.4.4. RTKLIB can use this kind of option in real time processing with a NTRIP connection and also can be used as input file in case of post processing mode. This option is only available for real-time processing.
- Broadcast ephemerids.

Application of corrections

Antenna correction can be applied. In this case, is it also possible to introduce different files for the receiver's antenna and the satellite's antennas.

DCB files can be also used (see explanation in WaPPP section). In this case, only one DCB file can be used.

There are different options for ionospheric corrections:

- OFF : Not apply ionospheric correction.
- Broadcast: Apply broadcast ionospheric model.
- SBAS: Apply SBAS ionospheric model.
- Iono-Free Linear Combination: Used with dual frequency measurements.
- IONEX TEC: Use IONEX TEC grid data in IONEX file.

Tropospheric corrections also have different options the user can choose:

- OFF: Not apply troposphere correction.
- Saastamoinen: Apply Saastamoinen model.
- SBAS: Apply SBAS tropospheric model.
- Estimate ZTD: Estimate ZTD (zenith total delay) parameters.
- Estimate ZTD+Grad: Estimate ZTD and horizontal gradient parameters.

With respect Earth Tides Correction, RTKLIB allows different options:

- OFF: Not apply earth tides correction.
- Solid: Apply solid earth tides correction.

- Solid/OTL: Apply solid earth tides, OTL (ocean tide loading) and pole tide corrections. If the option OTL is chosen, then BLQ and ERP files has to be introduced. BLQ file contains the corrections we have to apply to our coordinates due to the ocean tide loading. It is provided by the Onsala Space Observatory for free (<http://holt.oso.chalmers.se/loading/>). You have to send your station coordinates and you receive the corrections by email. ERP files contain information about the pole.

It is also possible to activate or not the option of applying the phase windup correction (see section 4.4.1).

Other processing parameters

Specific observations can be removed for the processing. Similar to WaPPP, satellites to exclude can be indicated using the constellation or the specific number of satellite. As a difference, RTKLIB also allows the user to exclude those satellites in eclipse, as they can degrade the PPP solution.

Elevation mask angle can be also set by user. By default, this value is 15 degrees.

The geoid model can be indicated and even the file containing it can be loaded and used.

The satellite's constellations that can be used in RTKLIB software are GPS, GLONASS, Galileo, SBAS, QZSS and BeiDou. It allows using RINEX 2.X and also RINEX 3.X versions.

Output options

Different options to configure the result of the processing can be established. Again it is not essential for PPP, so it is not described now, despite it has to be studied to analyse the results.

7.1.3 Comparison between WaPPP and RTKLIB for PPP

In the above sections, main parameters that affect PPP processing in both software engines has been shown. In general, RTKLIB allows the user a greater number of configurations regarding what kind of corrections to apply, as for example the corrections related with Earth Tides or Ocean loadings.

As a resume of what was exposed in previous sections, the Figure16 is presented. RTKLIB allows real-time processing, while WaPPP only allows post-processing, so the comparison is focussed on post-processing options.

		waPPP	RTKLIB
Orbit and clock information	sp3	Yes	Yes
	CLK	Yes	Yes
	Broadcast	Yes	Yes
	SSR corrections	No	Yes
Application of corrections	Antenna PCV	Satellite/Receiver	Satellite/Receiver
	DCB	-P1C1 -P1P2	Yes (1 file)
	Ionosphere correction	- IONEX file - Estimated from dual-frequency	- IONEX file - Estimated from dual-frequency - SBAS model - Broadcast
	Troposphere corrections	- Estimate ZTD + Grad - Estimate ZTD - Estimate Grad	- Estimate ZTD + Grad - SBAS - Estimate ZTD - Saastamoinen
	Earth Tides correction	No	- Solid -Solid OTL (BLQ + ERP)
	Phase Windup correction	No	Yes
Other processing parameters	Reject eclipsed satellites	No	Yes
	Elevation mask	Yes	Yes
	Geoid data	No	Yes
	Satellite constelltions	- GPS -GLONASS -GALILEO -SBAS	- GPS -GLONASS -GALILEO -SBAS - QZSS -BeiDou
	Rinex format	2.X	2.X, 3.X

Figure 16: Comparision between WaPPP and RTKLIB for PPP

7.2 Graphical user interface

According to the preliminary study (section 7.1) of both RTKLIB and WaPPP as software engines for PPP processing, in which different configuration parameters have been exposed, a user interface for new GOCA PPP module, developed under the context of this master thesis, has been designed. The user interface shown in this section is the complete version of the software, including real-time option.

The aim of this design is to be simple and functional, where different parameters can be easily changed or introduced by the user.

In this section, the proposed user interface is shown. In the figure provided in this section as draft of design, that controls that are enabled or disabled depending on other parameters are marked in red, while parameters that depend on the software engine are marked in blue.

The main window of the PPP software is shown in Figure 17. It includes one button to go to PPP mode settings dialog, one to make the process run another to stop the process (in case of real-time processing only), a text box that shows progress messages and a progress bar. This window has been made with the purpose to provide the PPP software a main window, so messages can be seen on it, but in case of integration with main GOCA GNSS module, it should be not necessary.

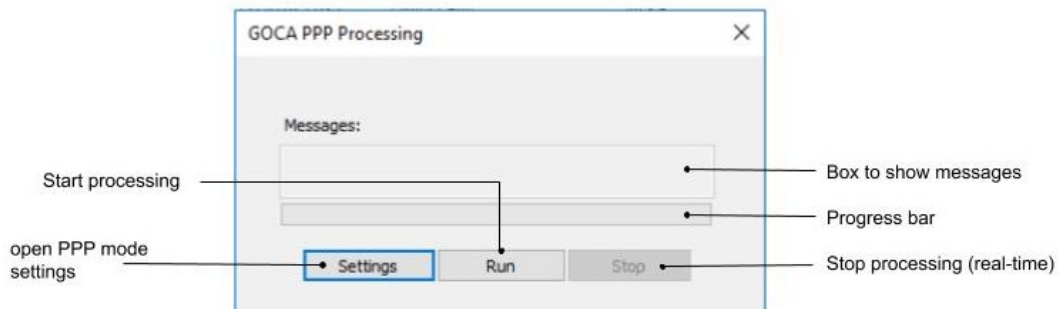


Figure 17: PPP software main window

When the button settings of the previous dialog is clicked, a new windows to choose the PPP processing mode (post-processing or real-time) is opened (Figure 18). In case of the post-processing only version of the software, this window is not included, so directly Figure19 is opened.

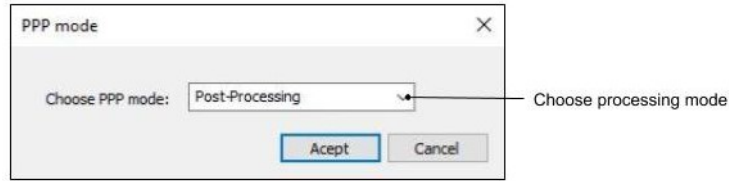


Figure 18: PPP processing mode

The main settings window for PPP in post-processing includes the configuration parameters that are used in both software engines. This design is shown in Figure 19.

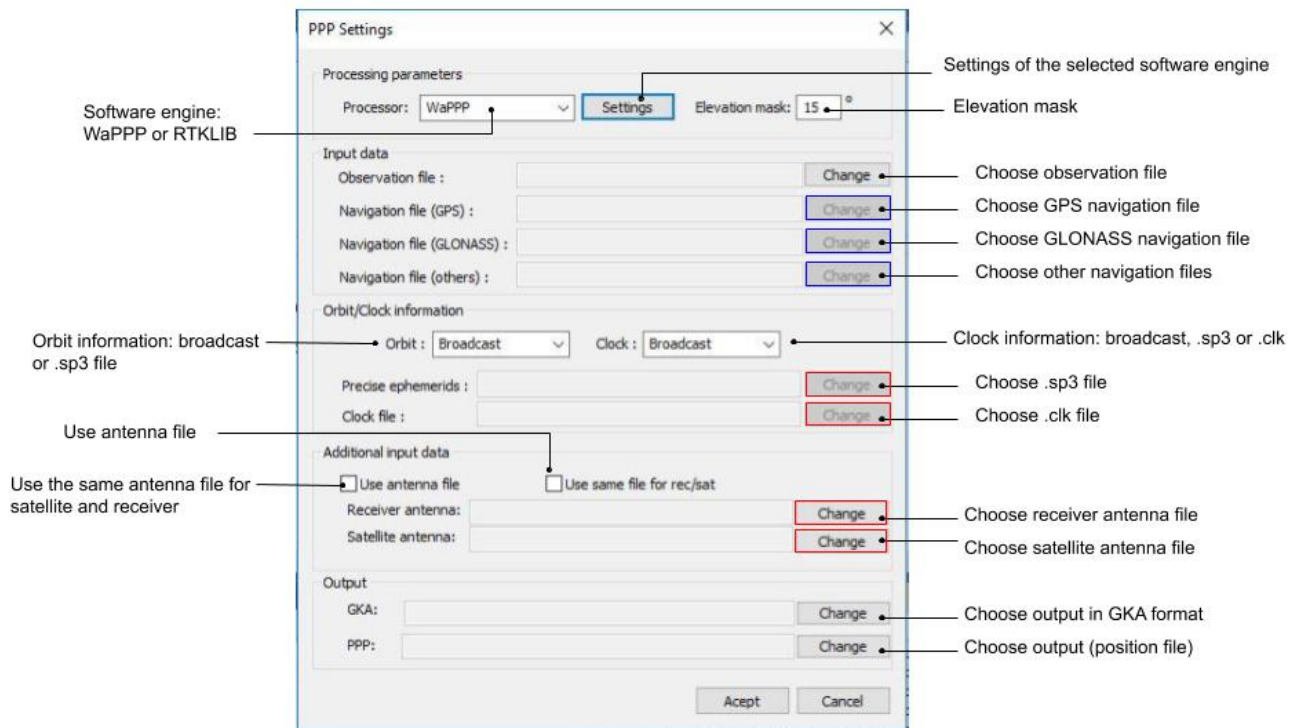


Figure 19: PPP Settings GUI

As can be seen in Figure 19, the first part is regarding the software engine to use and the elevation mask to apply. If RTKLIB is chosen, navigation file input buttons are enabled, while if the chosen processor is WaPPP this option is disabled (because this processor does not allow to specify navigation files, it just use them if they are in the working directory or in the same folder as the observation file). After that, is the main input data section (observation and navigation data).

In the orbit and clock information, user can choose what kind of orbit information to use (broadcast or precise) and also what kind of clock information (broadcast, .sp3 file or .clk file). When an option that requires that a file is chosen, corresponding buttons are enabled to choose the files. In case of RTKLIB, as shown in Figure 19, ephemerids and clock files can be chosen, while in case of WaPPP this orbit and clock part changes (Figure 20) in order to allow the user choosing a folder instead of specific files (According to the manual, also specific files can be chosen, but when it was trying during the development of this master thesis, this option didn't work properly).

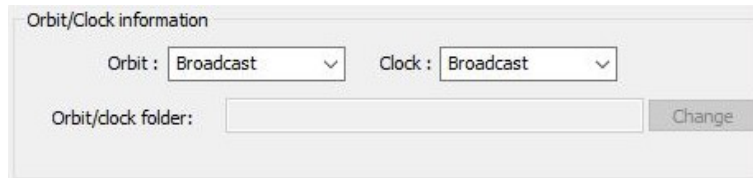


Figure 20: WaPPP orbit and clock information

Next information to be set is antenna corrections. It is chosen, user can introduce one correction file for receiver and for satellite antennas. Same file can be applied for both satellites and receiver if the corresponding check box is checked.

Last part of this dialog is focus on the output of the software. Both GKA (own format of GOCA) and position file must be set by the user. The first one will allow using the obtained data in GOCA main module and the second one allows the user analyse the results in more detail, having the coordinates and errors in each epoch.

The button settings allow the user to set up specific configuration of one processor. The interface in case of RTKLIB is shown in Figure 21.

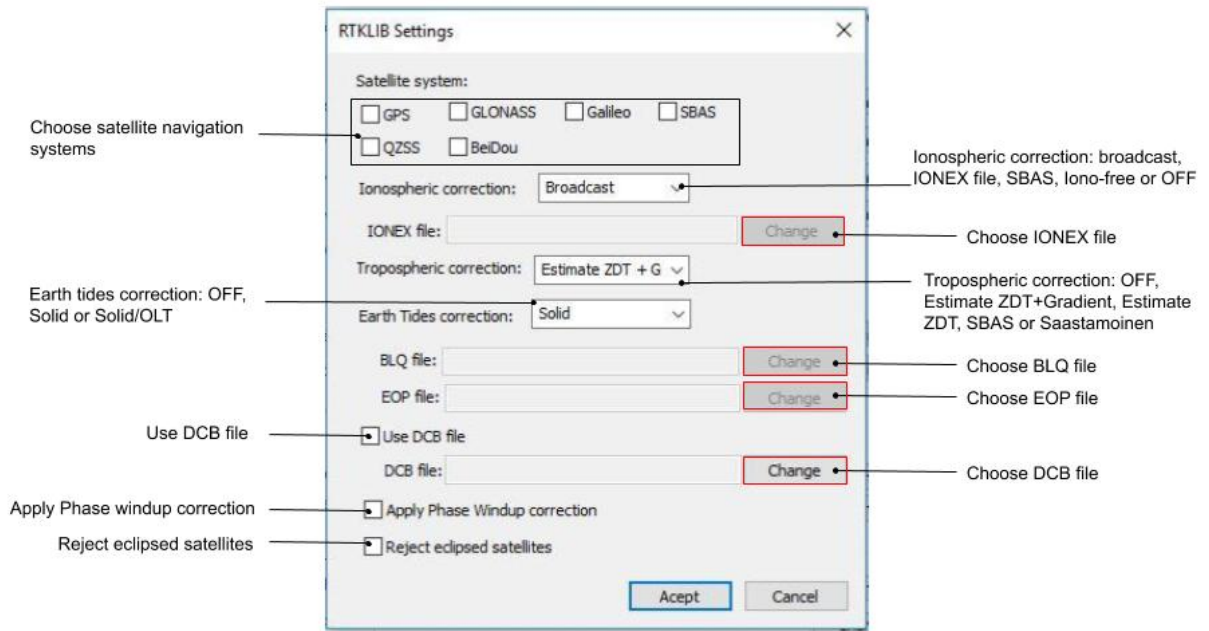


Figure 21: RTKLIB Settings GUI

Satellite systems to be used can be chosen with different checkboxes. By default, GPS and GLONASS are checked.

In this dialog, different corrections can be configured. The first one is the ionospheric correction, that can be broadcast, using IONEX file, SBAS, iono-free or not apply. In case of IONEX model, file has to be introduced.

Tropospheric correction can be chosen between different options (see Figure 21).

Earth tide corrections can be applied or not, and also EOP and BLQ files can be introduced to apply this correction. DCB file can be also used or not and phase wind up correction and the rejection of eclipsed satellites can be activated or not with the corresponding checkbox.

WaPPP software also has some options to be configured if chosen. This interface is shown in Figure 22.

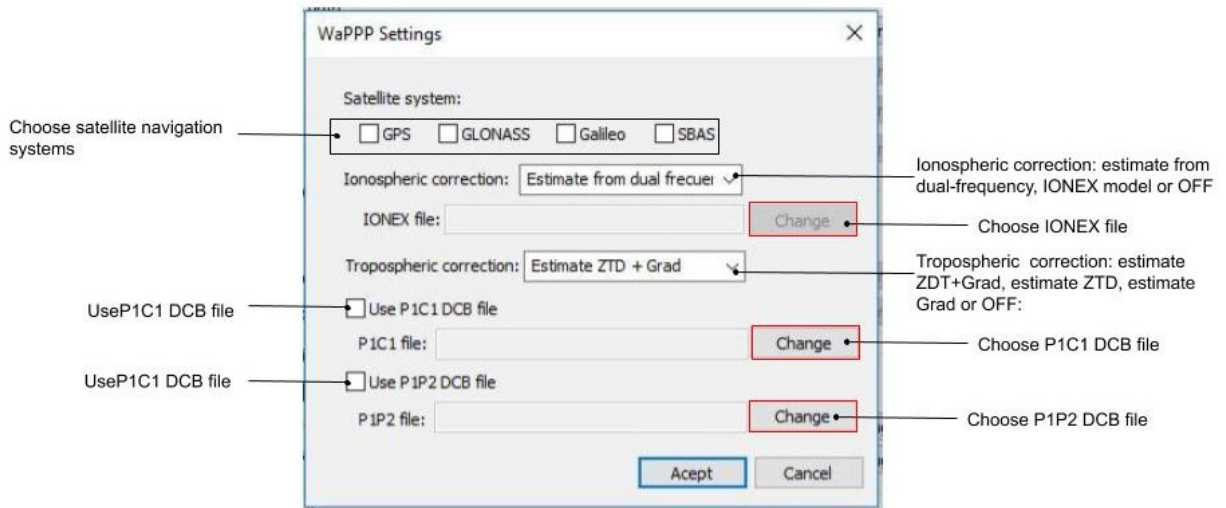


Figure 22: WaPPP Settings GUI

WaPPP interface is similar to the RTKLIB settings interface, but with less options. Navigation systems can be chosen, different ionospheric and tropospheric corrections (see Figure 22) and also DCB files can be introduced. In this case, two DCB files can be introduced (P1C1 and also P1P2).

If real-time option is chosen, then a dialog to set the TPC/IP address connection with the sensor (Figure 23).

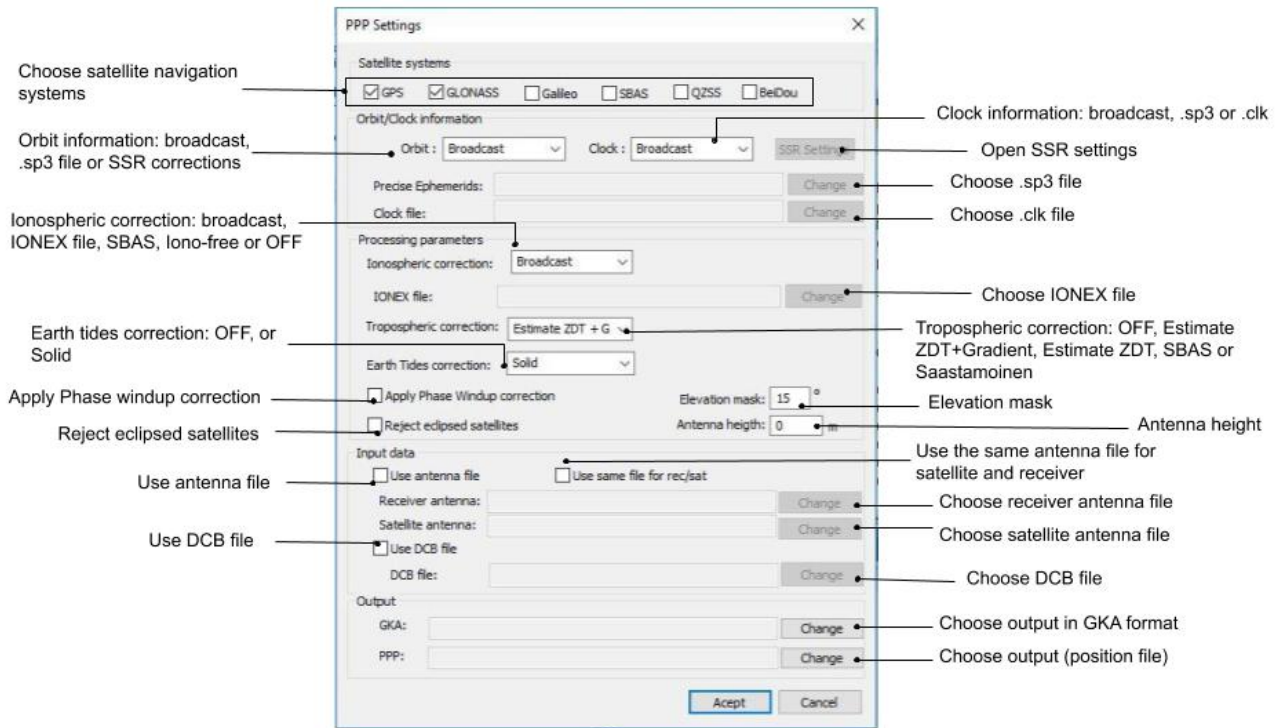


Figure 23: Real-time settings GUI

Main aspects of the settings for real-time processing are the same than in the post-processing case, no they are not going to be explained in detail. What is different, is the button that allows opening the SSR dialog if the option of SSR corrections is chosen in the orbit combo box. This dialog is shown in Figure 24, where there are different parameters to be introduced by the user in order to make the NTRIP connection to get SSR corrections.

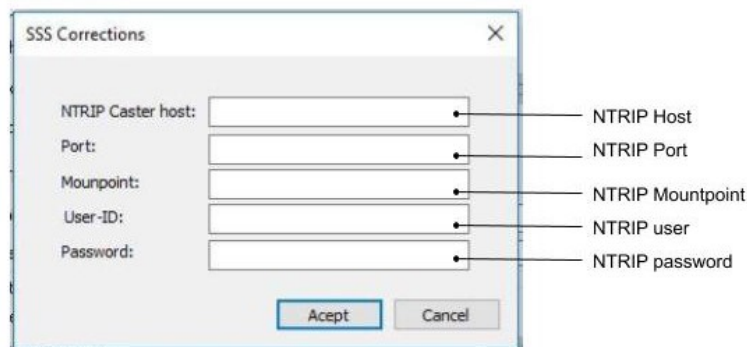


Figure 24: NTRIP connection for SSR corrections GUI

In this section, a general overview of the user interface has been provided. In case some part needs to be modified in future works, Annex A provides a list with the identification and names of all the variables involved in the graphical user interface.

7.2.1 Configuration error management

Following the idea of making the interface simple and useful for the user some possible configuration errors has to be managed. This management prevents errors when making software engines run.

First of all, before running the application it is checked if the settings are configured. If not, the program cannot be run.

In the next dialog, the one that allows choosing the processing mode (post-processing and real-time) also it is compulsory to configure the corresponding settings before accepting.

In case of post-processing mode, information that is checked in the main settings dialog is:

- Observation file is compulsory; user must introduce it.
- Navigation file (at least GPS one) is compulsory in case of RTKLIB.
- Output files are also compulsory.
- If precise ephemeris or clock file option are chosen, user must introduce corresponding files.
- If apply antenna corrections option is checked, antenna two files (or one file if apply same file is checked) have to be introduced.
- Specific processor settings has to be configured before running the program.

When clicking in the button “Accept”, if some of the above items are not correct, a message with the error information is shown (see example in Figure 25) and program cannot be run until there are no errors.

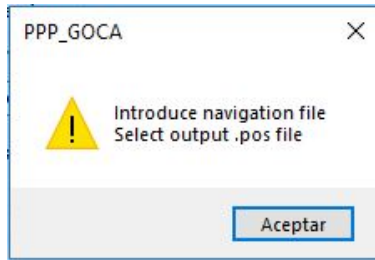


Figure 25: Configuration error example

In case of RTKLIB processor, errors that have to be checked are:

- At least one satellite system has to be selected.
- If IONEX model is chosen as ionospheric correction, file has to be introduced.
- If application of solid/OTL Earth tide corrections is chosen, both EOP and BLQ files has to be introduced.
- If use DCB file option is selected, file must be introduced.

Analogously, in case of WaPPP, following errors have to be checked:

- At least one navigation system has to be chosen.
- If IONEX model is chosen as ionospheric correction, file has to be introduced.
- If apply P1C1 DCB file option is checked, file must be introduced.
- If apply P1P2 DCB file option is checked, file must be introduced.

In case of real-time mode, first of all the TPC/IP address to connect with the sensor has to be introduced.

Then, the settings has to be configured. Errors managed in these settings are:

- Output files are compulsory.
- If precise ephemeris or clock file option are chosen, user must introduce corresponding files.
- If SSR corrections option is choosen, the corresponding configuration has to be set and all the fields on that SSR dialog are compulsory.
- If apply antenna corrections option is checked, antenna two files (or one file if apply same file is checked) have to be introduced.
- At least one satellite system has to be selected.

- If IONEX model is chosen as ionospheric correction, file has to be introduced.
- If use DCB file option is selected, file must be introduced.

Apart from these configuration errors, also processing errors have been taken into account, so if something goes wrong during the execution, a message is shown in the main PPP window (for example if the result cannot be opened to write GKA file).

7.3 Software architecture - General aspects

The developed PPP software has been explained from a point of view of user interface in section 7.2. In this section again the developed software is explained, but in this case as a technical guide of the software architecture.

The project contains different files (.h files, where the class declaration is done and .cpp files, that have all the corresponding code) that can be divided into three different groups: files containing functions, dialog classes and others. All of them that are relevant to PPP processing are exposed in this section. In addition, there exist other files that are created automatically to build the application, but they are not shown as they are created automatically and nothing has been changed. In case of integration with GOCA GNSS Control, these automatically created files are not needed.

In this section, those files that are used in both post-processing and real-time modes are shown. Specific aspects for each mode are exposed in further sections.

7.3.1 Function files

The first group of files that are included in the project are those that contain functions that are used in different parts of the code.

FileDirPPP.cpp

This file contains the functions that allow the user choose a file or a folder of the system. These functions are:

- `openFileFromDir`. This function allows choosing one file from the system. As inputs it requires the filter name and the filter extension. It returns the path name of the selected file or empty string if there is some problem. This function is shown in Figure 26.

```

// Function to open files from directory
CString openFileFromDir(CString FilterName, CString FilterExtension){

    CString Allfiles = _T("All Files (*.*)|*.*|"); // All files

    // Filter creation, from the function call and all files option
    CString FileFilter = FilterName + _T("|") + FilterExtension + _T("|") + Allfiles;

    // Create an Open dialog; the default file name extension is given when calling the function
    CFileDialog fileDlg(TRUE, FilterName, FilterExtension,
        OFN_FILEMUSTEXIST | OFN_HIDEREADONLY, FileFilter);

    // Display the file dialog. When user clicks OK, fileDlg.DoModal()
    if (fileDlg.DoModal() == IDOK)
    {
        CString pathName = fileDlg.GetPathName();

        return(pathName); // Return path name
    }
    else {
        return(_T(""));
    }
}

```

Figure 26: Open file from directory function

The way to call this function using extension filters is shown in Figure 27.

```

CString FilterName = _T("ATX file (*.atx)");
CString FilterExtension = _T("*.atx");
pathNameATX = openFileFromDir(FilterName, FilterExtension);

```

Figure 27: Open file from directory function call

- saveFileInDir. This function allows choosing also a file in the system, but in this case the file does not need to exist in advanced (NULL parameter in Figure 26), so is used to output files. Inputs are again filter name and filter extension.

```

// Create an Open dialog; the default file name extension is given when calling the function
CFileDialog fileDlg(TRUE, FilterName, FilterExtension,
    NULL | OFN_HIDEREADONLY, FileFilter);

```

Figure 28: Create file in directory function detail

- chooseFolder. This function allows choosing a folder. It has no inputs and returns the chosen path. This function is show in Figure 29.

```

CString chooseFolder() {
    CFolderPickerDialog folderDialog(NULL, 0, NULL, 0);

    if (folderDialog.DoModal() == IDOK)
    {
        CString pathName = folderDialog.GetPathName();

        return(pathName);
    }
    else {
        return(_T(""));
    }
}

```

Figure 29: Choose directory function

GKA_PPP.cpp

In this file functions to write GKA output files are included. There is one function to write data from RTKLIB and other to WaPPP. The structure of both functions is really similar, but the information is taken from a different place depending on the software engine used. Where each information is taken is going to be explained in section 7.6. Functions in this file are writeGKARTKLIB, writeGKAWAPP and writeGKART for real-time.

These functions receive as inputs:

- Path in which the file has to be written.
- Last line of results.
- Last time of observation (only in case of RTKLIB).
- Path of observation file (RINEX) (only in case of post-processing).

They are both Boolean functions, that returns true.

What these functions do is:

- Read information from last line of result. To do that it looks in specific positions of this line. Figure 30 shows an example of how this is done.

```

// Extract coordinates and errors from the last line of result (last epoch)
CString X = lastLine.Mid(26, 12);
CString Y = lastLine.Mid(42, 11);
CString Z = lastLine.Mid(56, 12);

```

Figure 30: Extract information of line

- Read RINEX heading to extract information. To prevent reading all the RINEX, it finishes when it finds the end of header. It looks in each header line for specific characters and takes the corresponding information with the position of the data in this specific line (Figure 31). This is not done in case of real-time processing because the RINEX is not included.

```

// Read RINEX to extract header information
if (file.Open(obsFile, CFile::modeRead, 0)) {
    while (file.ReadString(line)) {

        if (!line.Find(L"

        if (line.Mid(67, 4) == L"NAME") {
            name = line.Mid(0, 4); } // Get station name
    }
}

```

Figure 31: Read RINEX header information example

- Before continue, the RINEX file is closed.
- Assign default values in case there is no information in RINEX heading (name, receiver...).
- Cofactor matrix item calculation (according to section 7.6).
- Calculation of GPS time data (according to section 7.6).
- Write information in an output file. After that, the file is closed (Figure 32).

```

// Write GKA file
fileGKA.WriteString(L"#GOKA14 \n"); // header line

CString line1;
line1.Format(L"%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s \n",name,type,desc,gps_w,gps_e,
fileGKA.WriteString(line1); // line 1 of information

CString line2;
line2.Format(L"%s,%s,%s,%s,%s,%s,%s \n",sigma,qxx,qxy,qzx,qyy,qyz,qzz);
fileGKA.WriteString(line2); // line 2 of information (sigma and cofactor matrix)

fileGKA.WriteString(L"#END14 \n"); // last line

fileGKA.Close(); // Close GKA fil

```

Figure 32: Write GKA file example

PPP_Processing.cpp

This file contains the functions that are used to prepare the execution of both software engines. These functions are: PPP_RTKLIB, RTKLIBCommand and WAPPPCommand for post-processing. In case of real-time, functions in this file are PPP_RT and RTCommand.

PPP_RTKLIB:

This function writes the configuration file RTKLIB needs. As input, this function receives a rtklibSettings object that contains the properties to be used set by the user. It is a void function, so it does not return anything. The structure of this function is:

- Read information from settings object. This is done using the get methods implemented in this object.
- Read RINEX observation file header to get the antenna height, as in case of previous example of read RINEX. Close RINEX after getting height.
- Open configuration file and write, line by line, all the configuration. To do that, values extracted from object are used (example can be seen in Figure 33). Close this file at the end.

```
int tropo = rtklibSettings.getTropo();  
  
if (tropo == 0) value = L"est-ztdgrad";  
if (tropo == 1) value = L"est-ztd";  
if (tropo == 2) value = L"sbas";  
if (tropo == 3) value = L"saas";  
if (tropo == 4) value = L"off";  
buffer.Format(L"pos1-tropopt =%s #\n", value); // Troposphere correction  
file.WriteString(buffer);
```

Figure 33: Get property and write it on configuration file

RTKLIBCommand:

This function prepares the command that has to be executed to start RTKLIB processing. It doesn't need any input and it returns the complete command.

What it does is getting the information that needs from the global RTKLIB settings using get methods and concatenate the different information in the way RTKLIB command must be formed. As the main part of the configuration is in the configuration file, this function just need to get few additional parameters. These parameters are: configuration file, observation file, navigation files, orbit and clock files if they are used and output files. The command is obtained as shown in Figure 34.

```
L"rnx2rtkp -k " + conf + " " + obsFile + " " + navGpsFile + " " + navGlonassFile + " " + ephFile + " " + clkFile + " -o " + posFile
```

Figure 34: RTKLIB command

WAPPPCommand:

This function prepares the command needed to execute WaPPP software. As in the case of the previous one, this is done by getting the corresponding parameters of the global WaPPP settings using get methods and structure them as necessary.

In this case, WaPPP software does not need configuration file, so all parameters must be included in the command to execute. Based on the WaPPP manual, each parameter stored in the settings is translated into the corresponding WaPPP parameter (example in Figure 35).

```
int tropo = ::waSettings.getTropo(); // Troposphere information
CString trop;
if (tropo == 0) { trop = L""; } //Estimate ZTD + Grad
if (tropo == 1) { trop = L"-TG"; } // Estimate ZDT
if (tropo == 2) { trop = L"-T"; } // Estimate Grad
if (tropo == 3) { trop = L"-T -TG"; } // not apply
```

Figure 35: Example of WaPPP parameter in command

The full command is constructed as Figure 36.

```
CString wappCommand = L"wapp " + obsFile + " -e" + elev + " " + orb + " " + clk + " "
+ trop + " " + ion + " " + satSys + " " + dcb1 + " " + dcb2 + " " + atx1 + " " + atx2
+ " +L2 +PRO +P" + " +FP" + posFile + " " + "+EP" + ephFolder + " " + height + " ";
```

Figure 36: WaPPP command

This function does not need any input and it returns the WaPPP Command to be used in the execution.

The functions related to real-time processing are PPP_RT and RTCommand.

PPP_RT:

This function writes the configuration file RTKLIB needs, in this case for real-time processing. As input, this function receives a rtSettings object that contains the properties to be used set by the user. It is a void function, so it does not return anything.

What this function does is getting configuration parameters using get methods and write them in the corresponding configuration file as needed in case of each parameter, as was the case of PPP_RTKLIB.

RTCommand:

This function prepares the command needed to execute RTKLIB real-time module. In this case, all options are in the configuration file, so the command to execute is really simple (Figure 37).

```
CString rtCommand = L"rtkrcv_navka -o " + conf + "";
```

Figure 37: RTKLIB real-time command

7.3.2 Dialog files

This section includes the different classes that corresponds to different dialogs used in both processing modes.

GOCA_PPPIlg.cpp

This is the class corresponding to the main PPP dialog (Figure 17). This class has two main functions, corresponding to the three buttons in this dialog: OnBnClickedButtonPppSettings, OnBnClickedButtonPppRun and OnBnClickedButtonT. What these functions make is:

OnBnClickedButtonPppSettings:

It is executed when the button settings is clicked. It just open de PPP mode settings dialog as a modal window (Figure 38), to choose the post-processing mode or the real-time mode.

```
// Open PPP settings (button SETTINGS)
void GOCA_PPPIlg::OnBnClickedButtonPppSettings()
{
    PPPIlg pppdlg;
    pppdlg.DoModal();
}
```

Figure 38: Open dialog example

OnBnClickedButtonPppRun:

This function is executed when the button run is clicked. First, it checks for the chosen processing mode if settings have been set.

In case of post-processing mode, there are three different cases: if settings have not been set, RTKLIB is chosen and configured, and WaPPP is chosen and configured.

If RTKLIB has been chosen the actions that are made are: first the parameters need to start a process (that means to call an external .exe file) are prepared (Figure 39).

```
// Prepare variables to process
HANDLE hProcess = NULL; HANDLE hThread = NULL;
STARTUPINFO si; PROCESS_INFORMATION pi;
DWORD dwThreadId = 0; ZeroMemory(&si, sizeof(si)); ZeroMemory(&pi, sizeof(pi));
BOOL bCreateProcess = NULL;
```

Figure 39: Parameters to create process

After that, the function RTKLIBCommand is called and the corresponding command is obtained. Once the command is obtained, the program rnx2rtkp.exe is executed using CreateProcess function.

Then it is checked if the process is started or not. In case it cannot be started, a message is shown in the text window of the dialog.

In case the process has started correctly, as it takes some time, it is checked if the process still running. In that case the progress bar is moving and the message “Execution in progress...” is shown.

Once the process ends, the process variables are closed.

Then last line of the output is extracted and also the end time. If this file contains a solution, GKA file is created using the writeGKARTKLIB function. If this function ends successfully, the progress bar is completed and “DONE!” message is shown. If not, an error message is shown.

If WaPPP has been chosen, the process is analogous to the one above described. WaPPPCommand function is here called to prepare the execution and writeGKAWAPP to write the GKA file.

In case of real-time it is also checked if the corresponding settings have been set. If not, an message is shown.

As in the case of the post-processing mode, first the function to build the corresponding command is called. Once the command is obtained, the program `rtkrvc_navka.exe` is executed using `CreateProcess` function, as in previous cases. As a difference, the process started is a global variable, because the process does not stop until the user stop it.

If the process cannot be started an error message is shown and if it is has correctly started a execution message is shown. As the process has to be stopped by the user, this function also enables the stop button.

OnBnClickedButtonT:

This function is executed when the button stop is clicked. It only can be clicked when real-time processing is running. What it done in this function is:

- Terminate the real-time process using the global process variables.
- Solution file is read, as in case of post-processing. If the solution is not correct, an error message is shown. If it is correct, the function to write the GKA file is called.
- Check if the GKA file could be created and shown the corresponding message (error or finished).

PPPMode.cpp

This dialog allows choosing the processing mode (post-processing or real-time) with a combo box. It only has one function to choose the processing mode.

OnBnClickedOk:

this function opens with a `DoModal()` method the corresponding dialog depending on the mode chosen (0 in case of post-processing and 1 in case of real-time).

7.3.3 Other files

There is another file, that is not a dialog class or a file containing functions to be use in different parts of the software. This file is the one called `Globals`. In this file, those global variables, used along different parts of the software are defined.

These variables defined in this file are shown in Figure 40. Using global variables allows calling and using them from every part of the code.

```

// Setting objects
extern PPP_RTKLIBSettings rtklibSettings;
extern PPP_WaSettings waSettings;
extern RTSettings rtSettings;

extern int processor; // RTKLIB or WAPPP (post-processing mode)
extern int mode; // post-processing or real-time

extern bool waConf; // WAPPP configured? (post-processing)
extern bool rtklibConf; // RTKLIB configured? (post-processing)
extern bool runRT; // Real-time is running?

// Global variables to the real-time process
extern HANDLE hProcessRT; extern HANDLE hThreadRT;
extern STARTUPINFO siRT; extern PROCESS_INFORMATION piRT;
extern DWORD dwThreadIdRT;

```

Figure 40: Global variables

7.4 Software architecture - Post-processing

In this section, specific files of the PPP software that are related with the post-processing mode are explained.

7.4.1 Dialog files

In this section, specific dialog files for post-processing mode are explained.

PPPDlg.cpp

This is the class corresponding to the main PPP settings dialog (Figure 19). Functions included in this class are:

- OnInitDialog. This function is executed when the dialog is started. It is used to restore the options selected by the user, so if the window is closed and opened again, previous configuration is not lost. To do that, global settings are get and, depending on the values in the settings object (using get functions) different actions (as set combo boxes values and enable or disable buttons) are performance.
- OnBnClickedOK. This function is executed when the “Accept” button is clicked. First of all, it checks if the configuration the user has made is correct (according to configuration errors in section 7.2.1). These errors are controlled as shown as an example in Figure 41.

```

if (pathNameObs == "") {
    error_rtklib = error_rtklib + L"Introduce observation file \n"; correct_rtklib = FALSE;
    error_wappp = error_wappp + L"Introduce observation file \n"; correct_wappp = FALSE;
} // Observation file must be introduced

```

Figure 41: Errors and messages example

If the configuration of the chosen processor is correct, parameters are stored in the corresponding global setting object using set methods. After that, function PPP_RTKLIB (explained in previous section) is called in case of RTKLIB processor, so the configuration file is written.

- OnBnClickedCheckAntenna. When the check box of use antenna file is checked, this function is called. First, the data is updated. After that, if is checked buttons to choose .atx files are enabled and also the check box for using same file as receiver and satellite. If it is unchecked, these are disabled. Finally, the window is update. Figure 42 shows how this is done, and serves as an example for all the similar check boxes in the dialogs.

```

// Update data when antenna check box is clicked and enable buttons to choose files when it is checked
void PPPDlg::OnBnClickedCheckAntenna()
{
    UpdateData(TRUE);
    if (m_atx) {
        m_button_atxFile.EnableWindow(TRUE); m_button_atxFile2.EnableWindow(TRUE);
        GetDlgItem(IDC_CHECK_SAME)->EnableWindow(TRUE);}

    else {
        m_button_atxFile.EnableWindow(FALSE); m_button_atxFile2.EnableWindow(FALSE);
        CheckDlgItem(IDC_CHECK_SAME, FALSE); GetDlgItem(IDC_CHECK_SAME)->EnableWindow(FALSE);}

    UpdateWindow();
}

```

Figure 42: Check box code example

- OnBnClickedCheckSame. Similar to the previous one, data is updated and button of choosing a second .atx file is enabled or disabled.
- OnEnChangeEditElevationMask. When editable text of elevation mask is changed, the value of the elevation variable is updated.
- Buttons: these functions are executed when different buttons are clicked. All these functions call some of the functions in FileDir.cpp file. These functions are: OnBnClickedButtonAtxFile, OnBnClickedButtonAtxFile2, OnBnClickedButtonObsFile, OnBnClickedButtonGpsNav, OnBnClickedButtonGlonassNav, OnBnClickedButtonPreciseEph, OnBnClickedButtonClock, OnBnClickedButtonPppOut, OnBnClickedButtonGka.

As an example, Figure 43 shows the function `OnBnClickedButtonPreciseEph`, to collect precise ephemerids file. In this case, different function is called depending on the processor.

```

void PPPDlg::OnBnClickedButtonPreciseEph()
{
    if (comboPPP.GetCurSel() == 0) { // If RTKLIB is chosen, choose files
        CString FilterName = _T("Precise ephemerid file (*.sp3)");
        CString FilterExtension = _T("*.sp3");
        pathNameEph = openFileFromDir(FilterName, FilterExtension);
        m_ephFile.SetWindowText(pathNameEph);
    }

    else if (comboPPP.GetCurSel() == 1) { // If WaPPP is chosen, choose folder
        pathNameEph = chooseFolder();
        m_ephFile.SetWindowText(pathNameEph);
    }
}

```

Figure 43: Choose file or folder example

- Combo boxes: These functions are called when a combo box is changed. All these functions change the corresponding variable depending on the chosen option of the combo box and they enable or disable other components of the dialog depending the option. These functions are: `OnCbnSelchangeComboOrbit`, `OnCbnSelchangeComboClock`, `OnCbnSelchangeComboPpp`.

As an example, Figure 44 is shown. It corresponds to the combo box to choose the kind of orbit to use (precise or broadcast). If precise option is chosen, then button to choose the file (RTKLIB) or folder (WaPPP) is enabled, taking into account also the clock option.

```

// Update information of orbits when combo box is changed and enable or disable buttons
void PPPDlg::OnCbnSelchangeComboOrbit()
{
    selEph = comboOrbit.GetCurSel();

    if (comboPPP.GetCurSel() == 1) { // WaPPP
        if (selEph == 1) {m_button_orbit.EnableWindow(TRUE);}
        if (selEph == 0 && selClk == 0 ) {m_button_orbit.EnableWindow(FALSE);}
    }

    else { // RTKLIB
        if (selEph == 1) {m_button_orbit.EnableWindow(TRUE);}
        if (selEph == 0 && selClk != 1) {m_button_orbit.EnableWindow(FALSE);}
    }
}

```

Figure 44: Combo box code example

- OnBnClickedButtonSettings. This function creates a dialog object and open it as new modal window, depending on the chosen processor.

RTKLIBDlg.cpp and WAPPPDlg.cpp

These are the files corresponding to the RTKLIB and WaPPP settings dialogs. They are quite similar to the previous one, but containing settings specific to RTKLIB or WaPPP processor. The functions contained on them are:

- OnInitDialog. Analogous to the function with the same name in previous dialog, it takes the RTKLIB settings from the global object and restore the corresponding information.
- OnBnClickedOk. This function is executed when button "Accept" is clicked. First it controls there no configuration errors and then, if all is correct, information is saved in global settings objects using set methods.
- Other functions. In these files there are other functions, as in the previous one, that are executed when a check box is selected or when a button to choose a file is clicked. they are not detailed due to their similarity to the ones in other files explained before.

7.4.2 Other files

In addition to the previously explained files that contain functions and the code corresponding to dialogs, in the project there are other files with other functionalities. They are shown in this section.

PPP_WaSettings and PPP_RTKLIBSettings

These two files contain the definition of configuration classes. These classes contain all the properties that can be set and also corresponding definition of set and get methods.

In header files (.h) variables and methods are specified and in the code files (.cpp), these methods are defined.

Get functions don't receive anything and return the corresponding value (Example in Figure 45)

```
// Get functions to obtain values in RTKLIB settings
int PPP_RTKLIBSettings::getElevation() { return elevation; } // Elevation mask option (editable)
```

Figure 45: Get function example

Set functions receive a value, assign it to the corresponding parameter and return true (Example in Figure 45)

```
// Set functions to assign values to settings  
  
bool PPP_RTKLIBSettings::setElevation(int n) { elevation = n; return true; }
```

Figure 46: Set function example

The way that these classes are used is the following one: a global object of each class (one for WaPPP and other for RTKLIB) is created and their properties are defined with set functions in different dialogs of the software. After all the properties are in the object, this global object is used to extract these properties with get functions and this information is used to run the corresponding software engines.

Figures 47 (RTKLIB) and 48 (WaPPP) show all the different parameters that are included in each file, with their variable type and the corresponding set and get methods.

Variable	Type	Description	Set method	Get method
iono	int	iono combo box option	setIono(iono)	getIono()
tropo	int	tropo combo box option	setTropo(tropo)	getTropo()
tide	int	earth tide combo box option	setTide(tide)	getTide()
elevation	int	elevation mask	setElevation(elevation)	getElevation()
clock	int	clock combo box option	setClock(clock)	getClock()
navSystem	int	calculated number of navigation systems	setNavSys(navSystem)	getNavSys()
wind	bool	phase wind up check box	setWind(wind)	getWind()
eclipse	bool	reject eclipsed satellites check box	setEclipse(eclipse)	getEclipse()
gps, glonass, galileo, sbas, qzss, beidou	bool	navigations systems check boxes	setGps(gps), setGlonass(glonass)...	getGps(), getGlonass ...
dc_b	bool	DCB file check box	setDcb_b(dcb_b)	getDcb_b()
atx_b	bool	antenna file check box	setAtx_b(atx_b)	getAtx_b()
atx2_b	bool	antenna file 2 check box	setAtx2_b(atx2_b)	getAtx2_b()
pathNameATX	Cstring	antenna file path	setATX(path)	getATX()
pathNameATX2	Cstring	antenna file 2 path	setATX2(path)	getATX2()
pathNameBLQ	Cstring	BLQ file path	setBLQ(path)	getBLQ()
pathNameEOP	Cstring	EOP file path	setEOP(path)	getEOP()
pathNameDCB	Cstring	DCB file path	setDCB(path)	getDCB()
pathNameObs	Cstring	observation file path	setObs(path)	getObs()
pathNameGpsNav	Cstring	GPS navigation file path	setGpsNav(path)	getGpsNav()
pathNameGlonassNav	Cstring	GLONASS navigation file path	setGlonassNav(path)	getGlonassNav()
pathNameEph	Cstring	ephemerids file path	setEph(path)	getEph()
pathNameClk	Cstring	clock file path	setClk(path)	getClk()
pathNamePos	Cstring	position file path	setPos(path)	getPoc()
pathNameGKA	Cstring	GKA file path	setGKA(path)	getGKA()
pathNameIONEX	Cstring	IONEX file path	setIONEX(path)	getIONEX()

Figure 47: RTKLIB settings object parameters

Variable	Type	Description	Set method	Get method
iono	int	iono combo box option	setIono(iono)	getIono()
tropo	int	tropo combo box option	setTropo(tropo)	getTropo()
elevation	int	elevation mask	setElevation(elevation)	getElevation()
clock	int	clock combo box option	setClock(clock)	getClock()
gps, glonass, galileo, sbas	bool	navigations systems check boxes	setGps(gps), setGlonass(glonass)...	getGps(), getGlonass ...
dcb1_b	bool	DCB file check box	setDcb1_b(dcb_b)	getDcb1_b()
dcb2_b	bool	DCB file 2 check box	setDcb2_b(dcb_b)	getDcb2_b()
atx_b	bool	antenna file check box	setAtx_b(atx_b)	getAtx_b()
atx2_b	bool	antenna file 2 check box	setAtx2_b(atx2_b)	getAtx2_b()
pathNameATX	Cstring	antenna file path	setATX(path)	getATX()
pathNameATX2	Cstring	antenna file 2 path	setATX2(path)	getATX2()
pathNameDCB1	Cstring	DCB file path	setDCB1(path)	getDCB1()
pathNameDCB2	Cstring	DCB file 2 path	setDCB2(path)	getDCB2()
pathNameObs	Cstring	observation file path	setObs(path)	getObs()
pathNameEph	Cstring	ephemerids file path	setEph(path)	getEph()
pathNameClk	Cstring	clock file path	setClk(path)	getClk()
pathNamePos	Cstring	position file path	setPos(path)	getPoc()
pathNameGKA	Cstring	GKA file path	setGKA(path)	getGKA()
pathNameIONEX	Cstring	IONEX file path	setIONEX(path)	getIONEX()

Figure 48: WaPPP settings object parameters

7.4.3 Software architecture diagram

In order to provide a better overview of the software architecture, Figure 49 is provided. Here, the most important classes are included (so general classes automatically generated are not presented). Those classes in grey corresponds to dialog classes, classes represented in blue are files that contains functions and orange represents other classes.

In order make it easy to understand, not every single function is presented, some of them are group, for example all the functions that are executed when a file button is clicked, are represented by "file button functions". Arrows represent which functions call other functions. As explained in previous sections, when a dialog is opened, stored settings are taken, using get functions of the properties objects. These relationships are not included in the diagram, to simplify it.

In case of the first version of the software, in which only post-processing mode is available, the diagram is the same but without the dialog PPPMode, so PPPDlg is opened directly from GOCA_PPPLg.

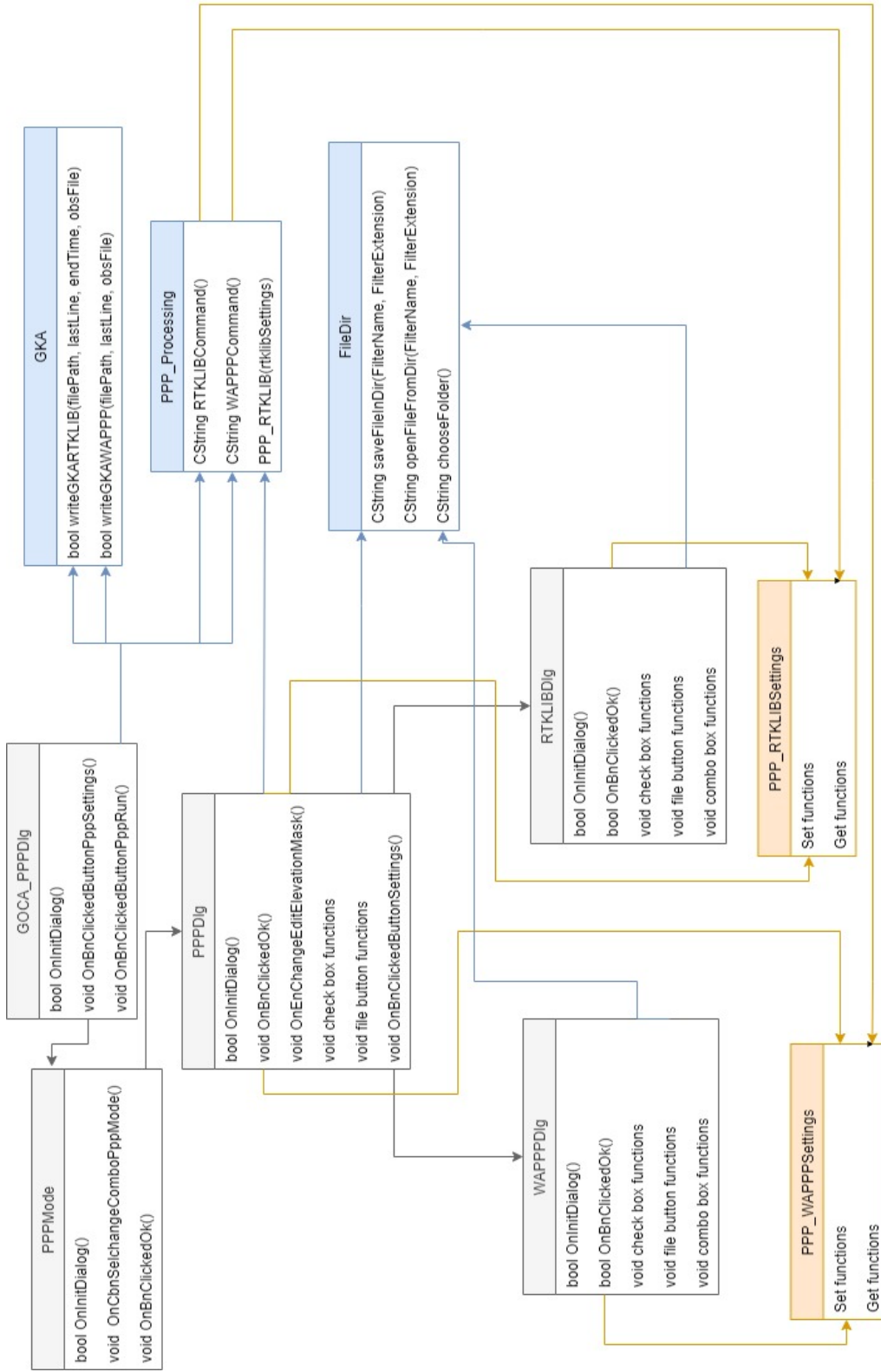


Figure 49: Software architecture diagram post-processing

7.5 Software architecture - Real-time

In this section, those files that are related with real-time processing are shown.

7.5.1 Dialog files

In this section, specific dialog files for real-time processing mode are explained.

PPPRTDlg.cpp

This is the class corresponding to the main PPP settings dialog in real-time option. Functions included in this class are:

- OnInitDialog. This function is executed when the dialog is started. It is used to restore the options selected by the user, so if the window is closed and opened again, previous configuration is not lost. To do that, global settings for real-time are get and, depending on the values in the settings object (using get functions) different actions (as set combo boxes values and enable or disable buttons) are performed.
- OnBnClickedOk. This function is executed when the “Accept” button is clicked. First of all, it checks if the configuration the user has made is correct, as it was the case of post-processing. If the configuration is correct, parameters are stored in the corresponding global setting object using set methods., if not, the corresponding errors are shown.
- OnBnClickedCheckAntennaRt. When the check box of use antenna file is checked, this function is called. It is the same than the case of post-processing: first, the data is updated. After that, if it is checked buttons to choose .atx files are enabled and also the check box for using same file as receiver and satellite. If it is unchecked, these are disabled. Finally, the window is updated.
- OnBnClickedCheckSameRt. Similar to the previous one, data is updated and button of choosing a second .atx file is enabled or disabled.
- OnEnChangeEditElevationMaskRT and OnEnChangeEditAntennaH. When editable text of elevation mask or height is changed, the value of the corresponding variable is updated.
- Buttons: these functions are executed when different buttons are clicked. The functionality is the same that the one explained in case of post-processing mode.

- Combo boxes: These functions are called when a combo box is changed. All these functions change the corresponding variable depending on the chosen option of the combo box and they enable or disable other components of the dialog depending the option.
- OnBnClickedButtonSsr. This function creates a SSR dialog object and open it as new modal window. This button is enabled or disabled depending on the option chosen in the orbit combo box.

SSTDlg.cpp

This class corresponds to the dialog to introduce data to use SSR corrections with NTRIP protocol. It contains the function to remember previous configuration (OnInitDialog), the one that is executed when button "Accept" is clicked (OnBnClickedOk) that check if the configuration is correct and then use set methods to save these variables and the functions that allow updating the different variables of the edit inputs.

TPCPPDlg.cpp

This is the class corresponding to the first dialog that is opened when real-time option is chosen. It contains the same function that in the previous case (OnInitDialog, OnBnClickedOk and functions to update variables). The difference in this case is than, hen clicking the button "Accept", apart from checking errors and saving configuration, the function PPP_RT is called in order to write the corresponding configuration file using all the options the user has choose.

It also contains a function to open the PPP real-time dialog (PPPRTDlg) when the corresponding button is clicked.

7.5.2 Other files

In addition to the previously explained files that contain functions and the code corresponding to dialogs, in the project there is other file with other functionalities also in case of real-time. It is shown in this section.

RTSettings

This file contains the definition of configuration class. This class contains all the properties that can be set and also corresponding definition of set and get methods.

As in case of post-processing, In header files (.h) variables and methods are specified and in the code files (.cpp), these methods are defined.

Get functions don't receive anything and return the corresponding value, analogous to the one shown in post-processing mode and Set functions receive a value, assign it to the corresponding parameter and return true.

The way that this class is used is the following one: a global object is created and their properties are defined with set functions in different dialogs of the software. After all the properties are in the object, this global object is used to extract these properties with get functions and this information is used to run the software.

Figure 50 shows all the different parameters that are included, with their variable type and the corresponding set and get methods.

Variable	Type	Description	Set method	Get method
iono	int	iono combo box option	setIono(iono)	getIono()
tropo	int	tropo combo box option	setTropo(tropo)	getTropo()
tide	int	earth tide combo box option	setTide(tide)	getTide()
elevation	int	elevation mask	setElevation(elevation)	getElevation()
orbit	int	orbit combo box option	setOrb(orbit)	getOrb()
clock	int	clock combo box option	setClock(clock)	getClock()
navSystem	int	calculated number of navigation systems	setNavSys(navSystem)	getNavSys()
height	double	antenna height	setHeight(height)	getHeight()
wind	bool	phase wind up check box	setWind(wind)	getWind()
eclipse	bool	reject eclipsed satellites check box	setEclipse(eclipse)	getEclipse()
gps, glonass, galileo, sbas, qzss, beidou	bool	navigrations systems check boxes	setGps(gps), setGlonass(glonass)	getGps(), getGlonass ...
dcb_b	bool	DCB file check box	setDcb_b(dcb_b)	getDcb_b()
atx_b	bool	antenna file check box	setAtx_b(atx_b)	getAtx_b()
atx2_b	bool	antenna file 2 check box	setAtx2_b(atx2_b)	getAtx2_b()
settings	bool	configured or not	setSettings(settings)	getSettings()
pathNameATX	Cstring	antenna file path	setATX(path)	getATX()
pathNameATX2	Cstring	antenna file 2 path	setATX2(path)	getATX2()
pathNameDCB	Cstring	DCB file path	setDCB(path)	getDCB()
pathNameEph	Cstring	ephemerids file path	setEph(path)	getEph()
pathNameClk	Cstring	clock file path	setClk(path)	getClk()
pathNamePos	Cstring	position file path	setPos(path)	getPoc()
pathNameGKA	Cstring	GKA file path	setGKA(path)	getGKA()
pathNameIONEX	Cstring	IONEX file path	setIONEX(path)	getIONEX()
NTRIPHost, NTRIPPort, NTRIPPMP, NTRIPUser, NTRIPPassword	Cstring	NTRIP connection information	setNTRIPHost (NTRIPHost)...	getNTRIPHost() ...
IPAddress, IPPort	Cstring	SSR connection information	setIONEX(path)	getIONEX()

Figure 50: Real-time settings object

7.5.3 Software architecture diagram

As in case of post-processing mode, real-time diagram (Figure 51) is shown to make clearer the software architecture. Again the most important classes are included (so general classes automatically generated are not presented). Those classes in grey corresponds to dialog classes, classes represented in blue are files that contains functions and orange represents other classes.

In order make it easy to understand, not every single function is presented, some of them are group, for example all the functions that are executed when a file button is clicked, are represented by "file button functions". Arrows represent which functions call other functions. As explained in previous sections, when a dialog is opened, stored settings are taken, using get functions of the properties objects. These relationships are not included in the diagram, to simplify it.

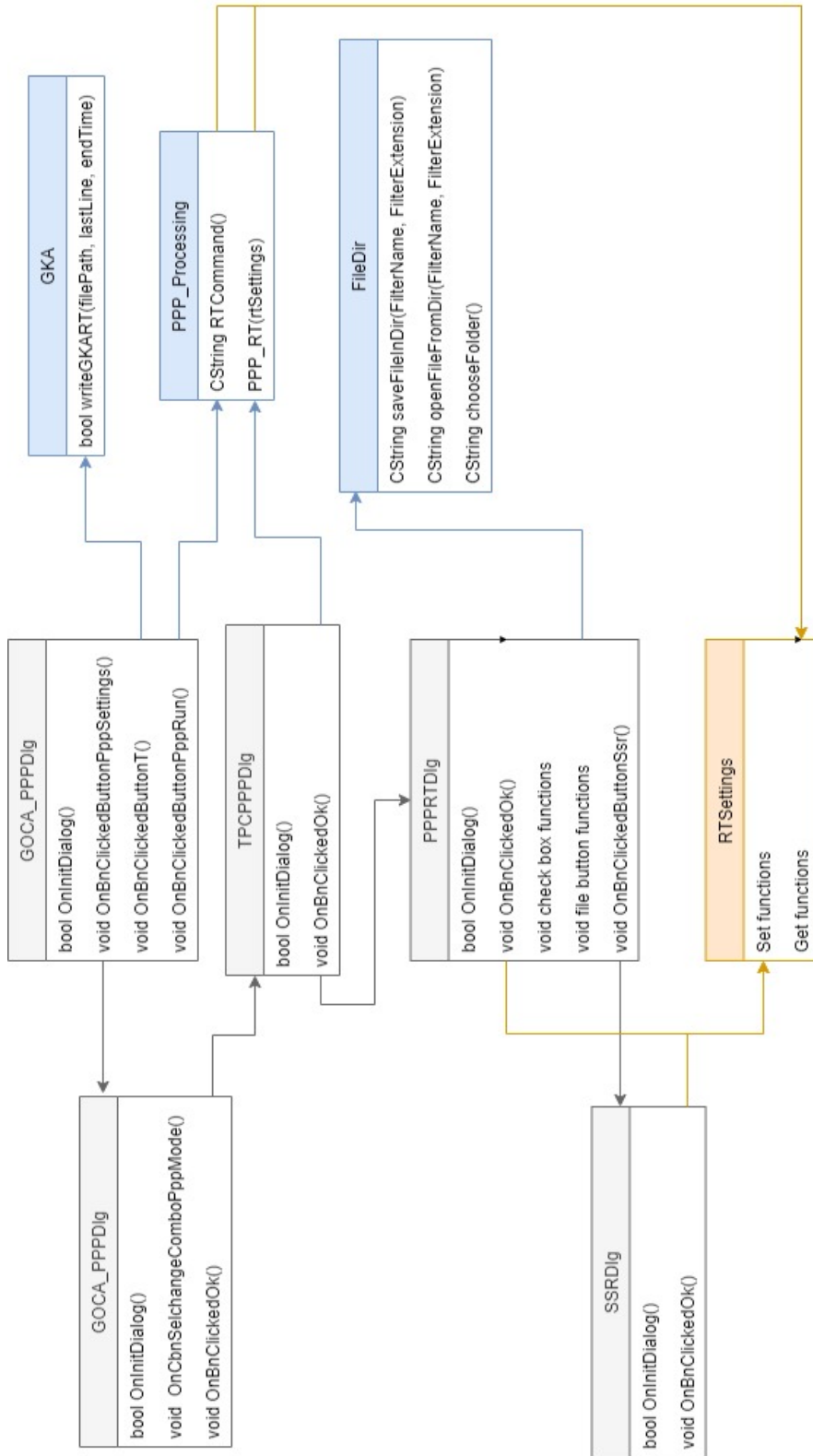


Figure 51: Software architecture diagram real-time

7.6 GKA information

In section 6.2 new GKA format defined for PPP results is exposed. All the information that has to be written in .gka file has to be extracted by the program. In this section, where this information is picked up from is described.

In case of post-processing option, some of the information is extracted from the RINEX observation file used in the computation, while other issues are get from the result file.

```

2.11      OBSERVATION DATA      M (MIXED)      RINEX VERSION / TYPE
teqc 2018Dec12      BKG Frankfurt      20190702 00:28:42UTCPGM / RUN BY / DATE
KARL      MARKER NAME
14216M001      MARKER NUMBER
GREFOP      BKG/AL/G3      OBSERVER / AGENCY
02042      JAVAD TRE 3 DELTA      3.7.6 APR,08,2019      REC # / TYPE / VERS
725092      LEIAR25.R4      LEIT      ANT # / TYPE
4146524.6600      613137.8250      4791516.9620      APPROX POSITION XYZ
0.0450      0.0000      0.0000      ANTENNA: DELTA H/E/N
1 1      WAVELENGTH FACT L1/2
14 C1 C2 C5 P1 P2 L1 L2 L5      D1# / TYPES OF OBSERV
D2 D5 S1 S2 S5      # / TYPES OF OBSERV
30.0000      INTERVAL
18      LEAP SECONDS
0      RCV CLOCK OFFS APPL
Forced Modulo Decimation to 30 seconds      COMMENT
Concatenated RINEX files (24)      COMMENT
teqc edited: GLONASS # 25 - 32 excluded      COMMENT
teqc edited: GLONASS # 33 - 50 excluded      COMMENT
2019 7 1 0 0 0.0000000      GPS      TIME OF FIRST OBS
Linux 2.6.32-573.12.1.x86_64|x86_64|gcc -static|Linux 64|=+      COMMENT
MAKERINEX 2.0.40699 NTRIPS13 238B30 2019-07-01 01:01      COMMENT
END OF HEADER

```

Figure 52: Rinex header example

The information extracted from the RINEX header (example in Figure 52), in both cases RTKLIB and WaPPP, is:

- Station name: from the header row where *MARKER NAME* appears. 4-character name (short name) is used.
- Receiver type: this information is placed in the RINEX header as *REC#/TYPE*. If this information is not included in RINEX file, default value that will appear in GKA file is “NONE”.
- Antenna height: extracted from the delta position information, indicated with *ANTENNA: DELTA H/E/N* in the RINEX header. The required information is the one corresponding to H.

In the real-time option, as the RINEX is not used, this information cannot be extracted from here. At the moment, as this part needs a little more testing, the antenna

height is directly set by the user as an input in the software and the marker name and receiver type is set as "NONE" by default. In further developments this has to be set, fixed or as additional inputs in the software.

The rest of information that has to be included in GKA file is extracted from the position file result. The structure for the resulting position file is different for each processor.

```

% program   : RTKLIB ver.2.4.3 demo5 b29b
% inp file  : C:\Users\Raque\Documents\GOCA\KARL_DATA_010719\karl1820.19o
% inp file  : C:\Users\Raque\Documents\GOCA\KARL_DATA_010719\karl1820.19n
% inp file  : C:\Users\Raque\Documents\GOCA\KARL_DATA_010719\karl1820.19g
% inp file  : C:\Users\Raque\Documents\GOCA\KARL_DATA_010719\igs20601.sp3
% obs start : 2019/07/01 00:00:00.0 GPST (week2060 86400.0s)
% obs end   : 2019/07/01 23:59:30.0 GPST (week2060 172770.0s)
% pos mode  : ppp-static

```

Figure 53: RTKLIB solution header example

The information corresponding to RTKLIB is in both the file header and the final result row (corresponding to last epoch of observation processed). Information in header is:

- GPS week: This information appear in the header of the result file (see Figure 53). The key that indicates this row is *% obs end*.
- GPS week second. This information is also in *% obs end* row. This information is used to obtain GPS week-day and GPS second. The week day is calculated as the integer part of the quotient between GPS week second and 86400. This value is the number of seconds in one day (24 hours * 60 minutes per hour * 60 seconds per minute), so the integer part of this quotient corresponds to the day of the week. The GPS second corresponds to the remainder of the above division, that means is the corresponding second in one specific day.

Remaining information is extracted from the last processed epoch (see Figure 54).

% GPST	x-ecor (m)	y-ecor (m)	z-ecor (m)	Q	ns	sdx (m)	sdym (m)	sdz (m)	sdxy (m)	sdzy (m)	sdzx (m)	age (s)	ratio
2019/07/01 01:33:30.000	4146523.0636	613138.3384	4791517.5484	6	6	2.9284	1.3721	3.8565	1.1202	1.2390	2.8494	0.00	0.0
2019/07/01 01:34:00.000	4146523.1833	613138.3015	4791517.6768	6	6	2.0795	0.9720	2.7309	0.7944	0.8769	2.0215	0.00	0.0
2019/07/01 01:34:30.000	4146523.4501	613138.1231	4791517.6086	6	6	1.7028	0.7948	2.2292	0.6494	0.7153	1.6526	0.00	0.0

Figure 54: RTKLIB solution format example

This information includes:

- X, Y and Z coordinates.

- Covariance matrix information, that is used in the calculation of cofactor matrix, which is included in GKA file (see 7.6.1).

In case of WaPPP result, all the information is extracted from the last epoch processed (Figure 55).

#week	sec	X	Y	Z	solution	#all	#fixed	pdop	s_X	s_Y	s_Z	cor_XY	cor_YZ	cor_ZX
# #1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15
2060	86430.00	4146524.2426	613138.3853	4791517.5139	LOT_float	10	0	3.8	0.2499	0.0983	0.3273	0.2865	-0.0184	-0.1252
2060	86460.00	4146524.3030	613138.4055	4791517.4288	LOT_float	10	0	3.8	0.2739	0.1075	0.3583	0.2902	-0.0227	-0.1336

Figure 55: WaPPP solution format example

This information is:

- GPS week and GPS week second (used as same way explained in RTKLIB part, for calculating the week-day and GPS second).
- X, Y and Z coordinates.
- Covariance matrix information, that is used in the calculation of cofactor matrix, which is included in GKA file (see 7.6.1).

Description, LStatus, LStat_Type and Stat.String are set by default in both cases (with values *NONE*, 0 and 0 respectively).

7.6.1 Cofactor matrix calculation

In GKA file, as explained in previous section, cofactor matrix is provided. In this section, the calculation of these terms of the cofactor matrix is explained.

First of all, the resulting files (both RTKLIB and WaPPP), provides the information to reconstruct the covariance matrix (C_{xx}). This symmetric matrix looks as:

$$C_{xx} = \begin{bmatrix} sdx * sdx & sdx * sdy & sdx * sdz \\ & sdy * sdy & sdy * sdz \\ & & sdz * sdz \end{bmatrix} = \begin{bmatrix} C_{xx}XX & C_{xx}XY & C_{xx}XZ \\ & C_{xx}YY & C_{xx}YZ \\ & & C_{xx}ZZ \end{bmatrix} \quad (7.1)$$

The covariance matrix C_{xx} can always be factorized as a product of variance factor $sigma * sigma$ and a cofactor-matrix (Q_{xx}):

$$C_{xx} = sigma * sigma * Q_{xx} \quad (7.2)$$

Variance factor can be calculated as:

$$sigma = \sqrt{C_{xx}XX + C_{xx}YY + C_{xx}ZZ} \quad (7.3)$$

Despite Equation (7.3), sigma can be also set as an arbitrary factor. The standard factor is often 0.001, because the mm is a standard accuracy level in engineering. In the context of this master thesis, 0.001 is going to be used as sigma value.

So, cofactor matrix (symmetric) can is defined as:

$$Q_{xx} \begin{bmatrix} q_{xx} & q_{xy} & q_{xz} \\ & q_{yy} & q_{yx} \\ & & q_{zz} \end{bmatrix} \quad (7.4)$$

where:

$$q_{xx} = \frac{C_{xx}XX}{sigma * sigma} \quad (7.5)$$

and the same with the other terms. In GKA file, these 6 terms are provided ($q_{xx}, q_{xy}, q_{zx}, q_{yy}, q_{yz}, q_{zz}$). Value of sigma is also provided in the resulting GKA file, so the covariance matrix can be reconstructed.

8. Integration proposal of PPP software with GOCA GNSS Control

The developed PPP software under the context of this master thesis was thought to be a part of GOCA GNSS Control. However, technical difficulties have prevented integration from taking place. In this section, a proposal for further integration is going to be done.

In this case, the integration proposal is related to the software's version that only contains post-processing, because this first version is the one that were developed and is tested deeper than the second version. In case of the whole software, the process should be analogous but adding all the corresponding C++ files.

In resume, the steps to follow are:

- Add executables (wapp and rnx2rtkp).
- Add source code files. Change resource file and application file.
- Add dialogs. Make sure dialogs are correctly linked to classes.
- Add new option in GOCA GNSS Control for new mode.
- Create corresponding code for this new mode (add, edit and delete).
- Re-write run function.

First of all, executables wapp and rnx2rtkp (RTKLIB) has to be added in the folder of GOCA GNSS Control where all the scripts are included, because they are need by the PPP software.

8.1 Add dialogs and source code files

Different source code files have to be added to the GOCA GNSS Control project. For that they have to be copied in the folder that contains all the GOCA GNSS Control code files and then, using Visual Studio, add them to the project, by adding existing files. Both .cpp and .h files have to be added. The name of these files are:

FileDirPPP, GKA_PPP, Globals, PPP_Processing, PPP_RTKLIBSettings, PPP_WaSettings, PPPDlg, RTKLIBDlg, WAPPPDlg

In case of GOCA_PPPDlg, the file has not to be added but the code on it has to be copied. This dialog has been designed to provide PPP software a main dialog, but in

case of GOCA GNSS Control is not needed, because the main PPP settings dialog is going to be directly opened.

Other files presented in the PPP software are not necessary for integrating it on other application. After that, some little changes have to be done in all these files in order to use them. In .cpp files, the resource file and the application file are included at the beginning of the scripts, in case of PPP software this is made by `#include "pch.h"` and `#include "PPP_GOCA.h"`. When including these files in GOCA GNSS Control, this has to be changed to `#include "stdafx.h"` (resource file) and `#include "GOCA_GNSSControl.h"` (application file).

Then, dialogs (user interface) has to be also included. To do that, resource files of both projects has to be opened in Visual Studio (GOCA_GNSSControl.rc and PPP-GOCA.rc) and copy the corresponding dialogs from PPP software to GOCA GNSS Control (IDD_DIALOG_PPP, IDD_DIALOG_RTKLIB, IDD_DIALOG_WAPPP dialogs are the one needed).

Once dialogs are copied, then in the GOCA project, corresponding classes have to be assigned to these dialogs if this is not automatically made. This can be made by right click on the dialog and then the option "Class assistant" (Figure 56).

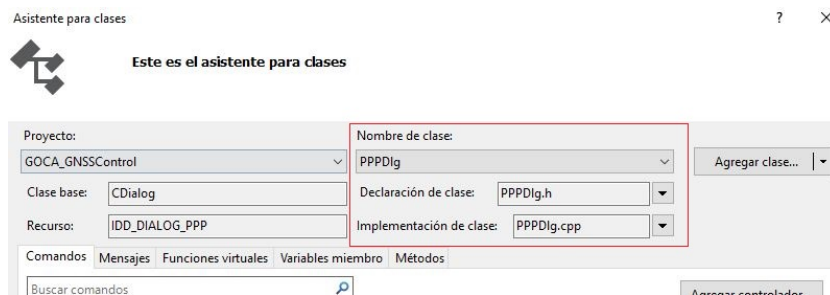


Figure 56: Class assistant Visual Studio

Corresponding classes for each dialog, that have to be correctly set, are:

- IDD_DIALOG_PPP: PPPDlg
- IDD_DIALOG_RTKLIB: RTKLIBDlg
- IDD_DIALOG_WAPPP: WAPPPDlg

After that, dialogs should be prepared to be used in GOCA GNSS Control.

8.2 Create new PPP mode in GOCA

New mode has to be added to PPP processing. The mode option has to be added in the dialog `IDD_SEL_MODE_TYPE` of GOCA GNSS Control. To do that, in the data section of properties of the `IDC_MODE` combo-box control, PPP mode has to be included by adding “;PPP” after what is currently written in this part (“RINEX-Datacollection;RINEX-Processing;RTK-Processing”). With that, this new mode will be the option 4.

8.3 Create code for PPP mode

In order to add PPP as new mode in GOCA GNSS Control, some code has to be added. This is done in the `MeasureList.cpp` and `MeasureList.h` files.

First of all, variables that are going to be used have to be defined in the header file. These variables, all of them with `CString` type, are: `m_messPPPProcessing`, `m_messPPPProcessingMode`, `m_messPPPProcessingModeCreated` and `m_messPPPProcessingModeEdited`.

After that, the values for these variables have to be initialized. The variable `m_messPPPProcessing` has to be initialized as: `m_messPPPProcessing = _T(“PPP-Processing”);`

The values of the other variables are set from the function `OnSetActive` as: `m_messPPPProcessingMode = _T(“PPP Processing Mode...”);` `m_messPPPProcessingModeCreated = _T(“PPP Processing Mode created”)` and `m_messPPPProcessingModeEdited = _T(“PPP Processing Mode edited”);`

In the function `DrawModeTable()`, new mode has to be added as one more if sentence: `if (pMode->m_iMode == 4) strMode = m_messPPPProcessing;`

Then all variables are ready for be used when creating, editing or deleting the new PPP mode. To do that, some code has to be added in these functions of the code.

First of all, the function responsible for adding a new mode is the one named `OnBnClickedModeAdd()`. In this function, new condition has to be set (`if (modus == 4)`). The process to follow is analogous to the other modes, but a little simpler as no time spans are used. An example of how this can be done is shown in Figure 57.

```

if (modus == 4) {
    // ...
    /* ... */
    PPPDlg dialogTest;
    int anzMode = (int)pApp->m_oaGNSSMode.GetSize(); // Previous mode number
    pMode->m_strName.Format(L"%s %d", m_messPPPProcessingMode, (anzMode + 1)); // "Processing mode PPP 4"
    CString m_strPPPMoName = pMode->m_strName;

    if(dialogTest.DoModal() == IDOK) // Start dialog and intercept OK
    {
        m_bDeleteMode = true;
        Sleep(100);
        // ...
        pMode->m_strName = m_strPPPMoName;
        pApp->m_oaGNSSMode.Add(pMode);
        pMode->m_strModeReport = m_messPPPProcessingModeCreated; // Report, "PPP Processing Mode created"
        pMode->m_iMode = modus; // Takes the mode control
        m_bDeleteMode = false;
    }
    else
    {
        pMode->DeleteMode(); // Modo de eliminación
        delete pMode;
    }
}
}

```

Figure 57: Code to add PPP mode

As can be seen in previous Figure, first of all a PPPDlg has to be created (so PPPDlg.h has to be included at the top of the file). After creating this dialog object, the number of the created mode has to be set and then the name of the mode has to be created using the name assigned in the variable `m_messPPPProcessingMode` plus the corresponding number.

Then, the dialog is opened and when the button OK is clicked, the mode is created and the corresponding message is shown in the report zone.

When the mode is edited, in the function `OnBnClickedModeEdit()` also a new condition to this new PPP mode has to be created. Following the structure of other modes, the code that has to be added is shown in Figure 58.

```

if (modus == 4)
{
    GOCA_PPPLg dialogTest;
    CString m_strPPPMoName = pMode->m_strName;

    if (dialogTest.DoModal() == IDOK)
    {
        m_bDeleteMode = true;
        Sleep(100);
        pMode->m_strName = m_strPPPMoName;
        pMode->m_strModeReport = m_messPPPProcessingModeEdited; // Report, "PPP Processing Mode edited"
        pMode->m_iMode = modus;
        m_bDeleteMode = false;
    }
    else
    {
    }
}
}

```

Figure 58: Code to edit PPP mode

8.4 Re-write run PPP function

After the mode is created, it has to run when the check box of the mode is checked (this is the reason why the main dialog is not used in case of integration with GOCA GNSS Control). To do that, the run function has to be re-written.

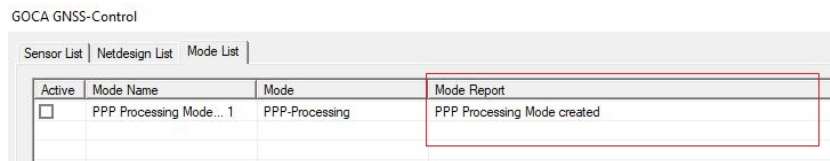
The mode is started with the function OnStartMode, in the file GNSSMode.cpp. There are different conditionals here, as the example in Figure 59. New condition has to be added, to check if the chosen mode is the PPP one (if ((pMode->m_iMode == 4))).

```
if((pMode->m_iMode == 2) && (pMode->m_pRinexProc != NULL))
{
    pMode->m_strModeReport = "start RINEX processing";
    if(pMode->m_pRinexProc->m_iProcessor == 0)
    {
        pMode->StartRINEXProcessing_NDT();
    }
    else
    {
        pMode->StartRINEXProcessing();
    }
}
```

Figure 59: If example to execute mode functions

Inside this if, the PPP run function has to be called. This function, that has to be re-written in the GNSSMode.cpp. The function, is the GOCA_PPPIlg::OnBnClickedButton-PppRun one in GOCA_PPPIlg.cpp file. This function has to be copied into a new function in the GNSSMode.cpp (called, for example StartPPPProcessing).

Besides copying the code, some things has to be modified to show messages in the corresponding place of GOCA GNSS Control interface (Figure 60) instead in the message box on PPP software.



Active	Mode Name	Mode	Mode Report
<input type="checkbox"/>	PPP Processing Mode... 1	PPP-Processing	PPP Processing Mode created

Figure 60: GOCA messages example

To do that, the command “m_messages.SetWindowText(“text”)” has to be replaced to “m_strModeReport = “text” ”, so all the messages shown in the PPP software will be shown in the corresponding place of GOCA GNSS Control.

As the progress bar is not needed, “m_PPP_progress.SetPos(i)” has to be removed or commented.

9. Field tests

As part of the objectives to be covered in the context of this master thesis different measurements have to be done in order to test the developed software and check the potential of PPP technic with real data and using a low cost receiver.

In this section, the methodology followed in each of the field test are exposed.

9.1 Test 1: 12-hour observation for post-processing

The first test consists on the performance of 12-hour measurement with the ublox ZED-F9P GNSS receiver and the NavXperience antenna, in the pillar number 300 (Figure 61) placed in the b-building of the Hochschule Karlsruhe Technick and Wirtschaft University of Applied Sciences.



Figure 61: Antenna at pillar 300

The date this observation and its characteristics is shown in Figure 62.

Data	GPS week	Day of year	Day of week
14/08/2019	2066	226	3

First observation (GPST)	Last observation (GPST)	Duration (h)	Time interval (s)
6:51:57	19:01:26	12:09:29	1

Figure 62: 12 hour observation information

How data was collected and the processing parameters chosen is explained in next sections.

9.1.1 Data collection

Data was taken using the module STRSVR of RTKLIB software, that is a communication server utility, with which user can configure input and output data stream [1]. So, with this raw data of the receiver can be stored in a file. There are two main options that has to be configured: the input as serial (the receiver) and the output as file (Figure 63).

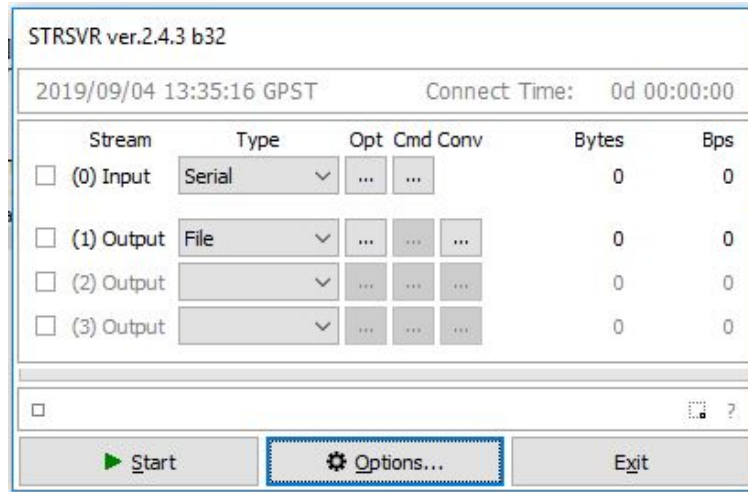


Figure 63: STRSVR configuration

In the input section, it is important to select the corresponding port of the receiver (this port automatically changes every time the receiver is connected to the computer) and the bit rate (Figure 64).

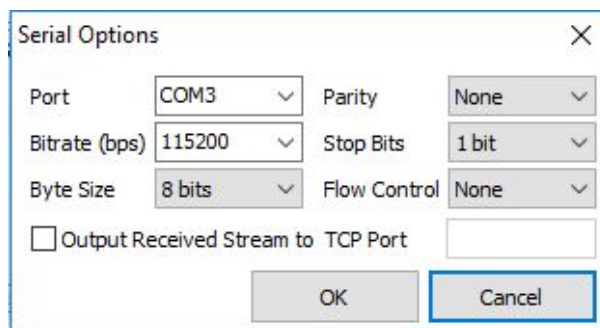


Figure 64: Input for STRSVR

In the output configuration, file in which raw data is going to be stored has to be defined (Figure 65). In this case, the file extension, as the observation was made with u-blox sensor, is .ubx.

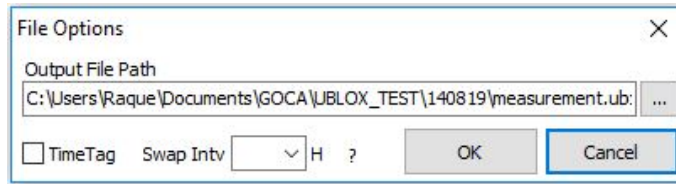


Figure 65: Output for STRSVR

After the measurement, an .ubx file is obtained. This file contains the raw data of the observation, in the u-blox specific format. In order to process this data in other software as RTKLIB and WaPPP, this file has to be converted into RINEX. To do this, another module of RTKLIB, called RTKCONV was used. This module allows different configurations (version of RINEX, observables to include, additional information to include in RINEX file, etc.). In this case, the main aspects of the configuration are:

- Station name fixed as p300 (pillar 300).
- RINEX version 2.11. Despite version 3 was also available, version 2.11 was chosen because WaPPP does not allow the use of RINEX 3.
- Antenna height was introduced, so it appears in the RINEX header.
- All constellations and observables available are used.

Once the RINEX was obtained, some additional information was manually added to the RINEX header, because the conversion program used does not include this information that is useful in the processing and also in the final GKA result file. This information is:

- Pillar name. Even it was indicated in RTKCONV and it is used in the RINEX file name, it was not included on it.
- Antenna model. This information is necessary to process the data with WaPPP software if antenna file wants to be use. If not, this software is not able to assign the corrections as doesn't know which antenna is it.
- Receiver type. This information is not compulsory, but as its included on GKA format it was indicated.

These items, with the antenna height can be shown in the RINEX header (Figure 66).

```

2.11          OBSERVATION DATA  M (MIXED)          RINEX VERSION / TYPE
RTKCONV 2.4.3 b32          20190815 080117 UTC      PGM / RUN BY / DATE
log: C:\Users\Raque\Documents\GOCA\UBLOX_TEST\140819\measu COMMENT
format: u-blox          COMMENT
p300          MARKER NAME
              MARKER NUMBER
              OBSERVER / AGENCY
              REC # / TYPE / VERS
              ANT # / TYPE
              APPROX POSITION XYZ
              ANTENNA: DELTA H/E/N
0.0000          0.0000          0.0000
0.2340          0.0000          0.0000
u-blox ZED-F9P
NAX3G+C          NONE

```

Figure 66: RINEX header

As mentioned above, data time interval is 1 second. This can make the processing slow, so in order to make tests easier and also have a comparison between different time intervals, RINEX was resampled to 30 seconds. To do this, TEQC software was used. This is a free software that allows [9]:

- Translate (convert) certain native binary formats to RINEX files.
- Check a RINEX file or files for compliance with the RINEX version 2 specification.
- Modify (edit) any existing RINEX header fields in a RINEX file.
- Quality check a valid RINEX.
- Window, cut, or splice two or more RINEX files.
- Create a new RINEX file with a longer sample interval.

Both the 1 second RINEX and the 30 second one have been processed. Results can be shown in results chapter.

Two different treatments are going to be done of this data: using ultra-rapid products (near real time processing) and final precise products. In order to process the data, different files have to be downloaded. These files and the source in which they are obtained is shown in Figure 67.

File	Name	Source
Ultra-rapid orbits	igu20663_18.sp3	gssc.esa.int/gnss/products
Final orbits	igs20663.sp3	gssc.esa.int/gnss/products
Clock file	igs20663.clk	gssc.esa.int/gnss/products
BLQ	p300.BLQ	Onsala webpage
Ultra-rapid EOP	igu20663_18.erp	gssc.esa.int/gnss/products
Final EOP	igs20667.erp	gssc.esa.int/gnss/products
IONEX file	igsg2260.19i	cddis.nasa.gov/gnss/products
DCB 1	p1c11907.cdb	igs.bkg.bund.de/dataandproducts
DCB 2	p1p21907.cdb	igs.bkg.bund.de/dataandproducts

Figure 67: Files to PPP processing

Not all previous files are needed, but they can be used depending on the processing parameters.

In addition, the RINEX observation file, navigation files and the antenna files (one specific for the receiver, and one general for satellites obtained from IGS) are going to be used in the processing.

9.1.2 Processing parameters

3 different processes have been done with each software engine (RTKLIB and WaPPP) using the developed PPP software. These are:

- Processing using final products.
- Processing using ultra-rapid products.
- Processing using ultra-rapid products and 30 second sample.

In order to achieve the best results, precise options have been chosen in all cases. These corrections have been explained at sections 4.4.1 and 4.3.1. That means:

- Use of precise information for orbits and clocks (no broadcast ephemerids are used).
- Use of iono-free lineal combination.
- Estimate tropospheric delay.
- Use of DCB files.
- Use of BLQ and EOP files (RTKLIB).

- Phase wind up correction and rejection of eclipsed satellites (RTKLIB).

Data has been processed using the developed software in the context of this Master Thesis. Example of configuration of PPP settings (Figure 68) and RTKLIB settings (Figure 69) are provided:

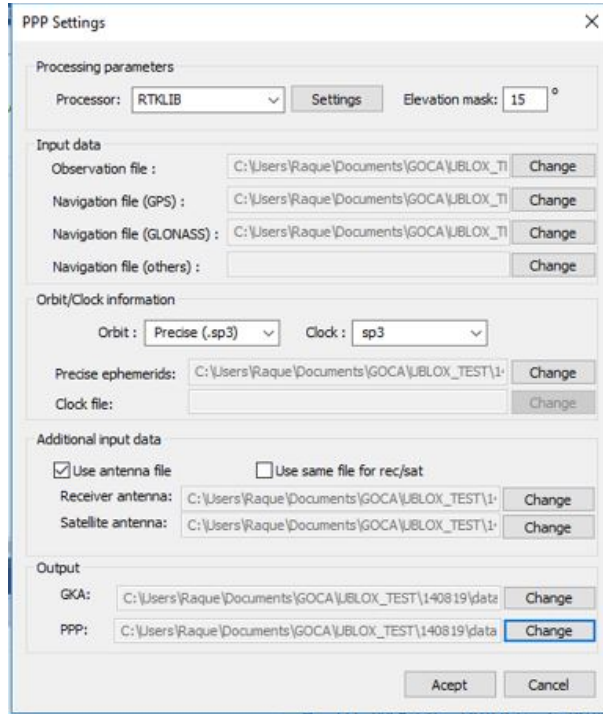


Figure 68: PPP settings example

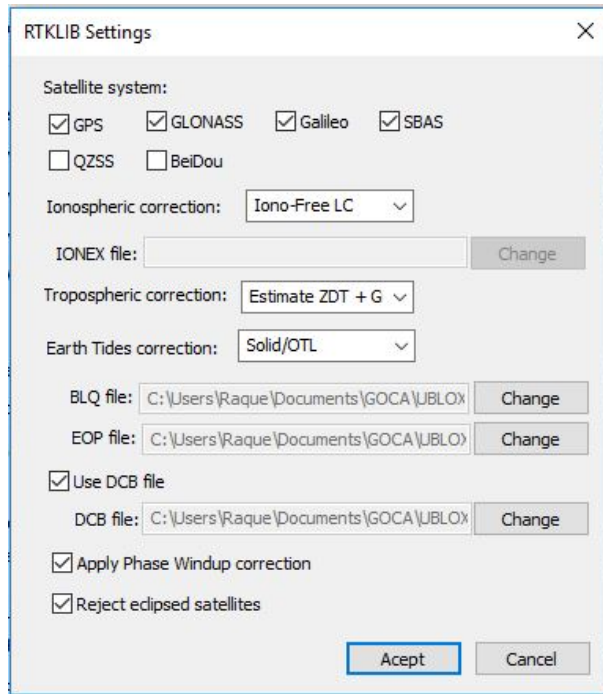


Figure 69: RTKLIB settings example

Processing with RTKLIB with 1 second RINEX takes about 8 minutes, while the same process with the 30 second RINEX takes less than a minute. In case of WaPPP the process is slower, taking even 20 minutes to process the 1 second RINEX and about 1 minute in case of 30 second resampled one.

9.2 Test 2: Real-time PPP processing

The second test consists on the performance of PPP real time processing also with the ublox ZED-F9P GNSS receiver and the NavXperience antenna, in the same pillar number 300.

The date this observation and its characteristics is shown in Figure 70.

Data	GPS week	Day of year	Day of week
24/09/2019	2072	267	2

First observation (GPST)	Last observation (GPST)	Duration (h)	Time interval solution (s)
9:26:36	10:56:43	1:30:08	0.2

Figure 70: Real-time observation information

9.2.1 Processing parameters

The aim of this test is to performance a real-time PPP processing, using SSR corrections, in order to analyze the potential of this technique together with the u-blox receiver and its later comparison with post-processing results. The software used to carry on this test is RTKLIB, specifically the module RTKNAVI.

When this test was performed, the real-time part of the developed PPP software was not available, as it was done after that in order to connect with the sensor using TPC-IP connection to test it on the TV tower of Stuttgart.

As input, the receiver has to be put as serial and, to get SSR corrections with NTRIP protocol, this has to be configured with a IGS NTRIP user in the section corrections of the input dialog (Figure 71).

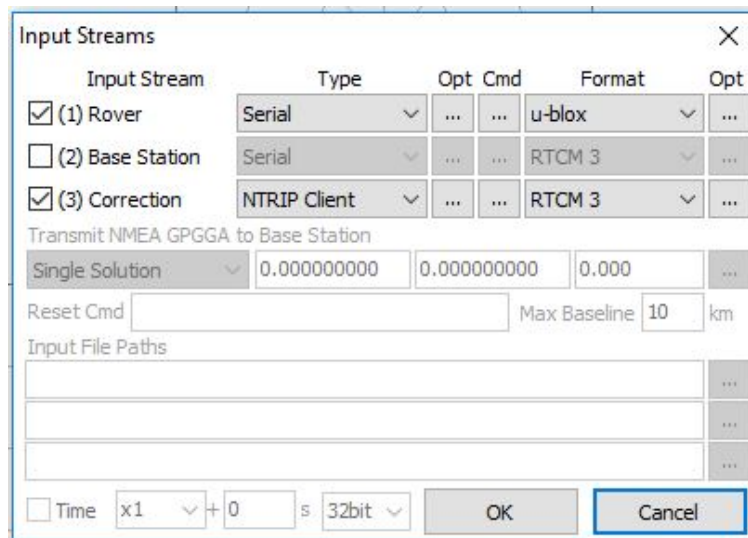


Figure 71: Real time observation input parameters

Main aspects of the configuration are shown in Figure 72. The most important parameters, based on previous literature review ([10], [26]), that have been used are:

- Positioning mode: PPP static.
- Apply Solid Earth corrections.
- Ionosphere correction: use iono-free combination.
- Troposphere correction: estimate ZTD and horizontal gradients.
- Satellite and clock information: broadcast + SSR APC. This is the configuration to use in order to apply SSR corrections to the orbit and clock information.

- Apply wind phase up correction.
- Reject eclipsed satellites.

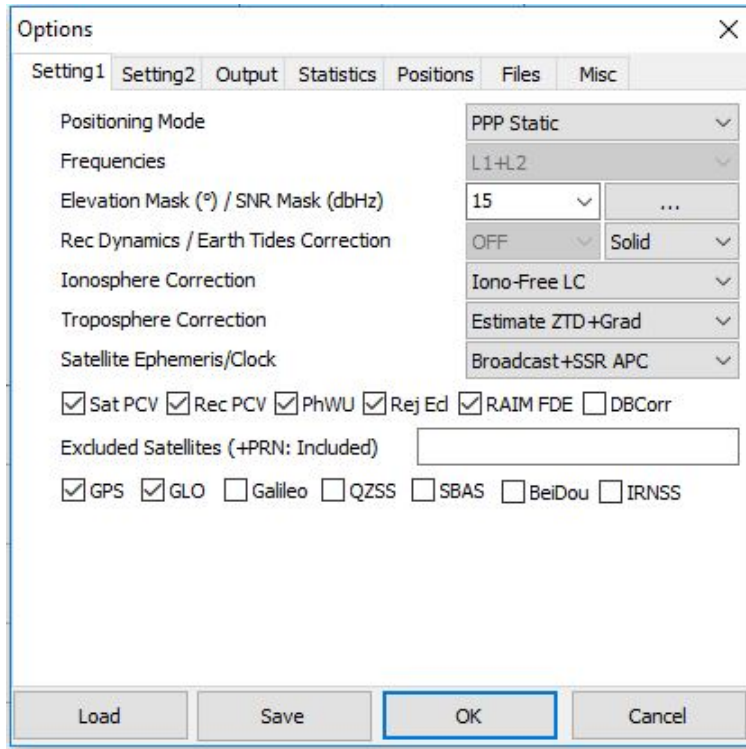


Figure 72: Real time observation parameters

In addition to previous parameters, also antenna files are used (Figure 73). For satellites, antenna file provided by IGS is used and for receiver, the specific file of it has been used.

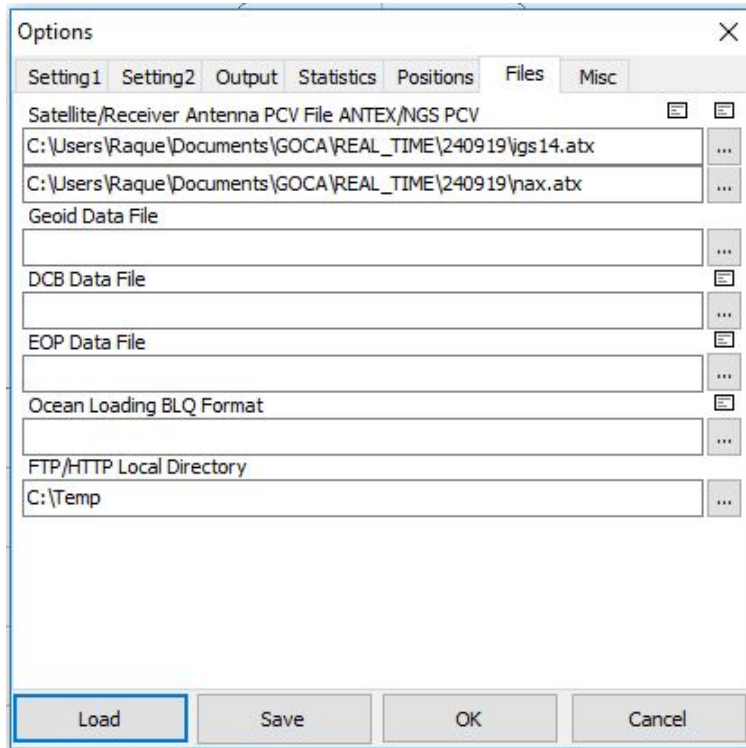


Figure 73: Real time observation files

Results

This chapter shows the result of the different test made under the context of this master thesis. These results are divided in post-processing observation and real-time observation.

In the part of post-processing, both software RTKLIB and WaPPP have been used, so both results are shown. Also, different products (final and ultra-rapid) where used, so also their results are exposed. All the parameters chosen for both kind of processing are explained in the previous chapter of development.

In this result section, numerical and graphical results are shown. Later, in the chapter of discussion, a more in-depth analysis of these results will be shown.

In all the processing approaches that have been done, apart from the position file in the own format of RTKLIB or WaPPP from which the graphs have been build, a GKA file is obtained. Figure 74 shows an example of the obtained file. The information included on it is described in section 6.2.

```
#GOKA14
p300,u-blox ZED-F9P      ,NONE,2066,3,68486,4146282.0740,611664.0784,4791922.5782,0.2340,0,0,0
0.001,49,1.69,32.49,49,4.41,39.69
#END14
```

Figure 74: GKA output file example

10. Test 1: 12-hour observation

First test, according to section 9 was 12 hour observation. The results of the processing of this data is shown in this section.

10.1 RTKLIB as software engine

10.1.1 Ultra-rapid products

Results in terms of coordinates (cartesian and geodetic) and their corresponding errors are shown in Figure 75.

X	Y	Z
4146282.084	611664.120	4791922.589
dX	dY	dZ
0.007	0.004	0.006

lat	lon	h
49.016725	8.391811	190.478
d N	d E	d U
0.004	0.005	0.008

Figure 75: Coordinates and errors RTKLIB with ultra-rapid products

The evolution of geocentric coordinates along observation time obtained after processing the 12 hour observation data with RTKLIB using ultra-rapid products are shown in Figure 76. Coordinates along the observation time. Coordinates are shown in meters.

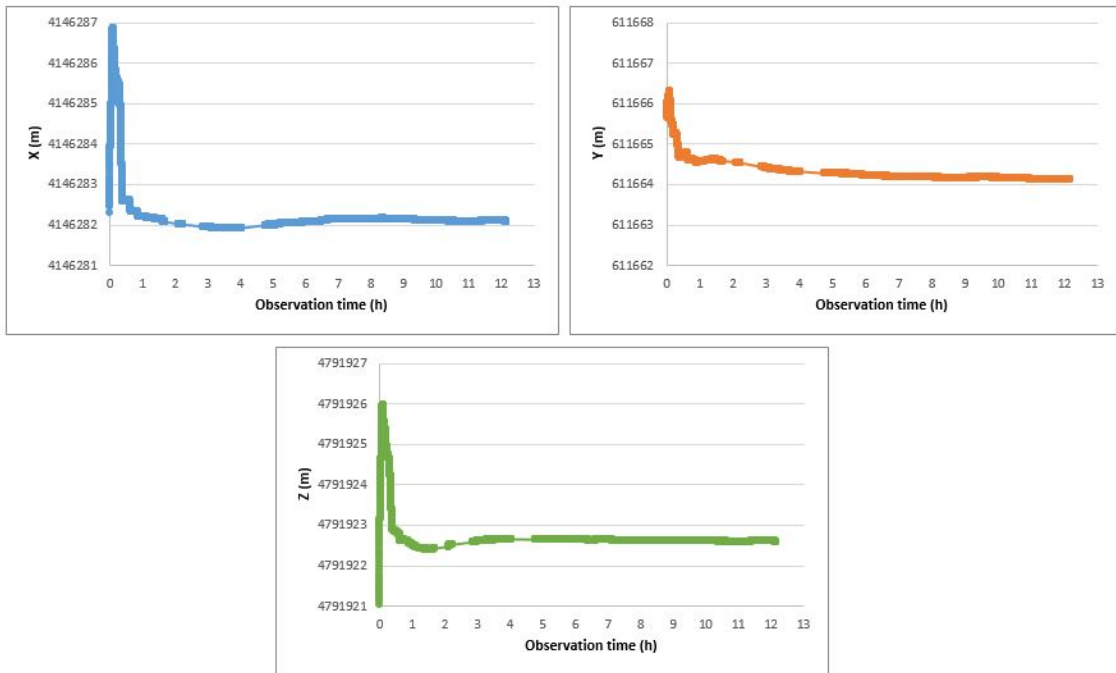


Figure 76: Coordinates along time RTKLIB with ultra-rapid products

In general coordinates seem to get stable after the first hour of observation. As in first epochs the result changes considerably, Figure 77 has been done, representing the same of the first one but in this case the Y axis shown includes only a range of 50 centimetres.

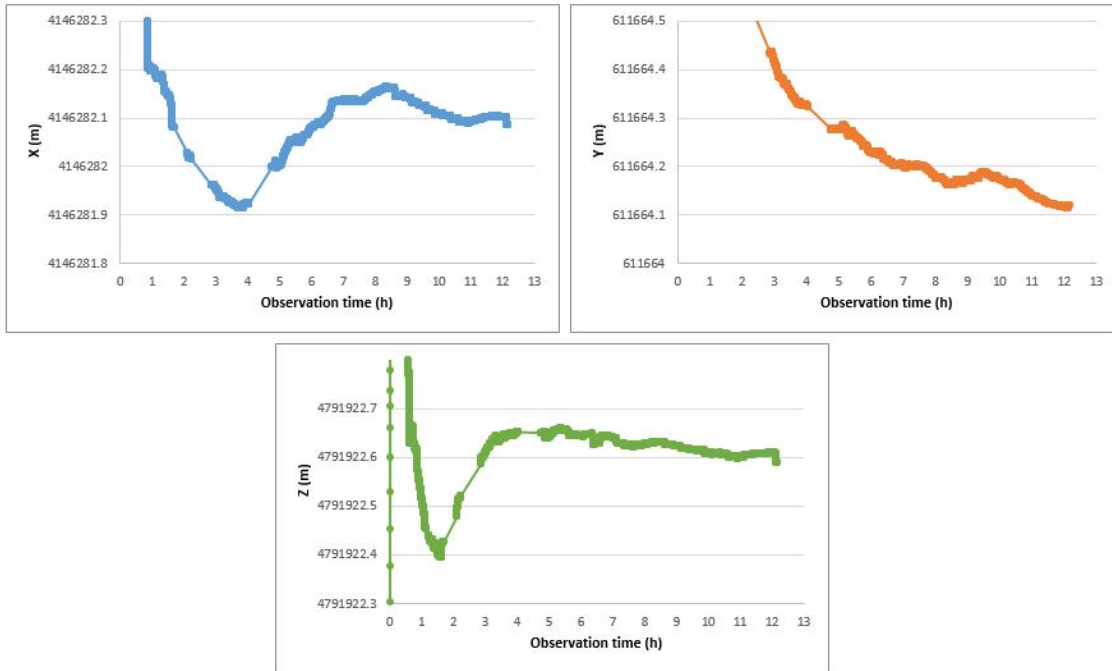


Figure 77: Coordinates along time RTKLIB with ultra-rapid products (detail)

According to previous Figure, coordinates get stable in a range of 50 centimetres after 1 hour of observation in case of X and Z and after 3 hours in Y coordinate. X coordinate gets stable in a range of 20 centimetres after 5 hours and around 10 centimetres after 7 hours. Y coordinate also seem stable at 20 centimetre level after 5 hours and the range of 10 centimetres is achieved after 7-8 hours, but the final hours are less stable than in case of the other coordinates. Z coordinate is stable at 10 centimetres level in approximately 3 hours.

Usually, in PPP processing, it is important to see the evolution of the error along time. This is shown in Figure 78. This error in first minutes of observation is quite high (around 4 meters) so the graph can not be fully interpreted.

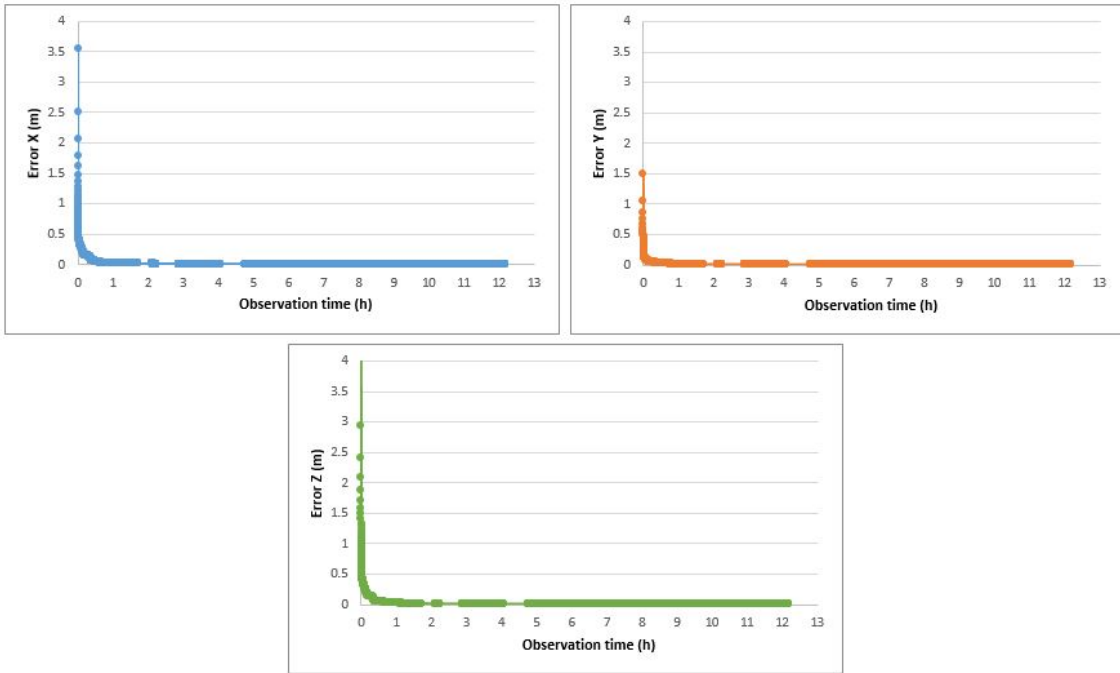


Figure 78: Errors along time RTKLIB with ultra-rapid products

In order to see errors in more detail, Figure 79 shows again the evolution of error along time but in this case with a more detailed Y axis.

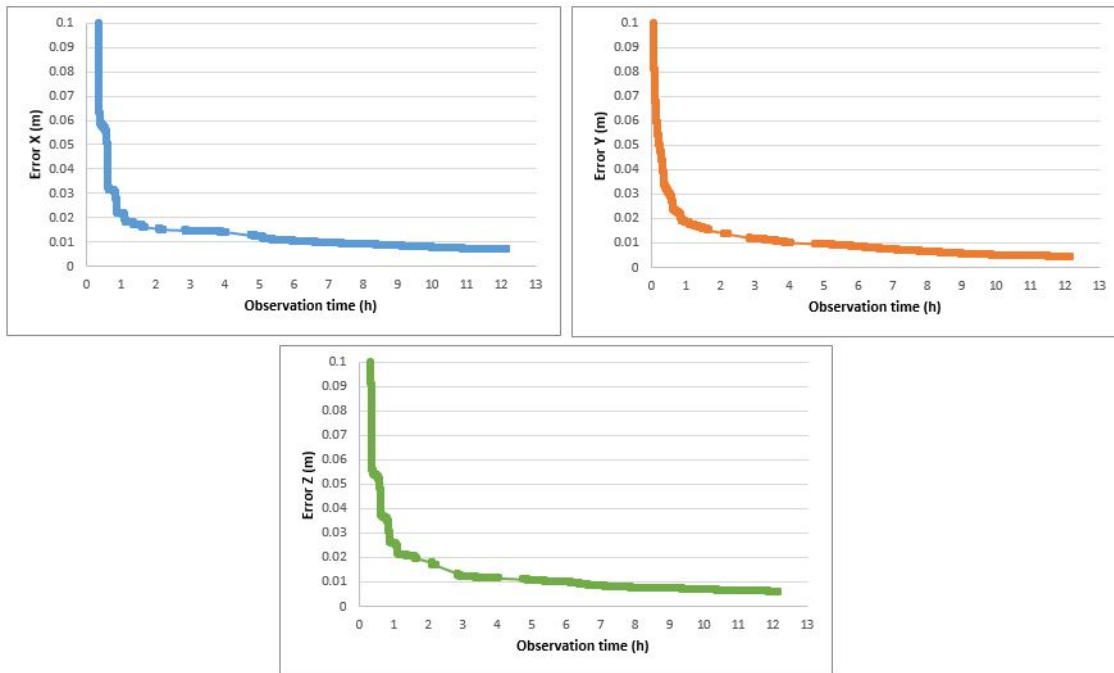


Figure 79: Errors along time RTKLIB with ultra-rapid products (detail)

After the first hour of observation, the estimated error is less than 2 centimetres and it remains approximately in 1 centimetre level from 5-6 hours, even less after 9 hours of observation.

RINEX resampled to 30 seconds

As mentioned in section 9.1, processing 1 second interval RINEX with 12 hour data can take high time. In order to test how the time interval affects the results, the same processing but this time with the resampled RINEX has been done. Results are shown in Figure 80.

X	Y	Z
4146282.009	611664.287	4791922.545
dX	dY	dZ
0.015	0.010	0.015

lat	lon	h
49.016725	8.391813	190.412
d N	d E	d U
0.006	0.010	0.020

Figure 80: Coordinates and errors RTKLIB with ultra-rapid products - 30 seconds

The evolution of coordinates with time in this case is shown in Figure 81. This graph is detailed at 50 centimetres in Y axis, the complete one is shown in Annex B (same with all the graphs from here on).

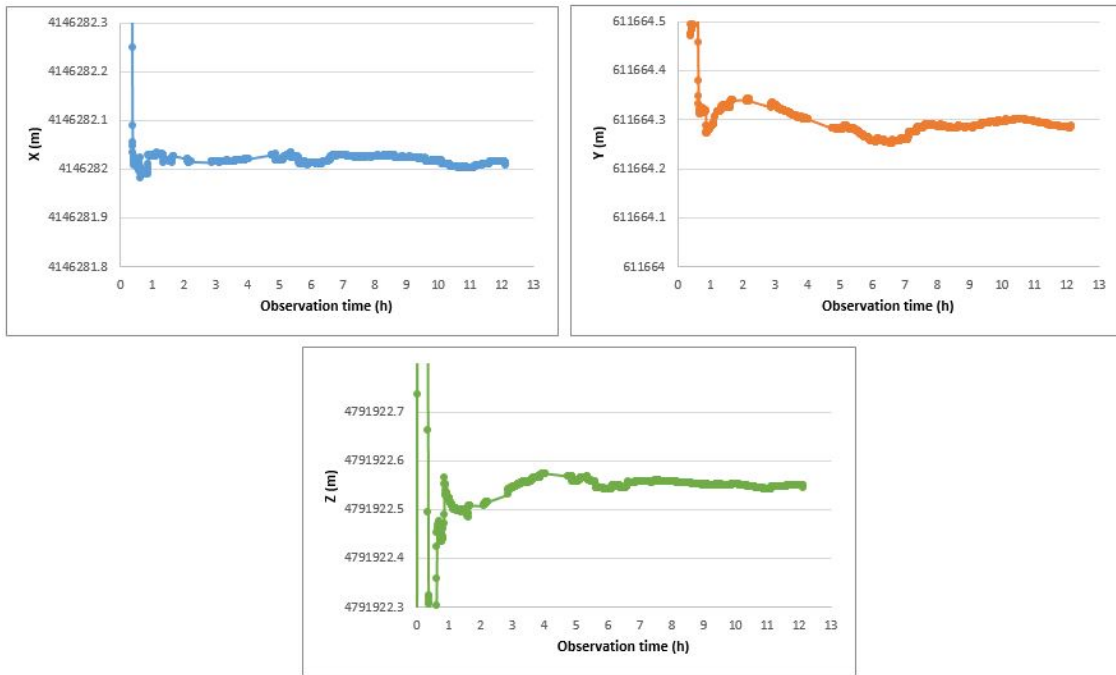


Figure 81: Coordinates along time RTKLIB with ultra-rapid products - 30 seconds (detail)

In this case, coordinates are more stable along time. After one hour of observation, coordinates are stable at around 10 centimetres and even better after 4-6 hours.

The evolution of errors along time in this case is shown in Figure 82 (complete one in Annex B).

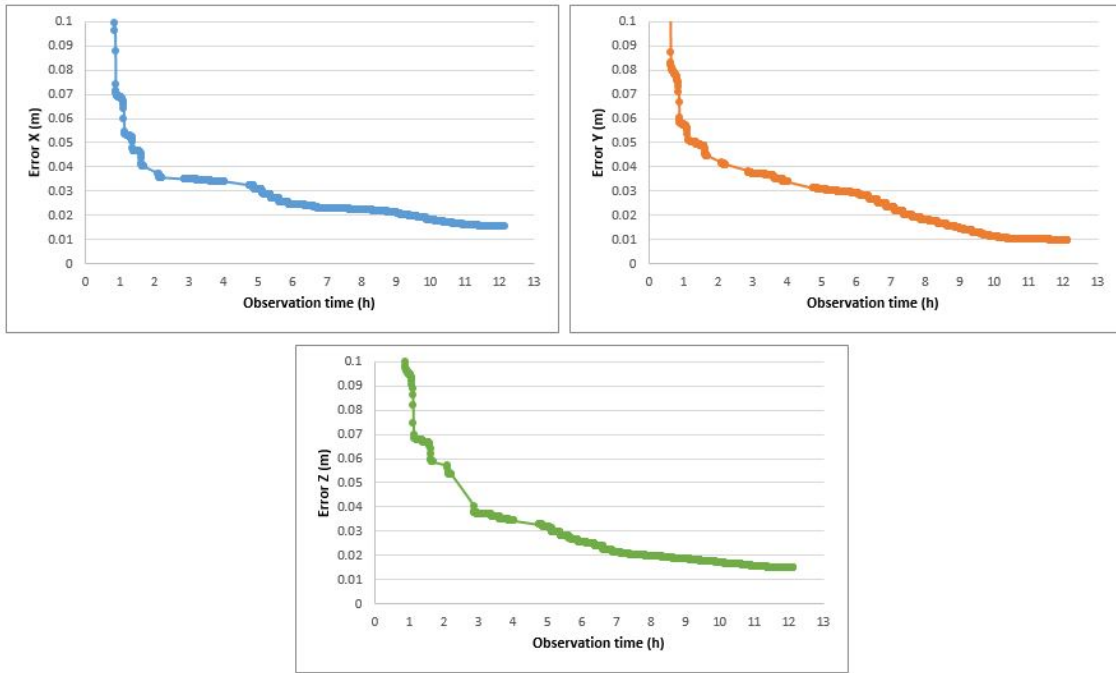


Figure 82: Errors along time RTKLIB with ultra-rapid products - 30 seconds (detail)

Errors have a value around 4 centimetres after 2 hours and they decrease to 2 centimetres after 7 hours of observation. In case of Y coordinate, it is around 1 centimetre from 10 hours on.

10.1.2 Final products

In this section results of using final products are shown. Complete graphs are included in Annex B. The obtained results are shown in Figure 83. Both geocentric and geodetic coordinates are shown.

X	Y	Z	lat	lon	h
4146282.074	611664.078	4791922.578	49.016725	8.391810	190.459
dX	dY	dZ	d N	d E	d U
0.007	0.005	0.006	0.004	0.005	0.009

Figure 83: Coordinates and errors RTKLIB with final products

Coordinates are shown in cartesian and geodetic coordinates, but all graphs are shown in cartesian ones. Figure 84 shown the evolution of coordinates along time.

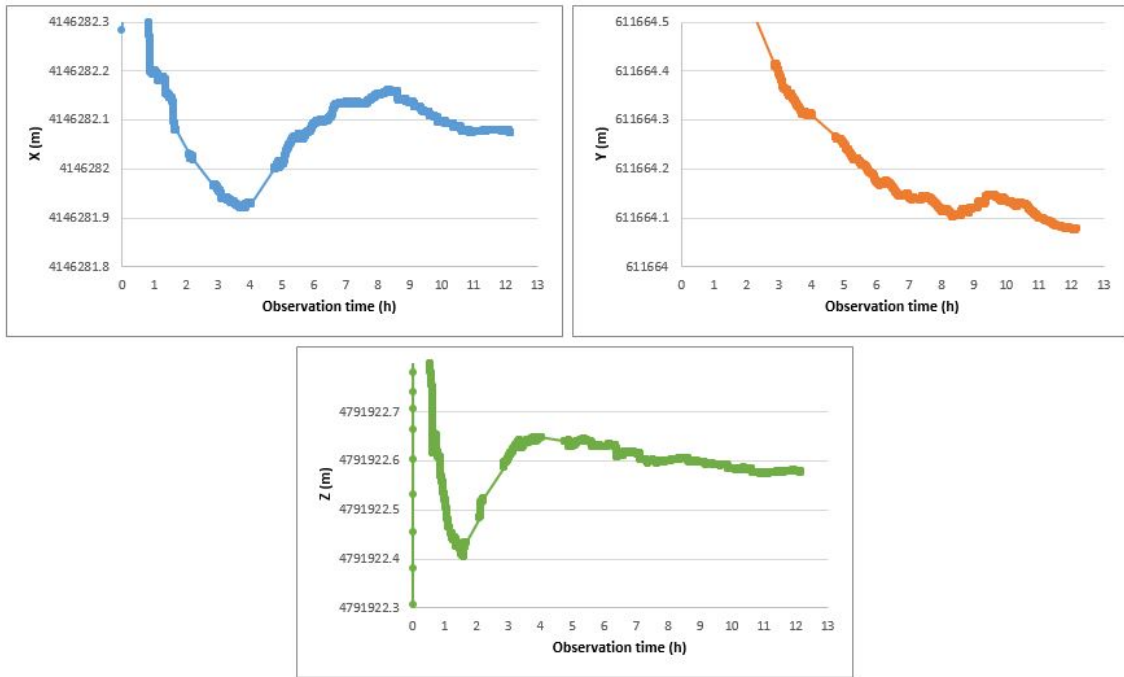


Figure 84: Coordinates along time RTKLIB with final products (detail)

As it happened in the case of ultra-rapid products, X and Z coordinate get stable at 50 centimetres level after 1 hour and in case of Y this time is around 3 hours. Coordinate X is stable at 20 centimetres level after 20 centimetres and the stability gets better (10-15 centimetres) in 7 hours. Case of Y coordinate is quite similar, 20 centimetres of variation after 5 hours and 10-12 centimetres in 7 hours of observation. Coordinate Z gets a 10-12 centimetres stability after 3 or 4 hours.

Evolution of errors is shown in Figure 85.

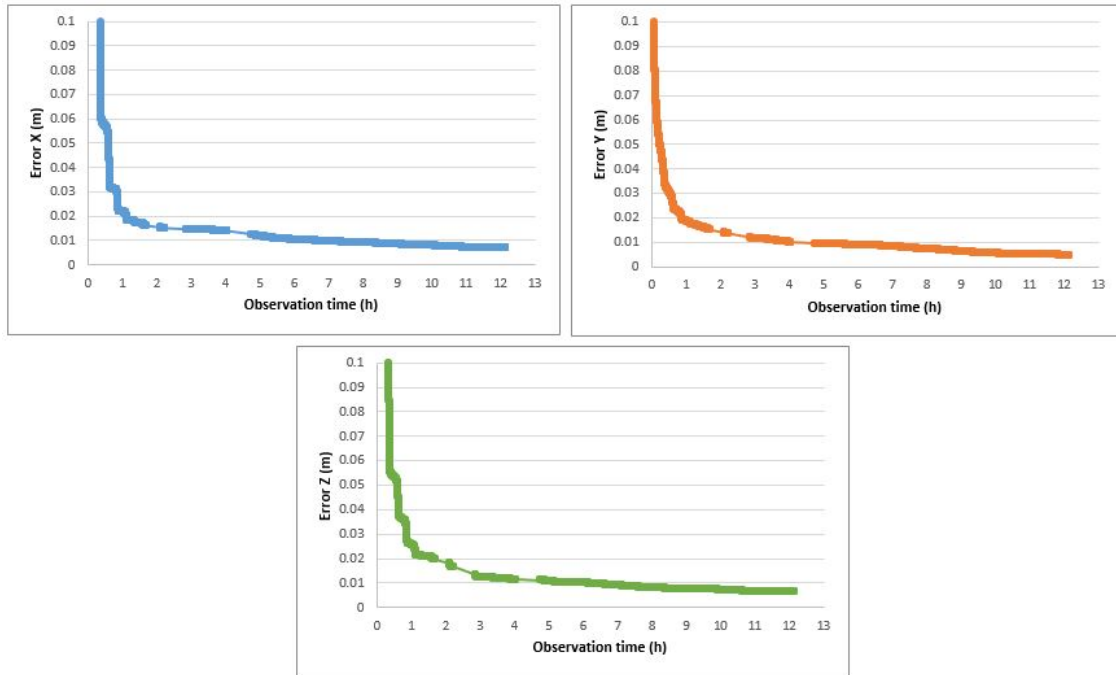


Figure 85: Errors along time RTKLIB with final products (detail)

After the first hour of observation errors are less than 2 centimetres and after 5 hours from the beginning they are around 1 centimetre, even less in case of Y coordinate.

10.1.3 RTKLIB results comparison

Previous sections show the results of different processing with RTKLIB. In order to compare them easier, all graphs have been put together: results with final products, ultra-rapid and ultra-rapid with a resample of 30 seconds. The full comparison, including WaPPP software and real-time is done in section 12.

Figure 86 shows the obtained Cartesian coordinates (detailed in Y axis). Red line represents the reference coordinates of the pillar number 300. From these graphs can be pointed out that, in general, the result with final and with ultra-rapid products is really similar.

In case of X coordinate, both final and ultra-rapid products provide almost the same result, but at the end of the observation little differences can be seen, being final products the ones that are closer to the reference X. With ultra-rapid products 30 seconds resampled, obtained X seems to be more stable, but its value is below the reference one. In case of Y coordinate, is the ultra-rapid 30 seconds resampled computation is the one that provides a closer value to the reference one. At last, Z coordinate is

similar to the case of X coordinate, being ultra-rapid product result the one that is closer to reference coordinate and ultra-rapid resampled one is under the reference value.

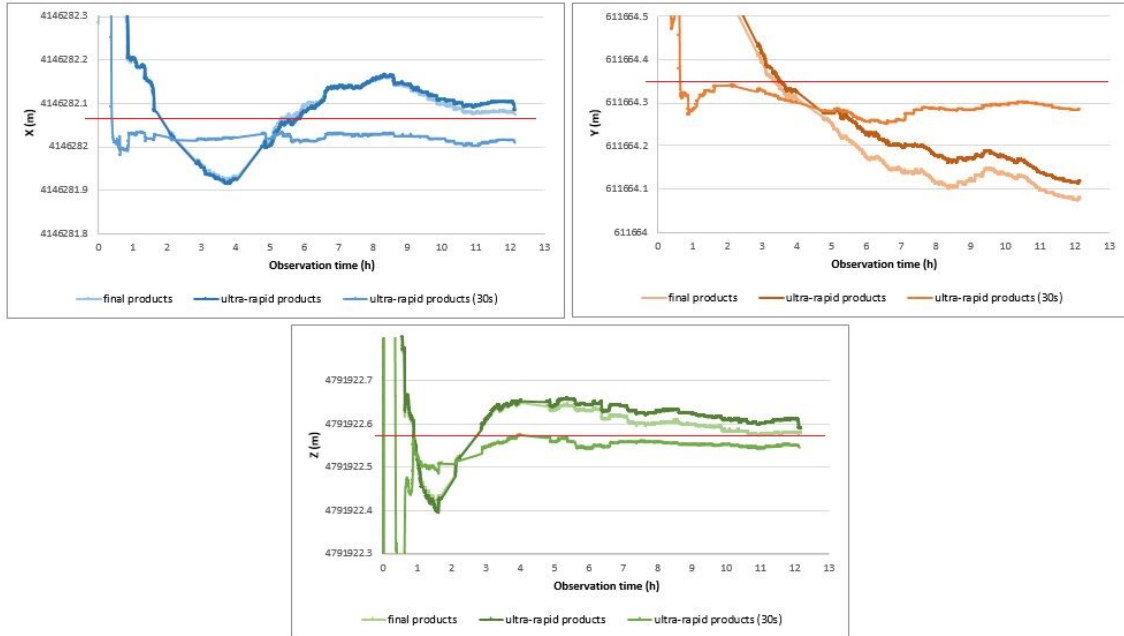


Figure 86: Cartesian coordinates pillar 300 with RTKLIB

Figure 87 shows the corresponding obtained errors. As can be seen, final and ultra-rapid products provide almost the same result while the resampled one has always a higher value and it takes more time to be stable as the other two.

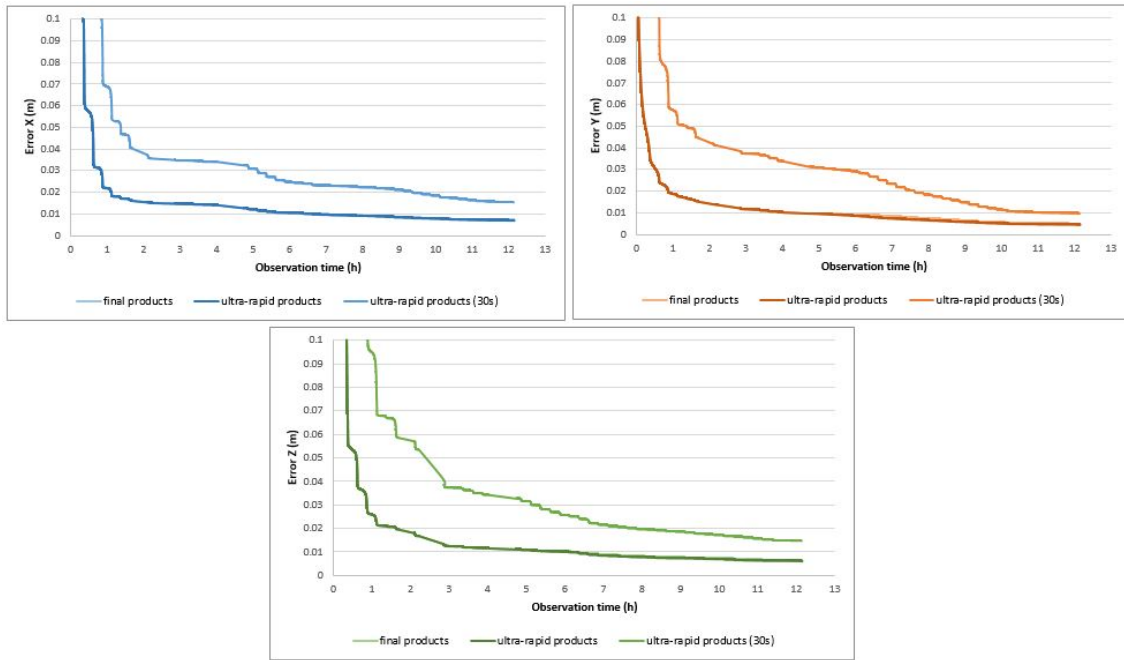


Figure 87: Errors in coordinates pillar 300 with RTKLIB

10.2 WaPPP as software engine

10.2.1 Ultra-rapid products

Results obtained when processing the data with WaPPP and ultra-rapid products are shown in Figure 88.

X	Y	Z
4146282.137	611664.338	4791922.651
dX	dY	dZ
0.049	0.031	0.031

lat	lon	h
49.016725	8.391813	190.580
d N	d E	d U
0.029	0.029	0.052

Figure 88: Coordinates and errors RTKLIB with ultra-rapid products

Evolution of coordinates obtained in each epoch can be seen in Figure 89. No tendency can be extracted from the graph and in the whole time of observations coordinates vary less than 1.5 meters. This does not provide useful information about convergence.

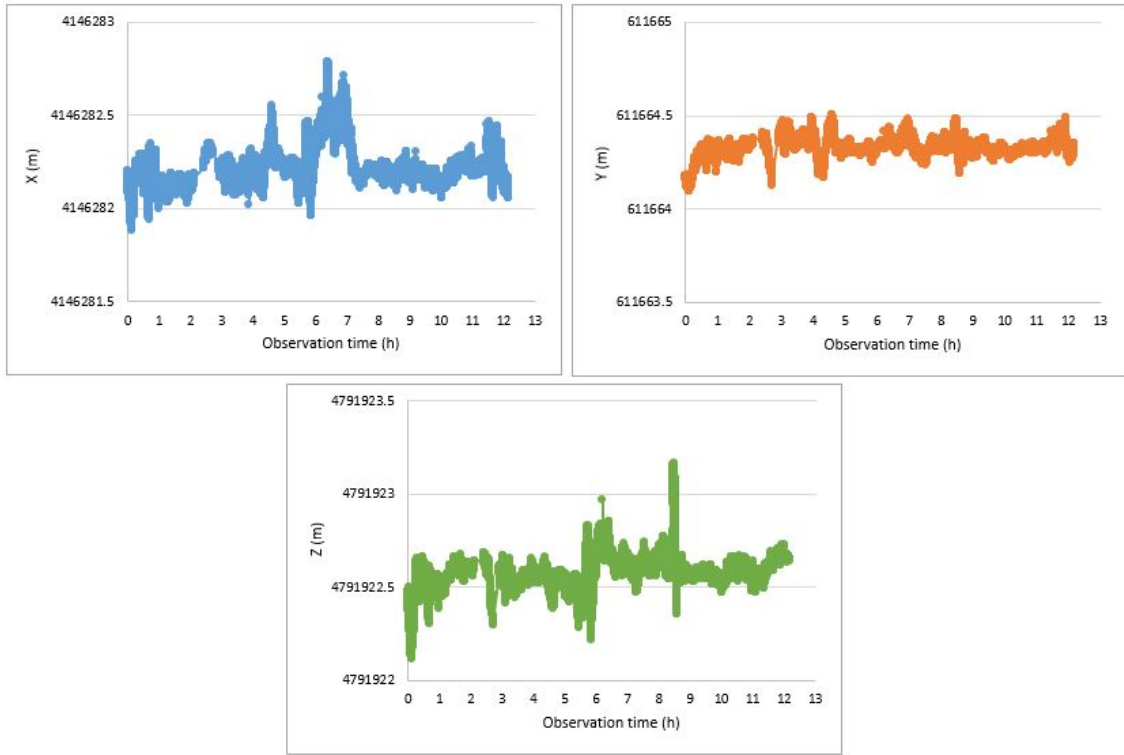


Figure 89: Coordinates along time WaPPP with ultra-rapid products

The same occurs with the variation of error along time (Figure 90). Errors are under 8 centimetres in that main part of time and do not exceed 18 centimetres at any epoch. Again this kind of graph is not really useful, so they are not included in further WaPPP sections.

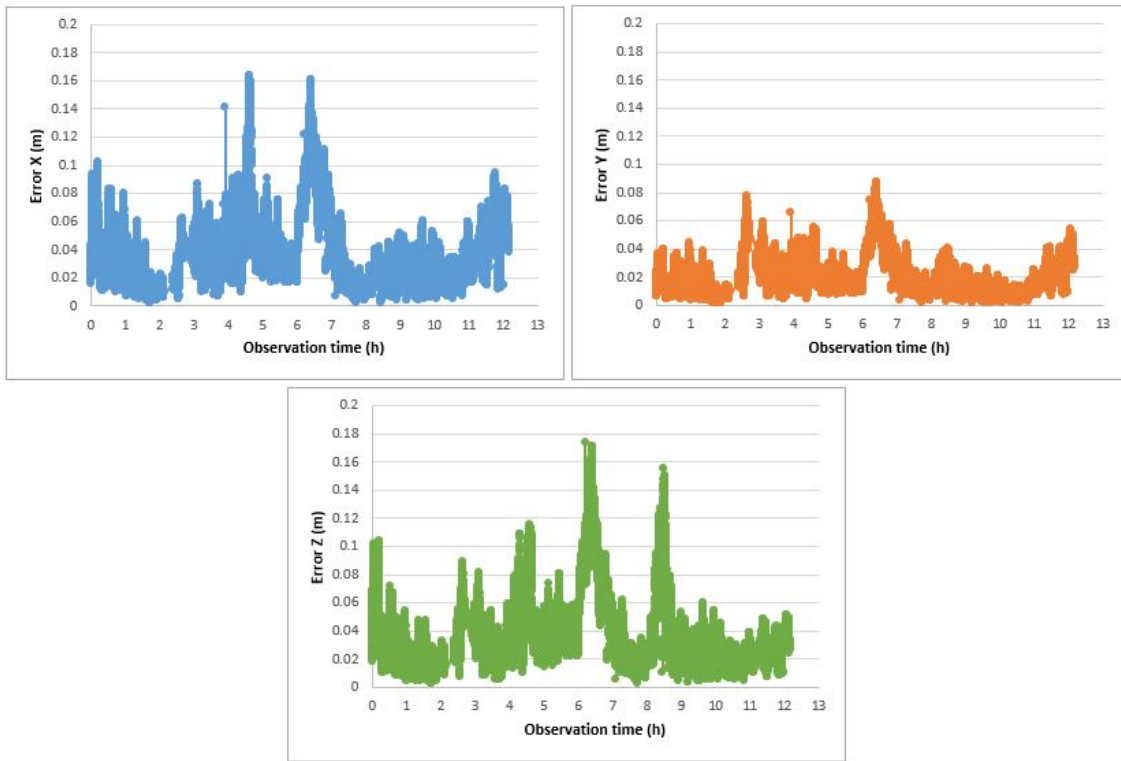


Figure 90: Errors along time WaPPP with ultra-rapid products

RINEX resampled to 30 seconds

When processing the resampled RINEX, obtained values are almost the same than using 1 second RINEX. Results are shown in Figure 91.

X	Y	Z	lat	lon	h
4146282.162	611664.331	4791922.670	49.016725	8.391813	190.609
dX	dY	dZ	d N	d E	d U
0.049	0.032	0.031	0.029	0.029	0.052

Figure 91: Coordinates and errors RTKLIB with ultra-rapid products - 30 seconds

Corresponding graphs, as they don't provide additional information, are included in Annex B.

10.2.2 Final products

Results obtained with WaPPP as software engine when using final products are shown in Figure 92.

X	Y	Z	lat	lon	h
4146282.042	611664.394	4791922.508	49.016724	8.391814	190.416
dX	dY	dZ	d N	d E	d U
0.037	0.024	0.023	0.021	0.022	0.039

Figure 92: Coordinates and errors RTKLIB with final products

Accuracy obtained when using final products is better than the one obtained when using ultra-rapid products also in case of WaPPP software. However, graphs are not relevant so they have been included in Annex B.

11. Test 2: real-time processing

The obtained results from the real-time PPP processing (section 9.2) are shown in Figure 93. The results are latitude, longitude and height and the corresponding errors.

Latitude (deg)	Longitude (deg)	Height (m)
49.016725	8.391811	190.170

sdn (m)	sde (m)	sdu (m)
0.008	0.008	0.019

Figure 93: Real time processing result

Estimated accuracy in North and East directions is 8 millimetres and in Up direction (corresponding to height) is about 2 centimetres.

The evolution of the coordinates along the observation time is shown in Figure 94. The red line represents the reference coordinates of pillar 300.

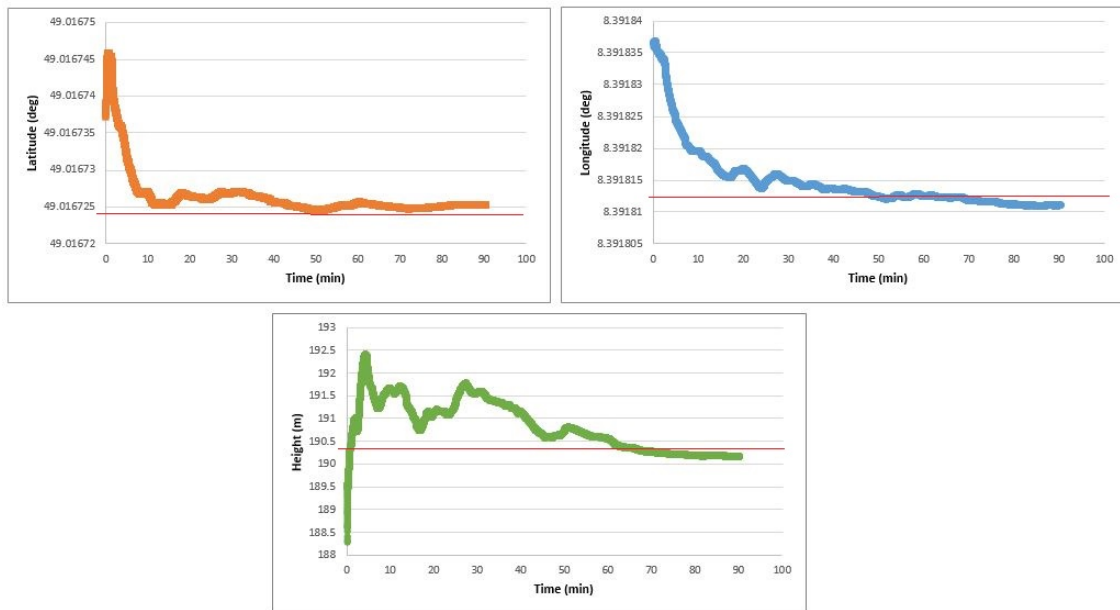


Figure 94: Real time processing result graph

As can be seen in above graph, both latitude and longitude get practically stable, under approximately 5 cm level, after 30 minutes of observation. However, height is less stable at it needs about 60 minutes of observation to get more or less stable in a

decimetre level.

The evolution of errors is shown in detail in Figure 95 (complete one in Annex B). Errors in horizontal coordinates are under 5 centimetres after 10 minutes of observation and get around 1 centimetre in 40-50 minutes. Height error is higher, under 15 centimetres after first 15 minutes and it takes 50 minutes to get under 5 centimetres. Finally, it is about 2 centimetres.

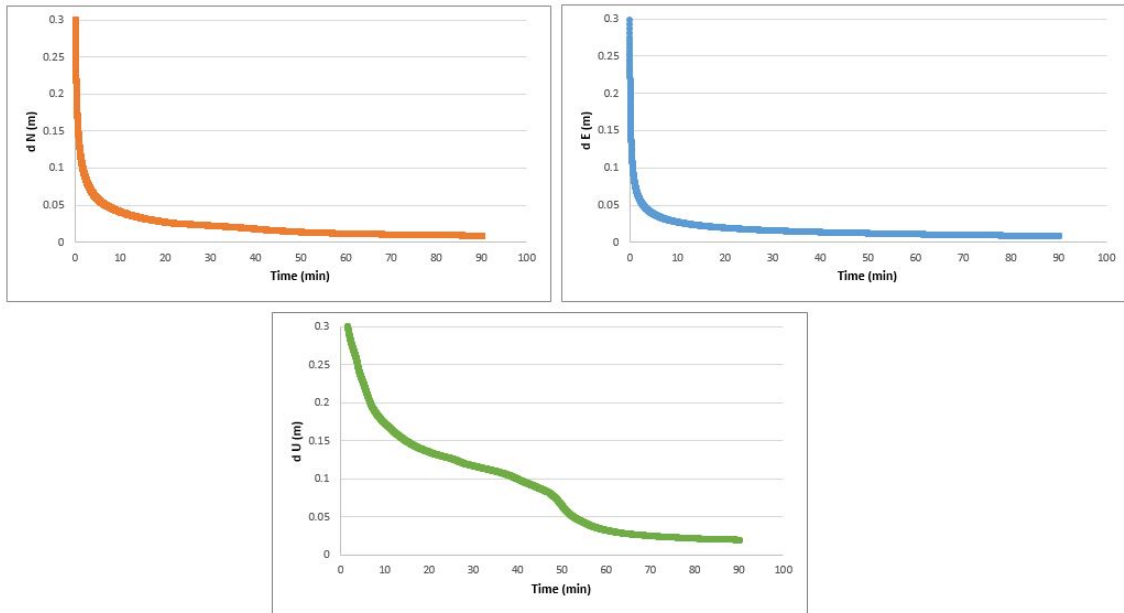


Figure 95: Real time processing result errors graph

12. Result comparison

In previous sections, obtained results using both RTKLIB and WaPPP software with ultra-rapid and final products have been shown. In order to analyse them deeper, in this section a comparison will be made between them. All coordinates are going to be compared with reference coordinates, obtained in a test field done by HSKA Laboratory on GNSS and Navigation, processed with an online PPP service, in the reference frame ITRF2014.2019.08.

Figure 96 shows all the obtained coordinates, included ones with RTKLIB and WaPPP in post-processing and also the ones obtained in the real-time observation.

	RTKLIB			WAPPP		
	X	Y	Z	X	Y	Z
Final products	4146282.074	611664.078	4791922.578	4146282.042	611664.394	4791922.508
Ultra-rapid products	4146282.084	611664.120	4791922.589	4146282.137	611664.338	4791922.651
Ultra-rapid products - 30s	4146282.009	611664.287	4791922.545	4146282.162	611664.331	4791922.670
Real-time	4146281.874	611664.121	4791922.362			

Reference coordinates ITRF2014.2019.08		
X	Y	Z
4146282.072	611664.334	4791922.591

Figure 96: Cartesian coordinates pillar 300

In case of RTKLIB processor, differences between obtained coordinates and reference ones are:

- X coordinate:
 - Final products: 2 mm.
 - Ultra-rapid products: 14 mm.
 - Ultra-rapid products 30 seconds: 63 mm.
 - Real-time: 198 mm.

So the most similar is the one with final products and the one that differs the most corresponds to the real time observation.

- Y coordinate:
 - Final products: 256 mm.
 - Ultra-rapid products: 214 mm.
 - Ultra-rapid products 30 seconds: 47 mm.

- Real-time: 213 mm.

This coordinate differs more with the reference one, being the most similar the one corresponding to the resampled RINEX and ultra-rapid products.

- Z coordinate:
 - Final products: 13 mm.
 - Ultra-rapid products: 2 mm.
 - Ultra-rapid products 30 seconds: 46 mm.
 - Real-time: 229 mm.

Real-time is again the one that differs more with respect the reference coordinate and, in this case. the ultra-rapid results are the most similar, but the final products also provide a quite similar solution.

In case of WaPPP processor, differences between obtained coordinates and reference ones are:

- X coordinate:
 - Final products: 30 mm.
 - Ultra-rapid products: 65 mm.
 - Ultra-rapid products 30 seconds: 90 mm.
- Y coordinate:
 - Final products: 50 mm.
 - Ultra-rapid products: 4 mm.
 - Ultra-rapid products 30 seconds: 3 mm.
- Z coordinate:
 - Final products: 83 mm.
 - Ultra-rapid products: 60 mm.
 - Ultra-rapid products 30 seconds: 79 mm.

It can be pointed out that the results in WaPPP are more stable, that means that all coordinates differ more or less the same with respect the reference ones, while RTKLIB provides more similar X and Z coordinates but in Y coordinate the results is more different. In terms of coordinates, there is not a clear pattern about what results are better, except of the case of real-time, that is always more different to the

reference coordinates that in any other case.

Corresponding errors are shown in Figure 97. In case of RTKLIB, both final and ultra-rapid products provide errors between 4 and 7 millimetres. With the resampled RINEX, errors get between 10 and 15 millimetres. In case of WaPPP, errors with final products are between 23 and 37 millimetres, between 31 and 49 in case of ultra-rapid products and also between 31-49 in the resampled RINEX case.

	RTKLIB			WAPPP		
	dX	dY	dZ	dX	dY	dZ
Final products	0.007	0.005	0.006	0.037	0.024	0.023
Ultra-rapid products	0.007	0.004	0.006	0.049	0.031	0.031
Ultra-rapid products - 30s	0.015	0.010	0.015	0.049	0.032	0.031

Figure 97: Cartesian coordinates errors pillar 300

Despite results of PPP are usually give as Cartesian coordinates, it is more intuitive to see geodetic coordinates, because we can analyse separately horizontal and vertical components, so this are also included in this section (Figure 98). In this case, the obtained results with the data of the 12 hour measurement processed using the CSRS PPP online service (<https://webapp.geod.nrcan.gc.ca/geod/tools-outils/ppp.php>) are also included.

	RTKLIB			WAPPP		
	lat	lon	h	lat	lon	h
Final products	49.016725	8.391810	190.459	49.016724	8.391814	190.416
Ultra-rapid products	49.016725	8.391811	190.478	49.016725	8.391813	190.580
Ultra-rapid products - 30s	49.016725	8.391813	190.412	49.016725	8.391813	190.609
Real-time	49.016725	8.391811	190.170			

Reference coordinates ITRF2014.2019.08		
lat	lon	h
49.016725	8.391813	190.491

	CSRS-PPP					
	lat	lon	h	d N	d E	d U
Final products	49.016725	8.391813	190.466	0.007	0.011	0.018
Ultra-rapid products	49.016725	8.391813	190.467	0.007	0.011	0.018

Figure 98: Geodetic coordinates pillar 300

Latitude and longitude are really similar in all results. In case of latitude, the only one that differs in the sixth decimal corresponds to WaPPP processor with final products. In case on longitude, there are some little differences in RTKLIB and also in WaPPP (final products).

Main differences can be noticed in height. Differences between reference one and obtained are:

- CSRS service, both final and ultra-rapid products: 24-25 mm.
- RTKLIB with final products: 32 mm.
- RTKLIB with ultra-rapid products: 13 mm.
- RTKLIB with ultra-rapid products - 30 seconds: 79 mm.
- RTKLIB real-time: 321 mm.
- WaPPP with final products: 75 mm.
- WaPPP with ultra-rapid products: 75 mm.
- WaPPP with ultra-rapid products - 30 seconds: 75 mm.

The corresponding errors, in N,E and U directions, are shown in Figure 99.

	RTKLIB			WAPPP		
	d N	d E	d U	d N	d E	d U
Final products	0.004	0.005	0.009	0.021	0.022	0.039
Ultra-rapid products	0.004	0.005	0.008	0.029	0.029	0.052
Ultra-rapid products - 30s	0.006	0.010	0.020	0.029	0.029	0.052
Real-time	0.008	0.008	0.019			

Figure 99: Geodetic coordinates errors pillar 300

The analysis of the meaning of all these results is made in the Discussion section.

Discussion

In the previous part, the different results obtained from the PPP processing using different products and software engines are shown. In this section, those results are going to be analysed.

First of all, it is notorious that the resulting graph of the processing using RTKLIB is not the same that the one obtained using WaPPP software. The result of the first one, RTKLIB, provides a kind of graph, with the traditional appearance of PPP graphs presented in papers and books (example in Figure 100). In this kind of graphs, it can be seen how obtained coordinates get more and more stable along time, and the same with the corresponding error, that is every time lower and more stable. However, WaPPP result graph is not like that and the in the obtained graph no pattern can be interpreted.

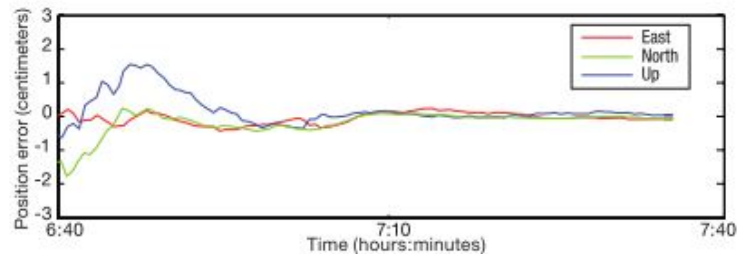


Figure 100: PPP error graph example [3]

RTKLIB uses extended Kalman filter, so the unknown parameters including the receiver position and velocity, the receiver clock bias, the troposphere parameters and the ionosphere-free LC carrier-phase biases are estimated with a measurement vector in a specific epoch [1]. This can explain the resulting graph, as each new computation is based in prediction and also contains all the previous epochs, so each time is smoother and more stable. In case of WaPPP the manual does not provide the algorithm used by the program, so no more conclusions can be extracted.

Some differences can be also seen between both software engines in terms of how the use of different products or sample intervals affects the results.

In case of RTKLIB, using final or ultra-rapid products does not make a big difference, as obtained coordinates are quite similar and their corresponding errors are practically the same. However, when data is resampled from 1 second to 30 seconds, the obtained coordinates change a little bit with respect the previous ones and errors get higher (approximately they become the double of the obtained with 1 second data processing). As RTKLIB computes a solution from a prediction it makes sense that, having less observations (in the resampled file), the result is worse than having a

larger number of epochs.

In case of WaPPP software, the result using final and ultra-rapid products is not the same. Using final products provides a lower error than using ultra-rapid ones. In this case, when the resampled file is processed, the results does not vary significantly with respect the previous one.

In terms of resulting errors, RTKLIB provides lower ones than WaPPP software. The range of RTKLIB is around half centimetre in horizontal and about 1 centimetre in height in 2 second data and below 1 centimetre in horizontal and around 2 centimetres in height in case of the resampled one. In case of WaPPP, errors are close to 2 centimetres in horizontal and 4 in vertical component when using final products and they come to 3 centimetres in horizontal and 5 in height in case of ultra-rapid products.

With respect the coordinate differences with the reference ones, in case of Cartesian coordinates no significant differences can be appreciated, as X and Z coordinates are better in RTKLIB but in Y coordinate, WaPPP results is closer to the reference one. When analysing geodetic coordinates, in terms of latitude and longitude they are quite similar, but in case of height RTKLIB seem to be closer to the reference one.

In real-time processing using SSR via NTRIP protocol, despite RTKLIB is used, cannot be included in the previous analysis as the way of processing is different. In this case, the obtained coordinates differ more with the reference ones with respect to the reference coordinates. The convergence time, is lower than in case of post-processing, after one hour and a half, the same errors that the ones obtained in the post processing mode of 12 hours of observation with the resampled RINEX and ultra-rapid products are achieved.

In general, the obtained results confirm the potential of PPP as technic for high accuracy applications, as errors obtained are, in all cases, in centimetre level, what is a nice result taking into account that the receiver used was a low cost one.

The obtained results are consistent with other results provided by other authors in the literature. As the case of [31], that shows an overall horizontal accuracy of 15 centimetres and a vertical accuracy of 25 centimetres after 10-minute initialization time using SSR corrections. Also in post-processing mode, other authors have pointed out accuracies below the centimetre when using large range of data (24 hours) and post-processing [3].

Results show, as mentioned, a high potential of the PPP technic. However, in the context of this master thesis, that is geomonitoring, some aspects have to be taken

into account to choose one or another kind of products and processing.

In permanent geomonitoring is needed to early detection of movements, final products could not be used, as their latency time is about 13 days. However, it can be useful to obtain precise coordinates in certain circumstances, as the case of the stable points used in the current version of GOCA GNSS Control, so they can be checked from time to time.

Both real-time and post-processing with ultra-rapid products can be used for geomonitoring. Real-time, theoretically can detect earlier changes, but as seen in the results section it less stable and also has some technical limitations as the need of internet connection. Using ultra-rapid products could be a good option to process, for example 12-hour package data when the expected movement is not really fast (as in case of some buildings or mountains, where the movement is quite slow).

Conclusions and future work

13. Conclusions

The main aim of this master thesis was the development of a dialog-based C++ software for PPP processing, using both RTKLIB and WaPPP as software engines in the context of the geomonitring software GOCA. This development. This development is complemented by the theoretical research of different PPP processing approaches and general GNSS processing issues.

In addition, two different measurements have been carried out to perform PPP processing. These measurements are: first one, 12-hour measurement in order to process static PPP in post-processing and a second one, 90 minutes, with real-time processing. Both measurements were done using a low cost receiver (U-Blox ZED-F9P), so under the context of this master thesis the possibility of using a low cost receiver to geomonitring applications would also be checked.

By last, this master thesis also pretended to make an analysis of differences found in different PPP processing aspects, such as using different processing engines, different IGS products (ultra-rapid and final ones) and even different PPP technics (post-processing and real-time).

In summary, this master thesis encompasses the whole process of PPP, from data collection to analysis of results using different approaches, going through the development of the necessary software to process the data.

Focusing on the development part of this work, it was initially raised as a part of the geomonitring software GOCA, in the module named GOCA GNSS Control, with the introduction of PPP processing as a new mode, with the purpose to obtain absolute coordinates as a new observation type on it. At this point some difficulties have been found, so the integration of the new PPP software with the already existing GOCA GNSS Control could not be fully developed.

The PPP software itself has been tested with different data, and during the implementation obtained results were compared with the ones obtained directly using RTKLIB user interface or WaPPP command line, in order to ensure they were the same. So, it can be concluded that the software developed under the context of this master thesis is functional and its main advantage regarding using directly the software engines is that it makes it easier to the user, using a unique interface for both. In case of WaPPP, having a user interface is ne, as the software engine itself works with command line. In case of RTKLIB, eliminating unnecessary options to PPP is helpful to the user comprehension, as the interface provided in RTKPOST have additional option for other kind of processing.

In relation to the results obtained from the field tests, analysed in the Discussion section, it can be concluded that, in general, this master thesis supports other studies that say the PPP technique has great potential. After evaluating two different software engines and different products and time intervals in case of post-processing, some aspects can be pointed out:

- RTKLIB provides, in general, lower errors than WaPPP software.
- RTKLIB provides a resulting graph in which we can see how coordinates an errors are getting more constant with time, while the graph obtained with the data processed using WaPPP is not interpretable.
- When using a large range of data (12-hours) and different settings to increase accuracy (Earth tide corrections, iono-free combination, etc.) final products does not make a great difference with respect ultra-rapid products.

With respect real-time observation, obtained results were worst in general. Although the results get good accuracy faster, the coordinates obtained differ more than the actual ones.

In view of the results obtained mentioned above, we can ensure that PPP technique can be used in the context of geomonitoring. Using IGS final products was for research reasons, as in practice the latency time is too high (around 2 weeks) to use them in a continuous monitoring. However, they can be used when some point wants to be precisely checked from time to time. Using ultra-rapid products is a good option for quasi-real time monitoring. Even it is post-processing, the low convergence time allows this kind of “quasi-real time”. That means that data can be processed, for example, each 12 hours, that is a time range enough to detect movements, as the kind of movements usually presented in geomonitoring are not really fast.

Real-time is of course an alternative to post-processing, but some more research has to be done to ensure good results and it is needed internet connexion, what can be a limitation in certain areas or applications.

The obtained results can with a low cost receiver are fully comparable with results in other studies using a more expensive receiver, so it can be also confirmed the potential of this U-Blox ZED-F9P receiver to high accuracy GNSS applications.

So, in general, this master thesis has covered the expected objectives. The still some limitations and future work to do (see Future work section) but it can contribute to PPP processing field, as it has pointed out good results using a low cost receiver and also an analysis of different software engines and processing parameters has been made.

14. Future work

Many different developments, tests, and experiments have been left for the future due to lack of time and the huge range of possibilities that can be studied in the context of PPP and also geomonitoring. Future work concerns integration issues with GOCA GNSS Control module, deeper analysis of particular PPP settings and further testing of absolute coordinates obtained from PPP in the context of geomonitoring. Some suggestions for future work are presented in this section.

In relation to the integration of the developed PPP software into GOCA GNSS Control, as explained before, some work is still needed to finish the integration. The most important aspect on this point is updating the existings GOCA_GNSS_Control software in order to make it fully functional and compilable under Visual Studio 2019. The limiting factor is that, even the .dll can be compiled and changes in functionality can be applied, the user interface does not get updated when re-compiling the software, so no one additional mode (PPP mode) can be added.

Processing parameters chosen to PPP processing has been set according a previous literature review, but using the ones available in existing software engines. It could be interesting making a deeper analysis of how specific parameters affect the results, as for example see what is the contribution of a specific parameter as could be the BLQ file or any other.

In addition, there exist some other approaches, as PPP with ambiguity resolution, that are not jet finished in RTKLIB or WaPPP, but some studies point out the potential of this approach, so it could be a good future work, implementing some of this technics and comparing the results with the ones presented in this document.

In relation to real-time PPP, some more research is needed in order to obtain more stable coordinates, especially in case of height, that has been the most problematic coordinate in this master thesis.

References

- [1] RTKLIB 2.4.2 Manual. 2013.
- [2] Neil Ashby. The Sagnac Effect in the Global Positioning System. In Guido Rizzi and Matteo Luca Ruggiero, editors, *Relativity in Rotating Frames. Fundamental Theories of Physics*. Springer, Dordrecht, 2004.
- [3] Sunil Bisnath and Yang Gao. Precise Point Positioning. A powerful technique with a promising future. *GPS World*, April 2009:43–50, 2009.
- [4] Mark Caissy, Loukis Agrotis, Georg Weber, Manuel Hernandez-Pajares, and Urs Hugentobler. The International GNSS Real-Time Service. *GPS World*, 2012.
- [5] Raquel M. Capilla, José Luis Berné, Angel Martín, and Raul Rodrigo. Simulation case study of deformations and landslides using real-time GNSS precise point positioning technique. *Geomatics, Natural Hazards and Risk*, 7(6):1856–1873, 2016.
- [6] Suelynn Choy, Sunil Bisnath, and Chris Rizos. Uncovering common misconceptions in GNSS Precise Point Positioning and its future prospect. *GPS Solutions*, 2016.
- [7] Collins. Undifferenced GPS Ambiguity Resolution Using the Decoupled Clock Model and Ambiguity Datum Fixing. 57(2):1–35, 2009.
- [8] ESA. Navipedia.
- [9] Lou Estey and Stuar Wier. Teqc Tutorial. Basics of Teqc Use and Teqc Products. *UNAVCO*, 2014.
- [10] Tim Everett. rtklibexplorer, 2018.
- [11] J. Geng, F. N. Teferle, X. Meng, and A. H. Dodson. Towards PPP-RTK: Ambiguity resolution in real-time precise point positioning. *Advances in Space Research*, 47(10):1664–1673, 2011.
- [12] Geo++ GmbH. State Space Representation. 2015.
- [13] Bernhard Hofman-Wellenhof, Herbert Lichtenegger, and Elmar Wasle. *GNSS: Global Navigation Satellite Systems. GPS, GLONASS, Galileo & more*. Springer-WienNewYork, 2008.
- [14] Chen Horng-yue, Kuo Long-chen, Chung Wang-shung, and Yu Shui-beih. Using quasi ionosphere-free post-processing algorithm on the medium-range kinematic high accuracy GPS relative positioning. *Wuhan University Journal of Natural Sciences*, 8(2):610–618, 2008.

- [15] Katrin Huber, Florian Heuberger, Christoph Abart, Ana Karabatic, Robert Weber, and Philipp Berglez. PPP: Precise Point Positioning – Constraints and Opportunities Katrin. *TS 10C - GNSS Modernisation and Trends*, (April):11–16, 2010.
- [16] IGS Antenna Working Group. IGS antenna files. Technical report, ESA, 2017.
- [17] International GNSS Service. IGS.
- [18] Reiner Jäger. GNSS / GPS / LPS based Online Control and Alarm System (GOCA) - Mathematical Models and Technical Realisation of a System for Natural and Geotechnical Deformation Monitoring and Analysis -. *Online*, 0:1–11, 2006.
- [19] Reiner Jäger. Methods and Approaches for Integrated Deformation Analysis. In *International Workshop "Integration of Point- and Area-wise Geodetic Monitoring for Structures and Natural Objects"*, number 0, Novosibirsk, 2014.
- [20] Reiner Jäger and Lyudmila Gorokhova. Integrated geomonitring with innovative sensor technology, IT and modelling on the reference object Fernsehturm Stuttgart.
- [21] Reiner Jäger, Tilman Müller, Heinz Saler, and Rainer Schwäble. *Klassische und robuste Ausgleichungsverfahren - Ein Leitfaden für Ausbildung und Praxis von Geodäten und Geoinformatikern*. Herbert Wichmann Verlag, Heidelberg, 2005.
- [22] Bofeng Li and Peter J.G. Teunissen. High Dimensional Integer Ambiguity Resolution: A First Comparison between LAMBDA and Bernese. *Journal of Navigation*, 64(S1):S192–S210, 2011.
- [23] A. Martín, A. B. Anquela, A. Dimas-Pagés, and F. Cos-Gayón. Validation of performance of real-time kinematic PPP. A possible tool for deformation monitoring. *Measurement: Journal of the International Measurement Confederation*, 69:95–108, 2015.
- [24] NASA. NASA CDDIS, 2019.
- [25] Thomas D Papanikolaou and Stavros Melachroinos. Quantifying mis-modelling effects in the GNSS yaw- attitude and phase wind-up. *International Global Navigation Satellite Systems Association IGNSS Conference 2016*, (December), 2016.
- [26] Mauricio Ernesto Paredes Wiedehold. *Procesamiento PPP de observaciones GNSS utilizando RTKLIB*. PhD thesis, Universidad de Santiago de Chile, 2013.

- [27] M. Oswald R. Jäger, A. Hoscislawski. GNSS/LPS/LS based Online Control and Alarm System (GOCA) - Mathematical Models and Technical Realization of a Scalable System for Natural and Geotechnical Deformation Monitoring and Analysis -. *Geodetic Deformation Monitoring: From Geophysical to Engineering Roles: IAG Symposium Jaén, Spain March 17–19, 2005*, 131(August):376–383, 2006.
- [28] Pierre Fridez Rolf Dach, Simon Lutz, Peter Walser. *Bernese GNSS Software, Version 5.2*, volume 47. 2015.
- [29] Junbo Shi. *Precise Point Positioning Integer Ambiguity Resolution with Decoupled Clocks*. PhD thesis, University of Calgary, 2012.
- [30] Junbo Shi and Yang Gao. A comparison of three PPP integer ambiguity resolution methods. *GPS Solutions*, 18(4):519–528, 2014.
- [31] Junbo Shi, Chaoqian Xu, Jiming Guo, and Gao Yang. A Performance Analysis of Real-Time Precise Point Positioning for Deformation Monitoring. 2013.
- [32] U-blox. ZED-F9P Data Sheet. *www.u-blox.com*, 2019.
- [33] Lambert Wanninger. WaSoft, 2019.
- [34] WaSoft. User ’ s Guide Wa1. (September), 2010.
- [35] WaSoft. WaSoft User ’ s Guide. 2(July), 2018.
- [36] Greg Welch and Gary Bishop. An Introduction to the Kalman Filter. *In Practice*, 7(1):1–16, 2006.
- [37] Walter M Welsch and Otto Heunecke. Models and terminology for the analysis of geodetic monitoring observations. *The 10th FIG Internaitonal Symposium on Deformation Measurements*, (25):390–412, 2001.
- [38] Gao Yang. GNSS Solutions: Precise point positioning and its challenges, aided-GNSS and signal tracking. *Inside GNSS*, 1(8):16–18, 2006.

Annexes

Annex A. GUI variables

In this annex, the different interface variables that have been used in the PPP software are indicating. For each variable, the ID, the variable name used in the code and the variable type is shown. The purpose of this annex is to provide a help in case something needs to be modified.

Variables on the main PPP window ¹:

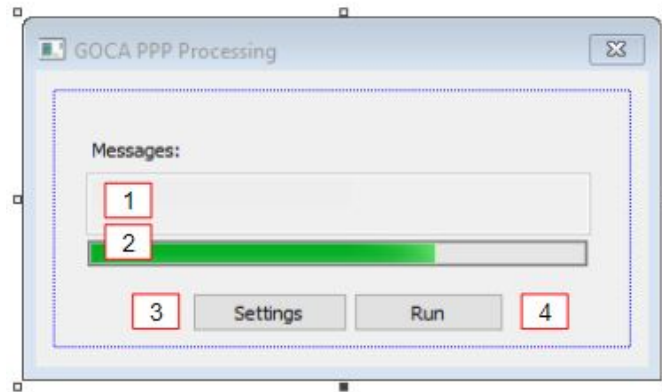


Figure A1: Main PPP dialog

Number	ID	variable name	variable type
1	IDC_EDIT_MSG	m_messages	CEdit
2	IDC_PROGRESS_PPP	m_PPP_progress	CProgressCtrl
3	IDC_BUTTON_PPP_SETTINGS	m_button_PPP_settings	CButton
4	IDC_BUTTON_PPP_RUN	m_button_PPP_run	CButton

Figure A2: Variables in main PPP dialog

¹This image corresponds to the first version of the software, in the second one stop button is added with id IDC_BUTTON_T

Variables corresponding to general PPP settings dialog in post-processing:

The image shows a software dialog box titled "Processing parameters" for PPP settings. It is divided into several sections: "Processing parameters", "Input data", "Orbit/Clock information", "Additional input data", and "Output". Each field or control element is marked with a red box and a number from 5 to 31. The "Processing parameters" section includes a "Processor" dropdown (5), a "Settings" button (6), and an "Elevation mask" field (7). The "Input data" section contains four rows for "Observation file" (8, 9), "Navigation file (GPS)" (10, 11), "Navigation file (GLONASS)" (12, 13), and "Navigation file (others)" (14, 15). The "Orbit/Clock information" section has "Orbit" (16) and "Clock" (17) dropdowns, "Precise Ephemerids" (18, 19), and "Clock file" (20, 21) fields. The "Additional input data" section features two checkboxes: "Use antenna file" (22) and "Use same file for rec/sat" (23), followed by "Receiver antenna" (24, 25) and "Satellite antenna" (26, 27) fields. The "Output" section includes "GKA" (28, 29) and "PPP" (30, 31) fields. At the bottom are "Accept" and "Cancel" buttons.

Figure A3: General PPP settings dialog

Number	ID	variable name	variable type
5	IDC_COMBO_PPP	comboPPP	CComboBox
6	IDC_BUTTON_SETTINGS	-	CButton
7	IDC_EDIT_ELEVATION_MASK	m_elevation	int
8	IDC_EDIT_OBS_FILE	m_obsFile	CEdit
9	IDC_BUTTON_OBS_FILE	-	CButton
10	IDC_EDIT_GPS_NAV	m_gpsNav	CEdit
11	IDC_BUTTON_GPS_NAV	-	CButton
12	IDC_EDIT_GLONASS_NAV	m_glonassNav	CEdit
13	IDC_BUTTON_GLONASS_NAV	-	CButton
14	IDC_EDIT_NAV	m_nav	CEdit
15	IDC_BUTTON_NAV	-	CButton
16	IDC_COMBO_ORBIT	comboOrbit	CComboBox
17	IDC_COMBO_CLOCK	comboClock	CComboBox
18	IDC_EDIT_PRECISE_EPH	m_ephFile	CEdit
19	IDC_BUTTON_PRECISE_EPH	m_button_orbit	CButton
20	IDC_EDIT_CLOCK	m_clkFile	CEdit
21	IDC_BUTTON_CLOCK	m_button_clock	CButton
22	IDC_CHECK_ANTENNA	m_atx	bool
23	IDC_CHECK_SAME	m_sameAtx	bool
24	IDC_EDIT_ATX_FILE	m_atxFile	CEdit
25	IDC_BUTTON_ATX_FILE	m_button_atxFile	CButton
26	IDC_EDIT_ATX_FILE2	m_atxFile2	CEdit
27	IDC_BUTTON_ATX2	m_button_atxFile2	CButton
28	IDC_EDIT_GKA_OUT	m_gkaFile	CEdit
29	IDC_BUTTON_GKA	m_button_gkaFile	CButton
30	IDC_EDIT_PPP_OUT	m_posFile	CEdit
31	IDC_BUTTON_PPP_OUT	m_button_pppFile	CButton

Figure A4: Variables in general PPP settings dialog

Variables corresponding to RTKLIB settings dialog (post-processing):

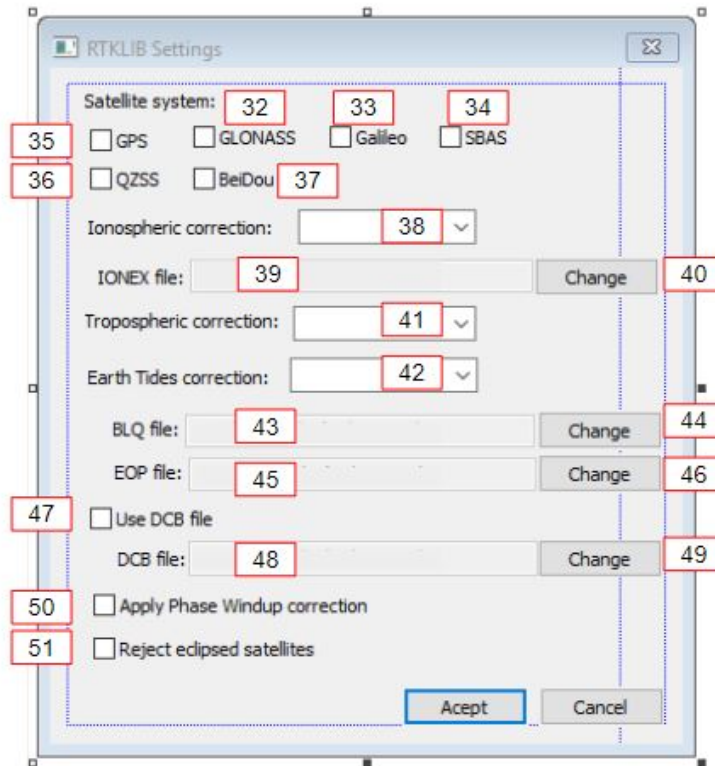


Figure A5: RTKLIB settings dialog

Number	ID	variable name	variable type
32	IDC_CHECK_GLONASS	m_check_glonass	bool
33	IDC_CHECK_GALILEO	m_check_galielo	bool
34	IDC_CHECK_SBAS	m_check_sbas	bool
35	IDC_CHECK_GPS	m_check_gps	bool
36	IDC_CHECK_QZSS	m_check_qzss	bool
37	IDC_CHECK_BEIDOU	m_check_beidou	bool
38	IDC_COMBO_RTKLIB_IONO	combo_rtkliblono	CComboBox
39	IDC_EDIT_RTKLIB_IONEX	m_IONEXFile	CEdit
40	IDC_BUTTON_RTKLIB_IONEX	m_button_rtkliblono	CButton
41	IDC_COMBO_RTKLIB_TROPO	combo_rtklibTrop	CComboBox
42	IDC_COMBO_RTKLIB_TIDES	combo_rtklibEarth	CComboBox
43	IDC_EDIT_RTKLIB_BLQ	m_BLQFile	CEdit
44	IDC_BUTTON_RTKLIB_BLQ	m_button_rtklibBLQ	CButton
45	IDC_EDIT_RTKLIB_EOP	m_EOPFile	CEdit
46	IDC_BUTTON_RTKLIB_EOP	m_button_rtklibEOP	CButton
47	IDC_CHECK_DCB_RTKLIB	m_dcb_rtklib	bool
48	IDC_EDIT_DCB_RTKLIB	m_dcbFile	CEdit
49	IDC_BUTTON_DCB_RTKLIB	m_button_dcb_rtklib	CButton
50	IDC_CHECK_RTKLIB_WINDUP	m_rtklibWind	bool
51	IDC_CHECK_RTKLIB_ECLIPSE	m_rtklibEclipse	bool

Figure A6: Variables in RTKLIB settings dialog

Variables corresponding to WaPPP settings dialog:

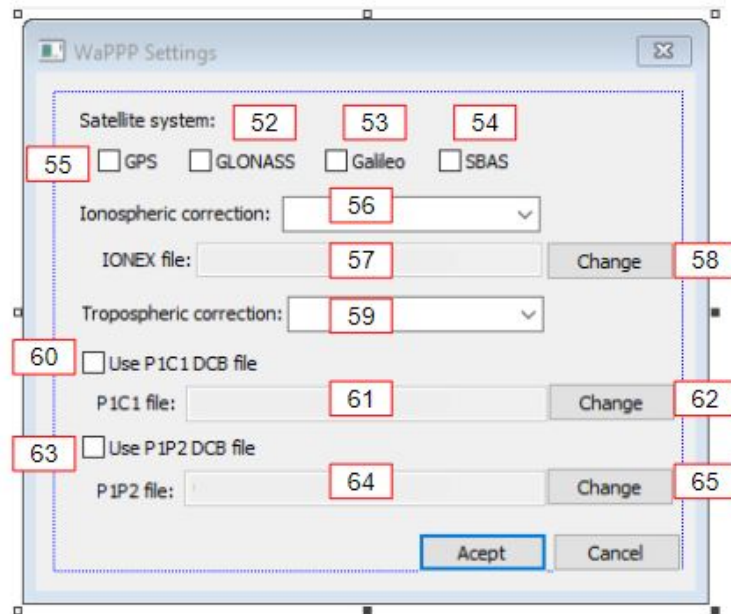


Figure A7: WaPPP settings dialog

Number	ID	variable name	variable type
52	IDC_CHECK_GLONAS_WAPPP	m_glonass_wappp	bool
53	IDC_CHECK_GALILEO_WAPPP	m_galileo_wappp	bool
54	IDC_CHECK_SBAS_WAPPP	m_sbas_wappp	bool
55	IDC_CHECK_GPS_WAPPP	m_gps_wappp	bool
56	IDC_COMBO_WA_IONO	combo_walono	CComboBox
57	IDC_EDIT_WA_IONEX	m_wa_ionex	CEdit
58	IDC_BUTTON_WA_IONEX	m_button_walono	CButton
59	IDC_COMBO_WA_TROPO	combo_waTropo	CComboBox
60	IDC_CHECK_p1c1	m_p1c1	bool
61	IDC_EDIT_p1c1	m_p1c1File	CEdit
62	IDC_BUTTON_p1c1	m_button_p1c1	CButton
63	IDC_CHECK_p1p2	m_p1p2	bool
64	IDC_EDIT_p1p2	m_p1p2File	CEdit
65	IDC_BUTTON_p1p2	m_button_p1p2	CButton

Figure A8: Variables in WaPPP settings dialog

Variables corresponding to main window to choose the processing mode:



Figure A9: Main window dialog

Number	ID	variable name	variable type
66	IDC_COMBO_PPP_MODE	m_ppp_mode	CComboBox

Figure A10: Variables in main window dialog

Variables corresponding to TPC dialog:

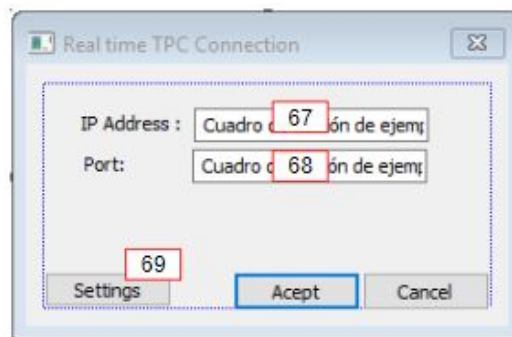


Figure A11: TPC dialog

Number	ID	variable name	variable type
67	IDC_EDIT_IP	m_IPAddress	Cstring
68	IDC_EDIT_PORT	m_lport	CString
69	IDC_BUTTON1	m_button_RT_settings	CButton

Figure A12: Variables in TPC dialog

Variables corresponding to PPP settings (real-time) :

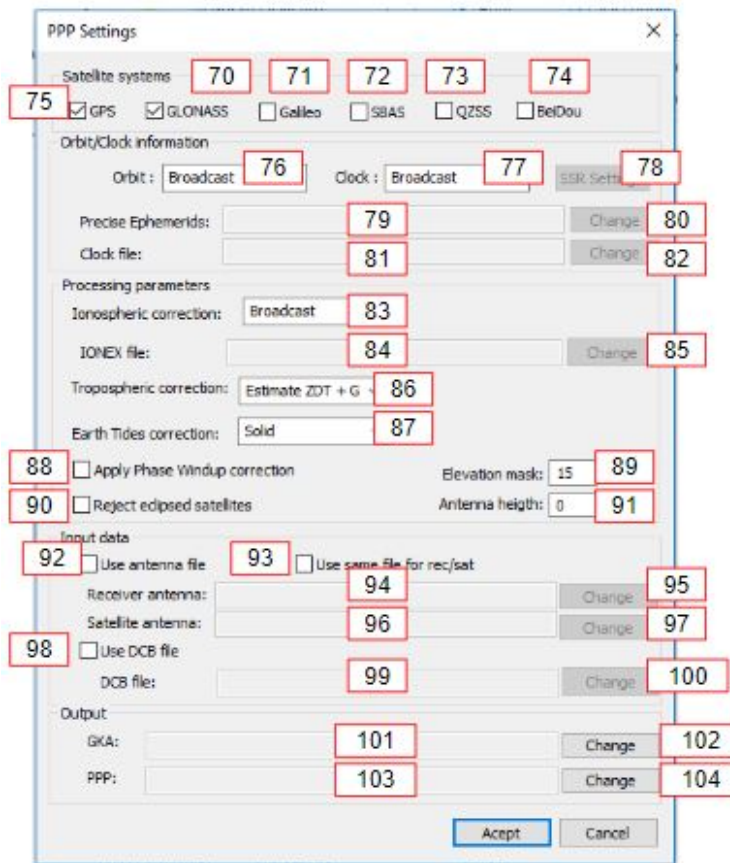


Figure A13: PPP settings dialog (real-time)

Number	ID	variable name	variable type
70	IDC_CHECK_GPS_RT	m_check_gps_RT	bool
71	IDC_CHECK_GLONASS_RT	m_check_glonass_RT	bool
72	IDC_CHECK_GALILEO_RT	m_check_galielo_RT	bool
73	IDC_CHECK_SBAS_RT	m_check_sbas_RT	bool
74	IDC_CHECK_QZSS_RT	m_check_qzss_RT	bool
75	IDC_CHECK_BEIDOU_RT	m_check_beidou_RT	bool
76	IDC_COMBO_ORBIT_RT	comboOrbit_RT	CComboBox
77	IDC_COMBO_CLOCK_RT	comboClock_RT	CComboBox
78	IDC_BUTTON_SSR	m_button_ssr	Cbutton
79	IDC_EDIT_PRECISE_EPH_RT	m_ephFile_RT	CEdit
80	IDC_BUTTON_PRECISE_EPH_RT	m_button_orbit_RT	CButton
81	IDC_EDIT_CLOCK_RT	m_clkFile_RT	CEdit
82	IDC_BUTTON_CLOCK_RT	m_button_clock_RT	CButton
83	IDC_COMBO_RTKLIB_IONO_RT	combo_iono_RT	CComboBox
84	IDC_EDIT_RTKLIB_IONEX_RT	m_IONEXFile_RT	CEdit
85	IDC_BUTTON_RTKLIB_IONEX_RT	m_button_iono_RT	CButton
86	IDC_COMBO_RTKLIB_TROPO_RT	combo_Trop_RT	CComboBox
87	IDC_COMBO_RTKLIB_TIDES_RT	combo_Earth_RT	CComboBox
88	IDC_CHECK_RTKLIB_WINDUP_RT	m_Wind_RT	bool
89	IDC_EDIT_ELEVATION_MASK_RT	m_elevation_RT	int
90	IDC_CHECK_RTKLIB_ECLIPSE_RT	m_Eclipse_RT	bool
91	IDC_EDIT_ANTENNA_H	m_antenna_h	double
92	IDC_CHECK_ANTENNA_RT	m_atx_RT	bool
93	IDC_CHECK_SAME_RT	m_sameAtx_RT	bool
94	IDC_EDIT_ATX_FILE_RT	m_atxFile_RT	CEdit
95	IDC_BUTTON_ATX_FILE_RT	m_button_atxFile_RT	CButton
96	IDC_EDIT_ATX_FILE2_RT	m_atxFile2_RT	CEdit
97	IDC_BUTTON_ATX2_RT	m_button_atxFile2_RT	CButton
98	IDC_CHECK_DCB_RTKLIB_RT	m_dcb_RT	bool
99	IDC_EDIT_DCB_RTKLIB_RT	m_dcbFile_RT	CEdit
100	IDC_BUTTON_DCB_RTKLIB_RT	m_button_dcb_RT	CButton
101	IDC_EDIT_GKA_OUT_RT	m_gkaFile_RT	CEdit
102	IDC_BUTTON_GKA_RT	m_button_gkaFile_RT	CButton
103	IDC_EDIT_PPP_OUT_RT	m_posFile_RT	CEdit
104	IDC_BUTTON_PPP_OUT_RT	m_button_pppFile_RT	CButton

Figure A14: Variables in PPP settings dialog (real-time)

Variables corresponding to SSR dialog:

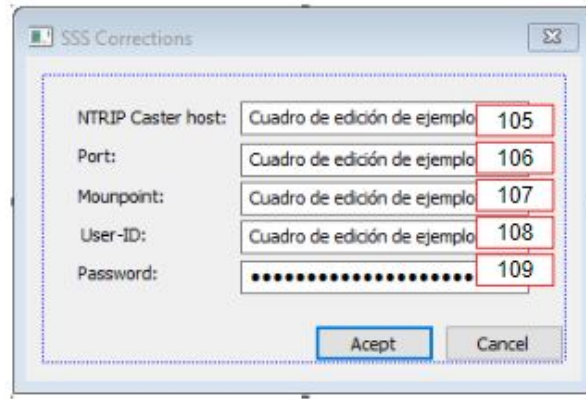


Figure A15: SSR dialog

Number	ID	variable name	variable type
105	IDC_EDIT_HOST	m_NTRIP_host	CString
106	IDC_EDIT_PORT	m_NTRIP_port	CString
107	IDC_EDIT_MP	m_NTRIP_mp	CString
108	IDC_EDIT_USER	m_NTRIP_user	CString
109	IDC_EDIT_PASSWORD	m_NTRIP_password	CString

Figure A16: Variables in SSR dialog

Annex B. Other results

In this Annex, graphs that are not included in section "Results" are provided.

RTKLIB with ultra-rapid products - 30 seconds

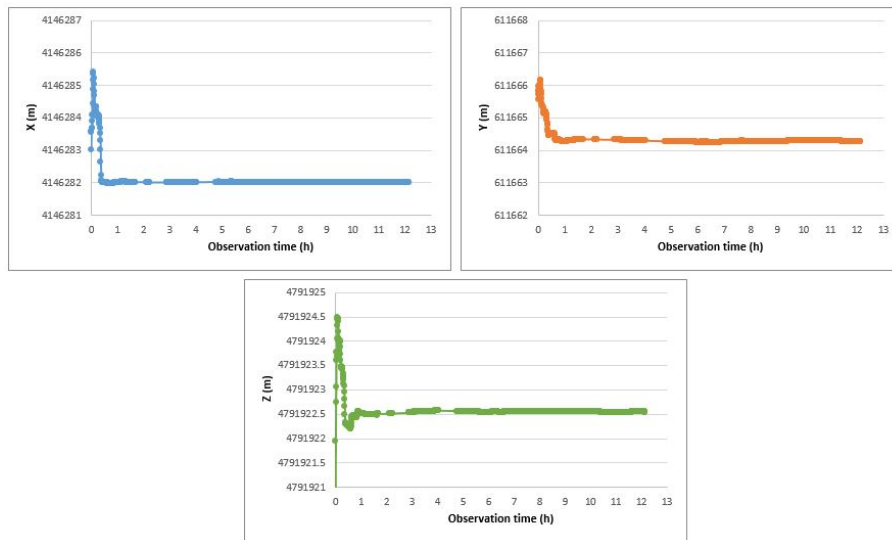


Figure B1: Coordinates along time RTKLIB with ultra-rapid products - 30 seconds

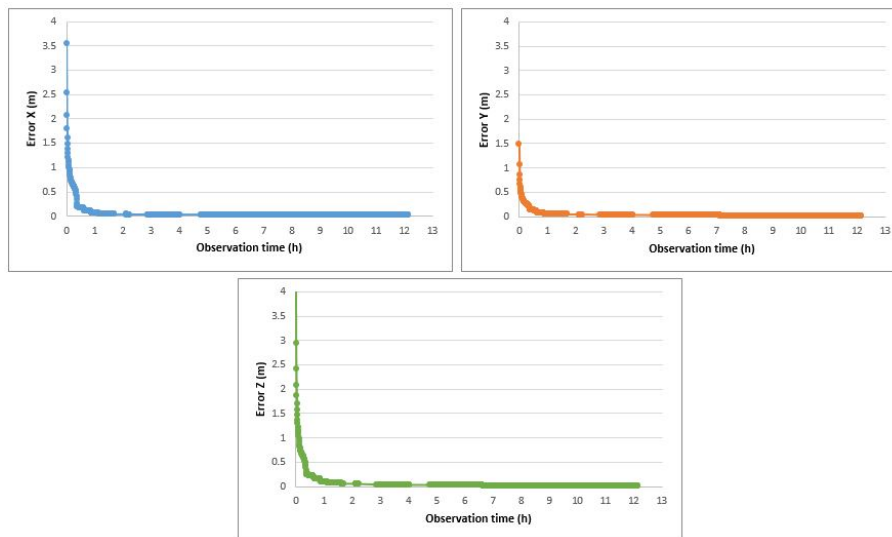


Figure B2: Errors along time RTKLIB with ultra-rapid products - 30 seconds

RTKLIB with final products

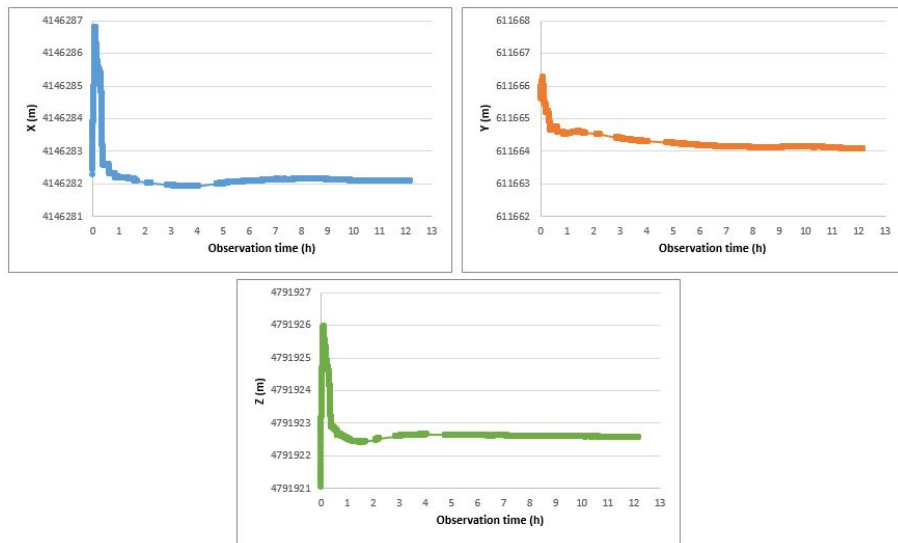


Figure B3: Coordinates along time RTKLIB with final products

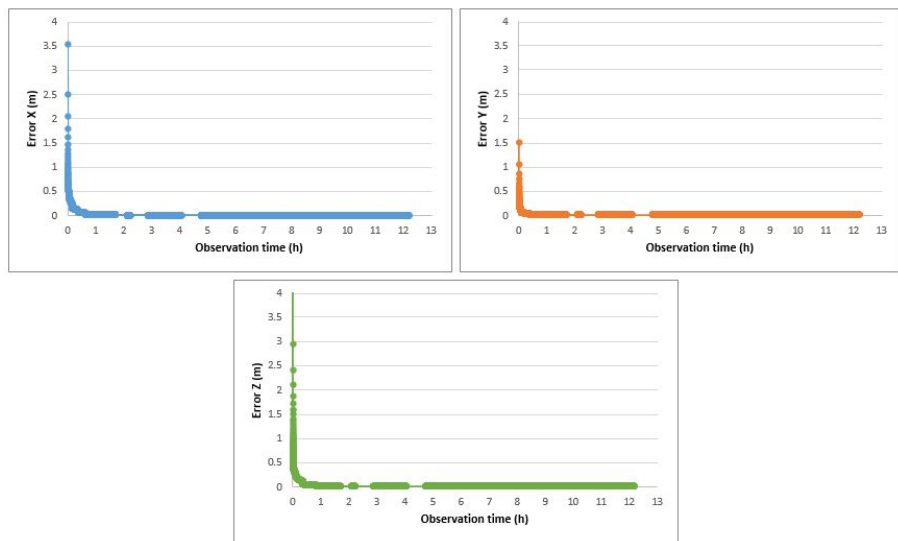


Figure B4: Errors along time RTKLIB with final products

WaPPP with ultra-rapid products - 30 seconds

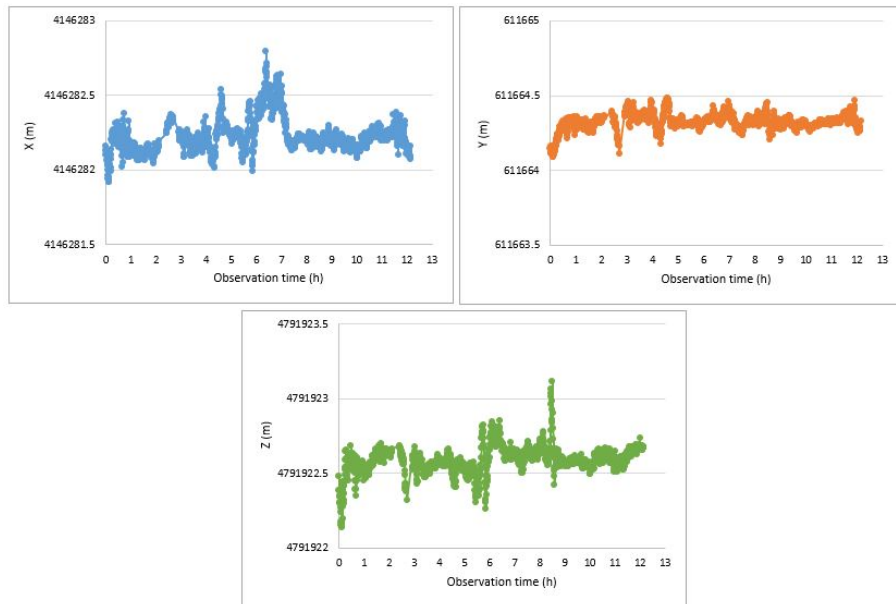


Figure B5: Coordinates along time WaPPP with ultra-rapid products - 30 seconds

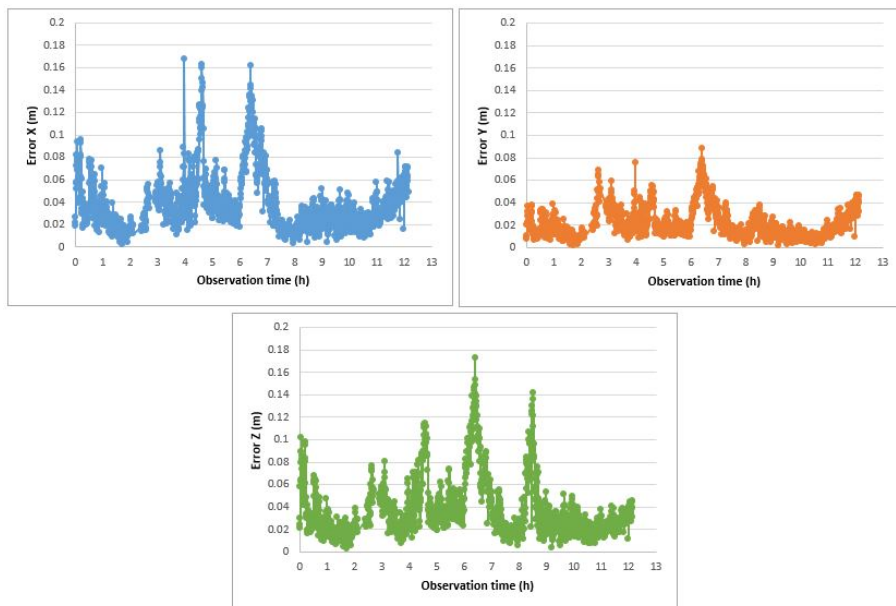


Figure B6: Errors along time WaPPP with ultra-rapid products - 30 seconds

WaPPP with final products

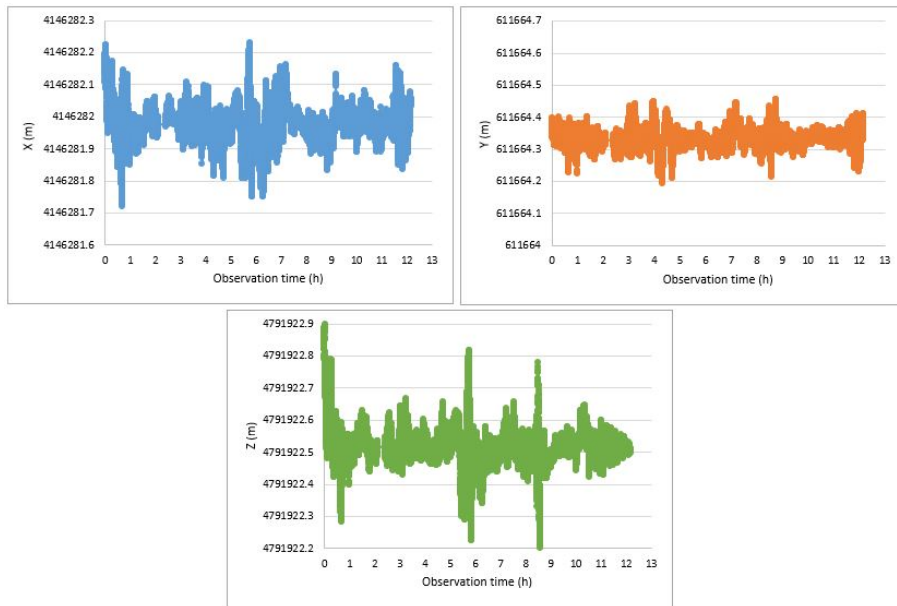


Figure B7: Coordinates along time WaPPP with final products

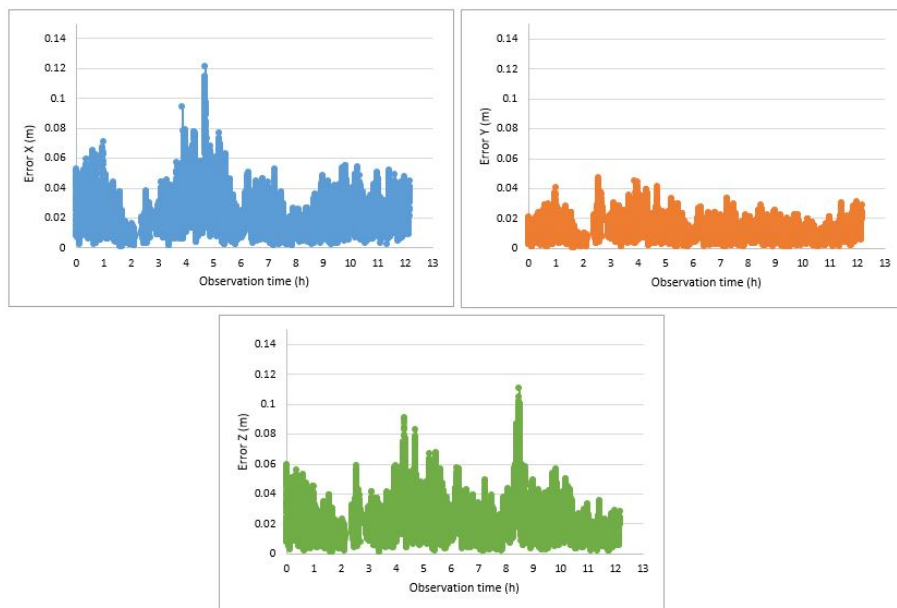


Figure B8: Errors along time WaPPP with final products