UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Karlsruhe University of
Applied Sciences

Fakultät für Informationsmanagement und Medien
Geomatics Master

Implementation of Moving-Base-GNSS in NAVKA Multisensor GNSS/MEMS/Optics Navigation Algorithms and Systems

---

Author:
Jorge Hernández Olcina

First Director:
Prof. Dr -Ing. Reiner Jäger
Second Director:
Prof. Dr -Ing. Ana Belén Anquela Julián

Supervisor:
Dipl. -Ing (FH) Julia Diekert

Approval and Declaration

This thesis titled *Implementation of Moving-Base-GNSS in NAVKA Multisensor GNSS/MEMS/Optics Navigation Algorithms and Systems,* was prepared and submitted by Jorge Hernández Olcina (*Matrikelnummber: 65612*) and has been accepted as partial fulfilment of the requirement for the Master of Geomatics in Karlsruhe University of Applied Science.

I declare that this thesis was made and produced based on my own research. I confirm that the thesis was done completely to achieve the master's degree and will not be sent to other institutes. The works contributed by others were clearly established with references in the thesis.

Jorge Hernández Olcina

First Director: Prof. Dr -Ing. Reiner Jäger

Second Director: Prof. Dr -Ing. Ana Belén Anquela Julián

Supervisor: Dipl. -Ing (FH) Julia Diekert

Abstract

The use of GNSS as a navigation technique is one of the most used methodologies when calculating mobile platform trajectories. This paper will be presenting several methodologies and algorithms related to Moving-Base-GNSS existing in the bibliography, of which one of them, due to be a better approach to reality, has been implemented and developed. Therefore, one of the main objectives of the work has been the implementation of a calculation routine in C++, which allows solving ambiguity fixation using the *Multivariate Constraint LAMBDA = MC-LAMBDA* algorithm. After its implementation, another objective of this work is to improve the RTKLIB open source library. This library consists of a series of routines and GNSS processing both in real time and postprocessing. In addition, the library itself has implemented a calculation option in Moving-Base, which is not entirely accurate, so with the MCLAMBDA algorithm it is intended to improve the calculation of the baselines between rover and master receivers. The implementation of said routine within the RTKLIB software in postprocessing allows the user to choose a series of options within the MCLAMBDA routine for the calculation of ambiguities in order to obtain a greater number of fixed solutions. Another problem to improve is the calculation of the coordinates of the rover from the coordinates of the master receiver. The RTKLIB library in its Moving-Base mode first calculates the coordinates of the master in single mode, and leaves the fixed station, to which the result of the baselines is added to the rover. This procedure has been improved taking into account that the master receiver is in motion together with the rover, and therefore, Python software has been developed that allows, from the coordinates of the master receiver, previously calculated with RTKLIB and the lines base between the rover and the master receiver, calculated with the MCLAMBDA routine, calculate the final coordinates of the rover in the e-frame. This process allows the user that the calculation of the coordinates of the master receiver can be performed in different postprocess modes such as DGNSS or PPPK, which allows you to obtain better accuracy than the single mode. The coordinates of the master receiver must be calculated in the e-frame (XYZ ECEF) and the baselines in the n-frame (ENU), so that a simple transformation between the n-frame and the e-frame is enough to Rover coordinate calculation. Finally, for the evaluation of all the procedures mentioned above, several tests were carried out using the GNSS/MEMS/camera navigation box, which was previously configured and repaired for use. This box has a dual frequency sensor L1 and L2 with dual antenna, and a UBLOX L1 sensor, which allows to mount a platform in Moving-Base situation considering that one of the antennas will be the master receiver and the other two will be rovers.

Content

Approval and Declaration ..... **I**
Abstract ..... **II**
List of Figures ..... **VI**
List of Tables ..... **VII**

1. Introduction **1**
1.1 Background 1
1.2 Statement of problem 2
1.3 Moving base concepts 3
1.3.1 Moving base situation 3
1.3.2 Leverarms 6
1.4 Objectives 7
1.5 Outline of dissertation 8

2. GNSS theory **9**
2.1 RTK 9
2.1.1 RTK Fundamentals 9
2.1.2 Technique 9
2.1.3 RTK Algorithm 10
2.2 DGNSS 13
2.2.1 DGNSS Fundamentals 13
2.2.2 The classical DGNSS technique 13
2.2.3 DGNSS Algorithm 13
2.2.4 RTCM 14
2.3 PPP 15
2.3.1 PPP Fundamentals 15
2.3.2 PPP benefits and Prospects 15

3. Existing moving base GNSS algorithms **17**
3.1 Introduction 17
3.2 Algorithms 17
3.2.1 Algorithm “MultiKin” 17
3.2.2 Algorithm “Multivariate Constraint LAMBDA = MC - LAMBDA” 22
3.2.3 Algorithm “Lambda with constraints” 28
3.2.4 Comparison and Analysis 31

4. Hardware and Software 35
4.1 Hardware 35
4.1.1 Novatel model OEM617D 36
4.1.2 Ublox M8T L1/E1/G1/B1 37
4.2 Software 38
4.2.1 RTKLIB 38
4.2.2 Novatel Connect 40
4.2.3 U-CENTER (U-blox) 41

5. Integer ambiguity resolution theory 43
5.1 Parameter estimation 43
5.2 Decorrelation technique 44
5.3 Integer rounding 45
5.4 Integer bootstrapping 45
5.5 Integer least squares (ILS) 47
5.6 ILS with Ratio Test 48

6. Implementation of MC-LAMBDA routine 49
6.1 The main MC-LAMBDA routine 49
6.2 Routines used by MC-LAMBDA 51
6.2.1 *chistart.cpp* 51
6.2.2 *decorrel.cpp* 52
6.2.3 *lsearch.cpp* 53
6.2.4 *ssearch.cpp* 54
6.3 Implementation in RTKLIB 55
6.3.1 *rtklib.h changes* 55
6.3.2 *RTKPOST GUI changes* 57
6.3.3 *Changes in RTKLIB source code* 58

7. Moving Base Software 61
7.1 Novatel reader 65

8. Test and results analysis 67
8.1 Test configuration 67
8.2 Static test 68
8.2.1 Single mode antenna 1 68
8.2.2 Static DGNSS and PPPS 70
8.2.3 Rover position calculation 74
8.3 Kinematic test 77

8.3.1 Single mode antenna 1 78
8.3.2 Kinematic DGNSS and PPPK 79
8.3.3 Rover position calculation 81

9. Conclusions and future work 85
9.1 Conclusions 85
9.2 Future work 86

10. References 87

Appendix A: Novatel and Ublox configuration 89
A.1 Novatel configuration 89
A.2 Ublox configuration 90

Appendix B: Data processing 91
B.1 DGNSS 91
B.2 PPP 94

List of Figures

FIGURE 1 GNSS RADAR FOR AERONAUTICAL APPLICATIONS. (LUO) 2
FIGURE 2: GNSS PLATFORM. (OWN SOURCE) 3
FIGURE 3: MULTISENSOR PLATFORM. (OWN SOURCE) 4
FIGURE 4: N-FRAME AND E-FRAME. (JÄGER) 5
FIGURE 5: LEVERARMS GENERAL CASE. (NAVKA) 6
FIGURE 6: SPECIAL CASE, PLATFORM = BODY. (NAVKA) 7
FIGURE 7 CONCEPT OF CLOSED (POLYGONAL) AMBIGUITY CONSTRAINTS. (LUO) 18
FIGURE 8 PROCEDURE OF AMBIGUITY DETERMINATION AND MONITORING USING MULTIPLE. (LUO) 19
FIGURE 9 PROCEDURE OF MULTIKIN. (LUO) 21
FIGURE 10: GNSS SIGNALS RECEIVED FROM THREE OR MORE ANTENNAS ONBOARD. (GIORGI) 22
FIGURE 11: PICTURE AND SCHEME OF THE FOUR ANTENNAS PLACEMENT DURING THE STATIC. (GIORGI) 32
FIGURE 12: GNSS/MEMS/CAMERA NAVIGATION BOX GNSS SENSORS AND CONNECTIONS. (OWN SOURCE) 35
FIGURE 13: NOVATEL OEM617D. (NOVATEL, OEM617D GPS) 36
FIGURE 14: RTKLIB GUI APS ON WINDOWS 7. (TAKASU) 39
FIGURE 15: GUI NOVATEL CONNECT. (OWN SOURCE) 40
FIGURE 16: GUI U-CENTER. (OWN SOURCE) 41
FIGURE 17: ACCEPTANCE REGIONS WITH RATIO TEST (BRIGHT GREEN AND RED). (DELFT) 48
FIGURE 18: MC-LAMBDA FUNCTION. (OWN SOURCE) 49
FIGURE 19: CHISTART FUNCTION. (OWN SOURCE) 51
FIGURE 20: DECORREL FUNCTION. (OWN SOURCE) 52
FIGURE 21: LSEARCH FUNCTION. (OWN SOURCE) 53
FIGURE 22: SSEARCH FUNCTION. (OWN SOURCE) 54
FIGURE 23: POSTPROCESSING MODES. (OWN SOURCE) 55
FIGURE 24: AMBIGUITY RESOLUTION MODES. (OWN SOURCE) 56
FIGURE 25: MCLAMBDA OPTIONS. (OWN SOURCE) 56
FIGURE 26: PRCOPT_T STRUCT DEFINITION. (OWN SOURCE) 56
FIGURE 27: AMBIGUITY RESOLUTION DEFINITION FUNCTIONS. (OWN SOURCE) 57
FIGURE 28: SETTINGS 1 RTKPOST OPTIONS. (OWN SOURCE) 57
FIGURE 29: SETTINGS 2 RTKPOST OPTIONS. (OWN SOURCE) 58
FIGURE 30: AMBIGUITY RESOLUTION DEFINITION FUNCTION. (OWN SOURCE) 59
FIGURE 31: CALL AMBIGUITY RESOLUTION FUNCTION. (OWN SOURCE) 59
FIGURE 32: CALL OF MCLAMBDA_EXEC. (OWN SOURCE) 59
FIGURE 33: MCLAMBDA_EXEC FUNCTION. (OWN SOURCE) 60
FIGURE 34: CALL MCLAMBDA ROUTINE. (OWN SOURCE) 60
FIGURE 35: MC-LAMBDA CALCULATION OPTIONS. (OWN SOURCE) 60
FIGURE 36: GUI OF MOVING BASE SOFTWARE. (OWN SOURCE) 61
FIGURE 37: TRANSFORMATION FUNCTION BETWEEN N-FRAME TO E-FRAME. (OWN SOURCE) 62
FIGURE 38: TRANSFORMATION FUNCTION BETWEEN ECEF TO GEOGRAPHIC. (OWN SOURCE) 63
FIGURE 39: ELLIPSOID FUNCTION. (OWN SOURCE) 63
FIGURE 40: "NU" FUNCTION. (OWN SOURCE) 63
FIGURE 41: SEARCH EPOCH AND CALCULATE ROVER COORDINATES. (OWN SOURCE) 64
FIGURE 42: CALCULATION PER ROVER. (OWN SOURCE) 64
FIGURE 43: NOVATEL READER. (OWN SOURCE) 65
FIGURE 44: TRANSFORMATION FUNCTION BETWEEN GEOGRAPHIC COORDINATES TO ECEF. (OWN SOURCE) 65
FIGURE 45: TEST PLATFORM CONFIGURATION. (OWN SOURCE) 67
FIGURE 46: SINGLE PROCESSING RTKLIB AND NOVATEL GROUND TRACK. (OWN SOURCE) 68
FIGURE 47: POSITION VISUALIZATION SINGLE MODE. (OWN SOURCE). 69
FIGURE 48: DGNSS AND PPPS GROUND TRACK. (OWN SOURCE) 70
FIGURE 49: POSITION VISUALIZATION DGNSS AND PPPS. (OWN SOURCE) 71
FIGURE 50: DGNSS ANT1 AND ANT2 GROUND TRACK. (OWN SOURCE) 72

FIGURE 51: POSITION VISUALIZATION DGNSS. (OWN SOURCE) 72
FIGURE 52: PPPS ANT1 AND ANT2 GROUND TRACK. (OWN SOURCE) 73
FIGURE 53: POSITION VISUALIZATION PPP. (OWN SOURCE) 73
FIGURE 54: FAH AND MCLAMBDA ANT2 SOLUTION GROUND TRACK. (OWN SOURCE) 75
FIGURE 55: POSITION VISUALIZATION FAH AND MCLAMBDA. (OWN SOURCE) 75
FIGURE 56: FAH AND MCLAMBDA UBLOX SOLUTION GROUND TRACK. (OWN SOURCE) 76
FIGURE 57: POSITION VISUALIZATION FAH AND MCLAMBDA. (OWN SOURCE) 77
FIGURE 58: SINGLE PROCESSING RTKLIB AND NOVATEL GROUND TRACK. (OWN SOURCE) 78
FIGURE 59: POSITION VISUALIZATION SINGLE MODE. (OWN SOURCE) 79
FIGURE 60: DGNSS AND PPPK ANT1 SOLUTION GROUND TRACK. (OWN SOURCE) 80
FIGURE 61: POSITION VISUALIZATION DGNSS AND PPPK SOLUTIONS. (OWN SOURCE) 80
FIGURE 62: FAH AND MCLAMBDA ANT2 SOLUTION GROUND TRACK. (OWN SOURCE) 81
FIGURE 63: POSITION VISUALIZATION FAH AND MCLAMBDA. (OWN SOURCE) 82
FIGURE 64: FAH AND MCLAMBDA UBLOX SOLUTION GROUND TRACK. (OWN SOURCE) 83
FIGURE 65: POSITION VISUALIZATION FAH AND MCLAMBDA. (OWN SOURCE) 83
FIGURE 66: UBLOX AND ANTENNA 2 TRAJECTORY MCLAMBDA. (OWN SOURCE) 84
FIGURE 67: NOVATEL CONFIGURATION. (OWN SOURCE) 89
FIGURE 68: RTKLIB CONFIGURATION SERIAL PORT IN RTKNAVI. (OWN SOURCE) 89

List of Tables

TABLE 1: APPLICABLE DOCUMENTS TO DGNSS.(ESA (EUROPEAN SPATIAL AGENCY)) 14
TABLE 2: "MULTIKIN" RELATIVE POSITIONING ACCURACY IN FIELD TEST 2. (LUO) 32
TABLE 3: STATIC FIELD TEST RESULTS. (GIORGI) 33
TABLE 4: TTFF STATISTICS FOR STATIC TEST. (ROTH ET AL.) 34
TABLE 5: UBLOX M8T CHARACTERISTICS. (UBLOX) 37
TABLE 6: RTKLIB APS. (TAKASU) 39

1. Introduction

1.1 Background

According to the ***European Space Agency*** (ESA (European Spatial Agency)), GNSS is defined as:

> *"**GNSS** stands for Global Navigation Satellite System and is the standard generic term for satellite navigation systems that provide autonomous geo-spatial positioning with global coverage. This term includes e.g. the GPS, GLONASS, Galileo, Beidou and other regional systems*"

To summarize, GNSS is the technology that allows us to position ourselves on the surface land. From observations to satellites, you can obtain the coordinates of a point on the surface. The basis of GNSS is to obtain the position of the satellites, that is, to obtain their coordinates in a reference system, in order to be able to calculate, from said coordinates, the position of the receiver.

GNSS positioning is a passive system where the receiver receives the signal from the decoded satellite, from which it calculates its position. Known positions of the satellites observed at the time of data collection (ephemeris), to obtain the receiver's position, it will be enough to measure the distances between the receiver and the satellites.

Each measurement (observable) is the distance between the satellite and the receiver and is based on the propagation of electromagnetic waves. Whereas the signal propagates in the vacuum at the speed of light ($3*10^5$ km/s) you can get the distance with the travel time of the signal, method known as calculation of the pseudorange (Hofmann-Wellenhof et al.).

Because it is not easy to determine the time of the signal, for higher measurements precision another method is employed.

Therefore, two methods of distance calculation, code measurement and measurement in phase (Berné Valero et al.).

> **Measurement in code:** Calculate the time elapsed between emission of the signal and reception of it. To determine this, the satellite emits a certain mark so that, when the signal that has been issued by the satellite, compares them and determines the increase in time that it has taken to receive it.
>
> **In-phase measurement:** The distance measurement can be calculated by measuring the not integer N of wavelengths and the non-integer part, and from there calculate the distance. It is not an easy method since the determination of N is not simple, problem that is known as determination of ambiguities.

To sum up, well-known the distance between the receiver and the satellites, it will be able to calculate the position of the receiver, previously known the coordinates of the satellites observed at the time of measurement.

1.2 Statement of problem

The differential carrier phase is a relative positioning technique, in which the inter-platform position vector between a reference station and a mobile station can be derived directly from the observables of the carrier phase. The absolute accuracy of a mobile station is based on the accuracy of the known coordinates of the reference station. Nowadays, in various applications, relative positioning instead of absolute positioning is the main concern. For most non-professional GNSS users, the absolute coordinates of an object, namely latitude, longitude and altitude in the World Geodetic System 1984 (WGS84) cannot give them a simple understanding of the object's location. However, if its location is provided relatively, such as 500 meters to the north and 1000 meters to the west of an established reference, the user can make an easier connection with the object's location (Luo).

The present investigation deals with the Implementation of Moving-Base-GNSS in NAVKA Multisensor GNSS/MEMS/Optics Navigation Algorithms and Systems. The main objective is to develop and improve a series of algorithms that allow obtaining better results in different platforms with GNSS sensors. For this purpose, different algorithms will be analyzed to obtain conclusions about them.

According to (Luo), the positioning search of multiple mobile platforms has the following characteristics:

1. In this application, the absolute positions of the objects are not important, but their relative positions, so the configuration of the reference station with precisely known coordinates is not mandatory.
2. High accuracy of relative positioning and reliability are required.
3. There are multiple platforms in the configuration, which implies that the multiplicity of platforms can improve the effectiveness of relative positioning.

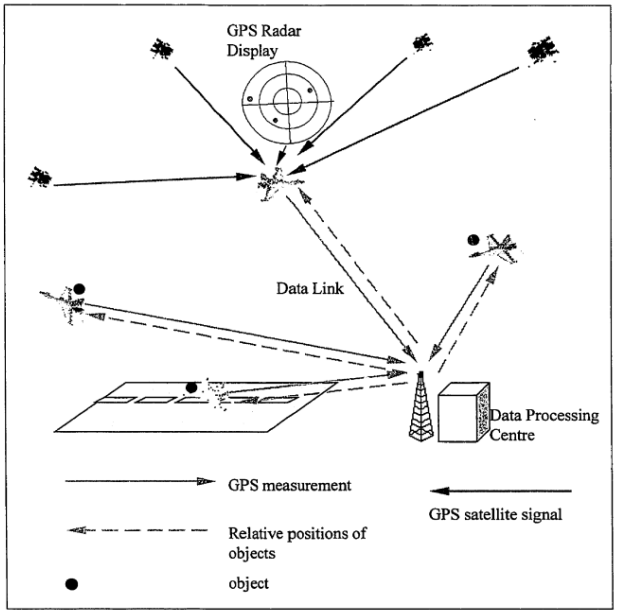

Figure 1 GNSS radar for aeronautical applications. (Luo)

1.3 Moving base concepts

Depending on the final objective that you want to obtain, there are different ways of designing and mounting a platform with different sensors (in this case, it concerns the situation of the GNSS receivers).

1.3.1 Moving base situation

As concerns the moving base situation, it considers the two cases, "GNSS direct on Body" and GNSS.

- Deep coupled motion model-free GNSS multi-sensor platform:

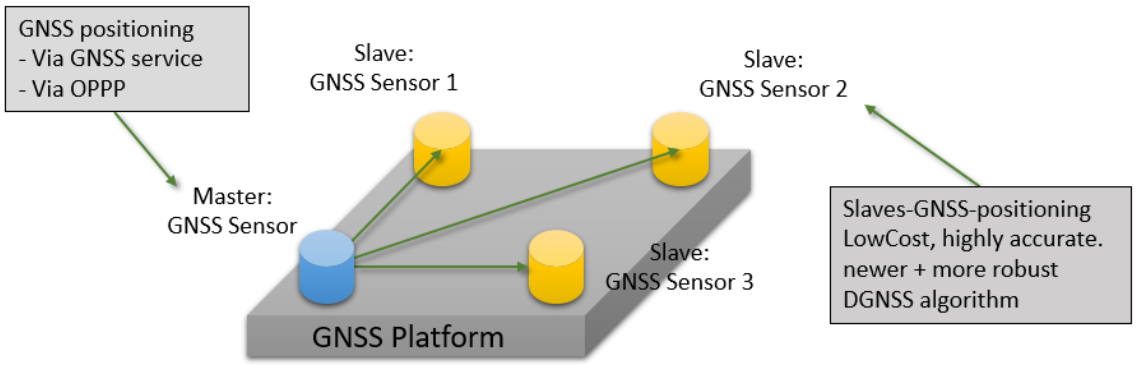

Figure 2: GNSS platform. (Own source)

In this case, two techniques can be used for GNSS positioning, one of which is to use a GNSS service, such as a network of permanent stations, which is nothing more than a set of GNSS receivers that are fixedly installed at points of known coordinates. They are connected to a network that provides services in deferred time or in real time, both raw data and processed data. Its purpose is to make measurements continuously to satellites. They are used, among other things, as geodetic networks.

- General interface:

There are three components to consider:

1) Component 1: sensor data connection opens multi-sensor design. Main sensors GNSS, INS, etc.
2) Component 2: research core of collaborative research basic algorithms for mobile multi-sensor platforms.
3) Component 3: application R & D mobile low-cost platform navigation and object georeferencing.

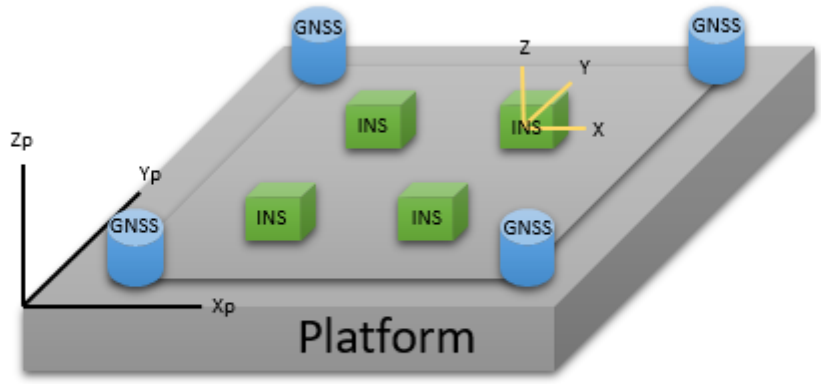

Figure 3: Multisensor Platform. (Own source)

The steps to follow for a good configuration and navigation calculation in GNSS-platform are (Jäger):

1) GNSS absolute positioning of one of the three receivers ("master") in the global terrestrial system (x, y, z). For example, with DGNSS or OPPP corrections.
2) Positioning of the remaining GNSS receivers ("slaves") with GNSS algorithms, which produce a high degree of relative accuracy for the master in ground-based systems (x, y, z).
3) Determination of the transformation parameters between the GNSS positions in the body frame (b) (x, y, z).in the terrestrial frame (e) (x, y, z). (through a three-dimensional similarity transformation.
4) Transformation of the body zero point in the fixed ground system (e) (x, y, z).
5) Calculation of (B, L (h)) of the zero point for the determination of the frame n Figure 4 and the associated rotation matrix n e R.
6) Transformation of the antenna positions (e) (x, y, z) in the n frame (x, y, z).
7) Determination of the elements of the rotation matrix R_b^n between the antenna positions in the frame (b) (x, y, z) and in the frame (n) (x, y, z). Of the elements of the matrix, also called attitude matrix

$$R_b^n = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\ cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta\,\phi & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}$$

(1.3.1)

$$R_b^n = C_b^n = \begin{pmatrix} C(11) & C(12) & C(13) \\ C(21) & C(22) & C(23) \\ C(31) & C(32) & C(33) \end{pmatrix} \qquad (1.3.2)$$

Then you can directly determine the aircraft spatially at n-frame angle Figure 4 oriented to Roll (ϕ) -, pitch (θ) - and Yaw (ψ).

$$\begin{pmatrix} roll \\ pitch \\ yaw \end{pmatrix} = \begin{pmatrix} \tan^{-1}[C_b^n(32)/C_b^n(33)] \\ \tan^{-1}[-C_b^n(31)/\sqrt{C_b^n(21)^2 + C_b^n(11)^2}] \\ \tan^{-1}[C_b^n(21)/C_b^n(11)] \end{pmatrix} \quad (1.3.3)$$

To determine the orientation in local navigation frame using right-handed ECEF frame (e-frame), the north (n^e), east (e^e), and down (d^e) axis are (see Figure 4):

$$d^e = \begin{bmatrix} -\cos\varphi\cos\lambda \\ -\cos\varphi\sin\lambda \\ -\sin\varphi \end{bmatrix}, \qquad n^e = -\frac{\delta d^e}{\delta\varphi}, \quad e^e = -\frac{1}{\cos\varphi}\frac{\delta d^e}{\delta\lambda}$$

(1.3.4)

And the rotation matrices from e-frame to n-frame and voiceovers are next equations (Groves):

$$R_e^n = \begin{bmatrix} -\sin\varphi\cos\lambda & -\sin\lambda\sin\varphi & \cos\varphi \\ -\sin\lambda & \cos\lambda & 0 \\ -\cos\varphi\cos\lambda & -\cos\varphi\sin\lambda & -\sin\varphi \end{bmatrix} \quad (1.3.5)$$

$$R_n^e = (R_e^n)^T \quad (1.3.6)$$

The final coordinates in n-frame will be obtained by:

$$x^n = R_e^n(\varphi,\lambda)\cdot(x^e - (x(\varphi,\lambda,h)^e) \quad (1.3.7)$$

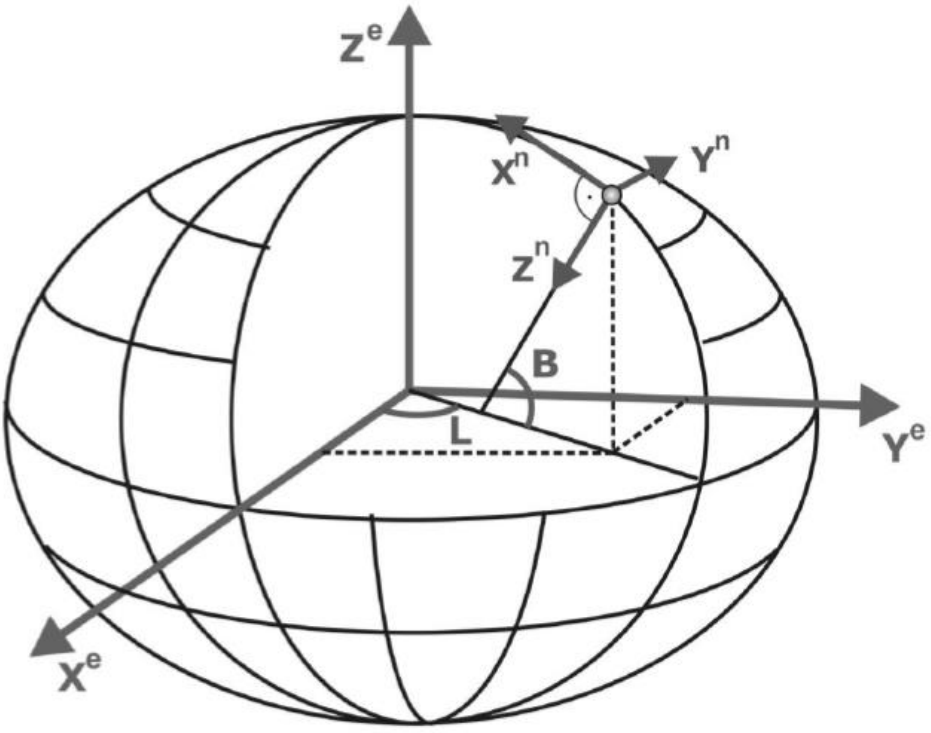

Figure 4: n-frame and e-frame. (Jäger)

1.3.2 Leverarms

As concern the leverarms there three different cases to be made:

1) Case 1: computation of a moving base configuration introducing only the distances between the receivers as conditions ("poor", but general configuration information).

2) Case 2, loose coupling: computation of a moving base configuration in a navigation scenario, (multisensor-multiplatform, or special case sensors on body). Here the sensor positions (and velocities), solved by case 1, mean an extension of the navigation state, and the leverarms equation is introduced as condition (in terms of an addition observation equation with small variance). But case 2 and case 1 are consistent, can be done and introduced independently.

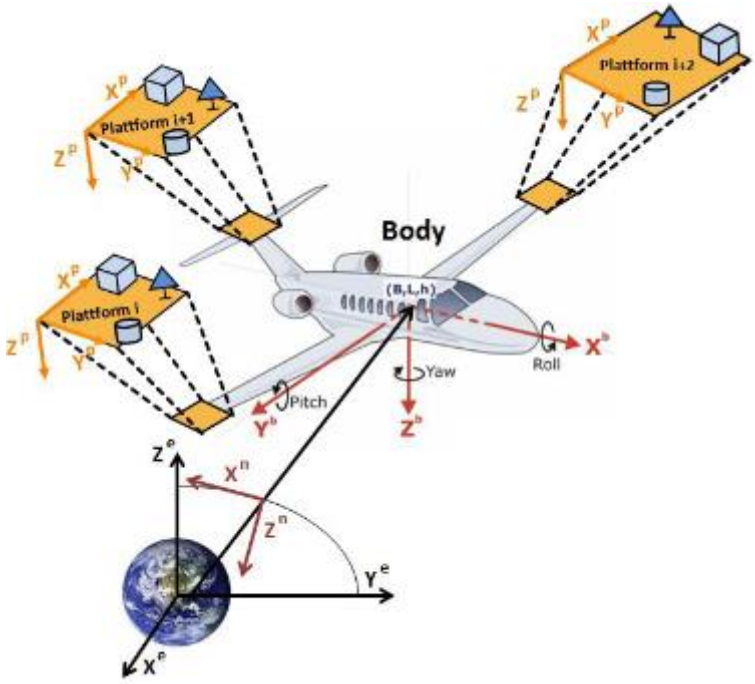

Figure 5: Leverarms general case. (Navka)

$$X^{e}_{GNSS-s_{i,j}} = x^{e}_{b} + R^{e}_{b}(r,p,y) * [t^{b}_{b,p_i} + R^{b}_{p_i} * t^{p_i}_{p_{i,},s_{i,j}}] \quad (1.3.8)$$

Where:

$X^{e}_{GNSS-s_{i,j}}$ Position vector of the GNSS on the e-frame.

x^{e}_{b} Position vector from b-frame to e-frame.

$R^{e}_{b}(r,p,y)$ Rotation matrix from b-frame to e-frame (roll, pitch, yaw).

t^{b}_{b,p_i} Platform position.

$R_{p_i}^b$ Platform orientation.

$t_{p_{i,}s_{i,j}}^{p_i}$ Sensor position on platform.

This formula refers to the general case of leverarms. In the previous figure it can be possible to see the construction of the general case.

The other possibility of construction is the special case where platform and body coincide, making the formula change.

$$X_{GNSS-s_{i,j}}^e = x_b^e + R_b^e(r,p,y) * t_{s_{i,j}}^b \qquad (1.3.9)$$

Where:

$t_{s_{i,j}}^b$ Sensor position on body.

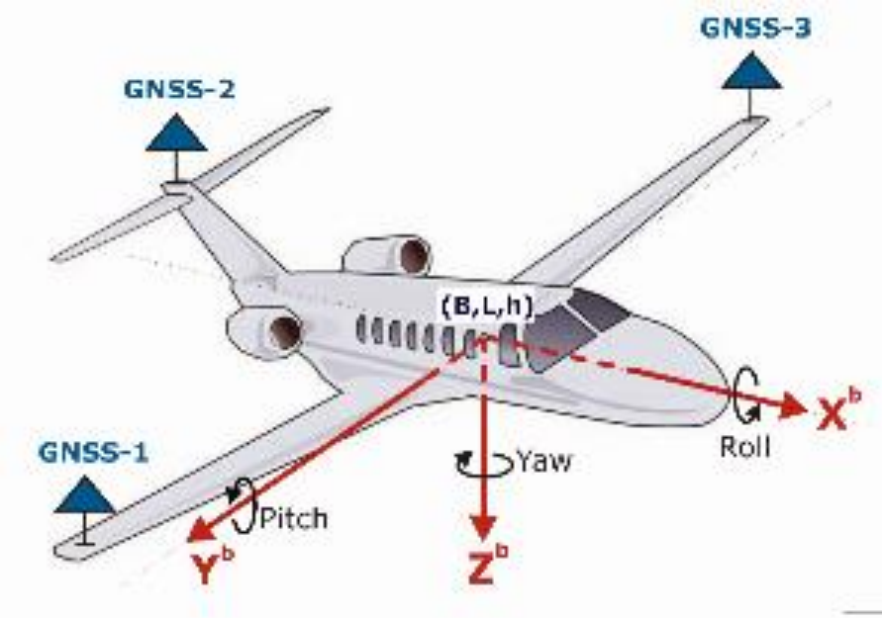

Figure 6: Special case, platform = body. (Navka)

3) Case 3, deep coupling based on case 2: it consists in the same process, but introducing the GNSS observations on code, phase, doppler related to the extended navigation state and the above leverarms conditions.

1.4 Objectives

A first target of the Master thesis is dealing with a review and evaluation of existing moving base GNSS algorithms.

This part is followed at second by real-data tests, and here at first using commercial moving base software in comparison with the moving base algorithm provided by the C/C++ open source software RTKLIB. For the moving base case of the RTKLIB, there

is an improved version provided by the NAVKA-team. As concerns the real data tests, at first the so-called GNSS/MEMS/camera navigation box shall be used. That multisensor navigation box has been developed by the NAVKA-team, and it is used for automotive out-/indoor navigation of special vehicles. The box carries - besides classical MEMS, a MEMS inclinometer and camera sensors - two Novatel L1/L2 GPS, and in addition one ublox M8T L1/E1/G1/B1 GNSS receiver. As concerns the Novatel GNSS component, the commercial moving base software from Novatel is available. The commercial Novatel software shall be used as a reference for the tests of the above mentioned RTKLIB moving base algorithm. For that purpose, the Novatel raw GNSS-data shall be logged in parallel and then being postprocessed, using RTKLIB (the RTKLIB can handle Novatel binary raw data) moving base algorithm along the same spatial 3D-trajectory (XYZ and NEH). By the simultaneous use and performance comparison of the two moving base software versions (Novatel/commercial and RTKLIB/open source), the open source software RTKLIB shall be improved in respect to the moving base settings, and the use of IGS-products (e.g. orbits & clocks) and SSR & OSR correction RTCM data her, both for the PPP-K and the DGNSS mode for the master-receiver. Further the optimum settings and data for the two-frequency DGNSS/GPS L1/L2 (Novatel), and the local one-frequency ublox M8T DGNSS/GPS L1 slave receivers shall be evaluated by the comparison of the Novatel and the RTKLIB results. In case, also further bug-fixing and changes in the C/C++ code of the RTKLIB must be considered.

1.5 Outline of dissertation

- Chapter 2: Short summary about the most important aspects of the GNSS theory used in this paper.
- Chapter 3: Analysis and study of several algorithms related to Moving-Base-GNSS, as well as its comparison and explanation of which has been implemented.
- Chapter 4: Description of the technical aspects of the hardware and software used in this work.
- Chapter 5: Development of the theory on which the implemented algorithm, formulation and calculation methodology is based.
- Chapter 6: Explanation of the routine implemented in C++, including technical and relevant aspects of programming, as well as implementation in RTKLIB.
- Chapter 7: Explanation of the software development for Moving Base, as well as the small software developed for reading Novatel data.
- Chapter 8: Evaluation and processing of tests performed in the field.
- Chapter 9: Conclusions and possible future lines

2. GNSS theory

Before starting with the analysis of the different existing algorithms, in the following section different theoretical concepts about GNSS necessary to better understand their operation are presented.

This theoretical part is an abstract from the website *Navipedia* (ESA (European Spatial Agency)).

2.1 RTK

2.1.1 RTK Fundamentals

With origin dating back to the mid-1990s, Real Time Kinematics (RTK) is a differential GNSS technique which provides high positioning performance in the vicinity of a base station. The technique is based on the use of carrier measurements and the transmission of corrections from the base station, whose location is well known, to the rover, so that the main errors that drive the stand-alone positioning cancel out. An RTK base station covers a service area spreading about 10 or 20 kilometers, and a real time communication channel is needed connecting base and rover. RTK, which achieves performances in the range of a few centimeters, is a technique commonly used in surveying applications.

2.1.2 Technique

From an architectural point of view, RTK consists of a base station, one or several rover users, and a communication channel with which the base broadcasts information to the users at real time.

The technique is based on the following high-level principles:

- In the neighbourhood of a clean-sky location, the main errors in the GNSS signal processing are constant, and hence they cancel out when differential processing is used. This includes the error in the satellite clock bias, the satellite orbital error, the ionospheric delay and the tropospheric delay. The main errors left without correction are multipath, interference and receiver thermal noise. Of the errors listed above, the only one which is truly constant with respect to the user location is the satellite clock bias; the rest will show a given dependency with the location as the rover moves away from the base station, being the tropospheric error the first to be fully de-correlated in a few kilometres from the base.
- The noise of carrier measurements is much smaller than the one of the pseudo-code measurements. The typical error of code pseudorange measurements is around 1 m, to compare with 5 mm for carrier phase measurements. However, the processing of carrier measurements is subject to the so-called carrier phase ambiguity, an unknown integer number of times the carrier wavelength, that needs to be fixed in order to rebuild full range measurements from carrier ones.

- The phase ambiguities can be fixed using differential measurements between two reference stations. There are different techniques available to fix them, some based on single frequency measurements with long convergence times, other taking benefit of dual frequency observables with shorter convergence. In general, the techniques either depend on a high precision knowledge of the ionosphere, or assume that the two stations are close enough so that the ionospheric differential delay is negligible when compared with the wavelength of the carriers, around 20 cm. The latter is the approached followed in RTK, limiting the service area to 10 or 20 km; the former is used in WARTK to cover big service areas with base stations separated around hundreds of kilometres away. The RTK approach needs continuity in the tracked measurements to avoid re-initialization of the phase-ambiguity filters; this is a severe limitation in urban environments due to the big number of obstructions.

The base station broadcasts its well-known location together with the code and carrier measurements at frequencies L1 and L2 for all in-view satellites. With this information, the rover equipment can fix the phase ambiguities and determine its location relative to the base with high precision. By adding up the location of the base, the rover is positioned in a global coordinate framework.

The RTK technique can be used for distances of up to 10 or 20 kilometres, yielding accuracies of a few centimetres in the rover position, to be compared with 1 m that is achieved with code-based differential GPS. Because of its high precision in controlled environments, RTK is extensively used in surveying applications.

2.1.3 RTK Algorithm

As stated in the previous section, one of the main problems in the RTK technique is fixing the phase ambiguities.

The RTK Algorithm is based on double differenced observables that can eliminate selective availability effects as well as other biases. The highlights of the algorithm are described next. At a given epoch, and for a given satellite, the simplified carrier phase observation equation is the following:

$$\phi = \rho - I + Tr + c(b_{Rx} - b_{Sat}) + [(N\lambda + \varepsilon)]_{\phi} \qquad (2.1.1)$$

Where:

I is the signal path delay due to the ionosphere;

Tr is the signal path delay due to the troposphere;

bRx is the receiver clock offset from the reference (GPS) time;

bSat is the satellite clock offset from the reference (GPS) time;

c is the vacuum speed of light;

λ is the carrier nominal wavelength;

N is the ambiguity of the carrier-phase (integer number);

$\varepsilon\phi$ are the measurement noise components, including multipath and other effects;

ρ is the geometrical range between the satellite and the receiver, computed as a function of the satellite (*xSat*,*ySat*,*zSat*) and receiver (*xRx*,*yRx*,*zRx*) coordinates as:

$$\rho = \sqrt{[(X_{Sat} - X_{Rx})]^2 + [(Y_{Sat} - Y_{Rx})]^2 + [(Z_{Sat} - Z_{Rx})]^2} \qquad (2.1.2)$$

For two receivers *a* and *b* making simultaneous measurements at the same nominal time to satellites *1* and *2*, the double difference observable is:

$$\phi_a^{12} - \phi_b^{12} = \rho_a^{12} - \rho_b^{12} - I_a^{12} + I_b^{12} + Tr_a^{12} - Tr_b^{12} + \lambda(N_a^{12} - N_b^{12}) + \varepsilon_a^{12} - \varepsilon_b^{12} \quad (2.1.3)$$

In the above equation receiver and satellite clock offsets and hardware biases cancel out. The single difference ambiguities difference $N12a - N12b$ is commonly parameterized as a new ambiguity parameter $N12ab$. The advantage of double differencing is that the new ambiguity parameter $N12ab$ is an integer because the non-integer terms in the GPS carrier phase observation, due to clock and hardware delays in the transmitter and receiver, are eliminated.

Although it would be possible to estimate the double difference ambiguity using a float approach instead of an integer one, this would lead to dm-level accuracy instead of cm-level. Hence, standard RTK fixes the ambiguities to integer figures.

Ambiguity Resolution: As stated in the section above, one of the keys to obtain the best accuracy from RTK is to fix the carrier phase ambiguities to integer numbers. Normally, this is done in three steps:

- The ambiguities are first fixed to float numbers using standard least-square techniques.
- The set of integer ambiguities is set to the one that optimizes the residuals in the surroundings of the float solution.
- The carrier measurements are corrected with the integer ambiguities and they are used to obtain the relative position of the rover to the base station.

Of these three steps, the second one is quite complex, because the float ambiguity covariance ellipsoid in the measurement space is extremely elongated. Therefore, the brute-force search process is inefficient, normally beyond the computational capabilities of the rover equipment. Several techniques have been developed to deal with this problem:

- **Carrier phase ambiguity fixing**

The carrier phase measurements are much more precise than the code pseudorange measurements (typically, about two orders of magnitude), but they contain the unknown ambiguities. If such ambiguities are fixed, thence the carrier phase measurements become as unambiguous pseudoranges, but accurate at the level of few millimetres.

- Double differenced ambiguity fixing

The carrier ambiguities will be considered in double differences between pairs of receivers and satellites. This is done in order to cancel out the fractional part of the ambiguities (b_{rec}, b^{sat}), being the remaining ambiguities integer number of wavelengths. That is, given

$$B_{rec}^{sat} = \lambda + N_{rec}^{sat} + b_{rec} + b^{sat} \qquad (2.1.4)$$

the double differences, regarding to a reference receiver and satellite, yield:

$$\Delta\nabla B_{rec}^{sat} = B_{rec}^{sat} - B_{rec_0}^{sat} - \left(B_{rec}^{sat_0} - B_{rec_0}^{sat_0}\right) = \lambda\Delta\nabla N_{rec}^{sat} \qquad (2.1.5)$$

where the satellite and receiver ambiguity terms (b_{rec}, b^{sat}) cancel-out.

- Undifferenced ambiguity fixing

The double-differenced ambiguities between pairs of satellites and receivers are integer numbers of wavelengths (see equation *(2.1.5)*). Indeed, the fractional part cancels in such double differences.

$$\Delta\nabla b_{rec}^{sat} = 0 \qquad (2.1.6)$$

An immediate consequence of previous equation *(2.1.6)* is the split of the fractional part of the ambiguities (for each satellite-receiver arch) in two independent terms, one of them linked only to the receiver and the other only to the satellite.

$$\Delta\nabla b_{rec}^{sat} = 0 \leftrightarrow b_{rec}^{sat} = b_{rec} + b^{sat} \qquad (2.1.7)$$

Notice that *(2.1.7)* means that fractional parts of the ambiguity (b_{rec}, b^{sat}) are not linked to a specific satellite-receiver arc, but b^{sat} depends only of satellite (and it is common to all carrier measurements of receivers tracking this satellite) and b_{rec} depends only to the receiver (and it is common to all satellites tracked by a given receiver).

2.2 DGNSS

2.2.1 DGNSS Fundamentals

The classical DGNSS technique is an enhancement to a primary GNSS system that consists of the determination of the GNSS position for an accurately surveyed position known as reference station. DGNSS accuracy is in the order of 1 m (1 sigma) for users in the range of few tens of km from the reference station.

2.2.2 The classical DGNSS technique

The standard DGNSS technique consists of the determination of the GNSS position from an accurately surveyed position known as reference station. The method takes advantage of the slow variation with time and user position of the errors due to ephemeris prediction, residual satellite clocks, ionospheric and tropospheric delays. Starting from the reference station, the system computes and broadcasts either corrections to the GNSS position or to the pseudorange measurements to the DGNSS users. In order to be able to apply these corrections, the receiver must be enabled for DGNSS and stay in the vicinity of the reference station to ensure that the two receivers (station and rover) observe the same GNSS satellite. Other uncorrelated errors (e.g. multipath) cannot be corrected by this method and specific techniques must be applied to mitigate them.

Variations of the method using corrections from multiple reference stations exist, leading to higher levels of accuracy.

2.2.3 DGNSS Algorithm

The classical DGNSS algorithm is based on single differences of pseudorange observables. At a given epoch, and for a given satellite, the simplified pseudorange observation equation is the following:

$$P = \rho + I + Tr + c(b_{Rx} - b_{Sat}) + \varepsilon P \qquad (2.2.1)$$

Where:

I is the signal path delay due to the ionosphere;

Tr is the signal path delay due to the troposphere;

bRx is the receiver clock offset from the reference (GPS) time;

bSat is the satellite clock offset from the reference (GPS) time;

c is the vacuum speed of light;

εP are the measurement noise components, including multipath and other effects;

ρ is the geometrical range between the satellite and the receiver, computed as a function of the satellite (*xSat*,*ySat*,*zSat*) and receiver (*xRx*,*yRx*,*zRx*) coordinates as:

$$\rho = \sqrt{[(X_{Sat} - X_{Rx})]^2 + [(Y_{Sat} - Y_{Rx})]^2 + [(Z_{Sat} - Z_{Rx})]^2} \qquad (2.2.2)$$

The next step is using a reference station at an accurately calibrated location (*xo*,*yo*,*zo*), the reference-to-satellite range can be calculated as:

$$R_0 = \sqrt{[(X_{Sat} - X_0)]^2 + [(Y_{Sat} - Y_0)]^2 + [(Z_{Sat} - Z_0)]^2} \qquad (2.2.3)$$

When single differencing of both pseudoranges, $\Delta P = Po - P$, the ionospheric and tropospheric delays cancel out and the satellite clock offset.

2.2.4 RTCM

The internationally accepted data transmission standards for DGNSS are defined by RTCM, particularly by its Special Committee SC-104. RTCM SC-104 is a standard that defines the data structure for differential correction information for a variety of differential correction applications. It was developed by the Radio Technical Commission for Maritime Services (RTCM) and has become an industry standard for communication of correction information. Note that RTCM is a binary data protocol.

The applicable documents to DGNSS systems are listed in the *Table 1* and constitute the current version of the core set of documents to be used for the development of a new DGNSS system.

Document Item	Reference	Issue	Comments
Recommended Standards for Differential GNSS (Global Navigation Satellite Systems) Service	RTCM 10402.3	2.3	This standard is used around the world for differential satellite navigation systems, both maritime and terrestrial.
Differential GNSS (Global Navigation Satellite Systems) Services	RTCM 10403.1	3.1	A more efficient alternative to RTCM 10402.3
Standard for Networked Transport of RTCM via Internet Protocol (Ntrip)	RTCM 10410.0	1	A more efficient alternative to RTCM 10402.3
Standard for Differential Navstar GPS Reference Stations and Integrity Monitors (RSIM)	RTCM 10401.2	2	A companion to RTCM 10402.3, this standard addresses the performance requirements for the equipment which broadcasts DGNSS corrections

Table 1: Applicable documents to DGNSS.(ESA (European Spatial Agency))

2.3 PPP

2.3.1 PPP Fundamentals

Precise Point Positioning (PPP) is a global service of precise positioning, since it requires the availability of accurate reference clock and satellite orbit products in real time using a network of GNSS reference stations distributed throughout the world. By combining the precise positions of satellites and clocks with a dual frequency GNSS receiver (to eliminate the first-order effect of the ionosphere), PPP can provide position solutions at the level of centimeter to decimeter, even less than 1 cm from level in static mode. PPP differs from double-difference real-time kinematic (RTK) positioning in that it does not require access to observations from one or more nearby reference stations inspected accurately and that PPP provides absolute positioning instead of relative location the reference station as RTK does. PPP only requires accurate orbit and clock data, calculated by a processing center with reference station measurements from a relatively dispersed network of stations (thousands of km away would suffice). This makes PPP a very attractive alternative to RTK for those areas where RTK coverage is not available. On the contrary, the PPP technique is not yet as consolidated as RTK and requires a longer convergence time to achieve maximum yields (of the order of tens of minutes). Currently, there are several consolidated postprocessing PPP services. On the contrary, real-time PPP systems are in an incipient phase of development. The PPP algorithm uses as input code and phase observations from a dual frequency receiver, and precise satellite orbits and clocks, to calculate the precise coordinates and clock of the receiver. Observations from all satellites are processed together in a filter that solves the different unknowns, namely the coordinates of the receiver, the clock of the receiver, the tropospheric zenithal delay and the phase ambiguities.

2.3.2 PPP benefits and Prospects

As it has been mentioned before, PPP technique offers significant benefits compared to differential precise positioning techniques:

- PPP involves only a single GPS receiver and, therefore, no reference stations are needed in the vicinity of the user.
- PPP can be regarded as a global position approach because its position solutions 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 reduces labor and equipment cost and simplifies operational logistics to field work since it eliminates the dependency on base station(s).
- PPP can support other applications beyond positioning. For example, as PPP technique estimates receiver clock and tropospheric effect parameters in addition to position coordinate parameter, it provides another way for precise time transfer and troposphere estimation using a single GPS receiver.

3. Existing moving base GNSS algorithms

3.1 Introduction

The following section consists of review and evaluation of existing moving base GNSS algorithms.

According to (Wikipedia), algorithm is defined as:

> "In mathematics and computer science, an **algorithm** is an unambiguous specification of how to solve a class of problems. Algorithms can perform calculation, data processing, automated reasoning, and other tasks."

It means, in this apart, different algorithm, designed to solve moving base GNSS problems, are compared.

3.2 Algorithms

3.2.1 Algorithm "MultiKin"

This section is an abstract from the doctorate thesis by (Luo).

"MultiKin" is a method developed for relative positioning, which can process the data from multiple GPS platforms at the same time. It has two characteristics. First, it does not need reference stations with precisely known coordinates. Second, it can make use of the constraints provided by the multiplicity of receivers to improve the OTF ambiguity resolution and hence to improve the efficiency and reliability of the relative positioning of each baseline.

The procedure of MultiKin consists of three steps:

1. Constraints construction

   1) The fixed baseline constraint

      A fixed baseline means that the inter-platform distance (baseline length) is precisely known. This can be used to aid in fixing ambiguities between the moving rover pair. Generally, the fixed baseline length is obtained from an external source other than GPS, such as a tape measurement.

   2) Attitude constraint

      When the integer ambiguities of a moving baseline are fixed, the fixed baseline can help fix the integer ambiguities between a reference station and the moving rover pair. Because the attitude of the "moving baseline" can be precisely estimated, the solved attitude can then be used as a constraint for the ambiguity resolution between the reference and rover stations.

   3) Approximate coordinate constraint

The most often used coordinate constraint is a height constraint. It is extremely useful in marine applications where the height is well known.

4) Ambiguity constraint

The concept of the ambiguity constraint is that the sum of the double-differenced ambiguities in a closed polygon is zero.

Since then, it has been widely used to aid ambiguity resolution:

$$\Delta\nabla N_{1,2} + \Delta\nabla N_{2,3} + \cdots + \Delta\nabla N_{m-1,m} + \Delta\nabla N_{m,1} = 0 \qquad (3.2.1)$$

Where $\Delta\nabla N_{i,j} = (N_i^A - N_j^A) - (N_i^B - N_j^B)$ is the double differenced integer ambiguities:

i,j are the indices of the GPS platforms (nodes of polygon),

A is the common satellite observed by all platforms, and

B is the common base satellite.

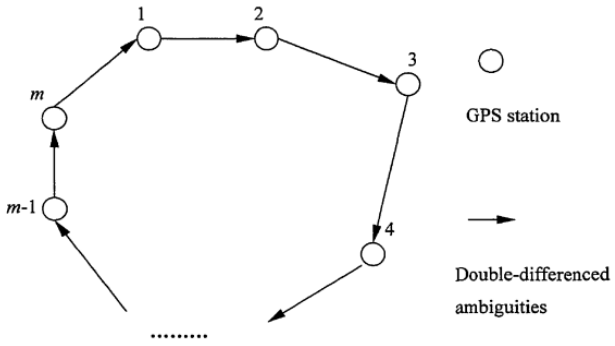

Figure 7 Concept of closed (polygonal) ambiguity constraints. (Luo)

Here are two prominent advantages of triangular constraints over polygonal constraints, **higher efficiency** and **higher reliability.**

2. Individual baseline resolution

The selection of baselines determines the effectiveness of an algorithm to construct ambiguity constraints. An optimal approach to select the baselines and construct constraint triangles must meet all the following requirements:

a. Reasonable computational burden
b. Effective use of constraints
c. Selection of the shortest baselines

In a GPS configuration containing *l* moving platforms, the numbers of baselines and triangles are *l(l - 1)/2 and l(l - 1)(l - 2)/6* respectively. When there are only three platforms in the configuration, the selection of baselines for ambiguity constraints is unique. However, when the number of platforms is more than three, the numbers of optional baselines and triangles increase dramatically. If

all the baselines and possible triangular constraints are used to aid ambiguity resolution, the extremely heavy computational burden can result in difficulties with real-time processing. For instance, selecting all the moving baselines and triangles under a configuration of 50 GPS platforms leads to simultaneous processing of 1225 moving baselines and 19600 triangular constraints. This requires a very high-speed processor and a very large amount of memory.

3. Enhancement of ambiguity fixing using constraints.

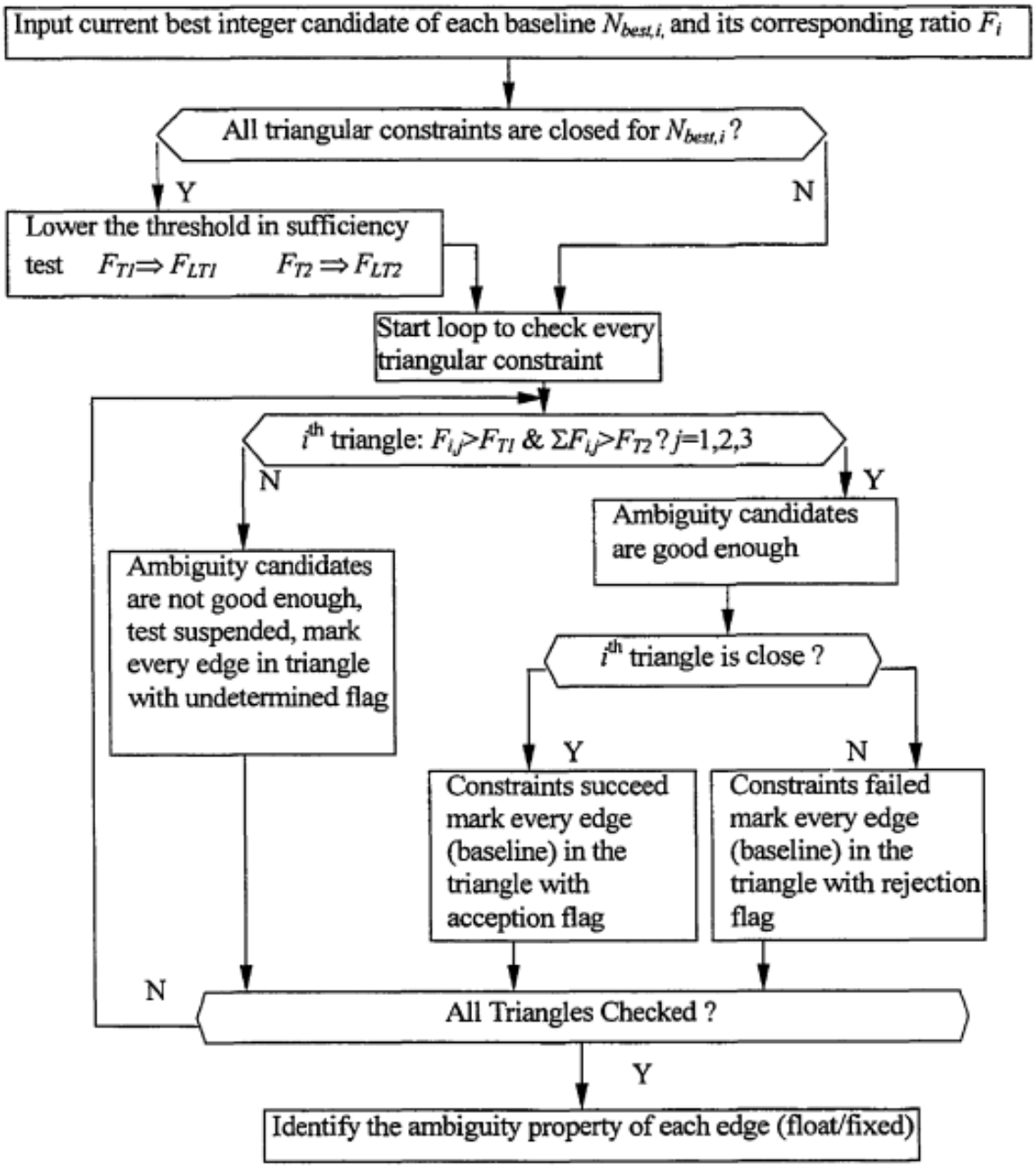

Figure 8 Procedure of ambiguity determination and monitoring using multiple. (Luo)

The overall procedure used in *"MultiKin"* is summarized in *Figure 9 .*First, *m* baselines connecting *t* moving platforms are selected by Delaunay triangulation to construct *n* triangles for applying ambiguity constraints. Second, those *m* ambiguity search modules try to fix ambiguity sets for each baseline individually. Each module outputs the float ambiguity set $\widehat{N}_i$, the best integer ambiguity set $\widetilde{N}_i$, and its corresponding ratio *Fi*. *Fi* is defined by:

$$F = \frac{\Omega(\widetilde{N}^2)}{\Omega(\widetilde{N}^1)} > F_r \qquad (3.2.2)$$

Where

$(\widetilde{N}^1)$ Is the best integer ambiguity candidate.

$(\widetilde{N}^2)$ Is the second-best integer ambiguity candidate.

$\Omega(\widetilde{N}) = (\widetilde{N} - N_{float})^T C^{-1}_{float|\widetilde{N}} (\widetilde{N} - N_{float})$

$C^{-1}_{float|\widetilde{N}}$ Is the conditional covariance matrix for the float ambiguities.

If the integer ambiguity set is successfully fixed to $\overline{N}_i$, it will be also output. In the third step, the algorithm for multiple-platform ambiguity determination is used to check whether the combination of the best integer candidates $\widetilde{N}_i$ can be the integer solution or whether $\overline{N}_i$, is wrongly fixed.

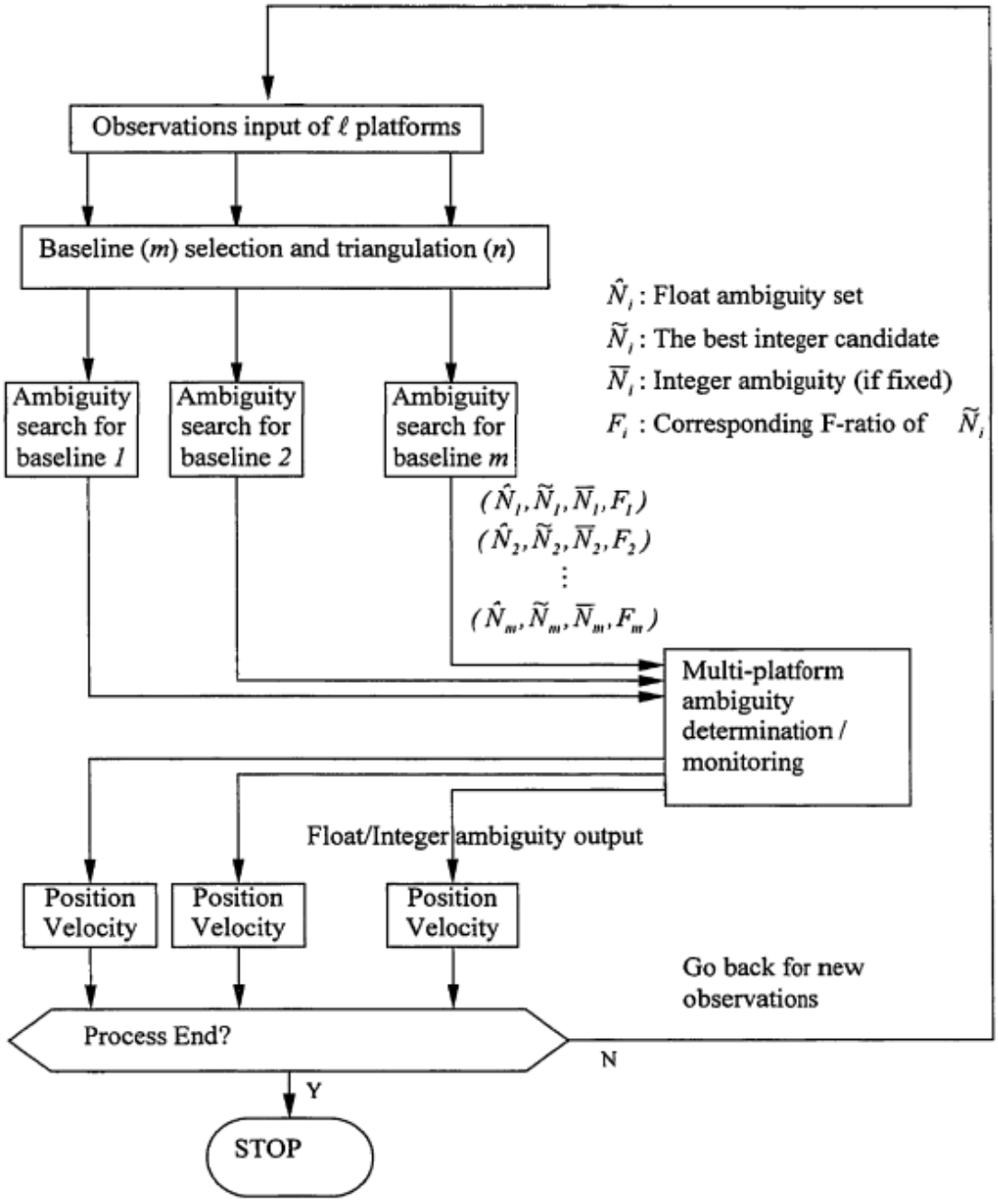

Figure 9 Procedure of MultiKin. (Luo)

3.2.2 Algorithm “Multivariate Constraint LAMBDA = MC - LAMBDA”

This section is an abstract from the article by (Giorgi).

The ambiguity resolution is the first step of any application based on carrier phase; the ambiguities inherent in the phase measurements must be resolved correctly.

This contribution restricts the focus on the theory of integer least squares (ILS), which applies to linear systems where a subset of the unknowns is integrated. A well-known implementation of the ILS principle is the LAMBDA method (Least-squares AMBiguity Decorrelation Adjustment), an ambiguity resolution algorithm widely used for unrestricted and linearly restricted applications. Although very effective, the method is not specifically designed for nonlinear restricted applications.

Attitude determination based on GNSS (see *Figure 10*) is a viable technique to estimate the orientation of a platform. To extract the attitude angles, the relative positions of the antennas are known in advance. This prior knowledge can be used in validation procedures, to test reference lengths and mutual orientations. Otherwise, nonlinear geometric constraints can be used directly to aid the fixing process. We focus on the last approach: the incorporation of restrictions requires a non-trivial modification of the LAMBDA method. The function that should be minimized when applying the ILS principle is modified with respect to the unrestricted case, which makes the whole procedure more complex. The modified method was first implemented through the restricted LAMBDA (C-LAMBDA) method, which is solved for restricted single baseline models, where only the relative distance between two antennas is known.

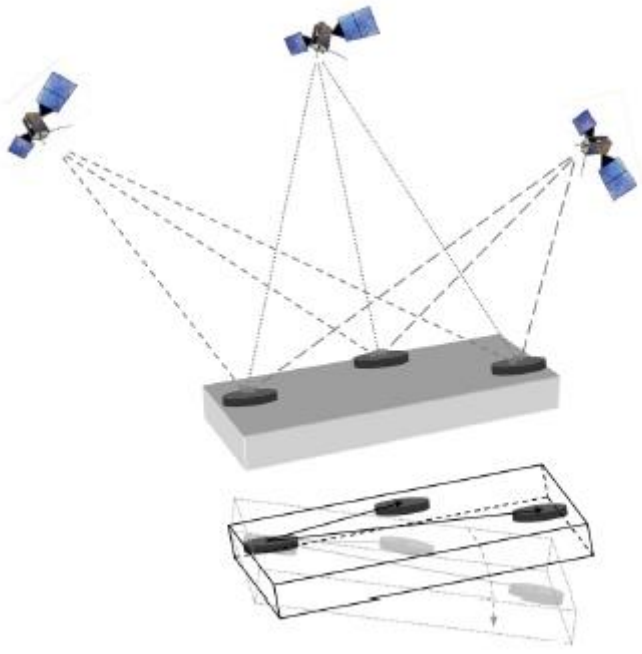

Figure 10: GNSS signals received from three or more antennas onboard. (Giorgi)

A generalization of the C-LAMBDA method, the Multivariate Constrained LAMBDA method (MC-LAMBDA), which is reviewed and tested in this contribution. This method is applicable to any number of baselines, with the complete set of nonlinear constraints integrated in the ambiguity resolution routines. Therefore, not only the reference lengths, but also the relative orientations between the antennas are modelled and incorporated as geometric constraints. The method integrally solves both the entire ambiguities and the orientation angles, maximizing the possibilities of correct correction.

- **The GNSS-BASED attitude model**

  - The unconstrained model

Assuming a set of three or more antennas, say *m* + 1, which track the same number of GNSS satellites (*n* + 1) on the same frequency, the double difference code (DD) and the observable phases for each baseline *j* = 1,..., *m* are released as:

$$E(y_j) = Az_j + Gb_j \qquad z_j \in \mathbb{Z}^n;\ b_j \in \mathbb{R}^p$$

$$D(y_i) = Q_y$$

E() is the expectation operator, while y_j is the vector of the GNSS observations collected at baseline *j*. z_j is the *j-th* vector of ambiguities of integer values (order *n*) and b_j the *j-th* vector of unknowns with real values remaining. We limit our analysis to short baselines (a few hundred meters), and atmospheric delays can be neglected. In addition, clock biases are cancelled after differentiation, and the only unknowns with actual values that are estimated are the baseline coordinates (*p = 3*). A contains the carrier wavelengths, while G is the matrix of the line-of-sight unit vectors. For antennas placed on the same platform, the same geometric matrix is used G. D() is the dispersion operator; it is assumed that the observable vector and *j* is affected by an error distributed in Gauss. This is described by the variance-covariance matrix (*v-c*) *Qy*, assumed identical for the different baselines. The complete set of observations collected in the different baselines can be formulated as the multivariate GNSS model:

$$E(Y) = AZ + GB \qquad Z \in \mathbb{Z}^{n*m};\ B \in \mathbb{R}^{3*m}$$

$$D(vec(Y)) = Q_Y \qquad (3.2.3)$$

where Y is the matrix whose *j-th* column is the vector of observations and *j*, Z is the matrix that contains the ambiguities of the carrier phase z_j and B is the matrix whose *j-th* column is the vector of the baseline coordinates b_j. The *vec* operator is introduced to define the matrix *v-c* of the observables: stack the columns of the *2n* matrix by m in a vector of order *2nm*. The dispersion of the vector *vec*(Y) is characterized by the matrix *v-c* QY.

If the antennas are firmly mounted on a structure (see Figure 10), an integral local coordinate system is used with the platform to define the antenna coordinates in relation to the body. The matrix of baseline coordinates expressed in the local frame

is indicated with F. If the movements of the platform are rigid, that is, the deformation of the structure carrying the antennas is insignificant, the local coordinates are linked to the coordinates global through the attitude matrix R, which describes the relative orientation between local and global coordinates B = RF. Therefore, model *(3.2.3)* is rewritten as:

$$E(Y) = AZ + GRF \qquad Z \in \mathbb{Z}^{n*m}$$
$$D(vec(Y)) = Q_Y \qquad (3.2.4)$$

- The constrained model

Model (b) does not explicitly explain the geometric constraint in the attitude matrix R. For rigid platforms, the rotation matrix belongs to the class of orthogonal matrices O^{3*q}, for which $R^TR = I_q$:

$$E(Y) = AZ + GRF \qquad Z \in \mathbb{Z}^{n*m}; R \in O^{3*q}$$
$$D(vec(Y)) = Q_Y \qquad (3.2.5)$$

The parameter *q* is entered to avoid loss of generality when only one (*q = 1*) or two (*q = 2*) baselines are available. When four or more antennas are considered, the parameter assumes the value *q = 3*. Note that in the case *m = q = 1*, model *(3.2.5)* formally matches the restricted simple baseline model. The solution of *(3.2.5)* must meet two different restrictions: the ambiguity matrix of the carrier phase is an integer value, $Z \in \mathbb{Z}^{n*m}$, and the attitude matrix is orthogonal, $R \in O^{3*q}$.

- **ILS Theory applied to the GNSS attitude problem**

In this section the solutions of the models *(3.2.4)* - *(3.2.5)* are given. Both solutions are derived following the same three-step procedure. First, the floating solution is derived, without considering any restrictions that arise in the unknown matrix. Then, an extensive search of the entire matrix of ambiguities is carried out, with the aim of minimizing the weighted square norm of least squares residues. As a last step, the attitude solution is adjusted according to the integer minimizer found. The two solutions differ only in the second step, since they are characterized by different cost functions that must be minimized. Since the ILS principle applies to vectors, it is convenient to convert the expressions *(3.2.4)* - *(3.2.5)* into a vector form, using the *vec* operator:

$$E\big(vec(Y)\big) = [(I_m \otimes A)\ (F^T \otimes G)] \begin{pmatrix} vec(Z) \\ vec(R) \end{pmatrix} \qquad (3.2.6)$$

where $\otimes$ denotes the product Kronecker.

- The float solution

The floating solution, that is, the least squares solution obtained without considering the restriction of integers in *(3.2.4)* - *(3.2.5)* and the restriction of orthonormality in *(3.2.5),* is derived as:

$$N * \begin{pmatrix} vec(\hat{Z}) \\ vec(\hat{R}) \end{pmatrix} = \begin{bmatrix} I_m \otimes A^T \\ F \otimes G^T \end{bmatrix} Q_Y^{-1} vec(Y)$$

(3.2.7)

$$N = \begin{bmatrix} I_m \otimes A^T \\ F \otimes G^T \end{bmatrix} Q_Y^{-1} [\, I_m \otimes A \quad F^T \otimes G]$$

$\hat{Z}$ is the floating estimator of the real value of the entire ambiguities, while $\hat{R}$ is the floating estimator, usually not orthogonal, of the attitude matrix. The accuracy of flotation estimators is characterized by the variance-covariance matrix (*v-c*):

$$\begin{bmatrix} Q_{\hat{Z}} & Q_{\hat{Z}\hat{R}} \\ Q_{\hat{R}\hat{Z}} & Q_{\hat{R}} \end{bmatrix} = N^{-1} \qquad (3.2.8)$$

Solving the systems *(3.2.4)* - *(3.2.5)* under the assumption that the ambiguities of the carrier phase are resolved, would give the solution of adjusted attitude

$$vec\left(\hat{R}(Z)\right) = vec(\hat{R}) - Q_{\hat{R}\hat{Z}} Q_{\hat{Z}}^{-1} vec(\hat{Z} - Z)$$

$$Q_{\hat{R}(Z)} = Q_{\hat{R}} - Q_{\hat{R}\hat{Z}} Q_{\hat{Z}}^{-1} Q_{\hat{Z}\hat{R}} \qquad (3.2.9)$$

- The ILS principle

The application of the ILS principle to *(3.2.6)* aims to minimize the norm of weighted residues (by the matrix *v-c* of the observables), which can be broken down into a sum of squares as follows:

$$\|vec(Y) - (I_m \otimes A) * vec(Z) - (F^T \otimes G) vec(R)\|_{Q_Y}^2 = \left\|\hat{E}\right\|_{Q_Y}^2 + \left\|vec(Z - \hat{Z})\right\|_{Q_{\hat{Z}}}^2 + \left\|vec(\hat{R}(Z) - R)\right\|_{Q_{\hat{R}(Z)}}^2$$

(3.2.10)

If we ignore the geometric restriction in R, the minimizer of *(3.2.10)* is minimizing only the second term on the right side, it being always possible to make the last term equal to zero by choosing R = $\hat{R}$ (Z). The minimization of the ambiguity term is achieved using the well-known LAMBDA method. However, if the orthonormality constraint is included in R, the last term on the right side generally differs from zero and minimizing *(3.2.10)* means minimizing the sum of two closely coupled terms. This makes the whole process more complicated, and a modification of the standard LAMBDA method is necessary. The advantage of incorporating the orthonormality constraint lies in the reinforced global model, which improves the correct ambiguity correction. Both solutions are given in the following sections.

- **The LAMBDA method**

The **LAMBDA method** is a reliable and widely used implementation of the ILS principle, suitable for least squares problems with unrestricted and linearly restricted integers. Considering only the integer restriction, minimizing *(3.2.10)* is the same as finding the minimizer of:

$$\check{Z}^U = \arg_{Z\in\mathbb{Z}^{n*m}} \min \left\| vec(Z - \hat{Z}) \right\|^2_{Q_{\hat{Z}}} \qquad (3.2.11)$$

There are no known closed-form solutions to the minimization problem *(3.2.11).* The search for the integer minimizer $\check{Z}^U$ is performed extensively in a subset of the integer matrix space $\mathbb{Z}^{n*m}$. This subset, that is, the search space, is formed by choosing a scalar to limit its size:

$$\Omega^U(\chi^2) = \left\{ \mathbb{Z}^{n*m} \middle| \left\| vec(Z - \hat{Z}) \right\|^2_{Q_{\hat{Z}}} \leq \chi^2 \right\} \qquad (3.2.12)$$

A careful choice is needed for the value of χ^2: it must be small enough to limit the computational load, but the lack of a vacuum of Ω^U must be guaranteed. In general, it is a good option to use the start value Z^b to set the size of Ω^U:

$$\chi^2 = \left\| vec(Z^b - \hat{Z}) \right\|^2_{Q_{\hat{Z}}} \qquad (3.2.13)$$

The search is carried out quickly through a decorrelation of the matrix $Q_{\hat{Z}}$, which softens the spectrum of eligible whole candidates. All entire candidates within the search space $\Omega^U(\chi^2)$ are examined and the minimizer of *(3.2.11)* is extracted.

- **The MC-LAMBDA method**

When considering the restriction of orthonormality in R, the minimization problem is modified as:

$$\check{Z}^C = \arg_{Z\in\mathbb{Z}^{n*m}} \min C(Z)$$

$$C(Z) = \left\| vec(Z - \hat{Z}) \right\|^2_{Q_{\hat{Z}}} + \left\| vec(\hat{R}(Z) - \check{R}(Z) \right\|^2_{Q_{\hat{R}(Z)}}$$

(3.2.14)

With

$$vec(\hat{R}(Z)) = \arg_{R\in O^{3*q}} \min \left\| vec(\hat{R}(Z) - R) \right\|^2_{Q_{\hat{R}Z}} \qquad (3.2.15)$$

The minimization process is complicated by the close coupling between the entire term and the attitude term. The evaluation of the cost function C(Z) for a given matrix of Z requires the solution of the problem of restricted least squares *(3.2.15).* This makes an extensive search strategy quite inefficient, especially if the search space size cannot be correctly limited.

$$\Omega^C(\chi^2) = \{Z \in \mathbb{Z}^{n*m} | C(Z) \leq \chi^2\} \qquad (3.2.16)$$

choosing a suitable value for the scalar χ. To overcome the problems associated with the greater computational load, two search strategies have been developed: The **Search and Shrink** approach, and the **Expansion** approach. Both strategies are based on the introduction of two delimitation functions for C(Z):

$$C_1(Z) \leq C(Z) \leq C_2(Z)$$

$$C_1(Z) = \left\| vec\left(Z - \hat{Z}\right) \right\|^2_{Q_{\hat{Z}}} + \lambda_m \sum_{i=1}^{q} (\|\hat{r}_i(Z)\| - 1)^2$$

$$C_2(Z) = \left\| vec\left(Z - \hat{Z}\right) \right\|^2_{Q_{\hat{Z}}} + \lambda_M \sum_{i=1}^{q} (\|\hat{r}_i(Z)\| + 1)^2$$

(3.2.17)

where $\hat{r}_i(Z)$ is the *i-th* column of $\hat{R}(Z)$ and λ_m, λ_M are the smallest and largest eigenvalues of the matrix $Q^{-1}_{\hat{R}(Z)}$, respectively. The inequalities are derived from the rules of the scalar product between vectors. The evaluation of $C_1(Z)$ and $C_2(Z)$ only requires the calculation of squared standards, and a search method based on these functions can proceed much faster than the use of the original cost function C(Z).

The **Search and Shrink** approach work by iteratively shrinking the search space:

$$\Omega_2(\chi^2) = \{Z \in \mathbb{Z}^{n*m} | C_2(Z) \leq \chi^2\} \subseteq \Omega^C(\chi^2)$$

Until you find the minimizer of $C_2(Z)$. The minimizer of C(Z), which may differ from that of $C_2(Z)$, is widely sought within the shrunken assembly:

$$\Omega^C(\bar{\chi}^2) = \{Z \in \mathbb{Z}^{n*m} | C(Z) \leq \bar{\chi}^2\} \supseteq \Omega_2(\bar{\chi}^2)$$

Where $\bar{\chi}^2 = C_2(\bar{Z})$, being $\bar{Z}$ the minimizer of $C_2(Z)$. The shrunken set $\Omega^C(\bar{\chi}^2)$ is usually small, because of which the computation of (m) is needed only a few times.

The **Expansion** approach works the other way around: starting with a small search space:

$$\Omega_1(\chi^2) = \{Z \in \mathbb{Z}^{n*m} | C_1(Z) \leq \chi^2\} \supseteq \Omega^C(\chi^2)$$

Scalar χ^2 is increased iteratively until $\Omega^C(\bar{\chi}^2)$ is not empty and the minimizer is removed.

Both the search strategies, which adaptively adjust the size of the search space by reducing or expanding the set of candidates as the search progresses, allow a quick and efficient search of the $\check{Z}^C$ integer minimizer.

- **The attitude solution**

Once the integer matrix of carrier phase ambiguities is found, the attitude solution is obtained as:

$$\check{R} = \arg_{R \in O^{3*q}} \min \left\| vec(\hat{R}(\check{Z}) - R) \right\|^2_{Q_{\hat{R}Z}} \qquad (3.2.18)$$

With

$\check{Z} = \check{Z}^U$ for the unconstrained model

$\check{Z} = \check{Z}^C$ for the constrained model

The rotation matrix $\check{R}$ is the orthogonal matrix that minimizes the distance to the matrix $\check{R}(\check{Z})$, in the metric of the *v-c* matrix $Q_{\hat{R}Z}$. The solution of *(3.2.18)* can be found with Newton's method, through an adequate parameterization of the rotation matrix, for which the orthonormality restriction is implicitly fulfilled. Examples of parameterizations are Euler's angles, Gibbs vector or Quaternion representations.

3.2.3 Algorithm “Lambda with constraints”

This section is an abstract from the article by (Roth et al.).

This section describes an integration of a single antenna and two antenna heading system with low-cost inertial field and magnetic field sensors to improve the availability and reliability of the GNSS pure attitude determination. This method calculates a redundant attitude solution in an error state Kalman filter using different sensor configurations. As a result, the ambiguity resolution process of the carrier phase is accelerated.

This not only reduces the repair time (TTFF), but also increases the reliability of fixed ambiguities.

- **The challenge of attitude determination**

The determination of the horizontal attitude poses a general problem for navigation applications. While the swing and tilt angles can be calculated from the accelerometer measurements of the gravity vector, the angle of rotation is poorly observable.

However, a GNSS compass provides trouble-free attitude information for any systematic compensation error. With a set of at least three antennas, the complete orientation of the antenna structure can be determined.

A generalized technique to identify ambiguities in the carrier phase is the LAMBDA (Least-squares AMBiguity Decorrelation Adjustment).

Using a floating estimate of ambiguities and the corresponding variance matrix, we can solve the problem of whole least squares in a very efficient way, achieving a double frequency data resolution within a few observation periods.

However, the use of relative positioning introduces new opportunities because additional information can be provided. A reduction of the ambiguity search space is achieved by considering the known reference length.

As a result, real-time resolution of double difference ambiguities from single frequency data is possible.

For a greater acceleration of the ambiguity identification process, an extension of the LAMBDA algorithm has been proposed to allow a perfect integration of the inclination angle and yaw restrictions in the LAMBDA method, which in turn produces an additional shortening of the time for the first repair.

- **Extended LAMBDA Method**

With the measurements of two receivers and the creation of wide-angle combinations, the LAMBDA algorithm can provide ambiguity resolution in a few times.

For a fixed system structure, the length of the baseline can be assumed as known. In addition, some GNSS compass applications allow restrictions on possible attitude angles.

As the ambiguity search space consists of the domain of multidimensional ambiguity, the restrictions given in the three-dimensional position domain can only be applied for three main ambiguities. From the doubly differentiated carrier phase model:

$$\phi_{dd} = \frac{(\vec{e}_i^T - \vec{e}_j^T)}{\lambda}\vec{r} + N_{dd,int} \qquad (3.2.19)$$

with the satellite unit vector *i* $\vec{e}_i^T$; the carrier wavelength, λ; and the entire ambiguity of double differentiation, N_{dd}, int, the base vector based on three primary ambiguities (*p* index) can be formulated in matrix notation as:

$$r_p = H_p^{-1}(\vec{\phi}_{dd,p} - \vec{N}_{dd,int,p}) \qquad (3.2.20)$$

Baseline Length Constraint. With the given baseline length, *l*, the restriction for the baseline length is defined by:

$$l^2 = \|\vec{r}\|\vec{r}^T\vec{r} \qquad (3.2.21)$$

Considering an error in the measurements of the carrier phase that produces a variation of the estimated reference length, the inequality:

$$(l - \Delta\mathrm{l})^2 \leq \left(\tilde{\vec{\phi}}_{dd,p} - \vec{N}_{dd,int,p}\right) H_p^{-1,T} H_p^{-1} \left(\tilde{\vec{\phi}}_{dd,p} - \vec{N}_{dd,int,p}\right) \leq (l + \Delta\mathrm{l})^2$$

(3.2.22)

It can be formulated from *(3.2.20)* and *(3.2.21).* This mathematical condition reduces the search space of the three main ambiguities to a spherical shell.

To incorporate the inequality *(3.2.22)* in the sequential processing steps of the LAMBDA method, we apply a Cholesky decomposition of the defined square and positive matrix $H_p^{-1,T} H_p^{-1}$. Like the deviation of the LAMBDA equations, it allows the definition of recursive limits for each ambiguity.

Yaw and Pitch Angle Constraints. The formulation of an equation for attitude angle constraints is based on the orthogonal projection of the normalized base vector. With the unit vector $\vec{e}_{up}$ pointing up, the angle of inclination pitch θ can be calculated as follows:

$$\sin\theta = \frac{\vec{e}_{up}^{T} * \vec{r}}{\|\vec{r}\|} \qquad (3.2.23)$$

With an estimate of the actual pitch angle, θ_0, and the known baseline length, *l*, as well as an allowable range, $\Delta\theta$ and Δl, respectively, a condition for orthogonal projection can be formulated:

$$\left|\vec{e}_{up}^{T} * \vec{r}\right| \leq (l + \Delta l)\sin(\theta_0 \pm \Delta\theta) \qquad (3.2.24)$$

Inserting *(3.2.20)* with $\rho_p = \left(\tilde{\vec{\phi}}_{dd,p} - \vec{N}_{dd,int,p}\right)$ into (f) results in:

$$|a_1\rho_1 + a_2\rho_2 + a_3\rho_3| \leq |(l \pm \Delta l)\sin(\theta_0 \pm \Delta\theta)| \qquad (3.2.25)$$

$H_{p,l}^{-1}$ is the definition of the *i-th* column of the matrix. Consequently, the coefficients a_1 in *(3.2.25)* hold:

$$a_1 = \vec{e}_{up}^{T} H_{p,i}^{-1} \qquad (3.2.26)$$

Since the signs of the coefficients, a_i, are unknown, only one limit can be defined for the third ambiguity (included in ρ_3). This results in the following conditions:

$$\frac{-|R_1\sin(\theta_0 - \Delta\theta)| - |a_1\rho_1 + a_2\rho_2|}{a_3} \leq \rho_3 \leq \frac{|R_2\sin(\theta_0 + \Delta\theta)| - |a_1\rho_1 + a_2\rho_2|}{a_3}$$

$$if\ a_3 > 0 \qquad (3.2.27)$$

And

$$\begin{aligned} R_1 &= l + \Delta l \ \ if\ \ \theta_0 - \Delta\theta < 0 \\ R_1 &= l - \Delta l \ \ if\ \ \theta_0 - \Delta\theta \geq 0 \\ R_2 &= l - \Delta l \ \ if\ \ \theta_0 + \Delta\theta < 0 \\ R_2 &= l + \Delta l \ \ if\ \ \theta_0 + \Delta\theta \geq 0 \end{aligned} \qquad (3.2.28)$$

To determine the restrictions for the angle of deviation instead of $\vec{e}_{up}$, we use a unit vector in the direction of the body-y:

$$e_y = (\sin\psi_0, \cos\psi_0, 0)^T \qquad (3.2.29)$$

Thus, ψ_0 is the actual yaw angle. The rest of the derivation is like the previous analysis. For the sake of completeness, the inequalities implemented for attitude restrictions are:

$$\frac{-|(l + \Delta l)\sin\Delta\psi| - |a_1\rho_1 + a_2\rho_2|}{a_3} \leq \rho_3 \leq \frac{|(l + \Delta l)\sin\Delta\psi| - |a_1\rho_1 + a_2\rho_2|}{a_3}$$

$$if\ a_3 > 0\ and$$

$$\frac{-|(l+\Delta l)\sin\Delta\psi| - |a_1\rho_1 + a_2\rho_2|}{a_3} \geq \rho_3 \geq \frac{|(l+\Delta l)\sin\Delta\psi| - |a_1\rho_1 + a_2\rho_2|}{a_3}$$

$$if\ a_3 < 0 \qquad (3.2.30)$$

To reject the region of $\psi_0 + \Delta\psi + 180º$, the condition

$$\frac{-|a_1\rho_1 + a_2\rho_2|}{a_3} \leq \rho_3 \qquad if\ a_3 > 0$$

$$\frac{-|a_1\rho_1 + a_2\rho_2|}{a_3} \geq \rho_3 \qquad if\ a_3 < 0$$

$$(3.2.31)$$

must also be fulfilled.

The extension of the LAMBDA method that we have described here forms the algorithmic basis of our improved attitude determination system. It allows a perfect integration of attitude restrictions in the highly efficient LAMBDA algorithm. In this way, not only can the time for the first solution be shortened, but also the reliability is improved, since the reduction of the ambiguity search space excludes possible incorrect solutions.

3.2.4 Comparison and Analysis

In this section, it is intended to compare the different algorithms described in the previous sections, in order to be able to reach a final conclusion of which of them may be a better approximation to the solution of the problem in question, which will be chosen to be implemented later.

To do this, the results of the tests that the authors of the articles and theses performed at the time to evaluate their work will be compared.

- **"MultiKin" results:** (Luo)

MultiKin simulation tests can be classified into three different types: efficiency tests, reliability tests and precision tests. Some field tests will also be presented to show the performance of MultiKin in the case of real GPS scenarios.

It can be found that the conclusions drawn from the simulation tests are validated by the field tests. For example, the improvement rate of ambiguity resolution efficiency decreases with an increasing number of platforms. Performance decreases with an increasing magnitude of differential errors. Since these two field tests were carried out under quite different conditions, such as configuration, reference lengths, error quantities, etc., it can be concluded that simulation tests are valid for evaluating the performance of MultiKin, This also implies that the design of the error models in the GPS software simulator are valid.

Baseline	Horizontal stand-alone positioning accuracy of Car i (RMS)	Residual of $\vec{r}_{ij}^{LL}$ (3D RMS)	Residual of $\vec{R}_{ij}^{LL}$ (3D RMS)	Residual of $\vec{r}_{ij}^{LL} - \vec{R}_{ij}^{LL}$ (3D RMS)
Car1-Car2	1.88 m	6.42 cm	6.29 cm	0.20 cm
Car2-Car3	2.25 m	5.67 cm	5.55 cm	0.2 1 cm
Car3-Car4	2.14 m	4.96 cm	4.87 cm	0.14 cm
Car4-Car5	2.16 m	6.59 cm	6.44 cm	0.30 cm
Car1-Car5	1.88 m	4.73 cm	4.69 cm	0.12 cm
Car2-Car5	2.25 m	6.58 cm	6.45 cm	0.29 cm
Car3-Car5	2.14 m	6.39 cm	6.30 cm	0.17 cm

Table 2: "MultiKin" Relative positioning accuracy in Field Test 2. (Luo)

- **"Multivariate Constraint LAMBDA = MC – LAMBDA" results** (Giorgi)

The author tested the MC-LAMBDA method on the data collected during a static experiment. The focus was on investigating the ability to resolve ambiguities of integers on all available baselines, without outside help, in the shortest possible time, and work on a single frequency. Therefore, he processed the data collected on the GPS L1, on a time-by-time basis, where information about previous times was not considered. A priori information on the attitude was not provided, and no mask angles, elevation-dependent models or dynamic models were applied.

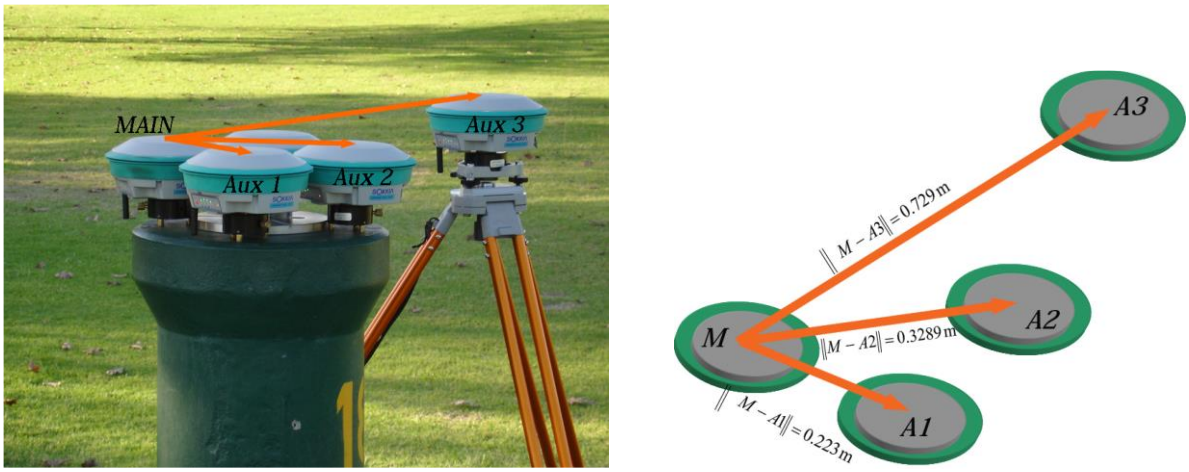

Figure 11: Picture and scheme of the four antennas placement during the static. (Giorgi)

The MC-LAMBDA method achieved surprising improvements. Already considering the single baseline processing, the success rate improved up to 100% in some of the data sets. As for the two baseline configurations, the multivariate restricted method marked success rates above 99% in most datasets, especially in a case of four satellites, where the standard method failed in almost all periods. In all the data sets examined, the incorporation of a priori geometric information in the ambiguity resolution process greatly increased the strength of the underlying model, which benefited the ability to correct the correct set of entire ambiguities. From Table 3 it is clear how the inclusion

of more baselines positively affected the success rate of the restricted solution. The difference between the MC LAMBDA method applied to the single baseline case and to the two-baseline case is quite large for some of the data sets. It is noteworthy that, apart from a data set, the MC-LAMBDA applied to two baseline configurations always guaranteed a success rate of over 90%.

Single baselines				**Two-baselines**			
Single-frequency, single-epoch unaided success rate [%]				Single-frequency, single-epoch unaided success rate [%]			
Baseline	Satellites	LAMBDA	MC-LAMBDA	**Baselines configuration**	Satellites	LAMBDA	MC-LAMBDA
M-A1	4 (a)	0.46	64.87	**M-A1-A2**	4 (a)	0.38	90.38
	4 (b)	0.69	87.19		4 (b)	0.1	97.94
	5 (a)	3.59	91.85		5 (a)	3.5	100
	5 (b)	3.2	100		5 (b)	2.97	100
	6	31.78	100		6	31.58	100
	7	60.53	100		7	65.88	100
M-A2	4 (a)	0.69	53.5	**M-A1-A3**	4 (a)	0.46	99.9
	4 (b)	0.53	97.33		4 (b)	0.53	98.69
	5 (a)	7.32	99.39		5 (a)	3.04	99.92
	5 (b)	14.56	99.92		5 (b)	2.97	100
	6	40.75	100		6	31.13	100
	7	60.88	100		7	64.69	100
M-A3	4 (a)	0.38	47.37	**M-A3-A2**	4 (a)	0.1	94.78
	4 (b)	0.61	66.11		4 (b)	0.46	87.68
	5 (a)	4.34	83.63		5 (a)	4.26	100
	5 (b)	11.27	94.67		5 (b)	11.49	99.77
	6	33.82	99.31		6	33.18	100
	7	70.26	100		7	74.8	100

Table 3: Static field test results. (Giorgi)

- **“Lambda with constraints” results** (Roth et al.)

Several measurement campaigns were carried out on the roof of the Systems Optimization Institute, Karlsruhe.

Time to First Fix. In the first static test, a baseline length of 20 centimetres was used and the construction was aligned approximately horizontally. During a 15-minute measurement campaign, inertial data and GNSS raw data were recorded. During this initial period, the GNSS compass did not rotate. They use offline processing to investigate the time of the first repair (TTFF). To obtain statistically sustainable results, a new filter was started with each GNSS period (one hertz) and the corresponding TTFF was recorded.

Constraint tolerance	Mean TTFF [s]	Number Fixes	Number wrong fixes
-	228.97	700	423 (39,6%)
Δl = 5cm	48.13	900	3 (0.3%)
Δl = 5cm Δθ = 15°	1.37	900	0 (0%)
Δl = 5cm Δθ = 15° Δψ = 30°	1.0155	900	0 (0%)

Table 4: TTFF statistics for static test. (Roth et al.)

- **Comparison.**

In order to compare the last two algorithms, both use the basis of **LAMBDA**, one with constraints (Roth et al.) and the other with multivariate constraint (Giorgi). In the analysis of the **MC–LAMBDA** It is possible to appreciate that both the LAMBDA method and the MC-LAMBDA method were applied in the tests, and it can be observed how the second one generates much better results.

Therefore, the LAMBDA with constraints algorithm presents simpler and not so approximate results, so it is discarded for implementation.

Since it is difficult to compare the results between **MultiKin** (Luo) and **MC-LAMBDA** (Giorgi), since both present completely different tests, and different calculation methodology, making a small review of both, it can be seen that in the case of several configured baselines, the results Ambiguity setting are better for the MC-LAMBDA algorithm, so this work will be implemented in order to improve the results of the moving base obtained through the RTKLIB open source library.

4. Hardware and Software

4.1 Hardware

For the field data collection, in navigation, a multi-sensor navigation box developed by the NAVKA team will be used, which is used for automotive outdoor/indoor navigation of special vehicles.

The box is composed of two Novatel L1/ L2 GPS sensors, in addition to an ublox M8T L1/E1/G1/G1/G1 GNSS receiver (which characteristics are described below). The box also has a MEMS sensor, inclinometer and camera (they will not be described as in this master thesis only the GNSS part is considered).

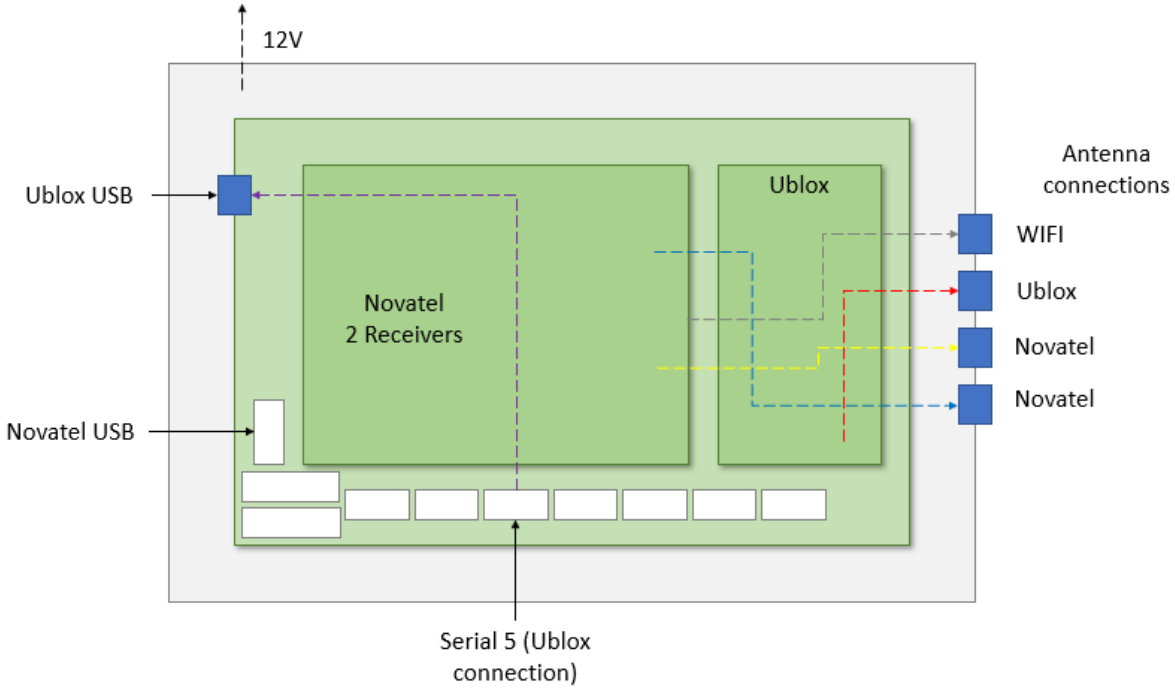

Figure 12: GNSS/MEMS/camera navigation box GNSS Sensors and connections. (Own source)

In the previous figure, you can see a diagram of the assembly of the GNSS sensors that will be used in the development of the project. Each of them has an output for connection with a GNSS antenna for signal reception (two for each of the Novatel, yellow and blue respectively; and an output for the Ublox, the red one). To capture the signal, the box has two USB outputs, one for both Novatel and one for the Ublox. The mains connection must be 12v and 0.5 amps. Next, the characteristics of both sensors will be detailed. For more details on the sensor configuration see Appendix A: Novatel and Ublox configuration.

4.1.1 **Novatel model OEM617D** (Novatel, *OEM617D GPS*)

The dual frequency OEM617D is part of NovAtel's powerful family of OEM6® receivers that offers precise heading and positioning for limited space applications. Compatible with previous versions of NovAtel's popular OEM615 ™ form factor, the OEM617D provides the most efficient way to quickly bring GNSS-capable navigation and positioning products to market. As with all receivers, Novatel OEM6, the OEM617D is ready for GPS, GLONASS and BeiDou signals.

Features:

- Increased satellite availability with BeiDou, GLONASS and Galileo* tracking.
- GLIDE smoothing algorithm.
- RT-2®, ALIGN and RAIM firmware options.

Benefits:

- Dual-frequency RTK with precise ALIGN heading+pitch/roll.
- Dual-frequency GPS+GLONASS BeiDou RTK and ALIGN heading solution.
- Easy to integrate.
- Compact size and low power.

The dual antenna and double frequency input allow the OEM617D to harness the power of NovAtel CORRECT with the RTK and ALIGN functionality. This makes the OEM617D ideal for land, marine or aircraft systems, as it provides heading and position data for multiple industry leading GNSS constellations in static and dynamic environments.

Figure 13: Novatel OEM617D. (Novatel, *OEM617D GPS*)

4.1.2 Ublox M8T L1/E1/G1/B1 (ublox)

The M8T concurrent GNSS modules offer high integrity and precise synchronization in various applications. The modules have support for the Beidou, GLONASS, Galileo and GPS constellations, which allows obtaining service with a global coverage. The improved sensitivity and signal reception of the different constellations, with joint solution, allow obtaining results in areas of difficult access for the GNSS signal.

- Product features:

		Model	
		NEO-M8T	LEA-M8T
Type	GPS / QZSS	x	x
	GLONASS	x	x
	Galileo	R	R
	Beidou	x	x
	Timing	x	x
	Dead Reckoning		
	Precise Point Positioning		
	Raw Data	x	x
Supply	1.65 V – 3.6 V		
	2.7 V – 3.6 V	x	x
	Lowest power (DC/DC)	x	x
Interfaces	UART	x	x
	USB	x	x
	SPI	x	x
	DDC (I2 C compliant)	x	x
Features	Programmable (Flash)	x	x
	Data logging	x	x
	Additional SAW	x	x
	Additional LNA	x	
	RTC crystal	x	x
	Internal oscillator	T	T
	Active antenna / LNA supply	x*	x
	Active antenna / LNA control	x*	x
	Antenna short circuit detection / protection pin		x
	Antenna open circuit detection pin		x*
	Frequency output		
Grade	Standard		
	Professional		
	Automotive		

Table 5: Ublox M8T characteristics. (ublox)

x = Optional, not activated per default or requires external components.*

T = TCXO R = Galileo ready

4.2 Software

For the treatment of the navigation data taken with the sensors mentioned above, it is necessary to have a software that processes them, from which a navigation solution is obtained. In this case, one of the objectives of this master thesis is to improve the algorithm implemented in the RTKLIB library/software for navigation. In addition, before performing the algorithm improvement, the results of several tests between the RTKLIB software and the different software belonging to each of the sensor supplier companies, which are Novatel Connect (for Novatel sensors) and U-CENTER (for the Ublox), will be compared. Next, the characteristics of the software will be detailed.

4.2.1 RTKLIB

According to (Takasu), RTKLIB is:

> "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 features of RTKLIB are: (Takasu)

1. It supports standard and precise positioning algorithms with:
    - GPS, GLONASS, Galileo, QZSS, Beidou and SBAS.
2. It supports various positioning modes with GNSS for both real-time- and post-processing:
    - Single, DGPS/DGNSS, Kinematic, Static, Moving-Baseline, Fixed, PPP-Kinematic, PPP-Static and PPP-Fixed.
3. It supports many standard formats and protocols for GNSS:
    - RINEX 2.10, 2.11, 2.12 OBS/NAV/GNAV/HNAV/LNAV/QNAV, RINEX 3.00, 3.01, 3.02 OBS/NAV, RINEX 3.02 CLK, RTCM ver.2.3, RTCM ver.3.1 (with amendment 1-5), RTCM ver.3.2, BINEX, NTRIP 1.0, NMEA 0183, SP3-c, ANTEX 1.4, IONEX 1.0, NGS PCV and EMS 2.0.
4. It supports several GNSS receiver's proprietary messages:
    - Novatel: OEM4/V/6, OEM3, OEMStar, Superstar II, Hemisphere: Eclipse, Crescent, u-blox: LEA-4T/5T/6T, SkyTraq: S1315F, JAVAD GRIL/GREIS, Furuno GW-10-II/III and NVS NV08C BINR.
5. It supports external communication via:
    - Serial, TCP/IP, NTRIP, local log file (record and playback) and FTP/HTTP (automatic download).
6. It provides many library functions and APIs (application program interfaces):
    - Satellite and navigation system functions, matrix and vector functions, time and string functions, coordinates transformation, input and output functions, debug trace functions, platform dependent functions, positioning models, atmosphere models, antenna models, earth tides models, geoid models, datum transformation, RINEX functions, ephemeris and clock functions, precise ephemeris and clock functions, receiver raw data functions, RTCM functions, solution functions, Google

Earth KML converter, SBAS functions, options functions, stream data input and output functions, integer ambiguity resolution, standard positioning, precise positioning, post-processing positioning, stream server functions, RTK server functions, downloader functions.

7. It includes the following GUI (graphical user interface) and CUI (command-line user interface) APs.

	Function	GUI AP	CUI AP	Notes
(a)	AP Launcher	RTKLAUNCH (3.1)	-	
(b)	Real-Time Positioning	RTKNAVI (3.2, 3.3, 3.5)	RTKRCV (3.11, A.1)	
(c)	Communication Server	STRSVR, (3.3)	STR2STR (3.11, A.5)	
(d)	Post-Processing Analysis	RTKPOST (3.4, 3.5)	RNX2RTKP (3.11, A.2)	
(e)	RINEX Converter	RTKCONV (3.6)	CONVBIN (3.11, A.4)	
(f)	Plot Solutions and Observation Data	RTKPLOT (3.7, 3.8)	-	
(g)	Downloader for GNSS Products and Data	RTKGET (3.9)	-	
(h)	NTRIP Browser	SRCTBLBROWS (3.10)	-	

Table 6: RTKLIB APs. (Takasu)

8. All the executable binary APs for Windows are included in the package as well as whole source programs of the library and the APs.

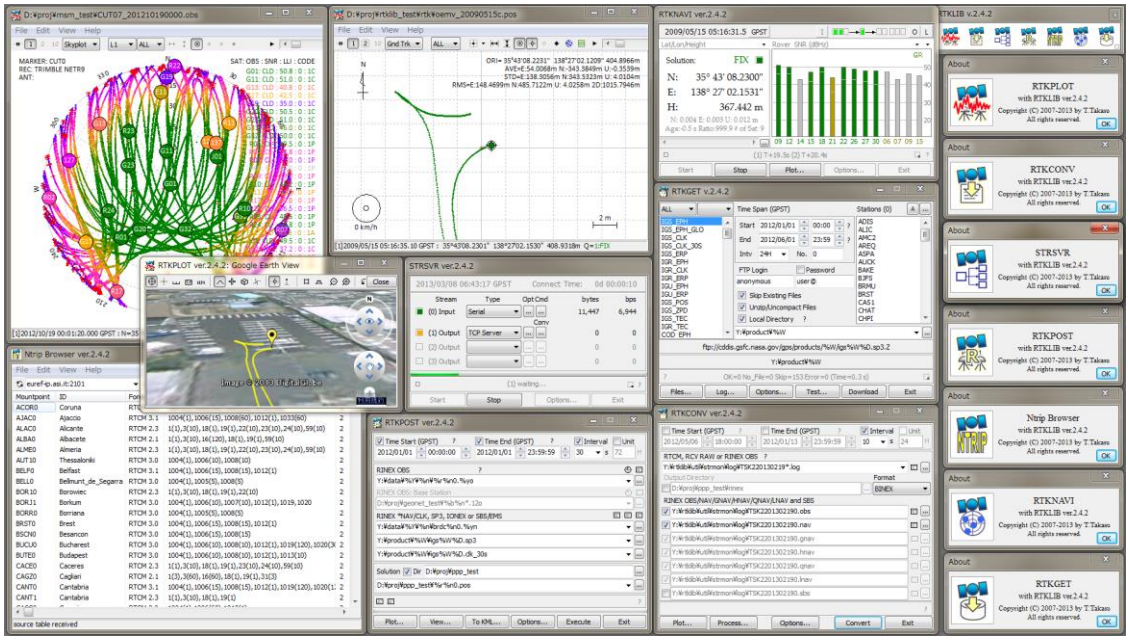

Figure 14: RTKLIB GUI APs on Windows 7. (Takasu)

RTKLIB employs EKF (Extended Kalman Filter) in order to obtain the final solutions in DGPS/DGNSS, Static, Kinematic and Moving-baseline modes in conjunction with the GNSS signal measurement models and the troposphere and ionosphere models.

The moving baseline mode is usually used if both rover and the base station receivers are moving and the only relative position of the rover with respect to the base station is required. The moving-base mode can be used to determine the precise attitude by mounting two antennas to a moving platform. In RTKLIB, the moving-base mode is applied if the processing option "Positioning Mode" is set to "Moving-Base". (Takasu)

4.2.2 **Novatel Connect** (*NovAtel Connect* | *Canal Geomatics*)

It is an easy-to-use Windows software that allows users of Novatel sensors to configure and control their devices. It has an easy-to-use GUI (Graphical User Interface) that allows users to capture data with their sensors through the USB port, serial port or Ethernet connection.

Novatel Connect allows its users a way to access their sensors without the need for complex software or a terminal emulator.

Features:

- Easy to use the GUI.
- Access your Novatel sensors via USB port, serial port or Ethernet connection.
- Connection Import Tool to import existing CDU connection settings.
- Compatible with Windows XP, Windows 7, Windows 8 and Windows 10.
- Great customer service.

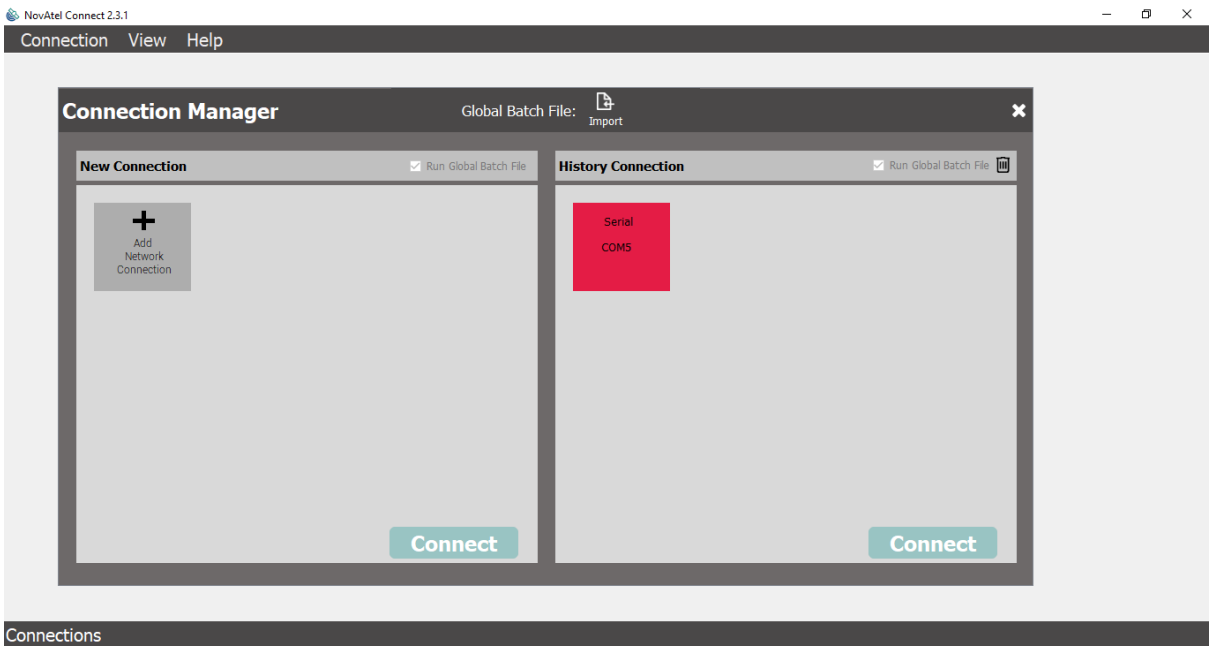

Figure 15: GUI Novatel Connect. (Own source)

4.2.3 **U-CENTER** (U-blox)

U-CENTER is the powerful GNSS evaluation and visualization tool for u-blox sensors that can be downloaded for free. It allows end users to evaluate and test U-blox GNSS positioning chips and modules for navigation and positioning performance.

The purpose of U-CENTER is to enable users to:

- Conduct performance tests on u-blox and other GNSS devices.
- Configure u-blox GNSS positioning chips and modules.
- Update the firmware on GNSS modules.
- Test the added performance provided by u-blox's Assist Now service.

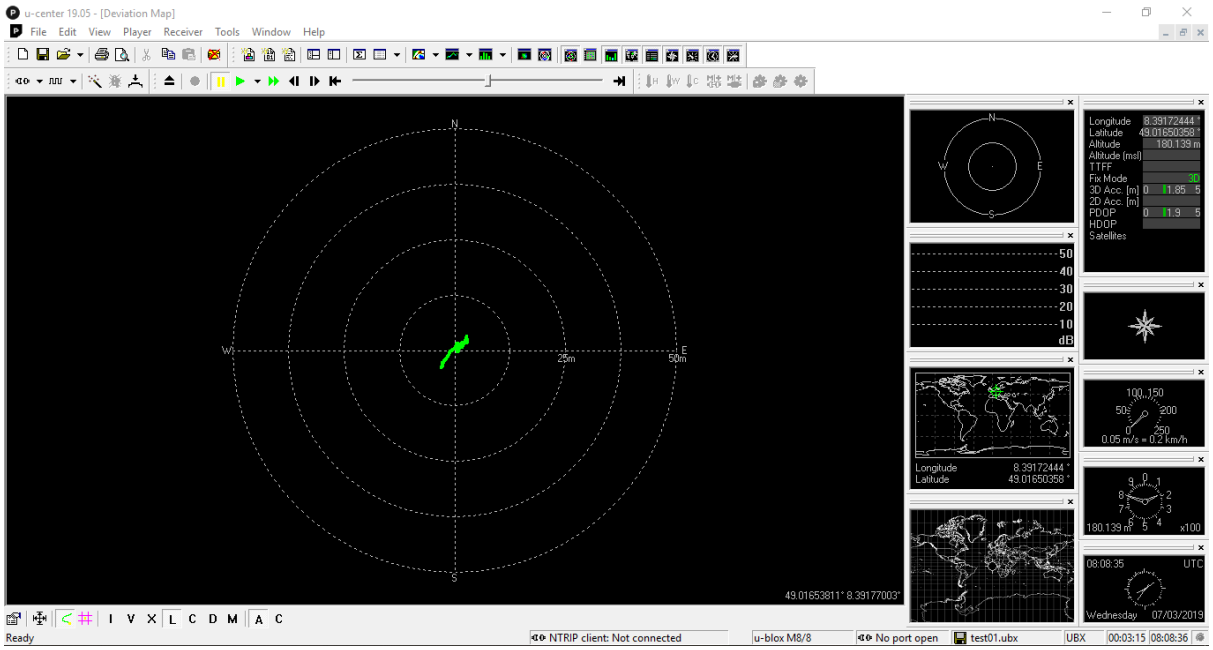

Figure 16: GUI U-CENTER. (Own source)

5. Integer ambiguity resolution theory

This section is an abstract from the article by (Delft).

In this chapter, a brief review of the entire ambiguity resolution theory will be given. The three integer estimators will all be described in the order of complexity, as well as the "optimum" in terms of the probability of correct fixation (success rate). This means that the rounding estimator is first described, followed by whole bootstrapping and integer least squares (ILS). The two ILS search strategies implemented will also be described. Finally, ILS will be presented with the Ratio Test.

5.1 Parameter estimation

GNSS ambiguity resolution is the process of solving unknown cycle ambiguities of the DD carrier phase data as integers. The GNSS models on which ambiguity resolution is based can be projected into the following conceptual framework of linearized observation equations:

$$y = Aa + Bb + \epsilon \qquad (5.1.1)$$

where $a \in \mathbb{Z}^n$ is the vector of integer parameters with whole ambiguities DD. $b \in \mathbb{R}^q$ is the vector of real value parameters, including baseline components and possibly tropospheric and ionospheric refraction parameters, etc. The coefficient matrices are $A \in \mathbb{R}^{n*m}$ and B $B \in \mathbb{R}^{n*q}$, with [*A B*] full column range. The observation vector and $y \in \mathbb{R}^m$ contain the least calculated observed pseudorange and those observable in the carrier phase, which are contaminated by the random noise vector ϵ. In general, it is assumed that ϵ is normally distributed with zero mean and variance-covariance matrix Q_{yy}.

In general, a four-step procedure is used to solve model based on the least squares' criterion.

- **Step 1: Float solution**

In the first step, the entire ownership of the ambiguities a is not considered and the so-called floating LS estimates are calculated along with their variance-covariance matrix

$$\begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} Q_{\hat{a}\hat{a}} & Q_{\hat{a}\hat{b}} \\ Q_{\hat{b}\hat{a}} & Q_{\hat{b}\hat{b}} \end{bmatrix} \qquad (5.1.2)$$

- **Step 2: Integer estimation**

In the second step, the floating ambiguity estimate $\hat{a}$ is used to calculate the corresponding integer ambiguity estimate, denoted as

$$\check{a} = S(\hat{a}) \qquad (5.1.3)$$

with $S: \mathbb{R}^n \mapsto \mathbb{Z}^n$, the integer mapping from the n-dimensional space of the reals to the n-dimensional space of the integers. In this step, there are different possible mapping

function options S, which correspond to the different integer estimation methods. Popular options are the integer least squares (ILS), integer bootstrapping (IB) and integer rounding (IR). ILS is optimal, since it can be shown that it has the highest success rate of all integer estimators. IR and IB, however, can also perform quite well, particularly after the MC-LAMBDA decorrelation has been applied. Its advantage over ILS is that an integer search is not required. Each of the methods will be discussed in more detail in the following subsections.

- **Step 3: Acceptance test**

The third step is optional. It consists in deciding whether to accept the entire solution once the entire estimates of the ambiguities have been calculated.

- **Step 4: Fixed solution**

In the fourth step, the floating solutions of the remaining real value parameters resolved in the first step are updated using the fixed integer parameters

$$\check{b} = \hat{b} - Q_{\hat{b}\hat{a}} Q_{\hat{a}\hat{a}}^{-1} (\hat{a} - \check{a}) \qquad (5.1.4)$$

If it can be assumed that the entire ambiguity solution is deterministic (if the success rate is very close to 1), the corresponding variance-covariance matrix of the fixed reference solution is obtained as:

$$Q_{\check{b}\check{b}} = Q_{\hat{b}\hat{b}} - Q_{\hat{b}\hat{a}} Q_{\hat{a}\hat{a}}^{-1} Q_{\hat{a}\hat{b}} \qquad (5.1.5)$$

5.2 Decorrelation technique

In theory, one can perform the mapping *(5.1.3)* in the original DD ambiguities. However, due to the high correlation between the elements of the ambiguity vector, as well as the poor accuracy of these elements, the entire solution obtained from the original DD ambiguities may not be reliable (in case of whole rounding or bootstrapping) or the calculation is time consuming. By reparametrizing the ambiguities, the accuracy of the elements of the ambiguity vector can be improved, while at the same time the correlation between ambiguities is greatly reduced. This reparameterization is known as the Z-transformation and transforms the original DD ambiguities into a new set of ambiguities such as

$$\hat{z} = Z^T \hat{a} \qquad (5.2.1)$$

The corresponding v-c matrices are transformed accordingly

$$Q_{\hat{z}\hat{z}} = Z^T Q_{\hat{b}\hat{b}} Z \quad and \quad Q_{\hat{b}\hat{z}} = Q_{\hat{b}\hat{a}} Z \qquad (5.2.2)$$

After the transformation, the floating solutions become

$$\begin{bmatrix} \hat{z} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} Q_{\hat{z}\hat{z}} & Q_{\hat{z}\hat{b}} \\ Q_{\hat{b}\hat{z}} & Q_{\hat{b}\hat{b}} \end{bmatrix} \qquad (5.2.3)$$

Now, the mapping function *(5.1.3)* corresponding to the entire method of choice is used to map $\hat{z}$ to its integers $\check{z}$. The inverse transformation provides the entire solution in terms of the original DD ambiguities

$$\hat{a} = Z^{-T}\hat{z} \qquad (5.2.4)$$

Note that the inverse transformation is not necessary for the calculation of the corresponding fixed baseline solution, since this solution can be obtained directly with:

$$\check{b} = \hat{b} - Q_{\hat{b}\hat{z}}Q_{\hat{z}\hat{z}}^{-1}(\hat{z} - \check{z}) \qquad (5.2.5)$$

5.3 Integer rounding

The simplest way to obtain an integer vector of the real value floating solution is to round each of the $\hat{z}$ entries to your nearest integer. The corresponding integer estimator reads

$$\hat{z}_R = ([\hat{z}_1], \dots, [\hat{z}_n])^T \qquad (5.3.1)$$

Where [.] means round to the nearest integer.

Note that if rounding is applied directly to the original ambiguities $\hat{a}$, the result may be different from $Z^{-T}\hat{z}$, with the last result in a greater probability of correct fixing.

5.4 Integer bootstrapping

The bootstrapped estimator still uses the rounding of integers but considers part of the correlation between ambiguities. The bootstrapped estimator is derived from a sequential adjustment of least squares and is calculated as follows. If there are *n* ambiguities available, we begin with the most precise ambiguity, here it is assumed to be the last ambiguity $\hat{z}_n$, and round its value to the nearest integer. The remaining flotation ambiguities are corrected by virtue of their correlation with the last ambiguity. Then, the estimate of real ambiguity but corrected, but now corrected, is rounded to its nearest whole number and all remaining ambiguities (*n - 2*) are corrected again, but now by virtue of their correlation with this ambiguity. This process continues until all ambiguities are considered. The bootstrapped $\check{z}_B$estimator components are given as

$$\begin{aligned}
\check{z}_{n,B} &= [\hat{z}_n] \\
\check{z}_{n-1,B} &= [\hat{z}_{n-1|n}] = [\hat{z}_{n-1} - \sigma_{\hat{z}_{n-1}\hat{z}_n}\sigma_{\hat{z}_n}^{-2}(\hat{z}_n - \hat{z}_{n,B})] \\
&\vdots \qquad (5.4.1) \\
\check{z}_{1,B} &= [\hat{z}_{1|N}] = \left[\hat{z}_1 - \sum_{i=2}^{m} \sigma_{\hat{z}_1\hat{z}_{i|I}}\,\sigma_{\hat{z}_{i|I}}^{-2}(\hat{z}_{i|I} - \hat{z}_{i,B})\right]
\end{aligned}$$

where the shorthand notation $\hat{z}_{i|I}$ represents the i-th ambiguity obtained through a conditioning in the previous *I = {i + 1,. . . , n}* sequentially rounded ambiguities. One must begin with the most precise floating ambiguity, which in this case is supposed to

be $\hat{z}_n$. The solution of sequential least squares of real value can be obtained by triangular decomposition of the variance-covariance matrix of the ambiguities: $Q_{\hat{z}\hat{z}} = L^T D L$, where *L* denotes a lower triangular matrix unit with inputs

$$l_{i,j} = \sigma_{\hat{z}_1 \hat{z}_{i|I}} \sigma^{-2}_{\hat{z}_{i|I}} \qquad (5.4.2)$$

and D a diagonal matrix with conditional variances $\sigma^2_{\hat{z}_{i|I}}$ as its diagonal elements.

Therefore, equation *(5.4.1)* can be expressed as:

$$\check{z}_{j,B} = [\hat{z}_{j|J}] = \left[\hat{z}_j - \sum_{i=j+1}^{n} \sigma_{\hat{z}_j \hat{z}_{i|I}} \sigma^{-2}_{\hat{z}_{i|I}} \left(\hat{z}_{i|I} - \hat{z}_{i,B}\right)\right] = \left[\hat{z}_j - \sum_{i=j+1}^{n} l_{i,j} \left(\hat{z}_{i|I} - \hat{z}_{i,B}\right)\right]$$

(5.4.3)

with $l_{i,j}$ of equation *(5.4.2).* Note that if one would like to start with $\hat{z}_1$ (being the most precise ambiguity), one should work with the decomposition $Q_{\hat{z}\hat{z}} = L D L^T$ instead.

The success rate of integer boot can be evaluated exactly

$$P_{s,B} = P(\check{z}_B = z) = \prod_{i=1}^{n} (2\Phi\left(\frac{1}{2\hat{z}_{i|I}}\right) - 1) \qquad (5.4.4)$$

with ***z*** the real ambiguity vector and $\Phi(\mathrm{x})$ the cumulative normal distribution:

$$\Phi(\mathrm{x}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left\{-\frac{1}{2} t^2\right\} dt$$

It was already mentioned that the starting procedure should start with the rounding of the most precise floating ambiguity. In addition, from equation *(5.4.1)* starting will generally result in different results if applied to reparametrized ambiguities. It is known that bootstrapping works almost optimally if it is applied to ambiguities related to decoration $\hat{z} = Z^T \hat{a}$. This is because sequential conditional variations are greatly reduced by decorrelation.

The Z-transformation that relates to decoration is implemented in such a way that the n-th ambiguity is the most accurate. In addition, the decorrelation algorithm is implemented in such a way that after each decorrelation step a rearrangement is performed, which guarantees that

$$\sigma_{\hat{z}_{i|I}} = \sigma_{\hat{z}_{j|I}} \qquad for\ j < i \qquad (5.4.5)$$

It should be noted that for real decoration, this property is not a prerequisite. However, it is important in the case of bootstrapping, since it means that the i-th transformed ambiguity has the smallest conditional variance possible, where the conditioning is in the previous *I = {i + 1,. . . , n}* transformed ambiguities.

5.5 Integer least squares (ILS)

By solving the GNSS model of equation *(5.1.1)* in a sense of least squares, but now with the additional restriction that the ambiguity parameters must have an integer value, the whole estimator of the second step in the procedure becomes:

$$\breve{a} = \min(\hat{a} - z)^T Q_{\hat{a}\hat{a}}^{-1}(\hat{a} - z) \qquad (5.5.1)$$

It is known that this estimator is optimal, which means that the probability of a correct integer estimate is maximized.

This ILS procedure is efficiently machined in the LAMBDA method. Keep in mind that the success rate. The ILS estimate is independent of the parameterization of floating ambiguities. The Z transformation that relates to decoration is only required to greatly reduce the search time, so that the LAMBDA method is highly efficient. The integer minimizer of equation *(5.5.1)* is obtained through a search on the points of the entire grid of an n-dimensional hyper-ellipsoid defined by the variance-covariance matrix $Q_{\hat{z}\hat{z}}$ and with the centre $\hat{z}$:

$$F(z) = (\hat{z} - z)^T Q_{\hat{z}\hat{z}}^{-1}(\hat{z} - z) \leq \chi^2, \ \ with \ z \in \mathbb{Z}^n \qquad (5.5.2)$$

The positive χ^2 determines the size of the search ellipsoid. The integer point of the ***z*** grid within the hyper-ellipsoid that gives the minimum value of the function ***F(z)*** is the optimal solution of ILS $\breve{z}$.

Equation *(5.5.2)* can be rewritten using *LDL* decomposition:

$$(\hat{z} - z)^T L^{-1} D^{-1} L^{-T} (\hat{z} - z) \leq \chi^2 \qquad (5.5.3)$$

Remember that the diagonal elements of D are the conditional variations of the transformed floating ambiguities $\hat{z}_i$. The following notation will be used: $d_i = \sigma_{\hat{z}_{i|I}}$. Defining $\tilde{z} = z - L^{-T}(\hat{z} - z)$:

$$L^T(\hat{z} - z) = \hat{z} - z \qquad (5.5.4)$$

Inserting equation *(5.5.3)* in *(5.5.2),* the hyper-ellipsoid becomes:

$$(\hat{z} - z)^T D^{-1}(\hat{z} - z) \leq \chi^2 \qquad (5.5.5)$$

Or equivalently:

$$\frac{(\tilde{z}_1 - z_1)^2}{d_1} + \cdots + \frac{(\tilde{z}_i - z_i)^2}{d_i} + \cdots + \frac{(\tilde{z}_n - z_n)^2}{d_n} \leq \chi^2 \qquad (5.5.6)$$

Obviously, for any z that satisfies the linked equation *(5.5.5)*, the following individual limits are also met:

$$\tilde{z}_n - \sigma_{\tilde{z}_n}\chi \leq z_n \leq \tilde{z}_n - \sigma_{\tilde{z}_n}\chi$$

The two search techniques described earlier in MCLAMBDA (Giorgi) are used to solve it.

5.6 ILS with Ratio Test

The ratio test is in fact a discrimination test: it proves the proximity of the floating solution to the optimal whole solution compared to other whole candidates. The test is defined as:

$$Accept\ \check{a}\ if: \frac{F(\check{a})}{F(\check{a}')} \leq \mu \qquad (5.6.1)$$

where μ is the threshold value, for which it contains 0 ≤ μ ≤ 1 (since the optimal solution has the smallest squared norm), and $\check{a}'$ is the integer vector that returns the second smallest quadratic form *F(z).*

The principle of the Ratio Test is illustrated in Figure 17: the acceptance regions are the bright green and red regions centred at all points of the entire grid. The value of μ determines the size of the acceptance regions. Four cases can be distinguished: correct acceptance (success), incorrect acceptance (failure), unnecessary rejection (false alarm) and correct rejection. Obviously, a higher value for μ will result in a larger acceptance region and greater chances of failure and success.

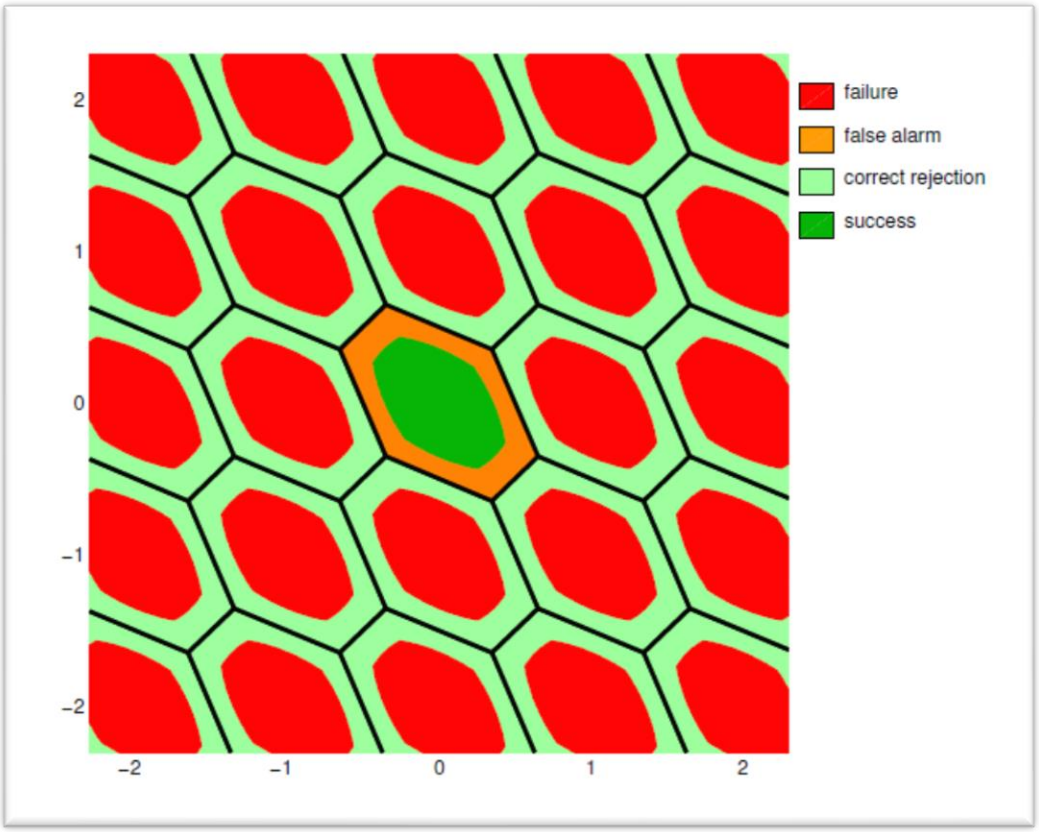

Figure 17: Acceptance regions with Ratio Test (bright green and red). (Delft)

6. Implementation of MC-LAMBDA routine

The next section will explain the most important aspects of the implementation of the MC-LAMBDA routine, related with the ambiguity resolution.

All the code generated and developed for its implementation has been made in C++, in order to facilitate a better implementation in the RTKLIB library, which is developed in C/C++.

For the implementation of the calculation methodology, several functions have been developed (which will be explained below) that perform the calculation of the processes mentioned in the previous section, in addition to a main function that includes all the procedures and depending on the Input configuration returns some results or others.

In addition, the following section can guide the use of the routine as a separate RTKLIKB module, or if you only want to use those functions that the user wants.

Finally, for the implementation of the routine, article (Delft) has served as a support guide for the development of the algorithm.

6.1 The main MC-LAMBDA routine

```
// ----------------------------------------------------------------------
//                                mclambda
// ----------------------------------------------------------------------
//
// Arguments    : int n                          -> Number of float ambiguities
//                double ahat                    -> Vector of float ambiguities
//                const double Qahat             -> Variance/covariance matrix
//                                                  of ambiguities
//                double method                  -> Calculation methodology
//                double param                   -> Activate optional parameters
//                const emxArray_char_T *type    -> Type of optional calculation
//                double value                   -> Value for optional calculation
//                emxArray_real_T *afixed        -> Output afixed
//                emxArray_real_T *sqnorm        -> Output sqnorm
//                double *Ps                     -> Output Ps
//                double Qzhat[]                 -> Output Qzhat
//                double Z[]                     -> Output Z
//                double *nfixed                 -> Output nfixed
//                double *mu                     -> Output mu
//
// Return       : void
//
// ----------------------------------------------------------------------
void mclambda(int n, double ahat[], double Qahat[], double method, double
              param, const emxArray_char_T *type, double value, emxArray_real_T
              *afixed, emxArray_real_T *sqnorm, double *Ps, double Qzhat[],
              double Z[], double *nfixed, double *mu)
{
```

Figure 18: MC-LAMBDA function. (Own source)

- **Description:**

This is the main routine of the MC-LAMBDA algorithm. The ILS method will be used for integer estimation based on the provided **float ambiguity vector** *ahat* and associated **variance-covariance matrix** *Qahat*. However, the user may also select other methods: integer rounding, bootstrapping and, furthermore, there is the option to apply the Ratio Test to decide on acceptance of the fixed solution.

- **Inputs:**
  - n: Number of float ambiguities
  - ahat: Float ambiguities
  - Qahat: Variance/covariance matrix of ambiguities
  - method:
    - 1: ILS method based on search-and-shrink
    - 2: ILS method based on enumeration in search
    - 3: Integer rounding method
    - 4: Integer bootstrapping method
    - 5: ILS method with Ratio Test (uses search-and shrink)
  - param: Activate or deactivate the optional input arguments.
    - 0: Deactivated
    - 1: Activated
  - type: [Optional input argument] Define some specific arguments
    - *'ncands'*, value: Number of requested integer candidate vectors (only used with ILS, DEFAULT = 2)
    - *'P0'*, value: With method 5 (ILS + Ratio test): Fixed failure rate (available options: 0.01 or 0.001) [DEFAULT=0.001]
    - *'MU'*, value: Fixed threshold value for Ratio Test (value must be between 0 and 1)
  - value: [Optional input argument] Value for the previous methods.

- **Outputs**
  - afixed: Array of size (n x ncands) with the estimated integer candidates, sorted according to the corresponding squared norms, best candidate first. For integer rounding and bootstrapping: ncands = 1
  - sqnorm: Distance between integer candidate and float ambiguity vectors in the metric of the variance-covariance matrix Qahat. Only available for ILS.
  - Ps: Bootstrapped success rate.
    - If ILS is used, Ps is its lower bound.
    - If rounding is used, Ps is its upper bound.
    - If bootstrapping is used, Ps is the exact success rate.
  - Qzhat: Variance-covariance matrix of decorrelated ambiguities
  - Z: Transformation matrix with (*n x n*) dimension.
  - nfixed: Number of fixed ambiguities
    - with methods 1 to 4: will always be equal to n.

- with method 5 (ILS + Ratio test): will be equal to n if fixed solution is accepted, and 0 otherwise.

- mu: Threshold value used for Ratio Test.

6.2 Routines used by MC-LAMBDA

6.2.1 *chistart.cpp*

```
// --------------------------------------------------------------------------
//                              chistart
// --------------------------------------------------------------------------
//
// Arguments    : int n             -> Number of float ambiguities
//                const double D    -> Matrix D of LtDL-decomposition
//                const double L    -> Matrix L of LtDL-decomposition
//                const double ahat -> Vector of float ambiguities
//                double ncands     -> Requested number of candidates
//
// Return       : double Chi2       -> Size of the search ellipsoid
//
// --------------------------------------------------------------------------
double chistart(int n, const double D[], const double L[], const double ahat[],
                double ncands)
{
```

Figure 19: CHISTART function. (Own source)

- **Description:**

These routine computes or approximates the initial size of the search ellipsoid. If the requested number of candidates is not more than the dimension + 1, this is done by computing the squared distances of partially conditionally rounded float vectors to the float vector in the metric of the covariance matrix. Otherwise an approximation is used.

- **Inputs:**
  - n: Number of float ambiguities
  - L, D: LtDL-decomposition of the variance-covariance matrix of the float ambiguities (preferably decorrelated)
  - ahat: float ambiguities (preferably decorrelated)
  - ncands: Requested number of candidates
  - factor: Multiplication factor for the volume of the resulting search ellipsoid

- **Outputs**:
  - Chi2: Size of the search ellipsoid

6.2.2 *decorrel.cpp*

```
// -----------------------------------------------------------------------
//                              decorrel
// -----------------------------------------------------------------------
//
// Arguments    : int n              -> Number of float ambiguities
//                const double Qahat -> Variance-covariance matrix of ambiguitie
//                const double ahat  -> Vector of float ambiguities
//                double Qzhat       -> Output Qzhat
//                double Z           -> Output Z
//                double L           -> Output L
//                double D           -> Output D
//                double zhat        -> Output zhat
//                double iZt         -> Output iZt
//
// Return       : void
//
// -----------------------------------------------------------------------
void decorrel(int n, const double Qahat[], const double ahat[], double Qzhat[],
              double Z[], double L[], double D[], double zhat[],
              double iZt[])
{
```

Figure 20: DECORREL function. (Own source)

- **Description:**

This routine creates a decorrelated Q-matrix, by finding the Z-matrix and performing the corresponding transformation.

- **Inputs:**
  - n: Number of float ambiguities
  - Qahat: Variance-covariance matrix of ambiguities (original)
  - ahat: Original ambiguities (optional)

- **Outputs**:
  - Qzhat: Variance-covariance matrix of decorrelated ambiguities
  - Z: Z-transformation matrix
  - L: L matrix (from LtDL-decomposition of Qzhat)
  - D: D matrix (from LtDL-decomposition of Qzhat)
  - zhat: Transformed ambiguities (optional)
  - iZt: inv(Z')-transformation matrix

6.2.3 *lsearch.cpp*

```
// ------------------------------------------------------------------------
//                                lsearch
// ------------------------------------------------------------------------
//
// Arguments    : int n                   -> Number of float ambiguities
//                const double ahat       -> Vector of float ambiguities
//                const double L          -> Matrix D of LDL-decomposition
//                const double D          -> Matrix L of LDL-decomposition
//                double ncands           -> Requested number of candidates
//                emxArray_real_T *afixed -> Output afixed
//                emxArray_real_T *sqnorm -> Output sqnorm
//
// Return       : void
//
// ------------------------------------------------------------------------
void lsearch(int n1, const double ahat[], const double L[], const double D[],
             double ncands, emxArray_real_T *afixed, emxArray_real_T *sqnorm)
{
```

Figure 21: LSEARCH function. (Own source)

- **Description:**

This routine finds the integer vector, which is closest to a given float vector, in a least squares sense. This is the search-step in integer ambiguity resolution. It is best to perform this search only on ambiguities which have been decorrelated using MC-LAMBDA.

- **Inputs:**
  - n: Number of float ambiguities
  - ahat: Float ambiguities (should be decorrelated for computational efficiency)
  - L, D: LtDL-decomposition of the variance-covariance matrix of the float ambiguities ahat
  - ncands: Number of requested candidates

- **Outputs**:
  - afixed: Estimated integers
  - sqnorm: Corresponding squared norms

6.2.4 *ssearch.cpp*

```
// ------------------------------------------------------------------------
//                                ssearch
// ------------------------------------------------------------------------
//
// Arguments    : int n                   -> Number of float ambiguities
//                const double ahat       -> Vector of float ambiguities
//                const double L          -> Matrix D of LDL-decomposition
//                const double D          -> Matrix L of LDL-decomposition
//                double ncands           -> Requested number of candidates
//                emxArray_real_T *afixed -> Output afixed
//                emxArray_real_T *sqnorm -> Output sqnorm
//
// Return       : void
//
// ------------------------------------------------------------------------
void ssearch(int n1, const double ahat[], const double L[], const double D[],
             double ncands, emxArray_real_T *afixed, emxArray_real_T *sqnorm)
{
```

Figure 22: SSEARCH function. (Own source)

- **Description:**

Integer ambiguity vector search by employing the search-and-shrink technique.

- **Inputs:**
  - n: Number of float ambiguities
  - ahat: Float ambiguities (should be decorrelated for computational efficiency)
  - L, D: LtDL-decomposition of the variance-covariance matrix of the float ambiguities ahat
  - ncands: Number of requested candidates

- **Outputs**:
  - afixed: Estimated integers
  - sqnorm: Corresponding squared norms

6.3 Implementation in RTKLIB

Once the calculation functions of the MC-LAMBDA algorithm were implemented, the next step was the implementation of the calculation routine in the RTKLIB library.

To do this, two parts that must be modified for implementation will be differentiated. On the one hand, the integration of a series of options in the graphical program interface (GUI), which is developed in C++, and on the other hand the implementation of the calculation routine in the internal RTKLIB calculation code, which It is developed in C.

For the present work, and since the main idea is to improve the calculation of the moving-based trajectory, the only modified RTKLIB module is RTKPOST, which, as mentioned above, allows GNSS postprocessing tasks. Although it has not been carried out in this work, the implementation of the routine in the real-time module RTKNAVI, would be carried out in the same way.

6.3.1 *rtklib.h changes*

Before starting with the changes and modifications in the RTKLIB library, the first thing that must be carried out is the modification of the configuration file and definitions of global variables and functions "*rtklib.h*".

In the case that concerns us, several variables of the original library file have been added and modified.

```
#define PMODE_SINGLE 0                  /* positioning mode: single */
#define PMODE_DGPS   1                  /* positioning mode: DGPS/DGNSS */
#define PMODE_KINEMA 2                  /* positioning mode: kinematic */
#define PMODE_STATIC 3                  /* positioning mode: static */
#define PMODE_MOVEB  4                  /* positioning mode: moving-base */
#define PMODE_MOVEB_MCLAMBDA 5          /* positioning mode: moving-base mclambda */
#define PMODE_FIXED  6                  /* positioning mode: fixed */
#define PMODE_PPP_KINEMA 7              /* positioning mode: PPP-kinemaric */
#define PMODE_PPP_STATIC 8              /* positioning mode: PPP-static */
#define PMODE_PPP_FIXED 9               /* positioning mode: PPP-fixed */
```

Figure 23: Postprocessing modes. (Own source)

In the previous figure you can see the different postprocessing methods. A new postprocessing method (in red) has been added that will refer to the developed algorithm and will appear as a new option in the options menu.

Since the algorithm mainly has the ambiguity resolution as its main calculation, a new global variable that refers to the new method must be added. In the following figure (in red) this modification can be seen.

```
#define ARMODE_OFF  0                   /* AR mode: off */
#define ARMODE_CONT 1                   /* AR mode: continuous */
#define ARMODE_INST 2                   /* AR mode: instantaneous */
#define ARMODE_FIXHOLD 3                /* AR mode: fix and hold */
#define ARMODE_WLNL 4                   /* AR mode: wide lane/narrow lane */
#define ARMODE_TCAR 7                   /* AR mode: triple carrier ar */
#define ARMODE_MCLAMBDA 5               /* AR mode: MCLAMBDA Resolution ar */
```

Figure 24: Ambiguity resolution modes. (Own source)

Since the algorithm has been developed in order to be a little more precise than the calculation performed by the RTKLIB library itself, several options have been added that will be part of the input of the MC-LAMBDA routine calculation function. These options will appear so that the user can choose between each of them. The following figure shows the global variables added for each one.

```
#define MCLAMBDA_ILS_SAS 0              /* ILS search-and-shrink ("ncands" = 10) */
#define MCLAMBDA_ILS_ES 1               /* ILS enum in search ("ncands" = 10) */
#define MCLAMBDA_IRM 2                  /* Integer rounding method */
#define MCLAMBDA_IB 3                   /* Integer bootstrapping method */
#define MCLAMBDA_ILS_PO 4               /* ILS Ratio Test ("PO" = 0.01) */
#define MCLAMBDA_ILS_MU 5               /* ILS Ratio Test ("MU" = 0.5) */
```

Figure 25: MCLAMBDA options. (Own source)

Since the selection of one of the above options must be stored to be part of the function's input variables, a new variable has been added that stores this information within the structure defined in the file as *prcopt_t.* In the following figure (in red) you can see this change.

```
typedef struct {        /* processing options type */
    int mode;           /* positioning mode (PMODE_???) */
    int soltype;        /* solution type (0:forward,1:backward,2:combined) */
    int nf;             /* number of frequencies (1:L1,2:L1+L2,3:L1+L2+L5) */
    int navsys;         /* navigation system */
    double elmin;       /* elevation mask angle (rad) */
    snrmask_t snrmask;  /* SNR mask */
    int sateph;         /* satellite ephemeris/clock (EPHOPT_???) */
    int modear;         /* AR mode (0:off,1:continuous,2:instantaneous,3:fix and hold,4:ppp-ar) */
    int mcopt;          /* Options for MCLAMBDA ambiguity resolution */
    int glomodear;      /* GLONASS AR mode (0:off,1:on,2:auto cal,3:ext cal) */
```

Figure 26: prcopt_t struct definition. (Own source)

In addition, since the function of calculating the MC-LAMBDA routine (see 6.1 The main MC-LAMBDA routine) must be executed in the part of the necessary code, a new function has been added and defined to prepare the input data of the routine and call it and execute it. In the following figure you can see (in red) this definition.

```
/* integer ambiguity resolution ----------------------------------------------*/
EXPORT int lambda(int n, int m, const double *a, const double *Q, double *F,
                  double *s);
EXPORT int mclambda_exec(rtk_t *rtk, int n, int m, const double *a, const double *Q, double *F,
                  double *s);
EXPORT int lambda_reduction(int n, const double *Q, double *Z);
EXPORT int lambda_search(int n, int m, const double *a, const double *Q,
                         double *F, double *s);
```

Figure 27: Ambiguity resolution definition functions. (Own source)

The use of the function will be explained in later sections.

6.3.2 *RTKPOST GUI changes*

As mentioned in the previous section, there are three options added to RTKPOST that will be reflected in the GUI, on the one hand the new processing mode (Figure 28), and on the other hand the ambiguity resolution along with its calculation options (Figure 29).

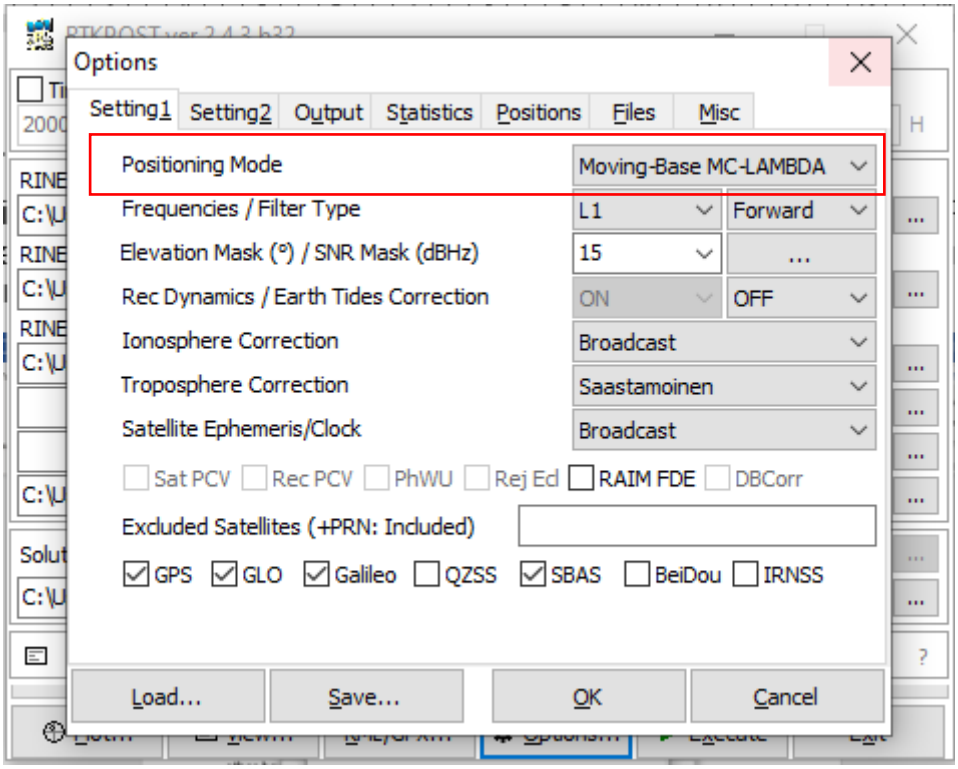

Figure 28: Settings 1 RTKPOST options. (Own source)

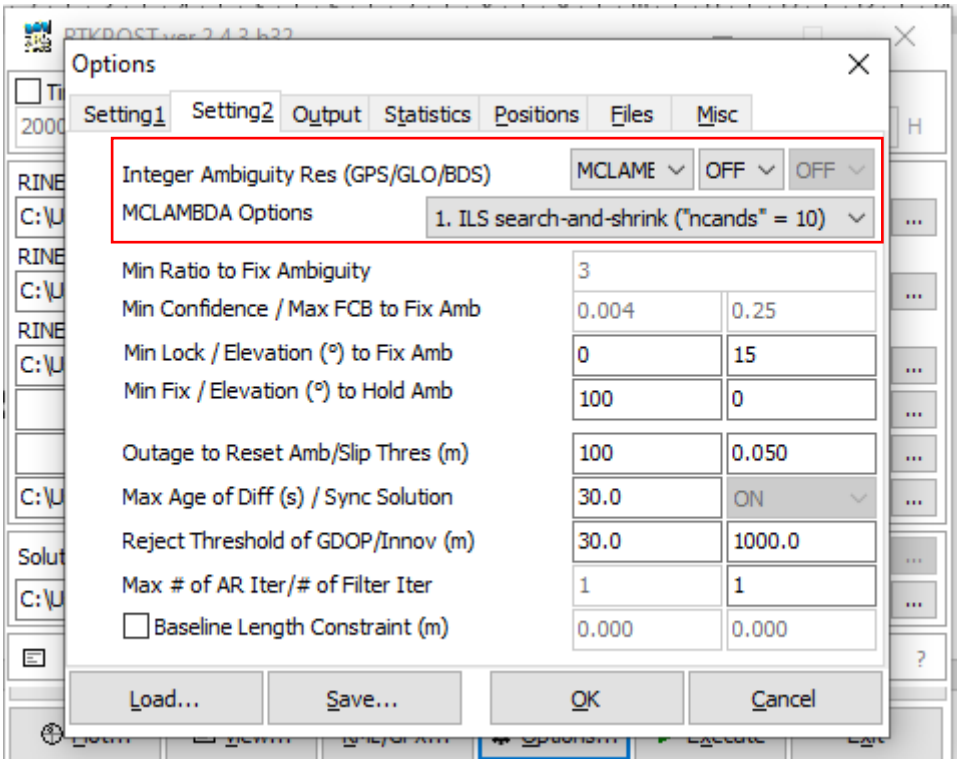

Figure 29: Settings 2 RTKPOST options. (Own source)

For the implementation of these options, following the RTKLIB manual (Takasu), the compilation of the program must be carried out by means of a specific IDE, which allows compiling together with the visual library used by the RTKLIB developer. This is the Embarcadero C++ Builder software (*C++Builder - Embarcadero Website*), which allows you to compile with VLC and which has been used to add the new options in the visual part as well as the new drop-down that allows you to select the calculation options of the routine.

Once the visual part of RTKPOST was modified, the next step was to make small modifications to the postopt.cpp and postmain.cpp files following the same configuration as the moving base mode already implemented.

6.3.3 *Changes in RTKLIB source code*

Following RTKLIB's own structure implemented for Moving-Base mode, the option and calculation of ambiguities have been implemented based on the function described in the section 6.1 The main MC-LAMBDA routine.

Previously, before proceeding with the explanation of the implementation of the new routine, since the RTKLIB code is developed in C and the tool has been developed in C++, the development of an API that allows the compatibility of both languages, that is, allow C to read all the features of C++. To do this, all the structures implemented in the new routine were modified in order to facilitate implementation.

The most relevant changes in the implementation in RTKLIB were made in the files *"rtkpost.c"* and *"lambda.c".*

- ***"rtkpost.c"* changes:**

In order to call the new ambiguity resolution function, you must first prepare the input data of the same, as well as incorporating it in the correct place of calculation. For this, within this C file, a new function has been created that prepares all the input data and makes a call to the new function created (Figure 27) for the execution of the algorithm routine. This function will be called in the calculation procedure.

```
// ------------------------------------------------------------------------
// resolve integer ambiguity by MC-LAMBDA
// ------------------------------------------------------------------------
static int resamb_MCLAMBDA(rtk_t *rtk, double *bias, double *xa)
```

Figure 30: Ambiguity resolution definition function. (Own source)

```
// ------------------------------------------------------------
// resolve integer ambiguity by MC-LAMBDA
// code added after
// ------------------------------------------------------------
else if (stat!=SOLQ_NONE&&resamb_MCLAMBDA(rtk,bias,xa)>1) {
    if (zdres(0,obs,nu,rs,dts,var,svh,nav,xa,opt,0,y,e,azel)) {
        /* post-fit reisiduals for fixed solution */
        nv=ddres(rtk,nav,dt,xa,NULL,sat,y,e,azel,iu,ir,ns,v,NULL,R,vflg);
        /* validation of fixed solution */
        if (valpos(rtk,v,R,vflg,nv,4.0)) {
            stat=SOLQ_FIX;
        }
    }
}
// ------------------------------------------------------------
// End of resolve integer ambiguity by MC-LAMBDA
// code added after
// ------------------------------------------------------------
```

Figure 31: Call ambiguity resolution function. (Own source)

The new function that will send all the input data of the MC-LAMBDA calculation routine is called within the *resamb_MCLAMBDA* function.

```
/* MCLAMBDA resolution */
if (!(info=mclambda_exec(rtk,nb,2,y+na,Qb,b,s))) {
```

Figure 32: Call of mclambda_exec. (Own source)

The result of this call to the new function is the solution of ambiguities through MC-LAMBDA, which will be used for the calculation of the baselines later.

- ***"lambda.c"* changes:**

Following the order of defining RTKLIB functions, it was decided to add the new function (Figure 27) to the file where the other ambiguity resolution methods are located. This function, as mentioned above, receives as input the necessary data for the execution of the MC-LAMBDA routine.

```
/* MC-LAMBDA RUN FUNCTION ----------------------------------------------------
* args   : rtk_t  rtk    I  Calculation options
*          int    n      I  number of float parameters
*          int    m      I  number of fixed solutions
*          double *a     I  float parameters (n x 1)
*          double *Q     I  covariance matrix of float parameters (n x n)
*          double *F     O  fixed solutions (n x m)
*          double *s     O  sum of squared residulas of fixed solutions (1 x m)
* return : status (0:ok,other:error)
* notes  : matrix stored by column-major order (fortran convension)
*-----------------------------------------------------------------------------*/
// ----------------------------------------------------------------------------
//                          MC-LAMBDA RUN FUNCTION
// ----------------------------------------------------------------------------
extern int mclambda_exec(rtk_t *rtk, int n, int m, const double *a, const double *Q, double *F,
                         double *s)
```

Figure 33: mclambda_exec function. (Own source)

Within the function, the input data is prepared, and the function call of the MC-LAMBDA routine is called (see 6.1 The main MC-LAMBDA routine).

```
// Execute MCLAMBDA Routine
mclambda(n, ahat, Qahat, method, param, type, value, afixed, sqnorm, &Ps, Qzhat, Z1, &nfixed, &mu);
```

Figure 34: Call mclambda routine. (Own source)

Since the function returns the fixed solutions of the ambiguities and the sum of squared residuals of fixed solutions these will be the output variables of the function necessary for the subsequent calculation of the baselines (see comment in Figure 33).

Following the recommendations of article (Delft), several pre-defined calculation options have been implemented so that the user can choose between them (see Figure 29).

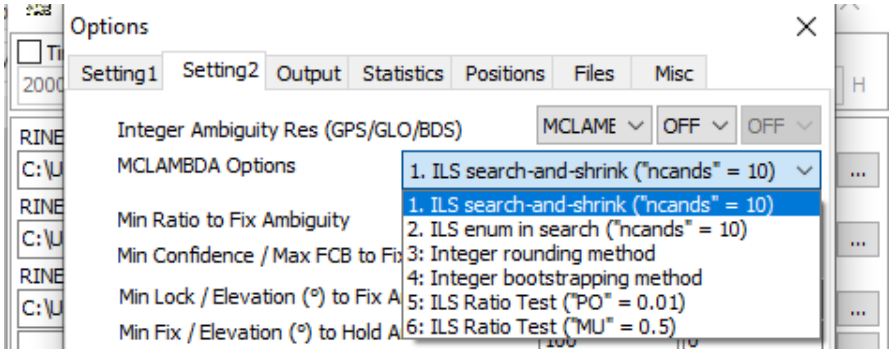

Figure 35: MC-LAMBDA calculation options. (Own source)

7. Moving Base Software

The RTKLIB library has a moving base processing mode which allows you to obtain the baselines and, from that, obtain the coordinates of the rover from the coordinates of the base station. The procedure is correct but to obtain the coordinates of the master station uses single position, which does not offer high precision, and does not allow the change to any other mode. In addition, it always uses a fixed position of the base station, so it does not consider its movement. The result generated is enough if the work area is not very large.

Since one of the objectives of the work is to improve this approach, an external software has been developed that allows the input of the base coordinates processed in any mode (single, DGNSS, PPP etc), processed in RTKLIB, in addition to the baselines (also processed with RTKLIB, using the resolution of ambiguities with MC-LAMBDA) of each of the rover (up to three rover); and from that get the coordinates of each of the rover. The result is a coordinate file which can be loaded into the RTKPLOT software for viewing and analysis.

The development of the tool has been carried out entirely in Python, because it is a simple and faster language for the development of applications, in addition to being able to use several tools during the thesis and to be able to make comparisons. The PyQt4 library has been used for the visual part.

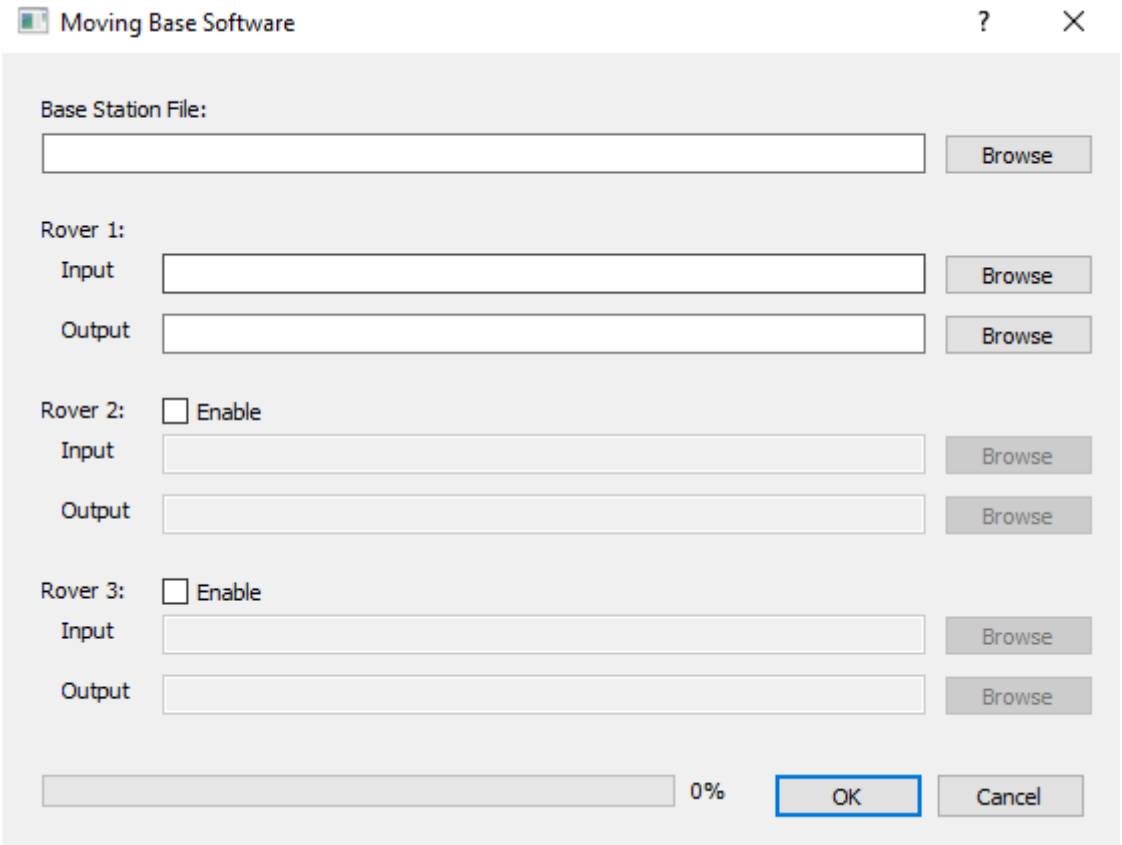

Figure 36: GUI of moving base software. (Own source)

The software allows the user to select the input files, (master and rover) and automatically in the same path of the input files generates the output path with the name of the output file, although it can be modified. In order to perform the calculation of several rover stations (up to three in total), each of the input dialog boxes must be activated.

The internal procedure consists of entering the input data, first they are read and stored in a format readable for Python. Once the data has been read, on the one hand, data of the teacher and, on the other, of the rover, a comparison of times between the rover and the master is made to find the coincidence between both, to finally obtain, of the coordinates from the master and the baseline of the rover at the same time. Once the ECEF coordinates of the master receiver and the baseline of the time are obtained, in ENU, a rotation of the same is done, using the equations *(1.3.5)* to *(1.3.7).* This rotation will allow to obtain the change of the coordinates of the baseline of the n -frame the e-frame, in order to have all the data in the same frame. For this, a programmed function is executed which receives the coordinates of the master receiver and the coordinates of the baseline, both in the form of a vector.

```
def enu2earth(coord_base, baseline):
    import numpy as np
    # Transform ECEF to Geo
    ellip = ellipsoid('WGS84')
    geo = xyzToGeo(coord_base[0], coord_base[1], coord_base[2], ellip)
    lat = geo[0]
    lon = geo[1]
    # Build Rne Rotation Matrix
    c1 = [-(np.sin(lat) * np.cos(lon)),-(np.sin(lon)), -(np.cos(lat) * np.cos(lon))]
    c2 = [ -(np.sin(lat) * np.sin(lon)), np.cos(lon), -(np.cos(lat) * np.sin(lon))]
    c3 = [np.cos(lat), 0, -np.sin(lat)]
    C = [c1, c2, c3] # Matrix Rne
    # Coordinate transformation
    V_ned = baseline # Coordinate vector NED
    Xearth = np.dot(C, V_ned) # Tranformation
    return Xearth # Vector IncX, IncY, IncZ ECEF
```

Figure 37: Transformation function between n-frame to e-frame. (Own source)

As can be seen in the previous figure, it is necessary to mount the rotation matrix so that the coordinates of the base station, the master receiver, are transformed to geographical coordinates.

To do this, a function is executed that receives the coordinates in ECEF and returns a vector with the geographic coordinates. In addition, the desired ellipsoid parameters must be sent in the function for projection. In this case, another function has been programmed that returns these parameters indicating the name of the work ellipsoid, which in this case, the WGS84 ellipsoid has been used. In addition, the radius of curvature of the first vertical must be calculated.

```python
def xyzToGeo(X, Y, Z, ellip):  # Transforms Cartesian geocentric to geographic coordinates
    import numpy as np
    # Process
    a1 = ellip[0]
    b1 = ellip[2]
    p = np.sqrt(X**2 + Y**2)
    az = np.arctan((Z*a1)/(p*b1))
    e2 = ellip[3]**2
    ep2 = ellip[4]**2
    lat = np.arctan2((Z + ep2*b1*(np.sin(az)**3)),(p - e2*a1*(np.cos(az)**3)))
    lon = np.arctan2(Y,X)
    nu1 = nu(lat, ellip)
    hel = (p/np.cos(lat)) - nu1
    geo = [lat, lon, hel]
    return geo
```

Figure 38: Transformation function between ECEF to Geographic. (Own source)

```python
def ellipsoid(ellip): # Returns the parameters of the ellipsoid
    import numpy as np
    # Vector with ellipsoid parameters
    v_elip = []
    # Ellipsoids
    if ellip == 'HAYFORD':
        v_elip.append(6378388.000)
        v_elip.append(1/297.000)
    elif ellip == 'WGS84':
        v_elip.append(6378137.000)
        v_elip.append(1/298.257223563)
    elif ellip == 'GRS80':
        v_elip.append(6378137.000)
        v_elip.append(1/298.257222101)
    else:
        return -1
    # Semi-axis and eccentricities
    b = v_elip[0] - v_elip[0]*v_elip[1]
    v_elip.append(b)
    e1 = np.sqrt(1 - (v_elip[2]**2)/(v_elip[0]**2))
    v_elip.append(e1)
    e2 = np.sqrt((v_elip[0]**2)/(v_elip[2]**2) - 1)
    v_elip.append(e2)
    return v_elip
```

Figure 39: Ellipsoid function. (Own source)

```python
def nu(lat,ellip): # Returns the radius of curvature of the first vertical
    import numpy as np
    # Process
    a = ellip[0]
    e2 = ellip[3]**2
    rpv = a/(np.sqrt(1 - e2*(np.sin(lat)**2)))
    return rpv
```

Figure 40: "nu" function. (Own source)

Once the coordinates of the baseline are rotated, the next step is simple, after rotation, the vector obtained is no more than a vector of coordinate increments in ECEF, so the sum of this to the coordinates of the base, will result in rover coordinates.

```
# Search the correct epoch for base station
coord_base = 0
for j in range(len(bcoord_base)):
    # Data of the base station coordinates, epoch
    day_base = date_base[j][0]
    hour_base = date_base[j][1]
    if day_rov == day_base: # Compare day
        if hour_rov == hour_base: # Compare epoch
            coord_base = bcoord_base[j] # Coordinate vector
            break
# Calculate rover position
if coord_base != 0:
    # Transform baseline to ECEF
    Xearth = enu2earth(coord_base, baseline)
    #print coord_base, baseline, Xearth
    pos = [coord_base[0] + Xearth[0], coord_base[1] + Xearth[1], coord_base[2] + Xearth[2]]
    rov_pos.append(pos)
else:
    list_i.append(i)
```

```
  # Search the correct epoch for base station
  coord_base = 0
  for j in range(len(bcoord_base)):
      # Data of the base station coordinates, epoch
      day_base = date_base[j][0]
      hour_base = date_base[j][1]
      if day_rov == day_base: # Compare day
          if hour_rov == hour_base: # Compare epoch
              coord_base = bcoord_base[j] # Coordinate vector
              break
  # Calculate rover position
  if coord_base != 0:
      pos = [coord_base[0] + baseline[0], coord_base[1] + baseline[1], coord_base[2] + baseline[2]]
      rov_pos.append(pos)
  else:
      list_i.append(i)
```

Figure 41: Search epoch and calculate rover coordinates. (Own source)

Once all the coordinates of the rover are obtained and stored, for each one of them, a function is executed that reads the stored data of the new coordinates and writes them in a new file, following the same RTKLIB format, since, as mentioned above, the file can be loaded into RTKPLOT for the analysis of the results.

```
# Rover 1 calculation
file_base = self.base.text()
file_rov1 = self.rov1.text()
file_out1 = self.rov1_2.text()
rov1_dcc = mba(file_rov1, file_base) # Calculate position
out = output(file_out1, rov1_dcc) # Print output
# Rover 2 calculation
if self.rov2On.isChecked():
    file_rov2 = self.rov1.text()
    file_out2 = self.rov1_3.text()
    rov2_dcc = mba(file_rov2, file_base) # Calculate position
    out2 = output(file_out2, rov2_dcc) # Print output
# Rover 3 calculation
if self.rov3On.isChecked():
    file_rov3 = self.rov1.text()
    file_out3 = self.rov1_4text()
    rov3_dcc = mba(file_rov3, file_base) # Calculate position
    out3 = output(file_out3, rov3_dcc) # Print output
```

Figure 42: Calculation per rover. (Own source)

7.1 Novatel reader

As mentioned above, a Novatel sensor has been used for this work, which allows raw data to be obtained so that everything is stored in a single file. In order to make a comparison between the results that the RTKLIB library itself performs with respect to an internal processing of the coordinates that Novatel itself performs, a Python software has been developed that allows obtaining the data and stores it in readable RTKLIB format. For proper operation, the BESTPOSA and HEADINGA logs must be activated. (See Appendix A: Novatel and Ublox configuration)

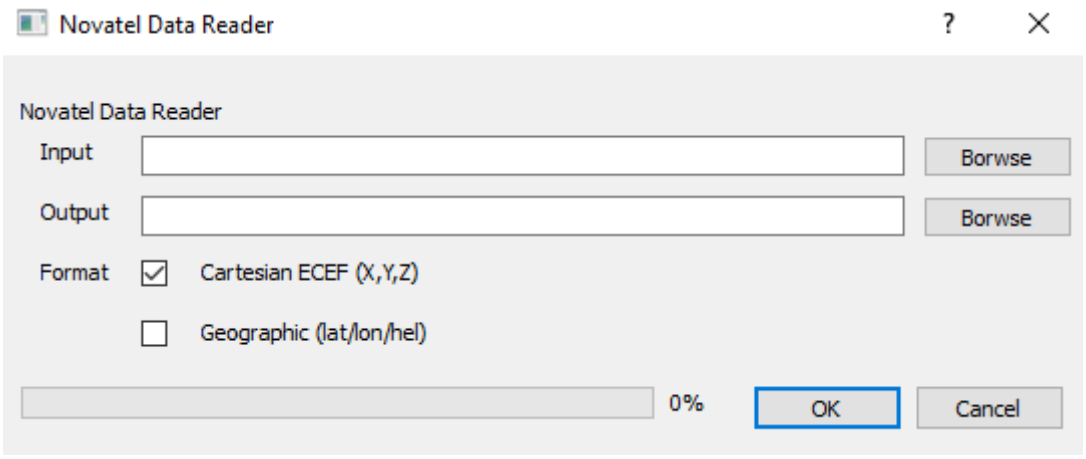

Figure 43: Novatel reader. (Own source)

The software not only allows the reading of the ".gps" files of Novatel, but also offers the user the possibility of displaying the coordinates of the receiver in ECEF or geographical coordinates, selecting the desired option. Since Novatel already directly offers geographic coordinates, a transformation to ECEF is necessary, for which a function like that of Figure 38 has been programmed, which performs the inverse process.

```
def geoToXYZ(lat, lon, hel, ellip): # Transforms geographic to Cartesian geocentric coordinates
    import numpy as np
    # Process
    nu1 = nu(lat, ellip)
    a2 = ellip[0]**2
    b2 = ellip[2]**2
    X = (nu1 + hel) * np.cos(lat) * np.cos(lon)
    Y = (nu1 + hel) * np.cos(lat) * np.sin(lon)
    Z = ((b2/a2) * nu1 + hel) * np.sin(lat)
    cart = [X, Y, Z]
    return cart
```

Figure 44: Transformation function between geographic coordinates to ECEF. (Own source)

Novatel returns the coordinates calculated in a single mode, from which latitude and longitude, altitude above sea level and geoid undulation are obtained. It also offers data on the heading and pitch of each of the periods.

8. Test and results analysis

In the next chapter, we will proceed to explain and detail everything related to the tests carried out in the field, both their configuration and their results.

8.1 Test configuration

All the tests carried out have been carried out on the roof of building B of the Hochschule Karlsruhe. For this, a platform was designed where three antennas fixed to a structure can be mounted, which allows to have fixed baselines between them. The structure had three antennas connected to the GNSS/MEMS/camera navigation box so that two of the antennas record data with the Novatel sensor and the other with the Ublox sensor. All this connected to a laptop which is configured following the Appendix A: Novatel and Ublox configuration

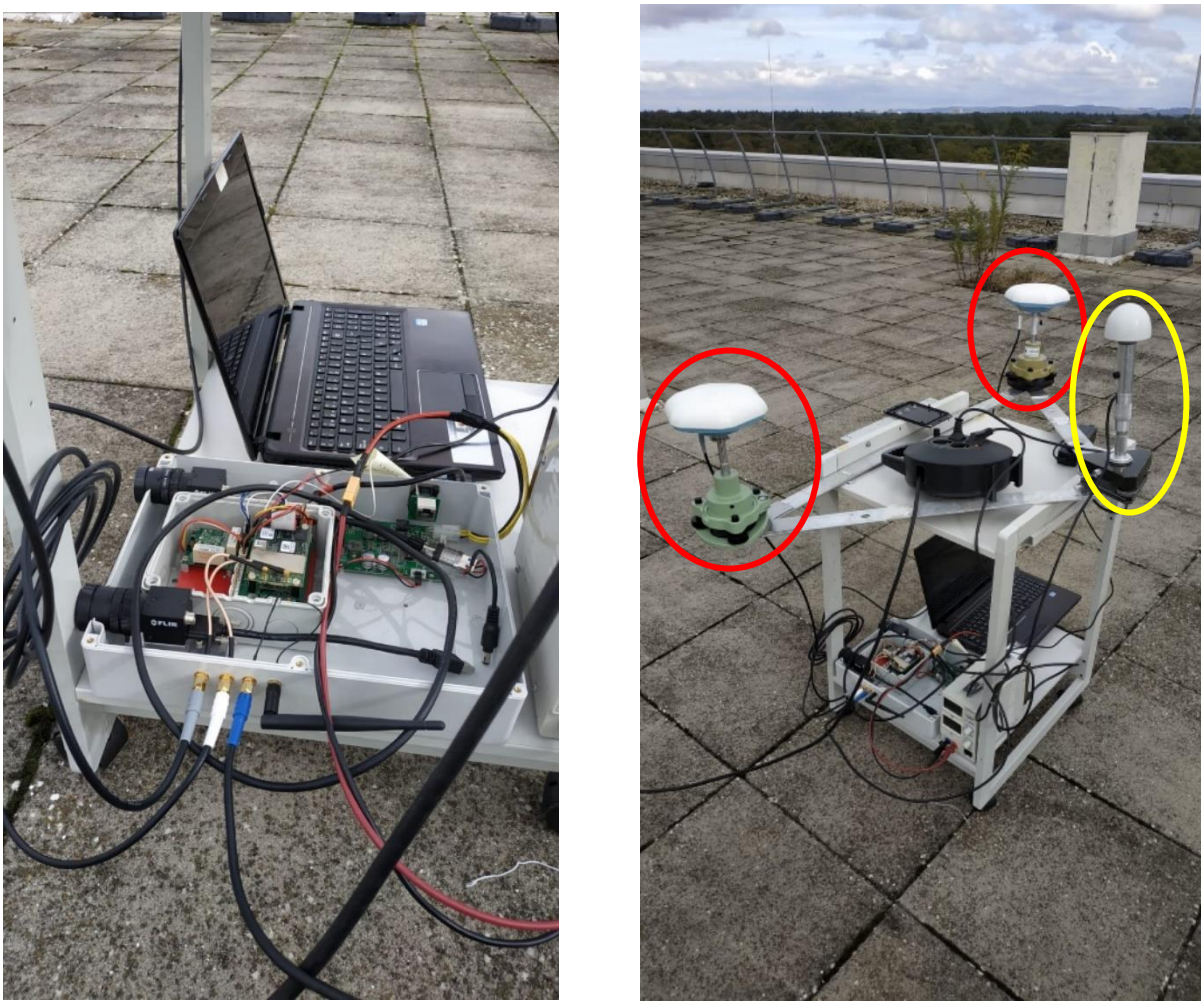

Figure 45: Test platform configuration. (Own source)

In red, the antennas connected to the Novatel sensor, in yellow the antenna connected to the Ublox sensor. From right to left antenna 1 and antenna 2 of Novatel respectively. For the test processing, the Novatel antenna will be used as the master receiver. Two tests were carried out in the field, a static survey of two hours and a kinematic of approximately half an hour with a static start.

8.2 Static test

A static survey was carried out with an approximate duration of two hours in order to better analyse in detail the operation of the sensors over a longer period. Its results are not entirely relevant in the case of Moving Base, since the platform is always static, but its analysis is intended to be interesting to see the operation of the platform as well as the possibility of using sensors in any type of application. Of course, the calculation of the baselines has been performed

8.2.1 Single mode antenna 1

As mentioned in section 7.1 Novatel reader, the Novatel sensor offers a single mode calculation of the coordinates of the antenna 1, so the first of the analysis consists of checking and comparing the results of said processing with those obtained by the same Single mode processing by RTKLIB. For this, a file with the Novatel software and another with the RTKLIB processing have been obtained.

For the analysis of the results, the RTKPLOT program will be used where up to two solution files can be loaded at the same time in order to make the comparison. In this case, the single mode processing of RTKLIB and Novatel will be analysed, as mentioned above.

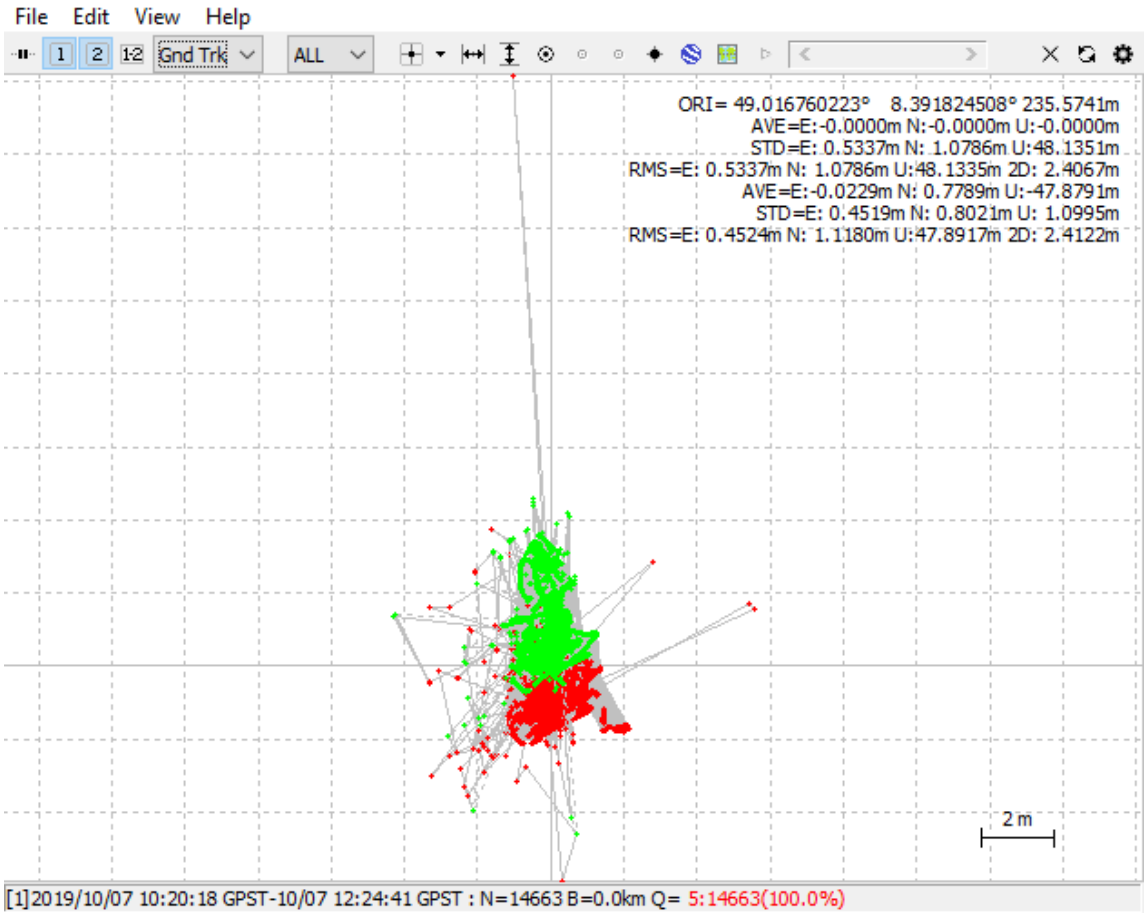

Figure 46: Single processing RTKLIB and Novatel Ground Track. (Own source)

In the previous figure, in red, solution 1 Novatel, in green solution 2 RTKLIB. At first glance you can see that there are quite a few differences in some of the points. Novatel solution presents some that opted outlayer, in addition to a greater dispersion of the points. As for the analysis of the standard deviations, we see how RTKLIB presents less error in E and N, although very similar, but we see that there is a lot of difference in U, so the analysis of each of the components is necessary to see what is occurring.

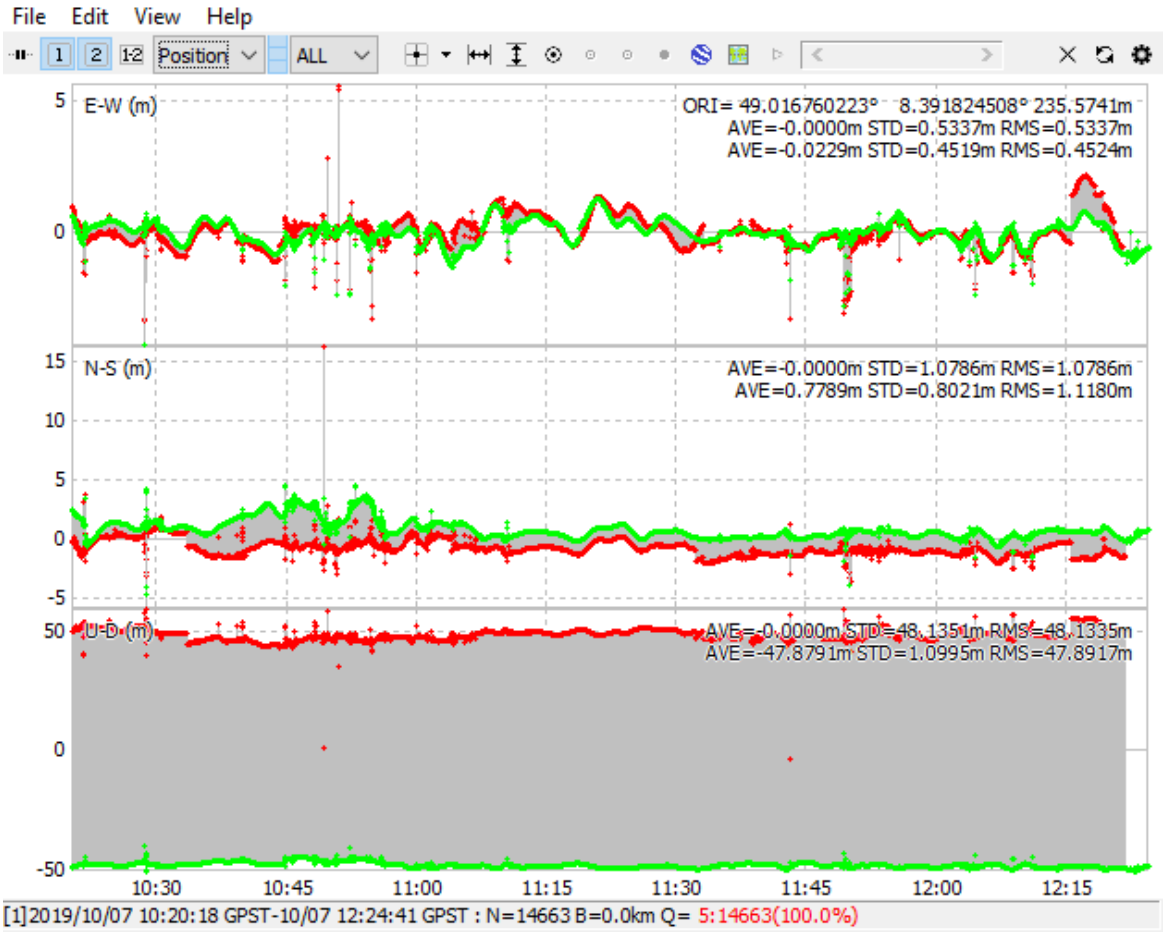

Figure 47: Position visualization Single mode. (Own source).

In the previous figure you can see that there is a huge difference between the U-D address. Analysing the results of the coordinates it has been observed that this occurs due to a large difference in ellipsoidal height, probably caused by a difference in geoid models. As for the rest of the E-W and N-S components, the results vary a bit, and the outlayers occur at the same times for both solutions, with the Novatel solution presenting the greatest error.

It can therefore be concluded that the Novatel solution presents greater errors than the RTKLIB solution. This may be due to RTKLIB having a better processing mode eliminating observables that present large errors, as well as satellites that are not valid.

8.2.2 Static DGNSS and PPPS

As mentioned in previous sections, the processing of the master receiver in a more precise way implies better results when calculating the position of the rovers associated with the same platform. In this work, Novatel antenna 1 has been processed, this being considered as the master receiver, in static DGNSS and PPPS modes. as a detail, indicate that for the DGNSS processing the permanent station of KARL belonging to EUREF (coordinates in ETRF2000 with its corresponding transformation to the current epoch) (Bruyninx et al.) has been used. For PPP the necessary files have been obtained to obtain a better precision.

For the comparison, both solutions will be loaded in RTKPOST, in red, solution 1 static DGNSS and in blue solution 2 PPPS.

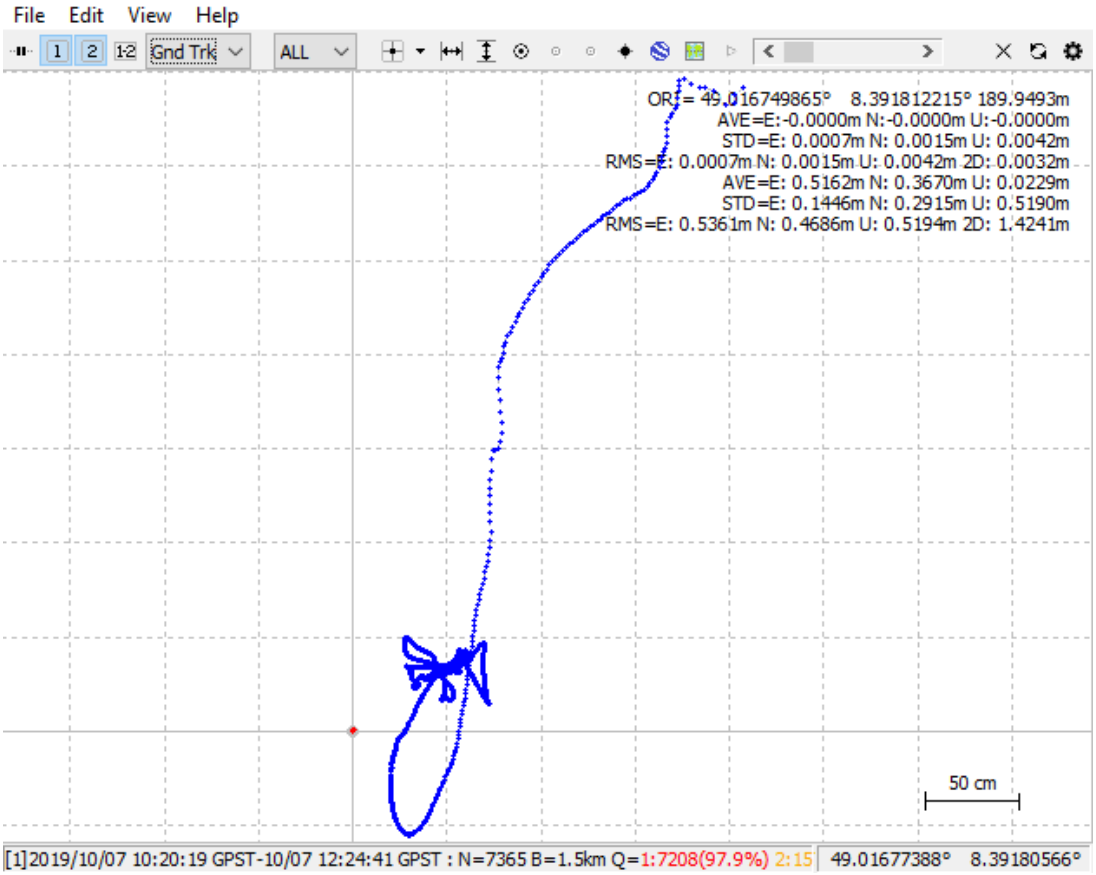

Figure 48: DGNSS and PPPS Ground Track. (Own source)

As can be seen in the previous figure, the PPPS takes a while to converge until a good solution is obtained, but still we see that this solution varies greatly with respect to the fixed point, since it is a static measurement. As for the DGNSS, it can be seen how each one of the points is concentrated in the same area, without presenting any outlayer.

Analysing the position graph in the following figure, you can see the convergence times of the PPP and the time it takes for the DGNSS to obtain fixed solutions, which we can see representing 97.9% of the points.

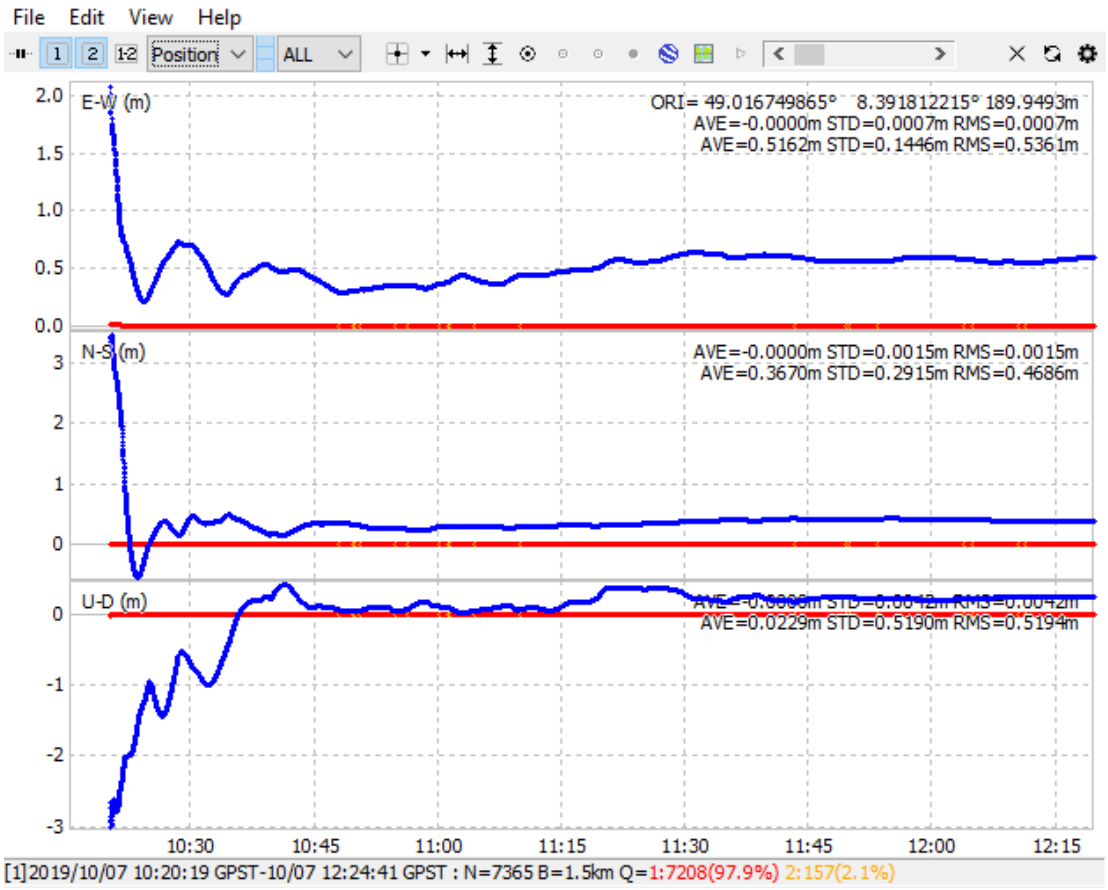

Figure 49: Position visualization DGNSS and PPPS. (Own source)

It can be seen how the PPP takes about 20 minutes to begin to converge, and shows the fluctuations mentioned above, while the DGNSS has stability throughout all periods.

It can therefore be concluded that the solution in DGNSS has better results compared to the PPPS. All these results are quite useful when considering a test with real-time processing, since, as seen in this static test, the results of the DGNSS are better than the PPP, but that PPP is gradually getting no it is necessary to depend on permanent stations, which would be ideal for navigation and the Moving Base.

As a test and comparison between both antennas that Novatel's own sensor registers, the observables of the second antenna have been processed in both DGNSS and PPPS. In the case of DGNSS it can be seen how the second antenna also registers and generates good results in the postprocess, being a little less precise than in the case of the first antenna but with good responses, very similar. In the following figure the points (in green) of the second antenna are also concentrated on the same area, in addition to being able to observe that the separation between the two antennas coincides with respect to reality.

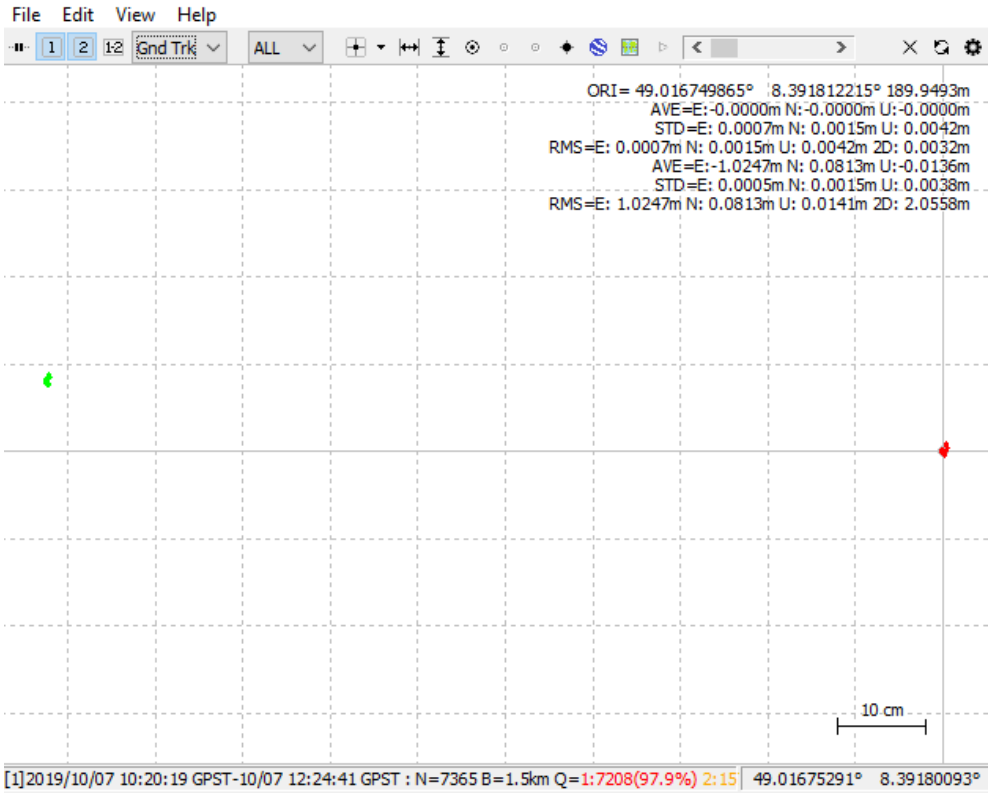

Figure 50: DGNSS ant1 and ant2 Ground track. (Own source)

If the graphs are compared in the position mode, both solutions have similar trends, setting points very quickly and with a very high percentage of them for the second antenna.

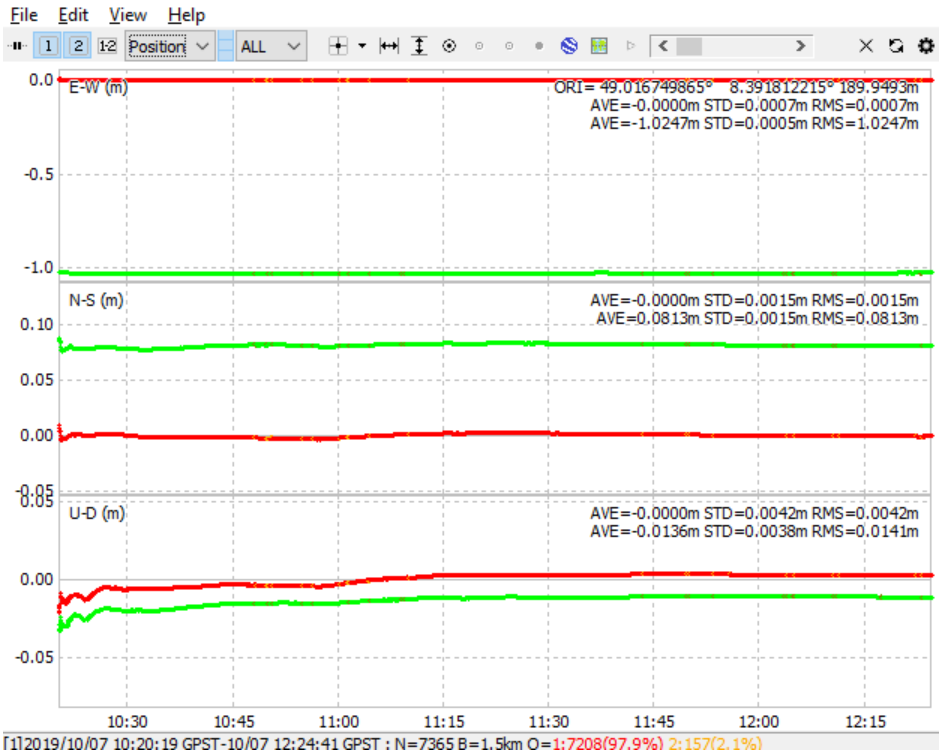

Figure 51: Position visualization DGNSS. (Own source)

In the case of PPPS the variation is greater. In the visualization of the trajectory of the points, the trend in convergence is similar for both antennas, (in blue the antenna2 and in orange the antenna 1) where the antenna 2 also generates all the points with greater separation between them.

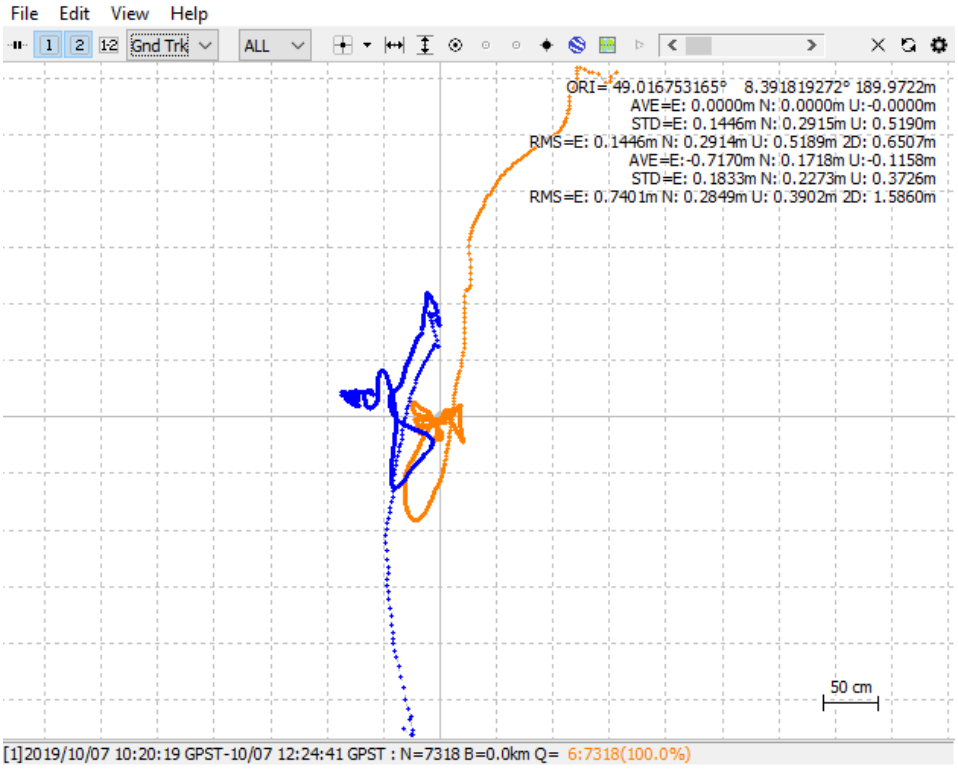

Figure 52: PPPS ant1 and ant2 Ground track. (Own source)

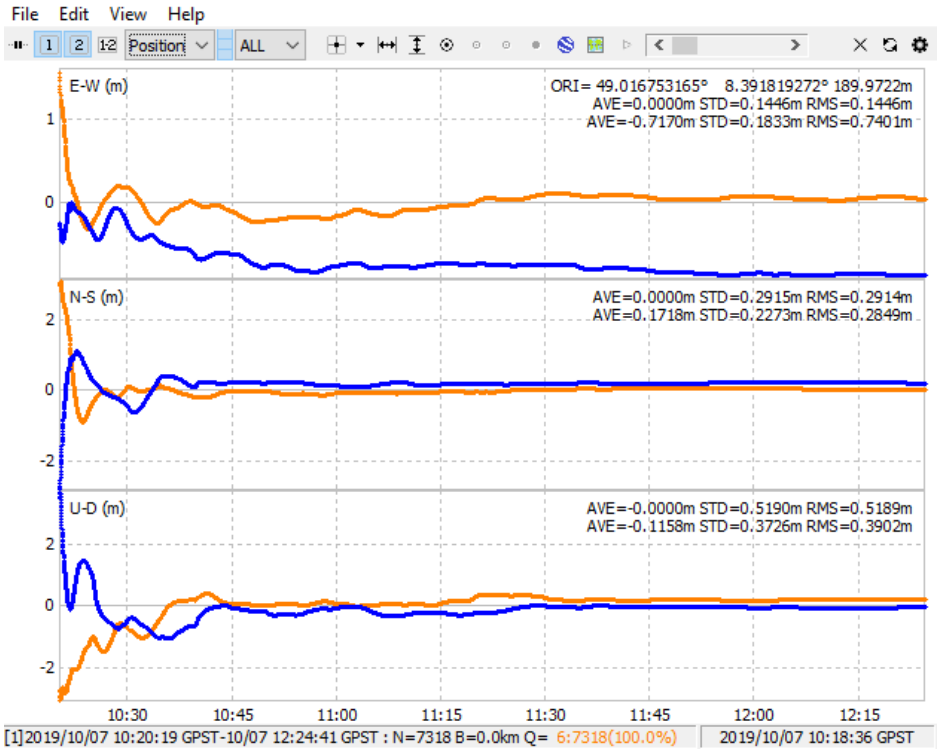

Figure 53: Position visualization PPP. (Own source)

In the previous figure the convergence times of PPPS for both antennas are quite similar, and in some cases the errors of antenna 2 are smaller than those of the primary antenna.

From all the data collected, it can be concluded that for the test that concerns us DGNSS data will generate better results when calculating the coordinates of the rovers from the baselines. PPP does not generate such precise results but perhaps with some improvement and adaptation in case of real-time measurements, its accuracy would be enough to perform navigation test in Moving Base mode.

8.2.3 Rover position calculation

With the coordinates of the master receiver already obtained (for the following case the DGNSS coordinates have been used since it has been shown that their results are better, although those of PPP could also have been used) the next step is to obtain the baselines of the two antennas, the secondary one of Novatel and the one connected to the Ublox and with it, with the Moving Base software created to generate and obtain the coordinates of the rovers. Since one of the principles was to program the MCLAMBDA ambiguity resolution algorithm, at the time of processing the baselines this option was chosen in addition to a second processing in "Fix and Hold" programmed in RTKLIB in order to compare both methods and see if in this case the resolution of ambiguities by MCLAMBDA generates better results or not from the baselines.

For the present test, since it is a static test, the results of the coordinates of the rovers are not entirely relevant, since the moving platform is not found, but it is interesting to analyse the results both for the resolution of ambiguities and for checking that the developed Moving Base software works properly.

The following figure shows the results of the Novatel antenna 2 on the same plot, on the one hand solution 1 with "Fix and Hold (FAH)" (in red points with fixed solution and in orange points with float solution), and on the other hand in solution 2 MCLAMBDA (in dark blue fixed points and in light blue points in float). First, the percentage of points set in each of the solutions has been analysed, appreciating that in the FAH solution the percentage is 93.8%, the rest being points in float, while in the MCLAMBDA solution the percentage is 99%. The MCLAMBDA solution offers better results in this test, set faster. Analysing the standard deviation (STD) and the mean square error (RMS) of each of the components (ENU), it can be seen that the MCLAMBDA solution presents less error compared to the FAH solution, except in the case of the standard deviation of the U component, where a greater error is appreciated, perhaps due to an outlayer, which can be seen at the bottom of the following figure. The values of STD and RMS can be seen in the upper right corner, those above corresponding to solution 1 (FAH) and those below to solution 2 (MCLAMBDA).

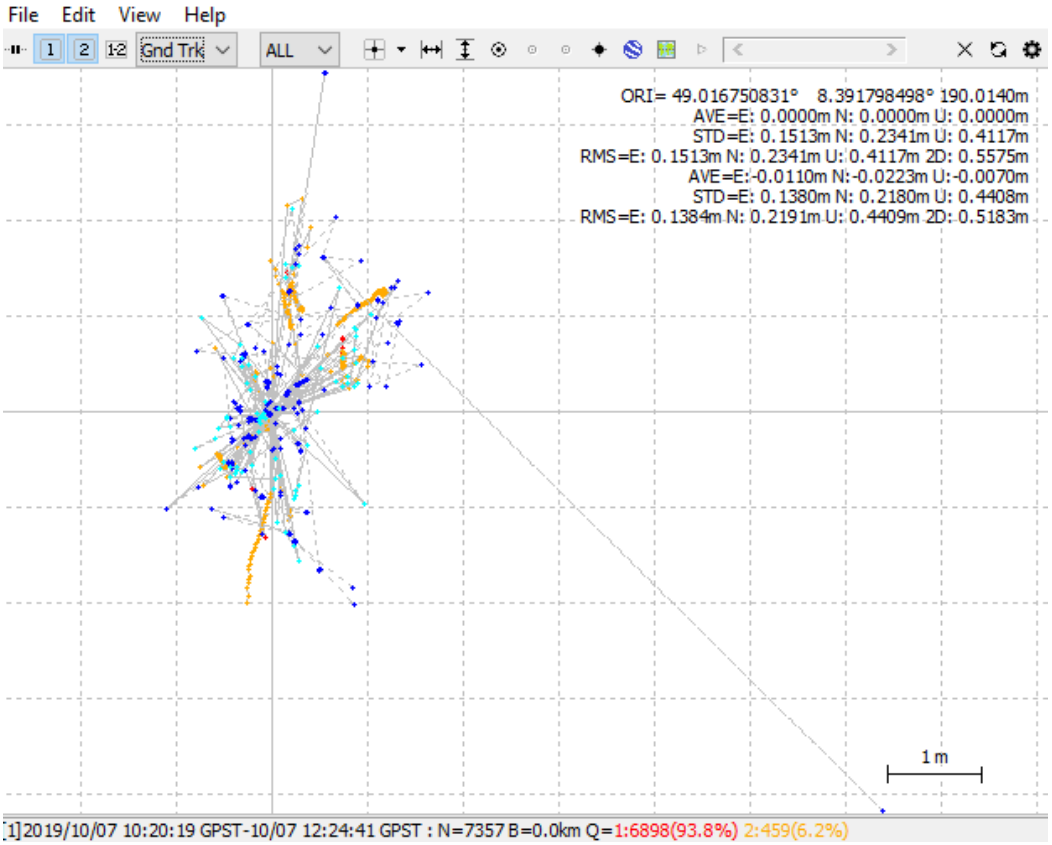

Figure 54: FAH and MCLAMBDA ant2 solution Ground track. (Own source)

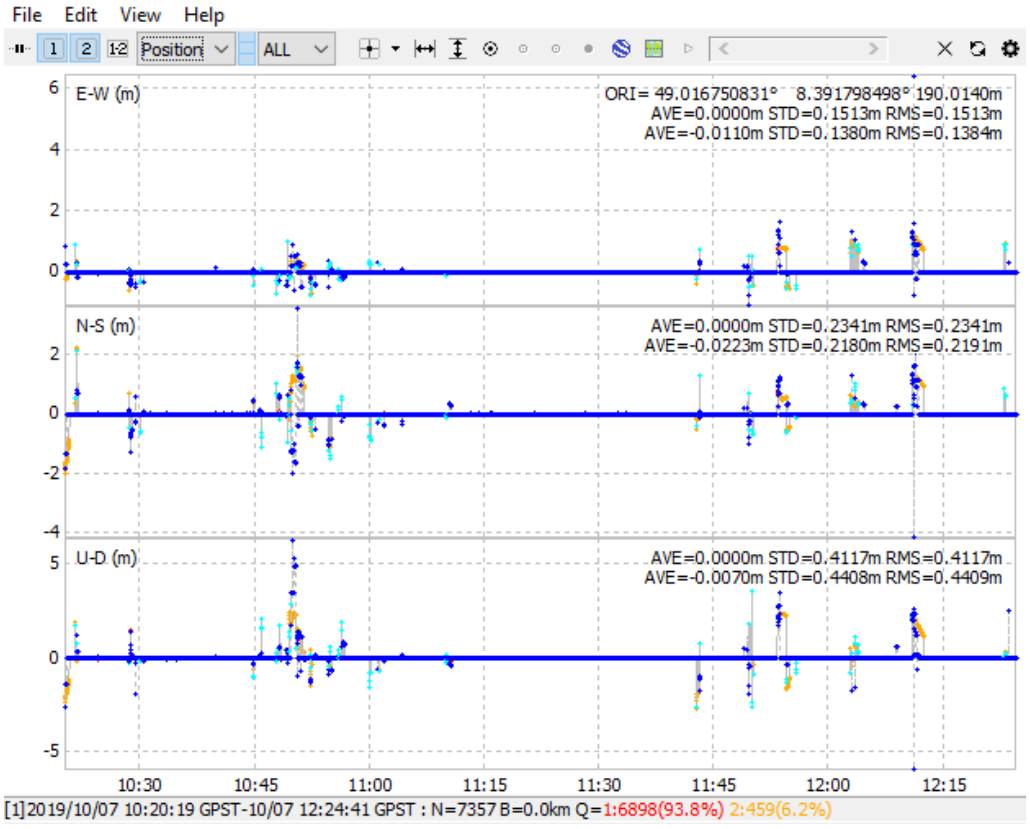

Figure 55: Position visualization FAH and MCLAMBDA. (Own source)

Analysing the previous fixation, you can see those points that present a greater error, where it is seen as in the case of the FAH solution the residues occur in greater rate, although here it can be appreciated that in one of the times there is an outlayer that It presents a large error in U for the MCLAMBDA solution, which corresponds to the one mentioned above.

The same analysis is carried out for the antenna connected to the Ublox sensor. In this case, something very interesting happens and the percentage of fixing points in FAH is much lower, of 65.2%, while in the case of MCLAMBDA it is 99.1%. It is also appreciated that in the case of Ublox the dispersion of the points is greater, perhaps because said sensor does not have L2, only measurements of L1.

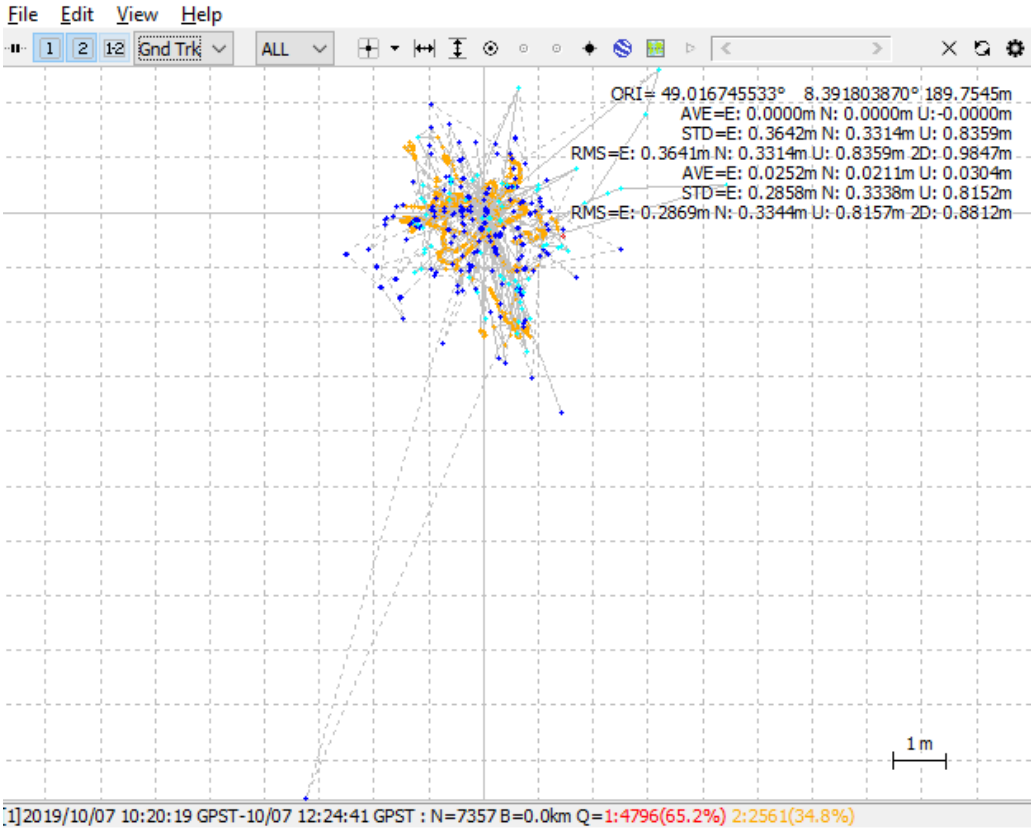

Figure 56: FAH and MCLAMBDA ublox solution Ground track. (Own source)

Analyzing the errors they are similar in both cases for each component being a little lower in the case of MCLAMBDA. If you compare the position graph you can see the dispersion mentioned above in the case of FAH, as well as some outlayers that appear in the case of MCLAMBDA, which are large.

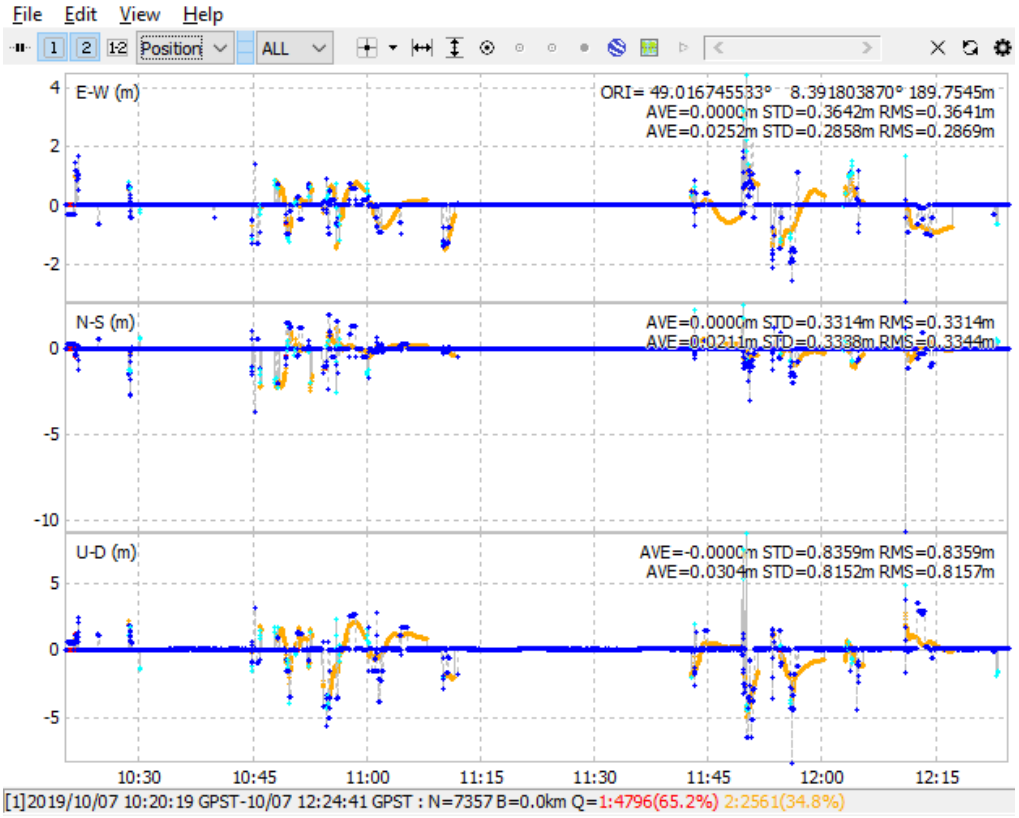

Figure 57: Position visualization FAH and MCLAMBDA. (Own source)

It can be concluded, therefore, that the MCLAMBDA solution in this test has better results than the FAH solution, although it seems to suffer in some observations. In addition, it can be concluded that Novatel antenna 2 has better results than Ublox antenna, perhaps caused by the presence of L2 in the observations and processing of the Novatel secondary antenna.

In the following section, the same analysis process will be carried out but in the case of the kinematic test, paying more attention to the results of the position of the rovers, since it has been shown that the solution of the master receiver is better in DGNSS than in PPP, although the solutions for each mode of said antenna will also be shown.

8.3 Kinematic test

A kinematic test was carried out considering that the entire platform was in motion at the same time, so the case of RTK disappears where the base station is fixed while the rovers move. In this case all the antennas are in motion. The duration of the test was approximately 40 minutes where at the beginning of the test the platform was left static in order to obtain a fixation time for the solutions for both DGNSS and PPP. After them, several movements of the platform were made following a line and following a square, to analyse its trajectory.

8.3.1 Single mode antenna 1

Following the structure of the previous section, the results of the Novatel antenna 1 in Single mode, RTKLIB and the solution provided by Novatel have been analysed, using Novatel Reader software.

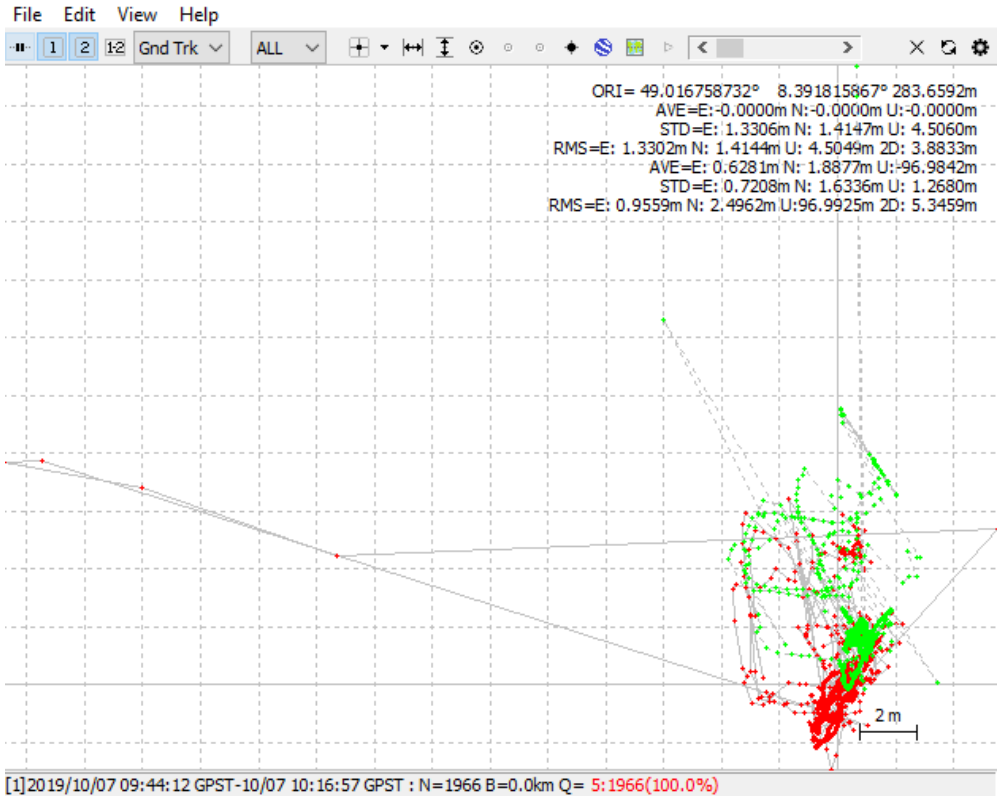

Figure 58: Single processing RTKLIB and Novatel Ground Track. (Own source)

As can be seen in the previous figure, the results of RTKLIB (in green) are much better than those of Novatel (in red). You can see the route in the RTKLIB solution as well as the initialization in static, while in the Novatel solution the results vary considerably with the trajectory followed, as well as the presence of outlayers in it, especially at the moment when the platform is in motion.

If the STD and RMS errors are analysed, it can be seen how the one analysed at a glance agrees with their results, since the values are higher in the case of the Novatel solution.

In the following figure, corresponding to the visualization of the position, you can see the differences between the two solutions, seeing how in the final part the difference is greater since it is the case in which the platform is moving. In addition, as in the case of the static test there is a big difference in the U-D component that, as in the previous case, is due to the ellipsoidal height obtained with the undulation and the height above sea level given by the Novatel solution.

Figure 59: Position visualization Single mode. (Own source)

8.3.2 Kinematic DGNSS and PPPK

As mentioned in the previous test, the data processing of the Novatel antenna 1 that acts as a master receiver is very important since its accuracy will determine a better result when obtaining rover positions. As in the previous case, the observables of the Novatel antenna 1 have been processed in both DGNSS and PPPK, following the same steps as in the previous test.

As in the previous test, the results of kinematic DGNSS and PPPK have been analysed at the same time (solution 1 in red for DGNSS and solution 2 in blue for PPPK). In this case something very curious happens and it is that there is not as much difference between DGNSS and PPPK as can be seen in the static test.

In the following ground track figure, you can see the initialization points and the path made quite clearly for both solutions. In both cases, outlayers are seen at the same times, perhaps due to a considerable loss of satellites at the time of measurement. The good thing about this is that it occurred at the time when the platform was static, so it does not affect the trajectory.

If the errors of both solutions are analysed together, it can be seen how in the case of the PPPK the errors are greater, perhaps due to the initialization that PPP needs at the beginning of the data collection. In DGNSS the number of fixed points corresponds to 97% of them, which can be considered quite acceptable.

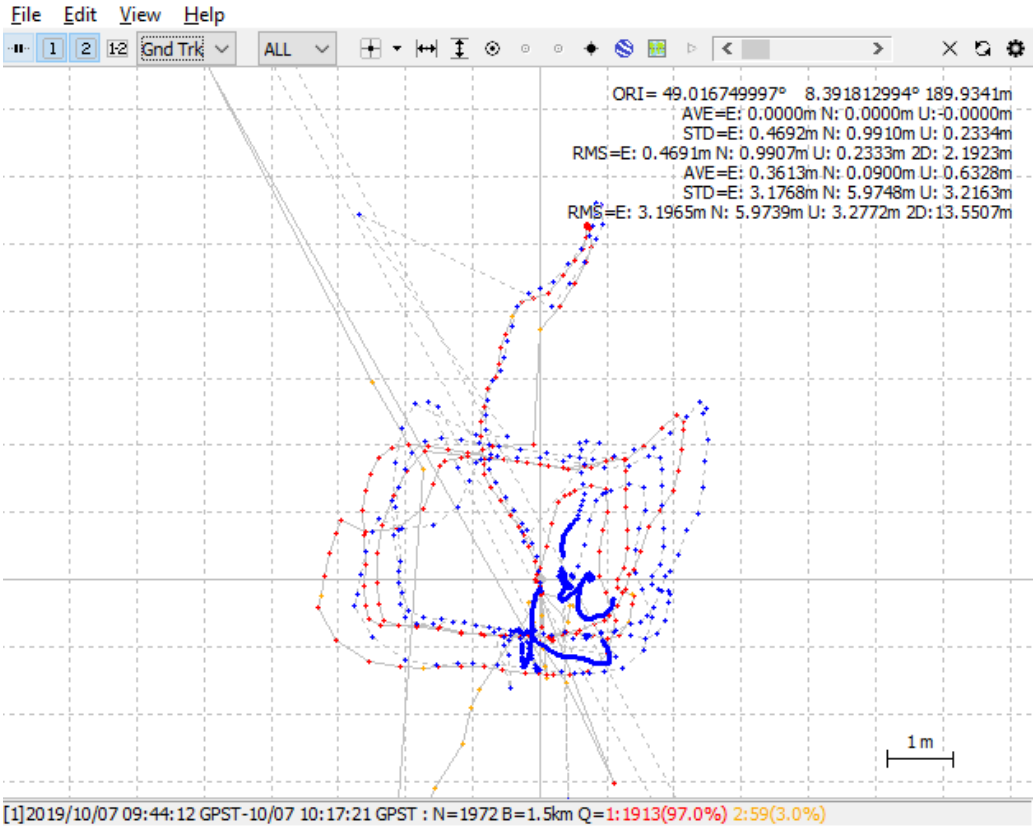

Figure 60: DGNSS and PPPK ant1 solution Ground Track. (Own source)

Figure 61: Position visualization DGNSS and PPPK solutions. (Own source)

In the previous figure of visualization of the position we see that the difference between both solutions is smaller than expected, compared to the static test, and PPP outlayers are appreciated at some times that are very large, so it generates that the RMS and the STD is triggered.

It can therefore be concluded that, although the DGNSS solution continues to present better results, the PPPK solution is quite good, since its results are quite acceptable and more when talking about navigation where centimetre accuracy is acceptable, and promising for future work where you don't want to depend on permanent stations.

It remains to be found out if the big difference in the static test in terms of comparing DGNSS and PPPS solutions is due to the internal processing of RTKLIB or data collection.

8.3.3 Rover position calculation

As in the case of the static test, once the coordinates of the master receiver (Novatel antenna) have been obtained and together with them the baselines of each of the receivers that act as rovers, both with "Fix and Hold (FAH)" (in red and orange) resolution as in MCLAMBDA (in light blue and dark blue), the next step is to obtain the coordinates of these rovers in the e-frame. As mentioned in the section of the Novatel software, since the baselines are in the n-frame (ENU) the software performs a transformation by means of a rotation and subsequently adds the increments of coordinates obtained to the corresponding point of the master receiver in the given time. This gives these coordinates and solutions that are going to be analysed next antenna to antenna.

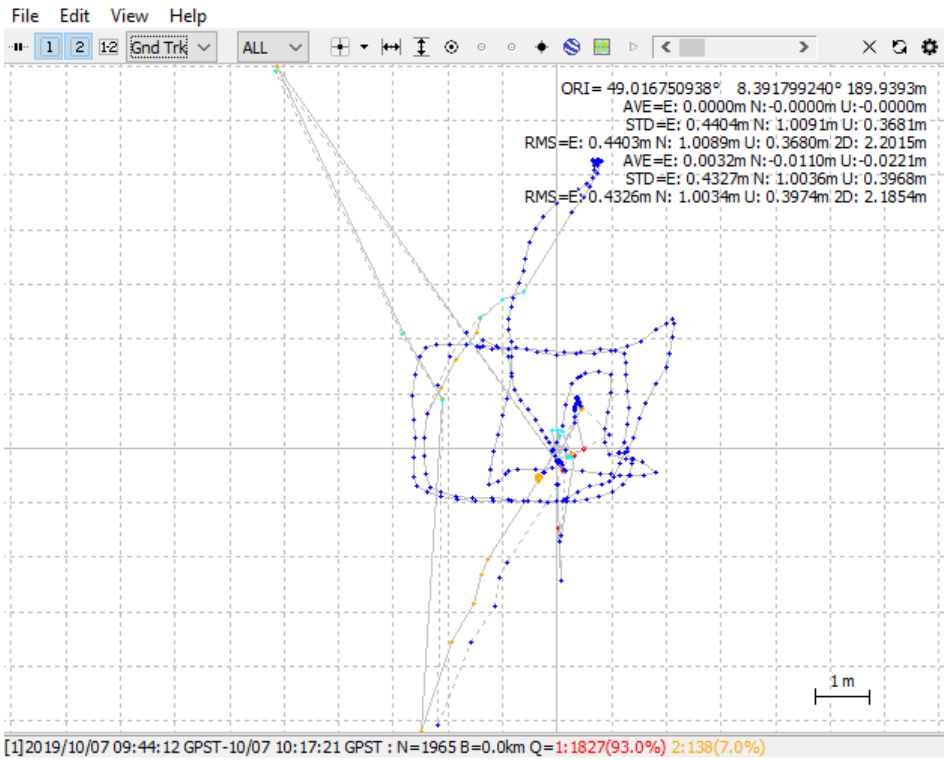

Figure 62: FAH and MCLAMBDA ant2 solution Ground track. (Own source)

In the previous figure it can be seen that the differences are minimal, seeing that the greatest variations occur in the initialization of the test, but subsequently both solutions are similar, the percentage of fixed solutions being greater in the case of MCLAMBDA, of 99.3%, compared to 93.0% of FAH. Regarding the errors we see that the trend is like the previous cases, the error in U is somewhat greater in MCLAMBDA while in the other two components it presents a greater FAH error.

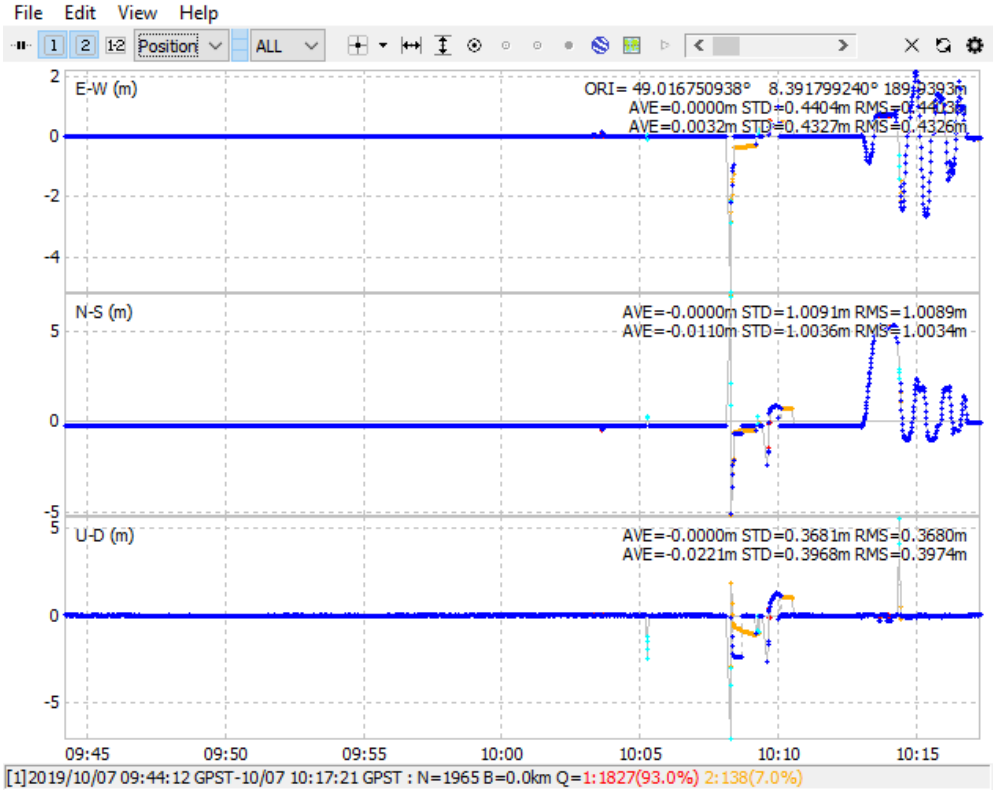

Figure 63: Position visualization FAH and MCLAMBDA. (Own source)

As for the position graph, the variations in position occur at the end of the position given that it is when the platform was in motion. Since the platform is always at the same height the U-D address does not change, except for those times that have outlayers, as we have seen in the previous point, where several observations are not entirely good, so errors appear. In this case both solutions could be given as good since both have very similar conditions.

The following analysis corresponds to the antenna connected to the Ublox sensor. Since this sensor only records L1 the results are somewhat worse, as has happened in the case of the static test, the difference between the FAH and MCLAMBDA solution being better appreciated. In the following figure you can see how the trajectory does not coincide as much as in the previous case, especially in the initial part of the movement in a straight line, where FAH has enough points in float solution. The percentage of solutions set for FAH is 50.2% while in the case of MCLAMBDA it is 99.2%.

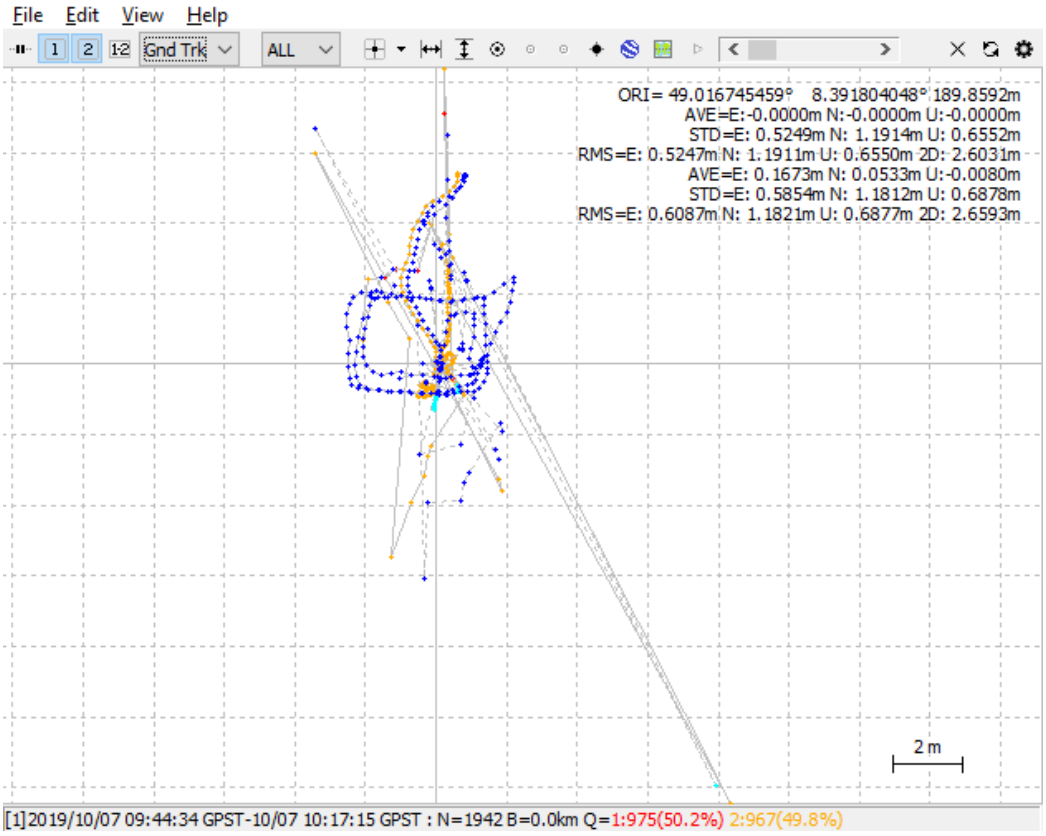

Figure 64: FAH and MCLAMBDA Ublox solution Ground track. (Own source)

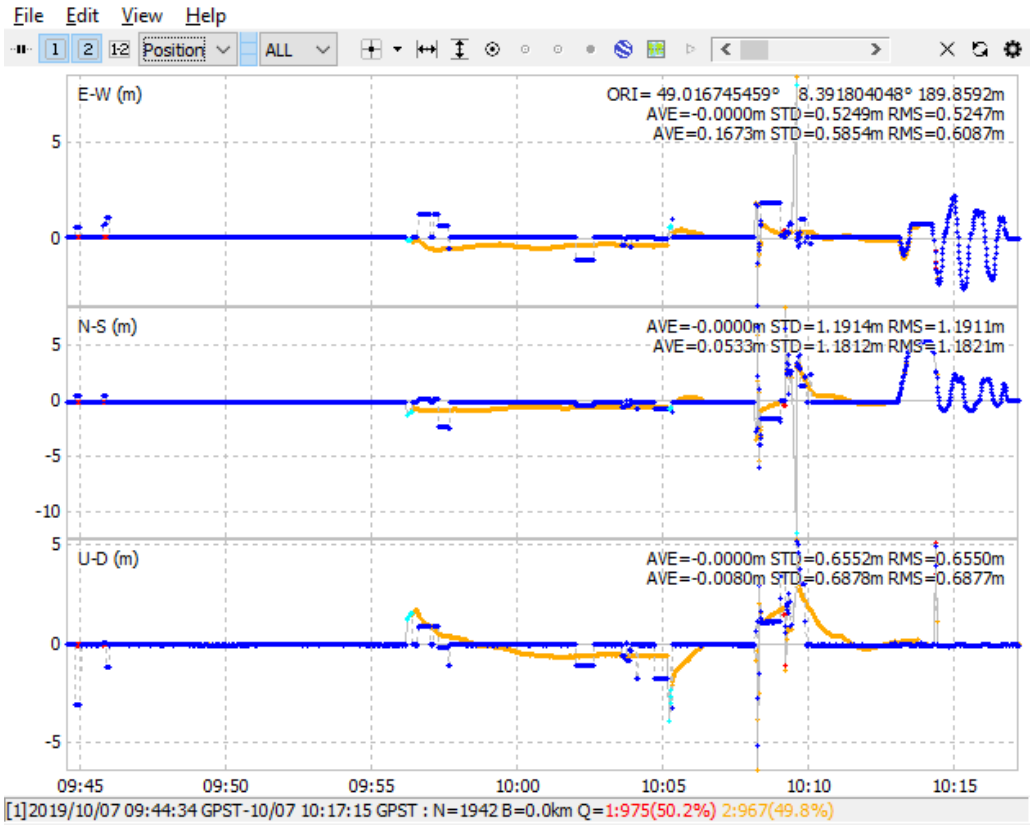

Figure 65: Position visualization FAH and MCLAMBDA. (Own source)

In the previous position graph, you can see what has been mentioned above, where it happens that both solutions have greater differences, and appreciating the peaks where the coordinates go since the coordinates of the master has a problem at sometimes. Therefore, it can be concluded that the solution with MCLAMBDA generates better results with a higher level of confidence.

Finally, as a final representation, the following figure shows the solutions of the secondary antenna of Novatel (in blue) and that of Ublox (in red) at the same time. In addition, the display time has been cut to show only the times that correspond to the moment when the platform was in motion. It can be seen how the marked path is practically identical in both cases, since the baselines are fixed on the platform, which are 0.72 m between antenna 2 and Ublox, antenna 1 and Ublox, and 1.02 m between antenna 1 and antenna 2. This distance is always preserved, so the results are correct.

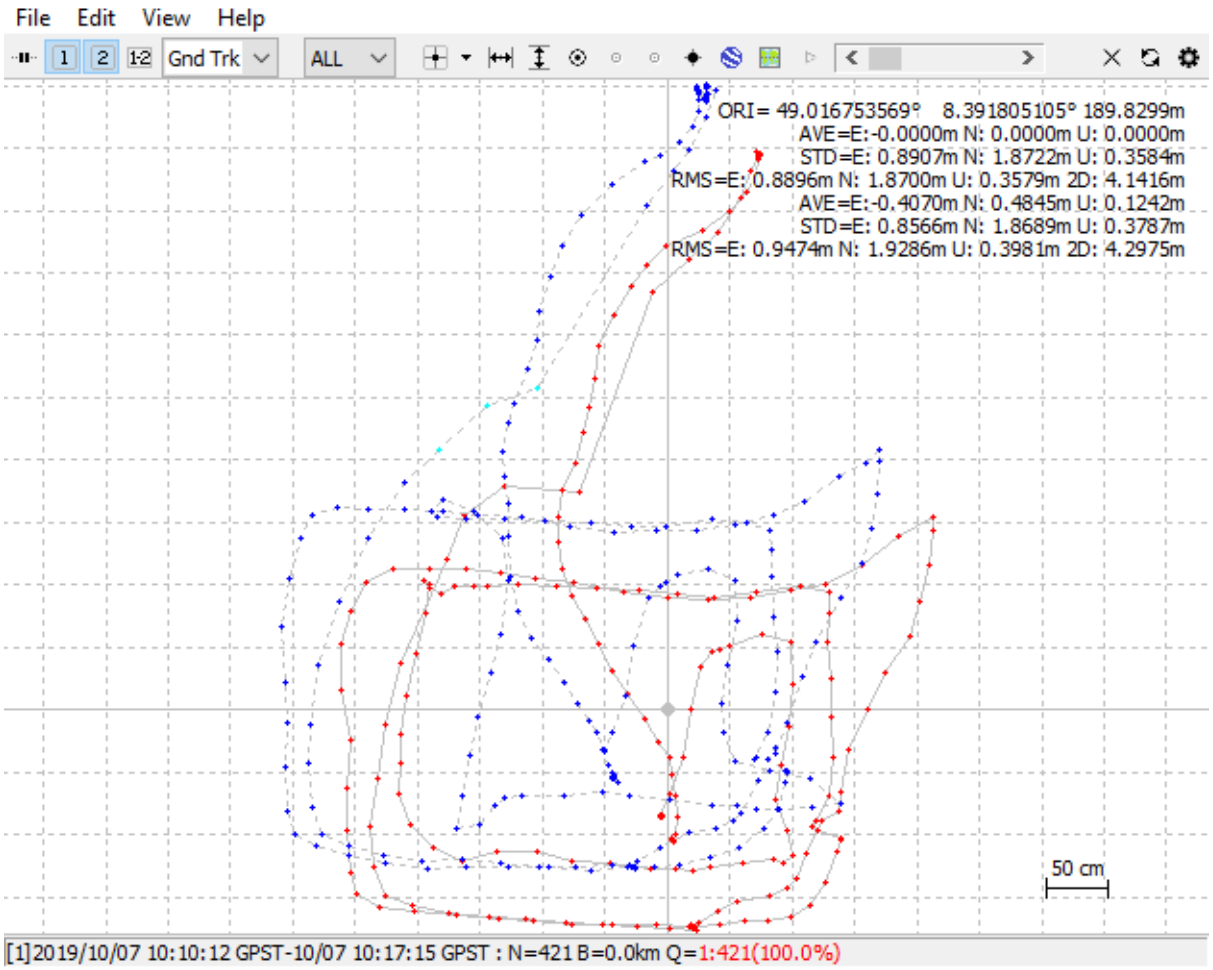

Figure 66: Ublox and antenna 2 trajectory MCLAMBDA. (Own source)

9. Conclusions and future work

9.1 Conclusions

The objective of this thesis included the contact and subsequent development of existing GNSS algorithms for navigation. For this, a study was made of the existing algorithms of which three of them were analysed, as they are the most important in all bibliographies, "Multikin", "MCLAMBDA" and "Extended LAMBDA method". After the analysis of the three, it was concluded that the results obtained by the authors of the algorithms showed that the MCLAMBDA algorithm had better results and great robustness when it came to resolving ambiguities. This algorithm was subsequently developed in the C ++ language in order to finally be able to implement its operation within the RTKLIB open source library. Once its implementation is developed, it allows, by means of the Moving Base mode of RTKLIB, to obtain the baselines between a rover and a base station, in the n-frame. Since the RTKLIB algorithm for the Moving Base solution is quite weak, new software was developed, because for its operation RTKLIB performs an antenna processing that acts as a master receiver in Single mode, and setting the position of the antenna, that is, at no time is it taken into account that the base receiver is moving. Said software which has been called "Moving Base Software", allows that, from the coordinates of a base station, obtained by RTKLIB by processing in PPP or DGNSS, and from the baselines obtained with the above-mentioned method, They can obtain the coordinates of up to three rovers simultaneously. The procedure is simple; given the coordinates of the master receiver in the e-frame and the baselines in the n-frame, you just have to perform a transformation of the latter by means of a rotation to obtain the XYZ increments in the e-frame, and thus be able to add said increases to the coordinates of the base for each of the periods. In addition, the software allows the results to be saved in a file with RTKLIB format in such a way that it is readable by the software itself for the analysis that the user wants to perform, such as displaying the data in RTKPLOT.

In addition, two tests were carried out to evaluate the operation of the tools; a static test and a kinematic test. For the tests two sensors installed in a GNSS/MEMS/camera navigation box were used, which were a double antenna sensor L1 L2 Novatel and a Ublox L1 sensor. As for the tests, a two-hour static measurement was carried out to subsequently correctly analyse not only the operation of the tools, but also the operation of the sensors themselves, since it was the first time that a test was achieved with these sensors A kinematic test was also carried out which presents a static initialization for a correct operation and subsequently several trajectories were made online and in square to observe the behaviour of the algorithms. For the tests a platform was mounted where three antennas were placed in a fixed situation, in a triangular metal structure where the baselines between the antennas were fixed, antennas which were connected to the Novatel and Ublox sensors. Regarding the test results, the processing of the Novatel antenna 1 was analysed, which has been used as a master receiver, in DGNSS and PPP modes. After analysing the results, it is concluded that those obtained with DGNSS are better and more precise than in PPP,

especially for the static test where there is a great variation and the convergence time of the PPP is very wide. However, in the case of the kinematic test, the difference between DGNSS and PPP is no longer so wide, which leaves good hope that one day the PPP will be the ideal processing mode of the master receiver and thus not depend so much on the permanent stations. Regarding the results of the positions of the rovers, antenna 2 of Novatel and Ublox, the operation of MCLAMBDA was analysed by comparing it with the solution of the baselines using "Fix and Hold (FAH)", which is already programmed default in RTKLIB. The results indicate that MCLAMBDA generates better accuracies and sets a greater number of points, so one of the objectives is met. On the other hand, analysing the operation of the MBS software, it is concluded that the desired objectives are met, since it is achieved that the master antenna is considered moving and is processed in a processing mode that offers better results.

Finally, as the Novatel sensor internally allows to record a single mode procedure as well as data on the heading, a small software was developed that allows to read the raw data files of Novatel and generate a final file with the Novatel solution readable for RTKLIB, to possible comparisons, although it has been shown that such processing is worse than the RTKLIB Single mode.

9.2 Future work

Although this work meets the objectives set, this section mentions possible improvement and implementation solutions for future work, perhaps for master candidates.

- Implementation of everything in the same software. Due to the complexity in the modification of RTKLIB the separation has been carried out in other software for the processing in Moving Base, but a possible solution would be to make RTKLIB directly solve the coordinates of the master receiver in more processing modes such as DGNSS or PPP and consider that these coordinates move along the rover. In addition, although very complex, it would be interesting to process several rovers at once, since RTKLIB only allows one to be processed.
- Perform different field tests with different sensors and conditions. It would be very interesting to be able to perform tests with other sensors and antennas as well as different situations, for example, that the baselines were larger or that they were not fixed. Perform larger movements where all ENU components are greatly affected, with changes in speed and therefore acceleration.
- Implement the entire process in real time. The next step would be for everything to work in real time, for example, to implement everything in RTKNAVI. The somewhat MCLAMBDA would have the same implementation that has been carried out in this project with RTKPOST. You could also try to combine with INS data or photogrammetry navigation.

10. References

Berné Valero, José Luis, et al. *GNSS GPS: Fundamentos y Aplicaciones En Geomática*. Ed. Universidad Politécnica de Valencia, 2016, www.lalibreria.upv.es.

Bruyninx, Carine, et al. “ETRF/ITRF Transformation.” *GPS Solutions*, vol. 23, no. 4, 2 Oct. 2019, p. 106, doi:10.1007/s10291-019-0880-9.

C++Builder - Embarcadero Website. https://www.embarcadero.com/es/products/cbuilder. Accessed 7 Oct. 2019.

Delft, Sandra Verhagen. *LAMBDA Software Package : Matlab*. no. June 2015, 2012, pp. 0–38.

ESA (European Spatial Agency). *Navipedia*. https://gssc.esa.int/navipedia/index.php/Main_Page. Accessed 7 June 2019.

Giorgi, Gabriele. *The Multivariate Constrained LAMBDA Method for Single-Epoch, Single-Frequency GNSS-Based Full Attitude Determination*. 2007.

Groves, P. D. *Principles of GNSS, Inertial, and Multisensor Integrated, Technology*. 2008.

Hofmann-Wellenhof, Bernhard, et al. *GNSS Global Navigation Satellite Systems. GPS, GLONASS, Galileo and More*. SpringerWienNewYork, 2008.

Jäger, Reiner. *GNSS Platform*. 2014.

Luo, Ning. *Precise Relative Positioning of Multiple Moving Platforms Using GPS Carrier Phase Observables*. 2001.

Navka. *Navka*. http://www.navka.de/index.php/de/. Accessed 7 June 2019.

Novatel. *NovAtel Connect User Guide*. no. October, 2018.

---. *OEM6® Family Firmware Reference Manual*. https://www.novatel.com/assets/Documents/Manuals/om-20000129.pdf.

---. *OEM617D GPS*. https://www.novatel.com/products/gnss-receivers/oem-receiver-boards/oem6-receivers/oem617d/. Accessed 8 Oct. 2019.

NovAtel Connect | Canal Geomatics. http://www.canalgeomatics.com/product/novatel-connect/. Accessed 24 July 2019.

Roth, Jochen, et al. “Improving GNSS Attitude Determination.” *InsideGNSS*, 2012, pp. 54–62.

Takasu, Tomoji. *RTKLIB Software Manual*. no. C, 2013.

U-blox. *GNSS Evaluation Software for Windows*.

ublox. *NEO/LEA-M8T u-Blox M8 Concurrent GNSS Timing Modules*. 2015, www.u-blox.com.

Wikipedia. *Wikipedia, the Free Encyclopedia*. https://en.wikipedia.org/wiki/Main_Page. Accessed 7 June 2019.

Appendix A: Novatel and Ublox configuration

The programs mentioned in section 4.2 Software have been used for the configuration of both sensors, in addition to the RTKLIB library, which allows the connection to the sensors for data collection.

A.1 Novatel configuration

The connection to the Novatel OEM617D sensor has always been made by serial port. For the connection in Novatel Connect (Novatel, *NovAtel Connect User Guide*) it is only necessary to select the port and indicate the corresponding baud rate for the connection. For the sensor installed in GNSS/MEMS/camera navigation box it is used:

- Serial: Depends on your computer USB connection.
- Baud Rate: 115.200

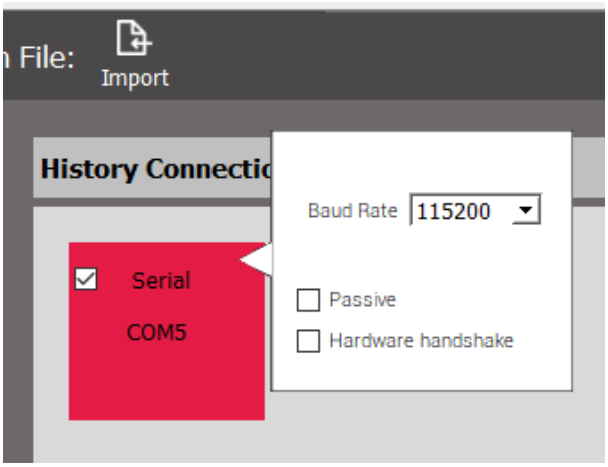

Figure 67: Novatel configuration. (Own source)

If you want to make the connection using RTKLIB, the configuration is the same.

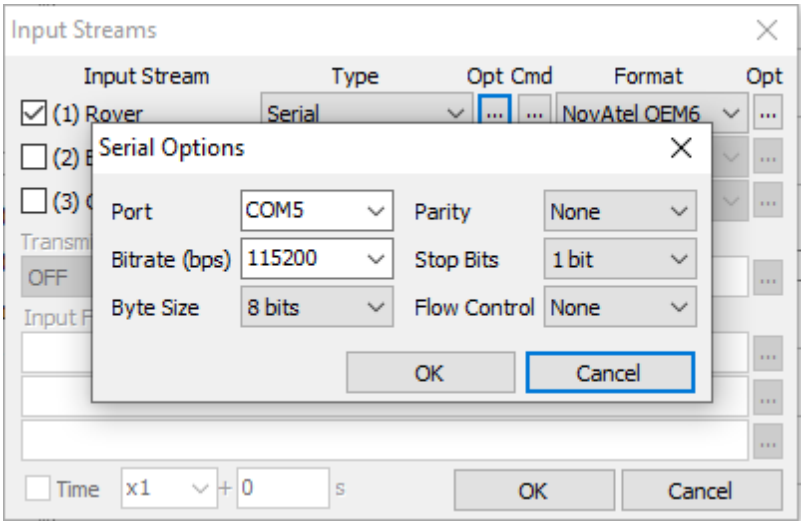

Figure 68: RTKLIB configuration serial port in RTKNAVI. (Own source)

To make the necessary log for the application, a "*.cmd"* file has been compiled that can be executed from RTKLIB in order to save a file with the necessary logs.

Recommended logs:

- unlogall
- log rangecmpb ontime 1
- log rawephemb onnew
- log ionutcb onnew
- log rawwaasframeb onnew
- log gloephemerisb onnew
- log qzssrawsubframeb onnew
- log qzssionutcb onnew
- log galephemerisb onnew
- log bdsephemerisb onnew
- log rangecmpb_1 ontime 1
- log bestposa ontime 1
- log headinga onchanged

For more information about other logs visit (Novatel, *OEM6® Family Firmware Reference Manual*).

Software versions:

- RTKLIB version 2.4.3
- Novatel Connect 2.3.1

The use of Novatel is recommended only for checking connections, in addition to checking whether both antennas receive data or not. For log data it is recommended to use RTKLIB. This is because the commands mentioned above are old and the new version of Novatel Connect does not recognize them correctly.

A.2 Ublox configuration

The configuration of Ublox is like that of Novatel, since it is a USB serial connection, but in this case Ublox's own software is used.

- Serial: Depends on your computer USB connection.
- Baud Rate: 921.600

Software versions:

- RTKLIB version 2.4.3
- U-Center 19.05

The use of U-Center for the capture of the data is recommended, since there may be conflicts with the formats when doing it with RTKLIB, although it is possible to do it with the latter. For more information about other logs visit (U-blox).

Appendix B: Data processing

For processing the data of the primary antenna of Novatel, the DGNSS and PPP processing modes have been used, which require a certain configuration to obtain better results. This appendix will present the most important aspects of the configuration as well as the result of the configuration files that RTKLIB allows to save.

B.1 DGNSS

For differential GNSS processing it is necessary to obtain data from a permanent reference station near the study area. In this case the permanent station KARL of EUREF has been used. To differentiate between kinematic DGNSS and static DGNSS, the kinematic and static modes must be chosen in the RTKLIB processing options. For best results it is interesting to load the antenna file containing the antenna model and its configurations. The coordinates of the reference station and the antenna height must also be added.

```
# rtkpost options (2019/10/28 11:51:16, v.2.4.3 b32)

pos1-posmode      =kinematic/static  # (0:single,1:dgps,2:kinematic,3:static,4:movingbase,5:fixed,6:ppp-kine,7:ppp-static,8:ppp-fixed)
pos1-frequency    =l1+l2     # (1:l1,2:l1+l2,3:l1+l2+l5,4:l1+l5)
pos1-soltype      =forward   # (0:forward,1:backward,2:combined)
pos1-elmask       =15        # (deg)
pos1-snrmask_r    =off       # (0:off,1:on)
pos1-snrmask_b    =off       # (0:off,1:on)
pos1-snrmask_L1   =0,0,0,0,0,0,0,0,0
pos1-snrmask_L2   =0,0,0,0,0,0,0,0,0
pos1-snrmask_L5   =0,0,0,0,0,0,0,0,0
pos1-dynamics     =off       # (0:off,1:on)
pos1-tidecorr     =off       # (0:off,1:on,2:otl)
pos1-ionoopt      =brdc      # (0:off,1:brdc,2:sbas,3:dual-freq,4:est-stec,5:ionex-tec,6:qzs-brdc,7:qzs-lex,8:stec)
pos1-tropopt      =saas      # (0:off,1:saas,2:sbas,3:est-ztd,4:est-ztdgrad,5:ztd)
pos1-sateph       =brdc      # (0:brdc,1:precise,2:brdc+sbas,3:brdc+ssrapc,4:brdc+ssrcom)
pos1-posopt1      =off       # (0:off,1:on)
pos1-posopt2      =off       # (0:off,1:on)
pos1-posopt3      =off       # (0:off,1:on,2:precise)
pos1-posopt4      =off       # (0:off,1:on)
pos1-posopt5      =off       # (0:off,1:on)
pos1-posopt6      =off       # (0:off,1:on)
```

```
pos1-exclsats     =          # (prn ...)
pos1-navsys       =5         # (1:gps+2:sbas+4:glo+8:gal+16:qzs+32:comp)
pos2-armode       =fix-and-hold # (0:off,1:continuous,2:instantaneous,3:fix-and-hold)
pos2-gloarmode    =on        # (0:off,1:on,2:autocal)
pos2-bdsarmode    =on        # (0:off,1:on)
pos2-arthres      =3
pos2-arthres1     =0.9999
pos2-arthres2     =0.25
pos2-arthres3     =0.1
pos2-arthres4     =0.05
pos2-arlockcnt    =0
pos2-arelmask     =0         # (deg)
pos2-arminfix     =10
pos2-armaxiter    =1
pos2-elmaskhold   =0         # (deg)
pos2-aroutcnt     =5
pos2-maxage       =30        # (s)
pos2-syncsol      =off       # (0:off,1:on)
pos2-slipthres    =0.05      # (m)
pos2-rejionno     =30        # (m)
pos2-rejgdop      =30
pos2-niter        =1
pos2-baselen      =0         # (m)
pos2-basesig      =0         # (m)
out-solformat     =xyz       # (0:llh,1:xyz,2:enu,3:nmea)
out-outhead       =on        # (0:off,1:on)
out-outopt        =on        # (0:off,1:on)
out-outvel        =off       # (0:off,1:on)
out-timesys       =gpst      # (0:gpst,1:utc,2:jst)
out-timeform      =hms       # (0:tow,1:hms)
out-timendec      =3
out-degform       =deg       # (0:deg,1:dms)
out-fieldsep      =
out-outsingle     =off       # (0:off,1:on)
```

```
out-maxsolstd     =0         # (m)
out-height        =ellipsoidal # (0:ellipsoidal,1:geodetic)
out-geoid         =internal   # (0:internal,1:egm96,2:egm08_2.5,3:egm08_1,4:gsi2000)
out-solstatic     =all       # (0:all,1:single)
out-nmeaintv1     =0         # (s)
out-nmeaintv2     =0         # (s)
out-outstat       =off       # (0:off,1:state,2:residual)
stats-eratio1     =100
stats-eratio2     =100
stats-errphase    =0.003     # (m)
stats-errphaseel  =0.003     # (m)
stats-errphasebl  =0         # (m/10km)
stats-errdoppler  =10        # (Hz)
stats-stdbias     =30        # (m)
stats-stdiono     =0.03      # (m)
stats-stdtrop     =0.3       # (m)
stats-prnaccelh   =10        # (m/s^2)
stats-prnaccelv   =10        # (m/s^2)
stats-prnbias     =0.0001    # (m)
stats-prniono     =0.001     # (m)
stats-prntrop     =0.0001    # (m)
stats-prnpos      =0         # (m)
stats-clkstab     =5e-12     # (s/s)
ant1-postype      =llh       # (0:llh,1:xyz,2:single,3:posfile,4:rinexhead,5:rtcm,6:raw)
ant1-pos1         =90        # (deg|m)
ant1-pos2         =0         # (deg|m)
ant1-pos3         =-6335367.6285 # (m|m)
ant1-anttype      =
ant1-antdele      =0         # (m)
ant1-antdeln      =0         # (m)
ant1-antdelu      =0         # (m)
ant2-postype      =llh       # (0:llh,1:xyz,2:single,3:posfile,4:rinexhead,5:rtcm,6:raw)
ant2-pos1         =49.0112424871129 # (deg|m)
ant2-pos2         =8.41125529807704 # (deg|m)
```

```
ant2-pos3        =182.89576605604 # (m|m)
ant2-anttype     =
ant2-antdele     =0          # (m)
ant2-antdeln     =0          # (m)
ant2-antdelu     =0.045      # (m)
ant2-maxaveep    =0
ant2-initrst     =off        # (0:off,1:on)
misc-timeinterp  =off        # (0:off,1:on)
misc-sbasatsel   =0          # (0:all)
misc-rnxopt1     =
misc-rnxopt2     =
misc-pppopt      =
file-satantfile  =(path)\igs14_1949.atx
file-rcvantfile  =
file-staposfile  =
file-geoidfile   =
file-ionofile    =
file-dcbfile     =
file-eopfile     =
file-blqfile     =
file-tempdir     =
file-geexefile   =
file-solstatfile =
file-tracefile   =
```

B.2 PPP

No reference station is necessary for PPP processing, but it is interesting to use fast, ultrafast or accurate ephemeris, as well as the clock file. In addition to that, the ionosphere, DCB, ERP, and BLQ oceanic charges must be added to the processing for better performance. Regarding the processing mode, they differ between PPP kinematic and PPP static

rtkpost options (2019/10/28 16:27:30, v.2.4.3 b32)

pos1-posmode =ppp-static/ppp-kine # (0:single,1:dgps,2:kinematic,3:static,4:movingbase,5:fixed,6:ppp-kine,7:ppp-static,8:ppp-fixed)

pos1-frequency =l1+l2 # (1:l1,2:l1+l2,3:l1+l2+l5,4:l1+l5)

pos1-soltype =forward # (0:forward,1:backward,2:combined)

pos1-elmask =15 # (deg)

pos1-snrmask_r =off # (0:off,1:on)

pos1-snrmask_b =off # (0:off,1:on)

pos1-snrmask_L1 =0,0,0,0,0,0,0,0,0

pos1-snrmask_L2 =0,0,0,0,0,0,0,0,0

pos1-snrmask_L5 =0,0,0,0,0,0,0,0,0

pos1-dynamics =off # (0:off,1:on)

pos1-tidecorr =off # (0:off,1:on,2:otl)

pos1-ionoopt =dual-freq # (0:off,1:brdc,2:sbas,3:dual-freq,4:est-stec,5:ionex-tec,6:qzs-brdc,7:qzs-lex,8:stec)

pos1-tropopt =est-ztdgrad # (0:off,1:saas,2:sbas,3:est-ztd,4:est-ztdgrad,5:ztd)

pos1-sateph =precise # (0:brdc,1:precise,2:brdc+sbas,3:brdc+ssrapc,4:brdc+ssrcom)

pos1-posopt1 =on # (0:off,1:on)

pos1-posopt2 =on # (0:off,1:on)

pos1-posopt3 =on # (0:off,1:on,2:precise)

pos1-posopt4 =on # (0:off,1:on)

pos1-posopt5 =on # (0:off,1:on)

pos1-posopt6 =on # (0:off,1:on)

pos1-exclsats = # (prn ...)

pos1-navsys =5 # (1:gps+2:sbas+4:glo+8:gal+16:qzs+32:comp)

pos2-armode =fix-and-hold # (0:off,1:continuous,2:instantaneous,3:fix-and-hold)

pos2-gloarmode =on # (0:off,1:on,2:autocal)

pos2-bdsarmode =on # (0:off,1:on)

pos2-arthres =3

pos2-arthres1 =0.9999

pos2-arthres2 =0.25

pos2-arthres3 =0.1

pos2-arthres4 =0.05

pos2-arlockcnt =0

pos2-arelmask =0 # (deg)

pos2-arminfix =10

pos2-armaxiter =1

pos2-elmaskhold =0 # (deg)

pos2-aroutcnt =5

pos2-maxage =30 # (s)

pos2-syncsol =off # (0:off,1:on)

pos2-slipthres =0.05 # (m)

pos2-rejionno =30 # (m)

pos2-rejgdop =30

pos2-niter =1

pos2-baselen =0 # (m)

pos2-basesig =0 # (m)

out-solformat =xyz # (0:llh,1:xyz,2:enu,3:nmea)

out-outhead =on # (0:off,1:on)

out-outopt =on # (0:off,1:on)

out-outvel =off # (0:off,1:on)

out-timesys =gpst # (0:gpst,1:utc,2:jst)

out-timeform =hms # (0:tow,1:hms)

out-timendec =3

out-degform =deg # (0:deg,1:dms)

out-fieldsep =

out-outsingle =off # (0:off,1:on)

out-maxsolstd =0 # (m)

out-height =ellipsoidal # (0:ellipsoidal,1:geodetic)

out-geoid =internal # (0:internal,1:egm96,2:egm08_2.5,3:egm08_1,4:gsi2000)

out-solstatic =all # (0:all,1:single)

out-nmeaintv1 =0 # (s)

out-nmeaintv2 =0 # (s)

out-outstat =off # (0:off,1:state,2:residual)

stats-eratio1 =100

stats-eratio2 =100

stats-errphase =0.003 # (m)

stats-errphaseel =0.003 # (m)

```
stats-errphasebl   =0        # (m/10km)
stats-errdoppler   =10       # (Hz)
stats-stdbias      =30       # (m)
stats-stdiono      =0.03     # (m)
stats-stdtrop      =0.3      # (m)
stats-prnaccelh    =10       # (m/s^2)
stats-prnaccelv    =10       # (m/s^2)
stats-prnbias      =0.0001   # (m)
stats-prniono      =0.001    # (m)
stats-prntrop      =0.0001   # (m)
stats-prnpos       =0        # (m)
stats-clkstab      =5e-12    # (s/s)
ant1-postype       =llh      # (0:llh,1:xyz,2:single,3:posfile,4:rinexhead,5:rtcm,6:raw)
ant1-pos1          =90       # (deg|m)
ant1-pos2          =0        # (deg|m)
ant1-pos3          =-6335367.6285 # (m|m)
ant1-anttype       =
ant1-antdele       =0        # (m)
ant1-antdeln       =0        # (m)
ant1-antdelu       =0        # (m)
ant2-postype       =llh      # (0:llh,1:xyz,2:single,3:posfile,4:rinexhead,5:rtcm,6:raw)
ant2-pos1          =49.011242487 # (deg|m)
ant2-pos2          =8.411255298 # (deg|m)
ant2-pos3          =182.895800001032 # (m|m)
ant2-anttype       =
ant2-antdele       =0        # (m)
ant2-antdeln       =0        # (m)
ant2-antdelu       =0.045    # (m)
ant2-maxaveep      =0
ant2-initrst       =off      # (0:off,1:on)
misc-timeinterp    =off      # (0:off,1:on)
misc-sbasatsel     =0        # (0:all)
misc-rnxopt1       =
misc-rnxopt2       =
```

misc-pppopt =

file-satantfile =(path) \igs14_1949.atx

file-rcvantfile =(path) \igs14_1949.atx

file-staposfile =

file-geoidfile =

file-ionofile =(path) \igrg2800.19i

file-dcbfile =(path) \p1c11909.dcb

file-eopfile =(path) \igr20741.erp

file-blqfile =(path) \blq.blq

file-tempdir =

file-geexefile =

file-solstatfile =

file-tracefile =