Politecnico di Torino

Polytechnic University of Turin

Master's Degree in Aerospace Engineering – Space Orientation

Graduation session December 2022

Comparison of the performance achievable with different positioning techniques for lunar PVT estimation

Host structure:
Thales Alenia Space
Via Saccomuro 24, Rome

Candidate:
Francesco Ungaro

Company Tutor:
Massimo Eleuteri
Ileana Milani
Mattia Carosi

Candidate ID:
S280220

Academic Supervisor:
Prof. Fabio Dovis

A.Y. 2021/2022

Abstract

In the last few years, the interest in Moon exploration has grown significantly among several space agencies, both at institutional and commercial levels. Lunar expeditions will offer new opportunities for scientific discovery, economic benefits, and a multitude of other different disciplines, with the goal of the human return to the Moon as the first step toward deep space exploration.
This growing trend has motivated research on various systems which could increase the robustness of the navigation infrastructures and capabilities, aiming to the creation of an autonomous navigation system.

Nonetheless, the Earth-based techniques currently adopted for navigation in cislunar space are not able to cover all the needs for future missions, both in term of service accessibility and performance. Global Navigation Satellite Systems (GNSS) are currently used in space missions, and recent studies have shown relatively good performances also for satellites in GEO and HEO, demonstrating its applicability for a wide range of space missions. Despite this, the number of satellites foreseen in a lunar service is much smaller compared to the Earth-based GNSS, and precise data concerning the position of a user will become of vital importance for positioning over Moon surface.

Because of these reasons, different complementary technologies and studies are ongoing worldwide to define infrastructures able to support Lunar missions in term of navigation services. In particular, it is of interest to determine a useful algorithm for the estimation of Position, Velocity and Time (PVT) considering the limited available resources, in order to improve and optimize the achievable navigation accuracy, toward the realization of an autonomous navigation systems capable of real-time and near real-time absolute positioning.

The purpose of this thesis is to provide an assessment of the performance achievable with different suitable positioning algorithms for Lunar PVT estimation, and compare them with the aim of allowing users to perform a positioning over the Moon surface using a limited number of navigation signals broadcasted by a dedicated Lunar Navigation System.
In particular, starting from the current state of art, the most used techniques of Least Squares (LS) and Extended Kalman Filter (EKF) have been analyzed and their performance and capabilities have been compared over both static and dynamic (landing) user. Accordingly, the Sensor Fusion (SF) technique have been implemented, taking into account the additional measurement of an altimeter. The outcomes of this work aim to combine the best features of each technique in order to define a unique tool that could perform a PVT estimation with the best possible performance and accuracy.

Acknowledgments

I would like to thank my company tutor and supervisors Massimo Eleuteri, Ileana Milani and Mattia Carosi for their fundamental help in the development of this work. Their constant presence, assistance and encouragement during this experience have been essential to the achievement of this goal. I cannot imagine a better support.

I wish to thank my academic supervisor from Politecnico di Torino, prof. Fabio Dovis, for making himself available and to oversee the thesis development.

I am very grateful to *Thales Alenia Space* for this extraordinary experience, giving me the opportunity to deal with a high-level work environment.

To my family
My Mother, My Father, My Brother
My Everything

Table of Content

List of Figures

List of Tables

Acronyms

ALS	Auto-variance Least Squares
AWGN	Additive White Gaussian Noise
BCE	Broadcast Ephemeris
CDF	Cumulative Distribution Function
CNSA	China National Space Administration
DOP	Dilution of Precision
DTE	Direct-to-Earth
ECEF	Earth Centered Earth Fixed
EKF	Extended Kalman Filter
ELFO	Elliptical Lunar Frozen Orbit
ESA	European Space Agency
EUV	Extreme Ultra-Violet
GEO	Geostationary Orbit
GLSDC	Gaussian Least Squares Differential Correction
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HDOP	Horizontal Dilution of Precision
HEO	Highly Elliptical Orbit
IMU	Inertial Measurement Unit
IPP	Ionosphere Pierce Point
ISECG	International Space Exploration Coordination Group
ISS	Information Satellite System
JAXA	Japan Aerospace Exploration Agency
KF	Kalman Filter
LAD	Least Absolute Deviation
LAMBDA	Least-Squares AMBiguity Decorrelation Adjustment
LEM	Lunar Excursion Module
LOS	Line of Sight
LQE	Linear Quadratic Estimation
LS	Least Squares
MM	MiniMax
NASA	National Aeronautics Space Administration
NCA	Nearly Constant Acceleration
NCV	Nearly Constant Velocity
PDOP	Position Dilution of Precision
PPP	Precise Point Positioning
PR	Pseudorange
PSKF	Partial-Update Schmidt Kalman Filter
PVT	Position, Velocity, Time
RINEX	Receiver Independent Exchange Format
RV	Reentry Vehicle
SDE	Stochastic Differential Equations
SF	Sensor Fusion
SNR	Signal-to-Noise Ratio
SP3	Standard Product 3 Orbit Format
SPP	Single Point Positioning

SR	Slant Range
SRF	Square Root Filter
SVD	Singular Value Decomposition
TDOP	Time Dilution of Precision
TOF	Time Of Flight
TOW	Time of Week
UKF	Unscented Kalman Filter
UTC	Universal Time Coordinate
VDOP	Vertical Dilution of Precision
VLSI	Very Large-Scale Integration
WLS	Weighted Least Squares
ZTD	Zenith Total Delay

1. Introduction

1.1. Moon Exploration

In the recent days, several space agencies are renewing the interest for Lunar exploration, which involves both the public and private sectors, and will offer new opportunities for a multitude of disciplines from planetary geology to astronomy and astrobiology. Of course, this renewing interest aim to the human return to the Moon, with the goal, among the others, of creation of lunar bases toward deep space expeditions like Mars.

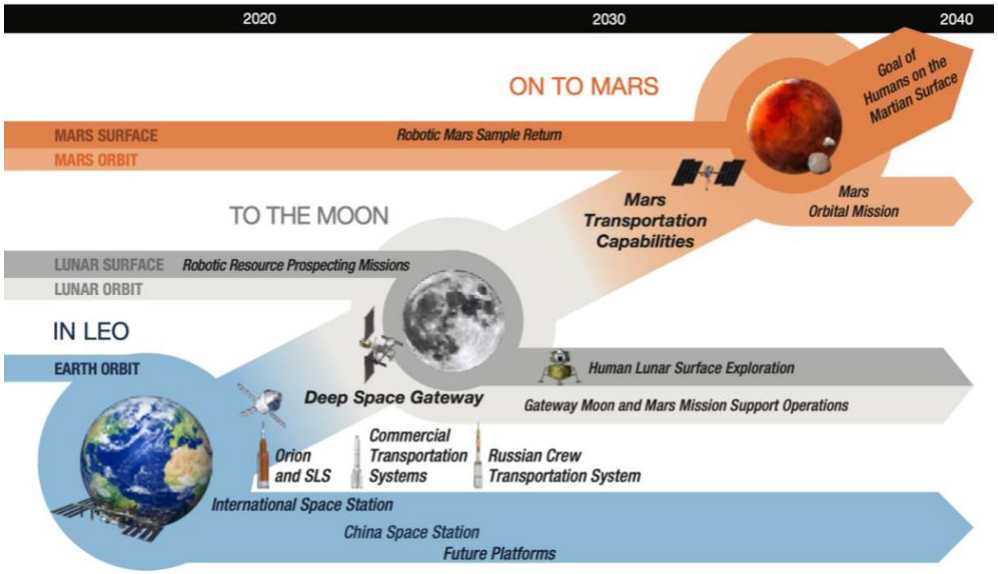

Figure 1.1: The Global Exploration Roadmap by ISECG (Source: [1])

This growing trend in the number of missions to the Moon is creating demand for various research on system which could increase the robustness of navigation architectures and improve their autonomous operation capabilities.

In the past, lunar expeditions have almost entirely relied on measurements from Earth and infrastructures used in terrestrial missions. The benefits of these relay infrastructures were also demonstrated by the recent far-side lunar mission, like for example the landing of the Chinese *Chang'E 4* mission (focused on relaying telemetry to ground rather than providing an independent orbit determination and navigation solution).

Moreover, Global Navigation Satellite Systems (GNSS) are currently used in space missions, not only as a navigation sensor but also as a science instrument. Although their use has been generally limited to orbits below the GNSS constellations, recent studies have shown that GNSS-based navigation for Geostationary Orbit (GEO) and Highly Elliptical Orbit (HEO) missions is feasible and with relatively good performances, demonstrating its applicability to a wide range of space missions. Therefore, these studies show that GNSS signals from Earth can be received at the Moon's altitude, effectively providing support for orbit determination and landing operations on the near side.

However, the Earth-based techniques currently adopted for navigation with satellites in cislunar space are not able to cover all the needs for future exploration, both in terms of service performance (i.e., need to land within 100 m of a predetermined location on the lunar surface) and accessibility. In fact, these technologies alone do not support far-side operations (the South Pole and the far side are not always accessible by Earth-based ground stations) and will not reach the accuracy required by the *Global Exploration Roadmap Critical Technology Needs* (ISECG) (*Figure 1.1*).

These problems are caused by the critical conditions of lunar environment, where the coverage is limited and the signals are weaker. Furthermore, for future missions, precise data concerning the position of rovers on the Moon surface will become of vital importance, and an autonomous navigation system capable of real-time absolute positioning on the Moon will be crucial for the future of the lunar exploration.

The topic has been widely discussed in the literature since the 1970s when Farquhar described how satellites in Earth-Moon libration points could be used to support satellite navigation in cislunar space. Other works have gone further in this argument, assessing different lunar navigation infrastructures based on Earth-Moon Lagrange point orbiters providing one-way Doppler measurements together with Earth GPS signals showing results better than 1 km for positioning and 5 cm/s for velocity in cislunar space [2].

For all these reasons, several space agencies have proposed dedicated systems to address these problems and provide navigation services to future lunar missions.
The Russian satellite maker, Information Satellite Systems (ISS) JSC, proposed a concept that envisions the deployment of a full constellation of 24 satellites around the Moon between 2036 and 2040; in the US, Lockheed Martin has proposed Parsec; JAXA (Japan Aerospace Exploration Agency) has recently launched a study which will consider possible lunar positioning satellite systems; China recently announced that its space agency (CNSA) is planning to set up a satellite constellation around the Moon to provide navigation services. On top of these initiatives, NASA has proposed the LunaNet framework to enable interoperability among different lunar navigation service providers [3].
In this context, the European Space Agency (ESA) has proposed a concept called *Moonlight* that aims to provide navigation services to institutional and commercial lunar missions. The ESA's vision represented by the Moonlight initiative is to foster the creation and development of dedicated lunar navigation services, to be delivered by private partners. These services will support the next generation of institutional and private lunar exploration missions, including enhancing the performance of those missions currently under definition and creating new possibilities [2].

1.2. PVT estimation

The determination of the user PVT is one of the most important issues of satellite-based navigation systems. PVT is an acronym that stands for Position (Latitude, Longitude, Height), Velocity (North, East, Up) and precise Time (in Universal Time Coordinated UTC).
GNSS receivers determine the user position, velocity, and precise time (PVT) by processing the signals broadcasted by satellites. Since the satellites are always in motion, the receiver has to continuously acquire and track the signals from the satellites in view, in order to compute an uninterrupted solution, as desired in most applications. Any navigation solution provided by a GNSS receiver is based on the computation of its distance to a set of satellites: this means measuring the propagation time it takes for an incoming signal transmitted by a satellite at a known location to reach a user receiver at the speed of light, according to local clocks of satellite and receiver. Multiplying this time interval by the speed of light in the vacuum (299 792 458 m/s), the time difference is transformed into a very rough estimate of the emitter-to-receiver distance, called Pseudorange:

$$Pseudorange = PR = c \cdot dt$$

This value is not the true range between satellite and user, since it has to be corrected taking into account a number of phenomena and errors. Furthermore, measuring different pseoduranges from signals broadcast from multiple satellites, the receiver position can be determined more precisely.

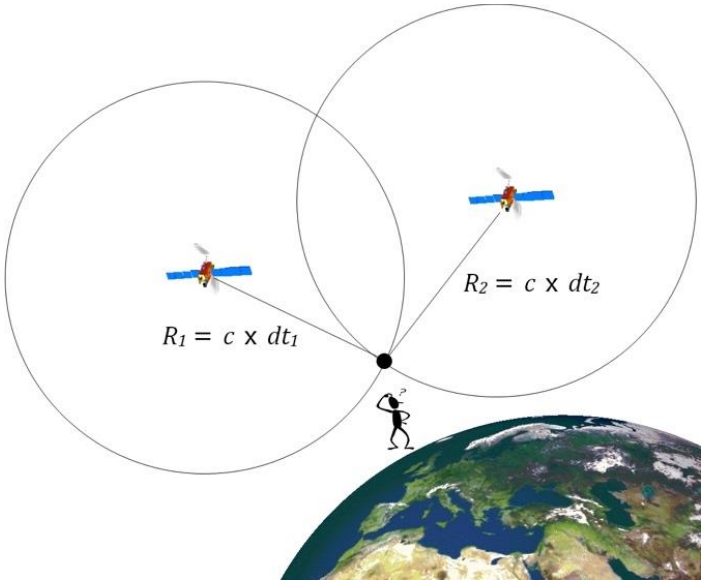

Figure 1.2: General concept of PVT determination (Source: Navipedia [4])

Of course, there are many complications. The major one is that the receiver is using its own clock to tag the received time, often supplied by a very inexpensive crystal oscillator. The speed of light is about $3 \cdot 10^8$ meter per second, thus very small errors in the receiver's clock can cause large range errors. This is solved by including the time bias of the receiver clock in the set of unknowns. Therefore, there are 4 unknowns at each time step where a solution is computed: 3 for position and 1 for time. This is why the minimum number of satellites in view for the determination of a solution is four and also why the receiver can provide such good time: at each timeline, usually once per second, a new estimate of accurate time (GPS time) is generated in every receiver.
As already said, the location of the satellites is needed to determine the PVT. The signal that allows to know location, velocity, and clock state of the satellite is the message data, which provides a series of numbers that are used in a fixed set of equations (a model).
The information is divided into two pieces: the Broadcast Ephemeris (BCE) and the Almanac. The BCE provides information on the satellite position and velocity, which is very accurate and stays that way for a day or so. The information about the bias of the satellites onboard clock is provided too, but, since the atomic clock of the satellites wanders a few nanoseconds per day, the inaccuracy in the clock parameters in the BCE are a major error source. It takes a maximum of 3 minutes tracking to get the ephemeris from a satellite, so this data repeats every 3 minutes, and they cannot be used for generating a solution until the BCE is completely received.
The Almanac is a lower accuracy set of numbers is provided for all the satellites in orbit, in order to help receivers plan satellite tracking and acquire satellites signals. This data cycles more slowly and takes 12.5 minutes to repeat. Usually, all satellites broadcast the same almanac and among other parameters, there are the values needed to convert the GPS Time used by the satellites to Universal Time Coordinated (UTC) [5].

As said before, to estimate the position of a user, the state vector consists of 4 unknowns variables:

$$u = [x\,, y\,, z\,, cdt]$$

The positioning problem is generally stated as:

$$y = h(u) + n$$

where *y* is the measurement vector (that is, the observables obtained from the GNSS signals of a set of m satellites), *h(u)* is the function that relates states with measurements, and *n* models measurement noise. Depending on the models, assumptions, available measurements, and the availability of a priori or externally provided information, many positioning strategies and algorithms can be used [6]. One of these positioning modes is the Single Positioning Mode, in which the vector of unknown states is defined as:

$$u = ({r_r}^T, c \cdot dt_r)^T$$

where r_r is the receiver's antenna position in an Earth-Centered Earth-Fixed (ECEF) coordinate system (in meters), *c* is the speed of light, and dt_r is the receiver clock bias (in seconds).
The measurement vector is defined as:

$$y = \left(Pr^{(1)}, Pr^{(2)}, \dots, Pr^{(m)}\right)^T$$

Where $Pr^{(s)}$ is the pseudorange measurement of the *s* satellite, that can be expressed as:

$$Pr^{(s)} = {\rho_r}^{(s)} + c(dt_r(t_r) - dT^{(s)}(t^{(s)})) + {I_r}^{(s)} + {T_r}^{(s)} + \epsilon_P$$

where:

- $Pr^{(s)}$ is the pseudorange measurement (in meters)
- $\rho_r^{(s)}$ is the true range from the satellite's to the receiver's antenna (in meters)
- *c* is the speed of light (in m/s)
- dt_r is the receiver clock offset from GNSS time (in seconds)
- t_r is the signal reception time (in seconds)
- $dT^{(s)}$ is the satellite clock offset from GNSS time (in seconds)
- $t^{(s)}$ is the signal transmission time (in seconds)
- $I_r^{(s)}$ is the ionospheric delay (in meters)
- $T_r^{(s)}$ is the tropospheric delay (in meters)
- ϵ_P models measurement noise, including satellite orbital errors, receiver's and satellite's instrumental delays, effects of multipath propagation, thermal noise and others (in meters).

Instead, considering Precise Point Positioning mode, the state vector to be estimated is defined as:

$$x = \left({r_r}^T, {v_r}^T, c \cdot dt_r\ ,\ Z_r\ ,\ G_{N_r}\ ,\ G_{E_r}\ ,\ {B_{LC}}^T\right)^T$$

Where:

- Z_r is the Zenith Total Delay (ZTD)
- G_{Nr} and G_{Er} are the north and east components of tropospheric gradients
- $B_{LC} = ({B_{r,LC}}^{(1)}, {B_{r,LC}}^{(2)}, \dots, {B_{r,LC}}^{(m)})$ is the ionosphere-free linear combination (in m).

Besides from these introductive definitions, the PVT estimation can be computed with the implementation of many positioning algorithms, which could follow similar procedures, but demonstrate several variants and differences both in the implementation and the performance.

1.3. Purpose and Development of the work

The main goal of this thesis is to provide an assessment of the performance achievable with different suitable positioning techniques for Lunar PVT estimation, considering both a static user on Moon surface and a dynamic user landing on predetermined point on the Moon.
The final objective is to combine the best features of each algorithm and define a unique tool able to perform PVT estimation with the best possible accuracy.
Chapter 2 presents the current state of the art of the positioning method used for several application on Earth with GNSS constellation, but also introducing some of the studies done for lunar applications. Therefore, the most used and common positioning algorithms are introduced: Least Squares estimation, Kalman Filter and Sensor Fusion technique.
The third chapter focuses on the lunar environment, showing the typical case studies for the PVT estimation on the Moon and pointing out the differences with respect to the terrestrial environment.
In chapter 4, the emphasis is on the implementation of the various techniques covered. First, the constellation considered and the input data are described. Subsequently, the formulations implemented considering each positioning technique are explained. Finally, the script produced based on these formulations are validated considering input data referred to an Earth constellation.
Finally, in chapter 5, the results obtained with the implementation of the different algorithms presented are discussed. The analysis focuses on the outcomes achieved with each one of them in the different scenario considered, so that their performance and capabilities can be evaluated. Afterwards, these results are compared, with the aim of showing the main differences and highlighting the algorithms that provide the best performance, in order to combine the best features of each technique and define a unique tool to perform the PVT estimation with the best possible performance and accuracy.

2. State of Art of GNSS positioning method

2.1. Least Squares

The Least Squares (LS) method is one of the oldest and most widely used statistical tools for linear models and its theoretical properties have been extensively studied in the past [7].
Least Squares is a batch estimation technique used to find a model that closely represents a collection of data and allows for the optimal determination of values or states within a system. This estimation technique can be applied to both linear and nonlinear system and is utilized in many different applications.
Many real-world applications often contain an assortment of sensors that can be used to determine various parameters of interest in the system. These parameters of interest are typically referred to as states and can be anything needing to be tracked in a system, such as the position of a spacecraft or the level of saltwater in an aquarium tank.
Each of the states in a system are stored in a vector known as the state vector, x. The assortment of sensors in a system is used to provide insight into what is actually happening in the system and how the system is changing over time. Each sensor yields a measurement which is stored in a vector known as the measurement vector, $\tilde{y}$. However, these measurements often only provide information about a state indirectly and often require some type of conversion before being compared to the state vector. The measurement model matrix, H, describes this relationship between the measured values and the state values and is used to map the state vector, x, into the measurement vector, $\tilde{y}$, as shown in the equation, which also takes into account any noise, v, in the measurement.

$$\tilde{y} = H \cdot x + v$$

Ideally, when estimating a particular state, the error between the true value and the estimated value of the state should be minimized. However, in a real-world system, the true value of a state is never actually known due to various error sources, such as measurement errors and modeling errors. As a result, linear Least Squares seeks to minimize the residual error, i.e., the error between the actual measurements, $\tilde{y}$, and the measurements predicted from the model/estimated value of the state, $\hat{x}$. In this case, the optimal estimate of the state vector for a particular system is found as [8]:

$$\hat{x} = (H^T H)^{-1} H^T \tilde{y}$$

Despite this, the main focus of the Least Squares method is to find estimation parameters, $\hat{x}$, such that a model $h(x)$ has the best fit on measurements y.
The residuals are defined as the difference between observation data and model function:

$$\vec{r_i} = \vec{y} - \vec{h}(\vec{x})$$

Therefore, the goal is to find x, such that the sum of the squares of the residuals is minimized [9].

$$S = \sum_{i=1}^{n} {r_i}^2$$

2.1.1. Weighted Least Squares

The linear Least Squares solution determines the optimal estimate for each of the estimated state values by minimizing the residual error while weighing each of the measurements equally. However, many applications utilize numerous sensors that have varying performance specifications and uncertainties, so that weighing each measurement equally is not as useful.
A technique known as *Weighted Least Squares (WLS)* adds an appropriate weight to each measurement to account for the uncertainty in each of the measurements. The linear least squares solution then becomes:

$$\hat{x} = (H^T\ W\ H)^{-1}\ H^T\ W\tilde{y}$$

where *W* is a symmetric, positive-definite matrix that contains the appropriate weights for each measurement. While any user-defined weights can be used in *W*, usually this matrix is set equal to the inverse of the measurement covariance matrix, *R*, yields optimal results.

$$\hat{x} = (H^T\ R^{-1}\ H)^{-1} H^T\ R^{-1}\ \tilde{y}$$

Note that in this case, the $(H^T\ R^{-1}\ H)^{-1}$ term in the Weighted Least Squares solution is then equal to the state covariance matrix, $P = (H^T\ R^{-1}\ H)^{-1}$ [8].

2.1.2. Least Squares variants

2.1.2.1. Non-Linear LS

While linear Least Squares can be used in various applications, some systems cannot be described by a linear model. For these nonlinear systems, the linear Least Squares can be extended to a nonlinear Least Squares solution, also known as the *Gaussian Least Squares Differential Correction (GLSDC)*. Rather than directly solving for a closed form solution of the model with respect to the parameters, an iterative approach is taken by linearization of *h* around an initial value x_0, in order to estimate an updated parameter $x_{k+1} = x_k + \Delta x$.
The nonlinear Least Squares estimation process uses a model of the form:

$$\tilde{y} = h(x)$$

where *h(x)* represents the equations of a nonlinear system.
An optimal estimate for a nonlinear system can then be found by iterating the nonlinear least squares solution, until convergence is achieved observing the values in Δx:

$$\hat{x}_{k+1} = \hat{x}_k + ({H_k}^T\ H_k)^{-1}\ {H_k}^T\ (\tilde{y} - h(\hat{x}_k))$$

Where the *H* matrix is known as the Jacobian matrix, defined with the partial derivatives of the modeled measurements with respect to the estimation parameters.

$$H_k = \frac{\delta h(x)}{\delta \hat{x}_k}$$

Weighted versions of this calculation follow the same formulation as the linear case: the weight matrix, *W* ideally is the inverted covariance matrix of the observations Q_z^{-1}, but in reality, the inverse of estimated variances is used, as the actual observation covariances are not available during real-time navigation.
Though this iterative process requires more computation than the linear Least Squares estimation process, nonlinear Least Squares provide the advantage of optimizing a wide range of real-world systems [8].

2.1.2.2. Least Absolute Deviation (LAD)

Despite its many superior properties, the Least Squares estimate can be sensitive to outliers and, therefore, non-robust. Its performance, in terms of accuracy and statistical inferences, may be compromised when the errors are large and heterogeneous. In fact, the traditional LS estimator is rather sensitive to large noises, resulting in large estimation errors, due to the Gaussian error assumption. When outliers occur, the Least Squares estimation scheme may not be a good choice because the LS estimator minimizes the mean squared error of the observations. In order to perform robust positioning estimation, other criteria may be employed. The common alternatives might be L_1 *(Least Absolute Deviation, LAD)* or L_∞ *(MiniMax, MM)* estimators.

These are known as two of the most robust estimators, and *LAD* in particular is also known to be able to produce approximately the maximum-likelihood estimation. As compared to the Least Squares, the *LAD* method is less sensitive to outliers and produces more robust estimates. In cases when the maximum-likelihood estimator is obtained by minimizing the mean absolute deviation, rather than the mean square deviation, it can perform more effective estimation. Even if the desired signals are corrupted by unknown errors, it tends to be impervious to unexpected large errors. Due to developments in theoretical and computational aspects, the *LAD* method has become increasingly popular, with many applications in econometrics and biomedical studies, among many others.

Moreover, different studies demonstrate that *LAD* approach shows a much better multipath resistance capability compared to the Least Squares method. However, *LAD* approach possesses more benefits when more satellites are in view and/or less satellite signals are corrupted by multipath. This is due to the fact that a larger number of multipath-corrupted signals will mislead the correct information, resulting in performance degradation [7].

2.1.2.3. Autocovariance Least Squares (ALS)

This algorithm is formulated to take into account the time-varying system dynamics and measurement matrices. The intrinsic feature of this algorithm is the account for the time correlation between measurement residuals, which exists due to the ambiguity of the measurement noise-covariance matrix, that is not known a priori.

Three approaches are commonly used for setting the covariance matrix of the GPS measurements:

- The first and the most common approach is to set the measurement covariance matrix as a predefined equally weighted diagonal matrix.
- Another approach is to set the GPS measurement covariance based on the elevation angle of the satellite.
- The third approach is to weight the GPS measurements based on the signal-to-noise ratio (SNR) or the carrier-to-noise power density ratio C/N0.

The last two methods, while giving a better gauge of the measurement quality, do not directly estimate the measurement covariance from the sampled measurements. For the second approach, neither the receiver quality nor adverse environments such as interference or jamming will affect the chosen measurement weighting. For the third approach, GPS manufacturers are not obliged to provide these values. Added to this fact is the lack of standards between GPS manufacturers in providing the SNR and the C/N0. In addition, it is unclear how the SNR or the C/N0 will be able to differentiate between the measurement and its noise without the feedback of the receiver dynamics measured by the inertial measurement unit (IMU). Therefore, it is seen that proposing a method to estimate the GPS measurement noise-covariance matrix from the sampled GPS measurements with the aid of the IMU-measured receiver dynamics is essential for high-integrity operation of GPS- dependent systems [10].

2.2. Kalman Filter

The Kalman Filter, also known as linear quadratic estimation (LQE), is one of the most important and common estimation algorithms. The filter is named after Rudolf E. Kálmán, who published his famous paper describing a recursive solution to the discrete-data linear filtering problem in 1960.
This algorithm uses a series of measurements observed over time, including statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe [4].

As it will be discussed in detail in chapter 4, the Kalman Filter works by a two-phase process. The first one is the prediction phase, in which the filter produces estimates of the current state variables, along with their uncertainties. Once the next measurement (including its errors) is observed, the estimates are updated using a weighted average, with more weight needed to obtain greater certainty. The algorithm is recursive, and it can operate in real time, using only present input measurements and the state calculated previously with its uncertainty, so that no additional past information is required.
There are different variants of the Kalman Filter: one of the most important is the *Extended Kalman Filter (EKF)*, which was developed for nonlinear applications such as navigation. Basically, it makes the nonlinear system linear around the current estimate of the Kalman Filter. This technique will be discussed more in detail also in chapter 4.
Another modified version is the *Unscented Kalman Filter (UKF)*, which is much more computationally intensive, but more accurate because it does not attempt the linearization of a nonlinear system. The main idea of this filter is that it is easier to perform a nonlinear transformation on a single point than a probability function and that it is possible to find a group of single points in state space whose sample probability density function approximates the true probability density function of the state vector. During the time update portion of the filter, twice as many sigma points as the length of the measurement vector are chosen, so that all together they have the same mean and covariance of x. The known nonlinear function is applied to each sigma point and the resulting transformed vectors are used to get a good estimate of the true mean and covariance. Hence, the procedure involves the calculation of the sigma points, and the time updated state vector and covariance matrix based on the sigma points. The subsequent steps are the calculation of the estimated measurements from the sigma points, the measurement covariance matrix, the measurement-state cross-covariance matrix, and finally the definition of the Kalman gain and the post measurement outputs [11].

2.2.1. Kalman Filter Issues

Two primary concerns emerge from the use of the Kalman Filter: the numerical precision of the filter and the filter's robustness for nonlinear systems or measurements. These problems arise since the error covariance matrix is updated directly and the covariance measurement update equation involves a subtraction of two positive definite matrices, which when performed with finite precision can represent potential numerical problems. These numerical inconveniences may lead to loss of accuracy and to the violation of the positive definiteness of the covariance matrix, principally for ill-conditioned problems, often leading also in divergence [12].
In addition, Kalman gain is estimated in the gain loop but acts as error correction feedback in the estimation loop. As long as the gain is accurate, this feedback into the estimation loop should correct for errors in the state estimate that are caused from roundoff, noise and a priori estimation errors. However, no such feedback exists in the gain loop, so that any errors, such as from computer roundoff, can accumulate and go unchecked in the computation of the state variance-covariance. Furthermore, there are about twice as many computer roundoff operations in the gain loop, compared to the estimation loop of the Kalman filter.

The main procedure used to overcome the issues of the Kalman Filter is the factorization of the covariance matrix, an operation almost as old as the filter itself. There are several factorization methods to stabilize the filter [13], factoring the variance-covariance matrix of states into:

- Products
    - Triangularization (QR decomposition)
        - Givens rotations
        - Householder transformation
    - Gram-Schmidt orthonormalization
- Square Root and UD filters
    - Carlson-Schmidt algorithm
        - Cholesky factors
    - Bierman-Thornton algorithm
        - Modified Cholesky factors
- Others

2.2.2. Kalman Filter variants

In this section, some of the variants mentioned before to overcome Kalman Filter issues are presented, with a brief introduction of the main points of interest, without dwelling on them.

2.2.2.1. Square Root Filters

In general, square root filters are more numerically stable than the conventional Kalman filter. First, the condition number for the square root of a covariance matrix is the square root of the condition number of the covariance matrix. Hence, these kinds of filters benefit both the numerical stability inherent in the square root filter and the robustness of the partial update while operating directly on the square root representation of the uncertainty. In fact, a low condition number is always desirable, mainly for the cases where the computer word length is limited (as in embedded systems), or when the filtering problem is poorly conditioned. Although these types of formulations are numerically more robust, it is at the cost of increasing the computational effort. Nevertheless, the number of extra computations can still be reasonable which allows the filter to be used in many applications.
The SR filter is mainly a covariance reformulation of the standard Kalman equations, and thus it is still a linear filter. Similar to what is done with a traditional filter, the square root KF can be applied in nonlinear systems through a linearized model. That is, the square root formulation does not enhance a filter's ability in addressing nonlinearity, it simply improves numerical conditioning [12].
In the estimation field, square root filtering refers to utilize a square root factorized representation of the error covariance matrix for purposes of propagation and correction of the estimation error. The goal of reformulating filters using such "square roots" or factorizations, is to increase the precision of the filter itself. By operating on the square root of the error covariance, the filter lowers the condition number of the uncertainty matrix, which is then less prone to numerical issues because fewer significant figures are required during the arithmetic operations.
The definition for the square root of a matrix is based on the idea of finding a matrix S that satisfies $P = SS^T$, where S will be referred to as the square root of the error covariance matrix P. Specifically, S is a lower triangular matrix, and S^T its transposed. Importantly, the product SS^T is naturally symmetric and positive semidefinite, regardless of the value of the lower triangular matrix S.
This means that numerical difficulties that could cause the covariance matrix P to become nonsymmetric or singular, cannot affect the product SS^T, thus preserving the theoretical properties of the covariance matrix P (within the machine precision). Also, as for scalars, the square root S is not unique, so there may be several solutions. One very well-known method to compute the matrix S is the Cholesky decomposition, which requires that the matrix to be factorized is positive definite and symmetric (this holds for P) and directly outputs the matrix S that will be triangular.

2.2.2.2. Potter Square Root Filters

The first square root filter development is due to Potter who developed in 1963 an algorithm for the limited case of uncorrelated scalar observations with no process noise. This filter was used in the Lunar Excursion Module (LEM) for the Apollo Program [14].
The main driver at the time was numerical precision, as computer words were only 8 bits long. Replacing the covariance by a square root matrix S, such as $P = SS^T$, reduces the spread of the elements of P bringing them closer to 1, doubling the numerical precision of the stored variable.
At the time, the Apollo Kalman filter was designed without any process noise, because computations required for inclusion of the process noise required too many computations. A very desirable by-product of this factorization is that the symmetry and semi-positive definiteness of the covariance are insured by construction and does not need to be checked or enforced to correct for numerical and round-off errors. It should be noted that this Apollo factorization was not a triangular square root matrix [15].

2.2.2.3. Bierman-Thornton UD Factorization

This is another more elegant formulation where the covariance matrix P is replaced by two factors: a diagonal matrix D and an upper triangular matrix U with ones on the main diagonal, such that

$$P = UDU^T \rightarrow \begin{bmatrix} p_{11} & p_{21} & p_{31} \\ p_{12} & p_{22} & p_{32} \\ p_{13} & p_{23} & p_{33} \end{bmatrix} = \begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & 0 \\ 0 & 0 & d_{33} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix}^T$$

The state variance-covariance matrix is never explicitly computed since the UDU^T factors are always propagated in each computational step through the filter instead of the variance-covariance matrix itself. More importantly, this factorization improves the computational stability and efficiency of large navigation filters, especially in situations prone to roundoff error, large a priori errors and asymmetry of the state variance-covariance matrix, which may all lead to filter divergence. This is possible because UDU factorization ensures symmetry of the covariance matrix by construction, and it requires a trivial check and correction to ensure semi-positive definiteness (it suffices to enforce that the diagonal elements of D remain non-negative). The UDU is classified as a square root filter, but technically, the formulation is free from square root operations, making it computationally cheaper, and for these reasons it has endured as one of the preferred practical implementations of Kalman filters in aerospace applications [15].
The UD filter is only capable of incorporating scalar observations, meaning that the observations must be uncorrelated for the solution to be rigorous. GNSS double-differenced phase observations are inherently mathematically correlated, which suggests why a UD carrier-phase float filter has not been reported in previous literature. This limitation can be overcome by diagonalizing the observation covariance matrix, thus decorrelating the components of the matrix. Although not initially intuitive, forming arbitrary linear combinations of the observations will not affect the integer nature of the estimated ambiguities provided we continue to estimate ambiguities corresponding to the original double-differences [13].

2.2.2.4. Partial-Update Schmidt Kalman Filter

The Partial-Update Schmidt Kalman Filter (PSKF or Partial-Update Filter for short) is a recent technique that is useful in accommodating measurement updates in nonlinear systems with mildly observable states.
The Partial-Update Schmidt Kalman filter is a straight-forward modification of the Extended Kalman Filter that effectively increases the range of uncertainties and associated nonlinearities that a filter can tolerate (compared to the EKF or Schmidt filter) while still producing accurate state estimates with appropriate co-variance bounds.

This form of the Kalman filter inherits the benefits of the Partial-Update formulation and combines them with the numerical robustness of the square root form. The result is a filter of higher numerical precision and increased tolerance to nonlinearities and uncertainty level with minimal additional computational burden [12].

2.2.2.5. Least-Squares AMBiguity Decorrelation Adjustment

Since *UDU* Bierman filter provides only a float solution, to solve for the integer ambiguities other algorithms will need to be used, such as *LAMBDA* (*Least-Squares AMBiguity Decorrelation Adjustment*). The resulting unit triangular matrix from the square root filter can be directly used into *LAMBDA* since the first step in this procedure is to decorrelate the double differenced ambiguities before the actual integer estimate of ambiguities is done. The decorrelation requires that the variance-covariance matrix of the ambiguity states be broken down into LDL^T (or equivalently UDU^T factors, which the Bierman-Thornton algorithm does). The overall process of going from a float solution to a fixed solution is resultantly more efficient. Furthermore, factorization of the variance-covariance matrix of the ambiguity states is only done once and the *UD* factors are propagated through the filter. The factorization only happens again if the ambiguity states change. This contrasts with *LAMBDA*, where factorization of the ambiguity states happens at every epoch regardless of changes in the ambiguity states or not. This further exemplifies the computational benefit and efficiency of the Bierman-Thornton method [13].

2.2.2.6. V-A Square Root Algorithms

These are new discrete-time (continuous/discrete and discrete/discrete) square root algorithms, which utilizes the spectral decomposition of the covariance matrix: *V* is the matrix whose columns are the eigenvectors of the covariance and *A* is the diagonal matrix of its eigenvalues. The new algorithms employ singular value decomposition (*SVD*), for which there exist today efficient and stable algorithms.

From a computational viewpoint, the new algorithms are more complex than other *SR* procedures that exist today, because of the reliance upon the *SVD* technique (as opposed to more efficient orthogonal transformations, on which other *SR* algorithms are based). Nevertheless, the new algorithms may be of great importance in certain applications, e.g., where loss of accuracy due to harsh numeric is expected, or where continuous monitoring of the eigen factors is necessary in order to reveal singularities as they occur and to identify those state subsets that are nearly dependent. It is believed that as the *SVD* is becoming today a tool of primary importance in control theory, further research will eventually lead to the development of new *SVD* algorithms of higher efficiency, to the benefit of the new *V-A* filters. Moreover, with the rapid emergence of Very Large-Scale Integration (VLSI), new parallel computing structures have been introduced for efficient, real-time implementation of matrix arithmetic algorithms such as Cholesky decomposition, eigenvalue decomposition etc.

The new update algorithm is free of explicit equations, a fact that may be advantageous in certain implementations, and it was shown to be numerically stable. The stability stems from the fact that it is based on the orthogonal *Householder* and *Givens* transformations, which are famous for their numerical stability and accuracy.

2.2.2.7. Hybrid type filters

These kinds of filters utilize alternately the covariance mode (in the time update stage) and the information mode (in the measurement update stage). Thus, because of the operation in both modes, the new filters possess the advantages of the covariance and information filters. These advantages are the ability to cope with the case of infinite initial covariance (no initial information), the efficiency of the covariance formulation in processing time updates and the efficiency of the information formulation in processing measurement updates. Moreover, because of the duality between the discrete time update of the covariance factors and the discrete measurement update of the information

matrix factors, the fact that the *V-A* filter operates in both modes implies algorithmic equivalence between the procedures used in the two stages of the filter. This equivalence introduces a saving factor in the implementation of the filter, because both stages use the same algorithm. This fact is also valuable for its simplification of the error analyses of specific implementations.
The conventional Kalman Filter cannot handle this relatively large initial covariance because of an algorithmic singularity, while using the *V-A SR* filter such initialization problems are not encountered, since the covariance factors are updated independently of the gain, and the gain itself can, this time, be computed using the a posteriori factors [16].

2.3. Sensor Fusion

The Sensor Fusion technique is a method that combines data from different sources of information in order to achieve better performance than can be achieved when each source of information is used alone.
The design of systems based on sensor fusion methods requires the availability of complementary sensors in order that the disadvantages of each sensor are overcome by the advantages of the others.
There are different applications of this technique, like air navigation, with several aerial vehicles which use various sensors usually noisy and biased, so that their combination can give optimized results [17]. Another interesting application of sensor fusion methods is motion tracking, since several sensor technologies are available, but none of them taken alone can give the best performances. This is true especially when motion is to be tracked without restrictions in space and time, and cost and compliance issues tend to restrict the range of potential candidates for applications like human motion tracking in biomedicine and healthcare [18].

The implementation of this work will consider the application of the Sensor Fusion technique combining the performances of an Extended Kalman Filter and the additional measurement given by an altimeter. The functioning of a laser altimetry (or laser ranging) works on the basic time-of-flight (TOF) principle. A pulse of laser energy is emitted, it reflects off a target surface, and the receiver (which could be the same that emitted the laser) detects the reflected energy. The time between pulse emission (start) and pulse reception (stop) provides a measure of the target distance, or range (R), based on the speed of light (c), through the simple relation [19]:

$$R = {}^{c\,\Delta t}/_{2}$$

3. PVT determination in Lunar Environment

3.1. Typical Use Cases

3.1.1. User orbiting around the Moon

The first typical use case in lunar environment is of course the determination of the PVT of a user that is orbiting around the Moon, following a specific trajectory previously determined. Nevertheless, in this work this case will not be taken into consideration, since it is not of interest for the analysis done and the algorithms developed.

3.1.2. Static User

The first scenario that will be analyzed in this study consider a static user, situated in a specific position on the lunar surface: this point is considered to be the South Pole of the Moon. As it will be deepened in the next chapter, the PVT estimation is achieved through the signals of four satellites in view, which is the least number of satellites needed for the determination of the user position. As it will be discussed, all four satellites need to be in view of the user, in order to allow to determine its position. Hence, if following their orbits around the Moon, one or more satellites are no longer in visibility of the user, the PVT estimation cannot be computed anymore, given the absence of sufficient inputs for the implementation of the positioning algorithms.

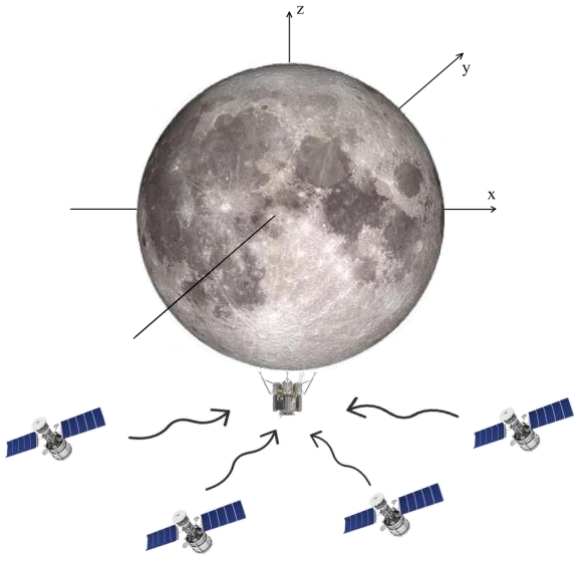

Figure 3.1: Static User configuration

The simulated configuration is depicted in *Figure 3.1*, but it can also be noticed the reference system considered for the analysis: it is the equivalent of the ECEF system on Earth, since its origin is in the center of the Moon and the *z* axis is directed toward the North Pole.

3.1.3. Dynamic User

The second scenario that will be discussed consider a user no longer on the lunar surface, but it is now in motion and following a defined trajectory. This trajectory includes a first phase in which the user is orbiting around the Moon, but then it starts approaching the lunar surface and finally a descending phase to land on a predetermined point, specifically the South Pole. Even in this scenario, the determination of the PVT of the user over time is provided through four satellites in view, which also follow a specific orbit around the Moon.
An approximation of the trajectory followed by the user is depicted in *Figure 3.2*. As for the static user case, also for this scenario the reference system considered has its origin in the center of the Moon, as shown in the figure.

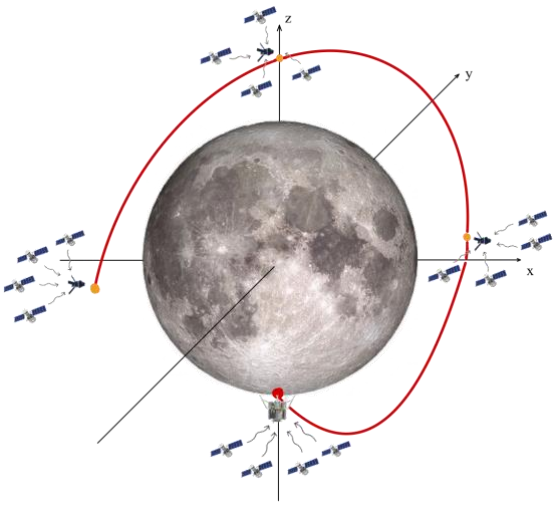

Figure 3.2: Dynamic User configuration

Landing on the Moon has been successfully performed since the initial phase of lunar exploration, both with human and robotic missions. However, recent failures have shown that landing on the Earth's natural satellite is actually not an easy task.
Despite this, recent studies on NASA's Jet Propulsion Laboratory lander vision system have demonstrated landing accuracies down to 40 meters and the also the latest landing on Mars has proven the level of reliability of this system [2].
This means that in theory the requirements expressed as part of the *Global Exploration Roadmap Critical Technology Needs* (ISECG) could be met with current state-of-the-art visual-based navigation technology such as the one used for the recent Mars landing. Though, a detailed assessment of the specific features and image quality of the lunar landing areas would still need to be performed. These also means a relatively expensive cost and heavy equipment dedicated to the landing phase, which is only partially reusable after touch down [2].

3.2. Differences with respect to Earth environment

3.2.1. Earth and Moon Applications

3.2.1.1. Earth

Nowadays, Kalman Filter technique is used in numerous and various applications on Earth, such as target tracking (Radar), location and navigation systems, control systems and computer graphics [20]. For example, calibration, alignment, and error correction of complex inertial navigation systems are

done with the implementation of Kalman filtering. Real-time online applications can include missile defense, estimation, and prediction of reentry vehicle (RV) position, while offline applications include estimation and correction of radar errors such as azimuth bias, elevation bias, and survey (base location) errors. Finally, space applications of these techniques comprehend estimation of the trajectories of thousands of Earth satellites and space debris, as well as augmentation systems and also every GNSS receiver uses an EKF to estimate its own position and velocity, and to synchronize the receiver clock with GPS time [21].

3.2.1.2. Moon

In lunar environment, the most important and known expedition that envisaged the implementation of the Kalman Filter technique is of course the APOLLO program. Nonetheless, other missions have demonstrated the likelihood of lunar positioning and also landing, as the Chinese Chang'E 4 expedition [22]. However, this mission (as almost every lunar mission in the past) has almost entirely relied on direct-to-Earth (DTE) ranging radiometric measurements for navigation, rather than providing an independent orbit determination and navigation solution.
Different studies have already shown that GNSS signals from Earth can effectively provide support for orbit determination and landing operations on Moon, but this technology alone does not support far-side operations and will not reach the accuracy required by the *Global Exploration Roadmap Critical Technology Needs* (ISECG). This topic has been widely discussed in the literature since the 1970s, with various papers describing how satellites in Earth-Moon Lagrangian points could be used to support satellite navigation in cislunar space.
To address these problems and provide navigation services to future lunar missions, in recent years several space agencies have proposed dedicated systems.
In particular, the Russian satellite maker, Information Satellite Systems (ISS) JSC, proposed a full constellation of 24 satellites around the Moon, while NASA has proposed LunaNET.
In this context, the European Space Agency (ESA) has proposed a concept called *Moonlight* that aims to provide communication and navigation services that will support the next generation of institutional and commercial lunar exploration missions, including enhancing the performance of those missions currently under definition and creating new possibilities [2].

3.2.2. Adaptation of the algorithms

In order to implement the positioning techniques, the differences between Earth and Moon must be considered, so that the algorithms need to be modified depending on the application environment. In this section the differences in the implementation will be described, while in the next chapter the simulation environment considered for the lunar application will be discussed more in-depth, together with formulations and implementation of the different positioning algorithms.

3.2.2.1. Earth

Considering Earth environment, the first step of the implementation is the check on the satellites in view at each observational epoch. In fact, as already discussed, in order to compute the user position, a minimum of four satellites in view is needed, otherwise the solution cannot be determined by the positioning algorithms.
This check is based on consideration on the elevation angle of the satellites: taking into account the line-of-sight vectors between the satellites and the user, the elevation angle of each satellite will be calculated. This will lead to the exclusion of the satellites which elevation angle is lower than 5 degrees, since they could invalidate to solution due to their relative position with respect to the user.

Given r_{sat} and r_{user} the geocentric positions of the satellite and the user, respectively, the Line-Of-Sight vector is defined as:

$$\widehat{LOS} = \frac{r_{sat} - r_{user}}{\|r_{sat} - r_{user}\|}$$

From the given coordinates of the user, latitude φ and longitude λ can be determined and used for the calculation of the unit vector $\hat{u}$ of the ENU reference system, through the rotation matrix from ECEF to ENU [*Appendix B.2 of ESA GNSS Book* [23]]:

$$\hat{u} = [\cos\lambda\cos\varphi\,, \sin\lambda\cos\varphi\,, \sin\varphi]$$

So that, as shown in *Figure 3.3*, the elevation angle is computed as:

$$E = \text{asin}(\widehat{LOS} \cdot \hat{u})$$

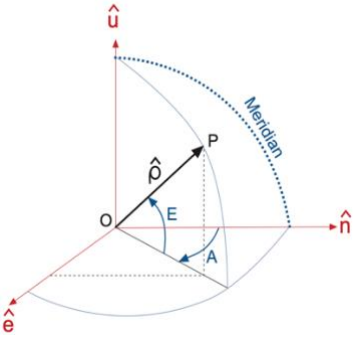

Figure 3.3: ENU coordinates and Elevation angle (Source: ESA GNSS Book [23])

Given this value for each satellite, the ones that result in elevation lower than 5 degrees will not be considered in the algorithm implementation.

Once this check has been done, the measurements considered in Earth application are the pseudoranges of the satellites in view. Given the coordinates and clock of each satellite $[x_i, y_i, z_i, t_i]$ and the coordinates of the user $[x_u, y_u, z_u]$, the pseudorange expression is:

$$Pseudorange = \rho_i = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} - c \cdot t_i$$

This measurement needs to be corrected taking into account various sources of errors, which can be referred either to the satellites or the receiver.
Considering satellites errors, each one will show:

- <u>*Satellite orbital error*</u>: this is given by the ephemeris parameters contained in the navigation message. In fact, these ephemeris data are broadcasted by the satellites with specific time intervals, so that the receiver calculates the satellite position through an estimation using a curve fit to predict the satellite orbit, which leads to residual errors relative to the actual orbit. It is estimated that they can result in up to ± 2.5 meters of position error [24]. In this work, this error will be modeled as an Additive White Gaussian Noise (AWGN) with zero mean and standard deviation defined, which is considered to be 10 meters [25].

- <u>*Satellite clock error*</u>: in the downlink data broadcasted by each satellite, they also provide the user and estimate of the offset between their clock and the receiver one. To obtain a more accurate position, the receiver needs to compensate this error, since this drift can lead to dramatic range errors in receiver measurements. In fact, clock errors in seconds are then multiplied by the speed of light, so that for example 20 nanoseconds of error results in 6 meters of position error. Satellite clock error model will be based on the theory of Stochastic Differential Equations (SDE):

$$\sigma_{clk} = \sigma_0 + \sqrt{{\sigma_1}^2 \cdot t \cdot \Delta + \frac{{\sigma_2}^2 \cdot (t \cdot \Delta)^3}{3}}$$

  Where the parameters are defined considering an application with Caesium HP clock [25], showed in *Table 3.1*.

Table 3.1: Clock error parameters

t = observational epoch in seconds (update interval: this value is set to zero every 12 hours)		*Δ = time difference between two epochs*
$\sigma_0 = 10^{-10}$	$\sigma_1 = 9.486 \cdot 10^{-12}$	$\sigma_2 = 1.643 \cdot 10^{-17}$

- *Sagnac Effect*: this consists of a relativistic error caused by the rotation of the Earth during the time of signal transmission between the satellite and the receiver. Ephemeris parameters provide information about the satellites position expressed in Earth-Centered Earth-Fixed (ECEF) frame at signal transmission time, but during signal transit time the Earth rotates, so that the signal reception time is not the same of the transmission one (*Figure 3.4*).

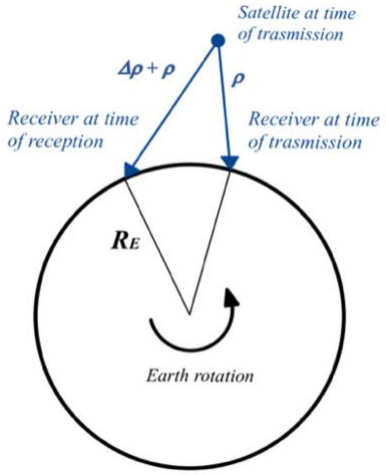

Figure 3.4: Sagnac effect

For each satellite, defined the observational epoch in seconds (T_{obs}) and the pseudorange corresponding to that epoch (ρ_i), given the speed of light c, the implementation of the Sagnac effect is given by the following procedure:

1. $T_S = T_{obs} - \frac{\rho_i}{c}$
2. $dtr = \frac{r \cdot v}{c^2}$, where the numerator is the scalar product between the position vector of the satellites $r=[x_i,y_i,z_i]$ and the velocity vector $v=[v_{xi},v_{yi},v_{zi}]$ determined by derivation.
3. $d_{ROT} = T_{obs} - (T_S - dtr)$
4. $\varphi = \omega_E \cdot d_{ROT}$, where ω_E is the Earth rotation velocity
5. $M_{SAGNAC} = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$
6. $\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}_{NEW} = M_{SAGNAC} \cdot \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$, which are the coordinates of the satellite corrected with the implementation of the Sagnac effect.

Regarding the user errors, the main error to consider is the *Multipath error*. Usually, the received signals is in direct line-of-sight (LOS) between satellite and receiver, but it arrives with one or more delayed echoes due to the reflection of the original signals depending on the surrounding environment and the relative satellite-receiver motion (*Figure 3.5*). On Earth, this delay can be caused by buildings or other objects, but also natural elements can cause this reflection, so that this effect has an impact also on lunar applications. These multipath errors can cause the receiver to calculate an incorrect position, up to pseudorange errors of 100 meters in the most severe conditions [26]. The implementation of the multipath error is obtained from Brenner's Multipath Model [25]:

$$\sigma_{MP} = a + b \cdot e^{c \cdot EL}$$

Where:

- $a = 0.1633$
- $b = 1.1846$
- $c = -0.0511$
- EL = *elevation angle*

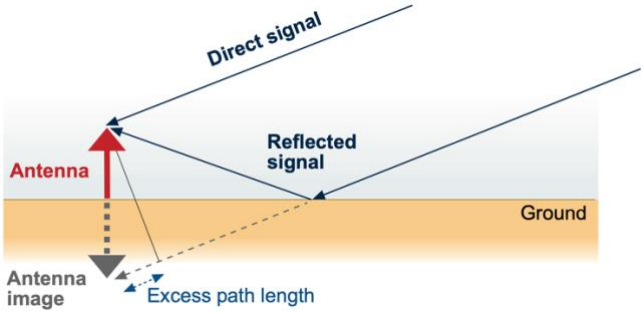

Figure 3.5: Multipath error (Source: ESA GNSS Book [23])

Moreover, another noise that must be taken into account is the *Receiver error*: this noise is a complex error generated at the receiver's side while measuring satellite signals. It covers different spectrum of noise types, including microwave radiations and it is present due to system components such as antennas, cables or amplifiers [26]. For the applications of this work, the receiver error will be modelled in the same way as previously defined for the satellite orbital error, which is an AWGN with zero mean and standard deviation of 10 meters.

Finally, in Earth environment one of the most important errors is given by the effect of the atmosphere. Among these atmospheric effects, the main correction to implement is the one that concerns the *Ionospheric effect*. The ionosphere is a dispersive medium located primarily in the region of the atmosphere between about 60 km and 1,000 km above the Earth's surface [27]. Within this region, as its name implies, there is a partially ionized medium due to Extreme UltraViolet (EUV) rays in the solar radiation and the incidence of charged particles. The propagation speed of GNSS electromagnetic signals in the ionosphere depends on its electron density, which is typically driven by two main processes. During the day, the Sun's radiation ionizes neutral atoms to produce free electrons and ions. During the night, the recombination process prevails, where free electrons are recombined with ions to produce neutral particles, which leads to a reduction in the electron density [23]. In fact, these free electrons influence electromagnetic wave propagation, including GNSS satellite signal broadcasts.

In order to correct this ionospheric delay, specific models need to be implemented. The one used in this paper is known as Klobuchar Model, initially developed for GPS, and usually used because of the simple structure and the convenience in calculations. This model is based on empirical approach, and it is estimated to reduce the ionospheric errors by about 50% worldwide [23]. As shown in *Figure 3.6*, the assumption assumes that the vertical ionospheric delay can be approximated by half a cosine function of the local time during daytime, whose amplitude and period are given as a function of the eight parameters broadcast in the GPS navigation message, and by a constant level during night-time (about 5 ns) [27].

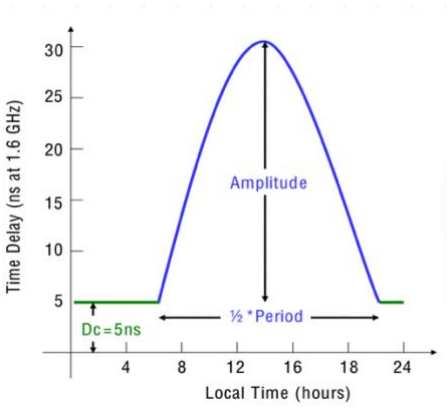

Figure 3.6: Ionospheric delay approximation (Source: ESA GNSS [23])

The Klobuchar model employs geomagnetic latitude on Ionosphere Pierce Point (IPP). As shown in *Figure 3.7*, it is assumed that the electron content is concentrated in a thin layer at 350 km in height (GPS), so that IPP is defined as the point where the line of sight that connects the GPS satellite to the signal reception point (slant delay in red in the figure) meets the single vertical layer (in blue). Since the change of IPP is affected by seasons and the geomagnetic field and closely related with the sunspot activity, it is varied following the solar activity in the period of 11 years [28].

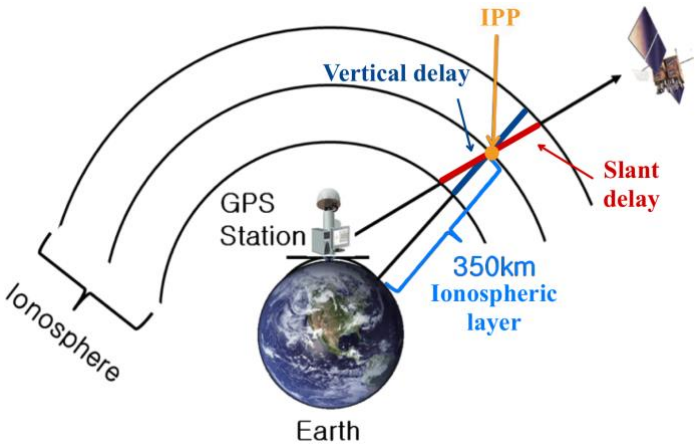

Figure 3.7: Ionospheric Pierce Point (Source: ESA GNSS Book [23])

To implement this correction, the input data for calculation of Klobuchar ionospheric delay are:

- Elevation angle *EL* and Azimuth AZ of the observed satellite (radians)
- Geodetic latitude φ and longitude λ (radians)
- Klobuchar coefficients α , β
- GPS time (seconds)

Klobuchar coefficients, as already said, can be found in the navigation data, while GPS time is determined from the observational epoch, using the classic conversion from Gregorian date (discussed more in depth in *appendix A.1*). As for the other terms, one of the possible solutions is the application of the Bowring iterative method *(Appendix C and* [29]*)*, which allows to determine latitude, longitude and altitude given the spatial coordinates, and then determine *EL* and *AZ* using trigonometric equations. Nevertheless, the solution implemented in this paper exploits the potential of MATLAB and its functions: given the coordinates of the user in the ECEF reference system, the function *ecef2lla* allows to determine the geodetic coordinates Latitude, Longitude and Altitude; moreover, the function *ecef2aer* coverts point locations from geocentric ECEF coordinates to local spherical coordinates Elevation and Azimuth given the coordinates of the satellite, latitude and longitude of the user and the reference ellipsoid. Now that all the input data are defined, the Klobuchar procedure can be applied, as explained in *ESA GNSS Book Vol. I* [23]:

1. Earth-centered angle: $$\psi = \frac{\pi}{2} - EL - \arcsin\left(\frac{R_E}{R_E+h}\cos EL\right)$$
   Where R_E *=6378 km* and *h=350 km*

2. Latitude of the IPP: $$\phi_{IPP} = \arcsin(\sin\varphi \sin\psi + \cos\varphi \sin\psi \cos AZ)$$

3. Longitude of the IPP: $$\lambda_{IPP} = \lambda + \frac{\psi \sin AZ}{\cos\phi_{IPP}}$$

4. Geomagnetic latitude of the IPP:
   given coordinates of geomagnetic pole $\phi_p = 78.3°$, $\lambda_p = 291$
   $$\phi_m = \arcsin\left(\sin\phi_{IPP}\sin\phi_p + \cos\phi_{IPP}\cos\phi_p\cos(\lambda_{IPP} - \lambda_p)\right)$$

5. Local time at the IPP: $t = 43200\ \frac{\lambda_{IPP}}{\pi} + t_{GPS}$

   Where $0 \leq t \leq 86400$. Therefore: $if\ t \geq 86400, subtract\ 86400;\ if\ t < 0, add\ 86400$.

6. Amplitude of the ionospheric delay:

$$Q = \sum_{n=0}^{3} \alpha_n \left(\frac{\phi_m}{\pi}\right)^n \qquad if\ Q < 0\ , then\ Q = 0$$

7. Period of the ionospheric delay:

$$P = \sum_{n=0}^{3} \beta_n \left(\frac{\phi_m}{\pi}\right)^n \qquad if\ P < 72000\ , then\ P = 72000$$

8. Phase of the ionospheric delay: $X = \frac{2\pi\,(t-50400)}{P}$

9. Slant Factor (Ionospheric mapping function): $F = \left[1 - \left(\frac{R_E}{R_E+h}\cos EL\right)^2\right]^{-\frac{1}{2}}$

10. Compute the ionospheric time delay:

$$dI = \begin{cases} [5 \cdot 10^{-9} + Q\ \cos X] \times F, & |X| < \pi/2 \\ 5 \cdot 10^{-9} \times\ F, & |X| \geq \pi/2 \end{cases}$$

The result *dI* (in seconds) is then multiplied by the speed of light *c* to give a measure in meters that is the actual ionospheric correction added to the calculation of pseudorange.

3.2.2.2. Moon

The considerations made in the previous section for the terrestrial environment are in part applicable also for lunar implementation, but there are some differences to take into account.
In this analysis, the check on the satellites in view is also important, even more than before since the in lunar environment the number of signals is considerable reduced, so that the possibility of blackout in the solution due to the absence of the minimum number of satellites in view is very high.

Considering the static user scenario, the procedure is the same already explained before for Earth environment, which takes into account the elevation of the satellites with respect to the user position.

Instead, for the dynamic user scenario, since the user is no longer on lunar surface, satellites elevation considerations lose their meaning. Hence, in this case the check to be done is that the user position is at each epoch below the satellites: given the position of the satellites $[x_i, y_i, z_i]$ and the user $[x_u, y_u, z_u]$ referred to the center of the Moon, this means assuring that the norm of user position is lower than the norm of the satellites:

$$norm\left(\begin{bmatrix} x_u \\ y_u \\ z_u \end{bmatrix}\right) < norm\left(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}\right) \qquad for\ each\ satellite\ i$$

Once this check is done, one of the differences about lunar environment is that the pseudorange measurements are not provided by the input data, so that these values are replaced by the calculation of the Slant Range between each satellite and user position. If the coordinates of each satellite are defined as $[x_i, y_i, z_i]$ and the coordinates of the user are $[x_u, y_u, z_u]$, the Slant Range is computed as:

$$Slant\ Range = SR_i = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2}$$

As for the pseudorange, this expression must consider difference sources of errors.
The first observation that can be done is that, in lunar environment, atmospheric effects are not present, so that the ionospheric effect discussed before does not intervene in slant range calculation.

Concerning satellite's side, the errors to consider are the same introduced for Earth environment:
- Satellite orbital error
- Satellite clock error
- Sagnac effect.

As for the user noises, multipath error will be implemented in the static user scenario, while of course it is not considered in dynamic user case, since it will be not in proximity of lunar surface. Receiver error, instead, will be implemented in both cases.

As already said, these errors are added to the pseudoranges/slant ranges expressions. However, simply adding this correction will consider that the measurement errors are implemented only in the direction of the line-of-sight between satellite and receiver, as depicted on the left in *Figure 3.8*. To be more accurate in the implementation of these errors, it will be considered the concept of sphere of uncertainty: as shown on the right in *Figure 3.8*, this means that the errors are applied considering a sphere centered in the satellites and one on the user. The radii of these spheres will correspond to the sum of the errors on each of them, which means orbital and clock errors on satellites and multipath/receiver errors on user.

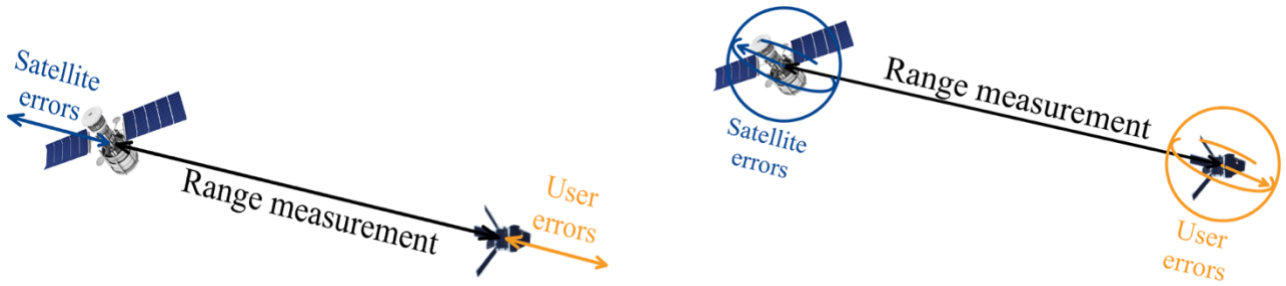

Figure 3.8: Application of errors on satellite and user without (left) and with (right) sphere of uncertainty

In *appendix B*, an example of the effect of the application of these errors is described considering either the implementation of each of them alone or in combination with each other.

4. Simulation Environment

4.1. Navigation system and inputs

The simulation environment considered in this work obviously takes into account all the difficulties described about lunar navigation. Therefore, the number of satellites in view will be reduced compared to Earth application, but also the signals will be weaker, and the input information will be lower than before.
The constellation used for the implementation of the positioning algorithms considers four satellites in view orbiting around the Moon, which is the minimum number needed to estimate the position of the user. This means that not only the amount of information is limited to the bare minimum, but also that the margin of error is minimal, since even a single failure no longer allows the positioning algorithm to work.
The input data provided are referred to four satellites following an Elliptical Lunar Frozen Orbit (ELFO). This category of frozen orbit provides greater coverage of lunar poles, since the satellite altitude remains constant over a long period of time at the same point in each orbit. The changes in inclination, position of the apsis of the orbit and eccentricity are minimized by the choice of the initial values, so that the perturbations are canceled out. This results in long-term stable orbit that minimizes the use of station-keeping propellant [4].
The constellation provided has been optimized in order to achieve high availability and minimize the Horizontal Dilution of Precision (that will be discussed later) at the South Pole.

For the application of the Sensor Fusion technique, an altimeter measurement is considered in addition to the four satellites in view. This sensor is considered to be inside the user, and it is activated when the user position is approximately 10 km away from Moon surface.
Given this additional measurement, one of the test cases that will be discussed, will consider the exclusion of one of the four satellites for the entire analysis and the substitution of the slant range measurement of this satellite with the altimeter reading.

For the validation of the codes, the input data has been taken from terrestrial satellites of the MOSE station, situated in Rome. The reason is that Earth environment is more controlled and known, since the number of input data is higher and the functioning of the algorithms has already been analyzed for different scopes, so that the reliability of the scripts produced can be verified.
For this scenario, the input data are contained into two important files:

- *The Receiver Independent Exchange Format (RINEX):* this is a file format for storing data from satellite navigation systems, which can provide *Observation data file* or *Navigation data file*. From the observation data, the input extracted will be the number of satellites in view and the pseudorange (in meters) of each satellite for each observation period (given as Gregorian date). The navigation data, instead, contain the Klobuchar coefficients described before, needed for the implementation of the ionospheric correction. The file provided contain 1 day data with observation every 30 seconds.
- *The Extended Standard Product 3 Orbit Format (SP3):* this is a file format containing other orbital information necessary for the implementation of the algorithm. In particular, from this file, the input data considered are the satellites coordinates (in kilometers) and clocks (in microseconds). The update rate of this data is 5 minutes, so they are not aligned with the RINEX data, then an interpolation will be needed to have all the information useful at the same epoch of observation (Lagrange interpolation is deepened in *appendix A.2*).

After the Earth validation analysis has been performed, the lunar data of the constellation described before will be considered. The input data are provided from four text files (one for each satellite) which contain the observation time (UTC), the positions (in km) and the velocities (in km/sec) of the satellites. For the dynamic user scenario, the lander characteristics are also provided with another text file in the same configuration just described for the satellites.
The data referred to the static user case are updated every minute, while for the dynamic user case the update rate is 1 second.

4.2. Implementation in MATLAB of PVT algorithms

4.2.1. Least Squares

Least squares estimation is a technique used to find a model that closely represents a collection of data and allows for the optimal determination of values or states within a system [8]. There are various parameters of interest in the system, which are typically referred to as states. To determine these parameters, an assortment of sensor is used, in order to provide information about what is actually happening in the system and how is changing over time.
In the analysis of this work, the measurements in input are the pseudorange ρ of the satellites, while the states to be determined are the position coordinates and time of the user *(x,y,z,t)*. The positioning problem is solved by linearizing the pseudorange observation equations, so that the first step is to define the actual observation as the sum of the modelled one plus an error:

$$\rho_{observed} = \rho_{model} + noise = \rho(x, y, z, t) + v$$

Then, the Taylor's theorem is applied to the model, ignoring the second and higher order terms

$$\begin{aligned}\rho(x, y, z, t) = {} & \rho(x_0, y_0, z_0, t_0) + (x - x_0)\frac{\partial \rho}{\partial x} + (y - y_0)\frac{\partial \rho}{\partial y} + (z - z_0)\frac{\partial \rho}{\partial z} + (t - t_0)\frac{\partial \rho}{\partial t} \\ & = \rho_{computed} + \Delta x \frac{\partial \rho}{\partial x} + \Delta y \frac{\partial \rho}{\partial y} + \Delta z \frac{\partial \rho}{\partial z} + \Delta t \frac{\partial \rho}{\partial t}\end{aligned}$$

The residual observation is defined as the difference between the actual observation and the computed one:

$$\Delta\rho = \rho_{observed} - \rho_{computed} = \Delta x \frac{\partial \rho}{\partial x} + \Delta y \frac{\partial \rho}{\partial y} + \Delta z \frac{\partial \rho}{\partial z} + \Delta t \frac{\partial \rho}{\partial t} + v$$

Which can be written in matrix form:

$$\Delta\rho = \begin{bmatrix}\frac{\partial \rho}{\partial x} & \frac{\partial \rho}{\partial y} & \frac{\partial \rho}{\partial z} & \frac{\partial \rho}{\partial t}\end{bmatrix} \cdot \begin{bmatrix}\Delta x \\ \Delta y \\ \Delta z \\ \Delta t\end{bmatrix} + v$$

This equation is valid for each satellite in view, so that considering *m* satellites this become a system of m equations in matrix form:

$$\begin{bmatrix}\Delta\rho^1\\ \Delta\rho^2\\ \vdots\\ \Delta\rho^m\end{bmatrix}=\begin{bmatrix}\frac{\partial\rho^1}{\partial x} & \frac{\partial\rho^1}{\partial y} & \frac{\partial\rho^1}{\partial z} & \frac{\partial\rho^1}{\partial t}\\ \frac{\partial\rho^2}{\partial x} & \frac{\partial\rho^2}{\partial y} & \frac{\partial\rho^2}{\partial z} & \frac{\partial\rho^2}{\partial t}\\ \vdots & \vdots & \vdots & \vdots\\ \frac{\partial\rho^m}{\partial x} & \frac{\partial\rho^m}{\partial y} & \frac{\partial\rho^m}{\partial z} & \frac{\partial\rho^m}{\partial t}\end{bmatrix}\cdot\begin{bmatrix}\Delta x\\ \Delta y\\ \Delta z\\ \Delta t\end{bmatrix}+\begin{bmatrix}v^1\\ v^2\\ \vdots\\ v^m\end{bmatrix}$$

Defining the state vector $u = [\Delta x, \Delta y, \Delta z, \Delta t]'$, the measurement vector $\Delta\rho = [\Delta\rho^1, \Delta\rho^2, \ldots, \Delta\rho^m]'$, the measurement noise $v = [v^1, v^2, \ldots, v^m]'$ and the Measurement Model Matrix H that represents the relationship between the measured values and the state values, the "linearized observation equations" can be written as:

$$\Delta\boldsymbol{\rho} = \boldsymbol{H}\cdot\boldsymbol{u} + \boldsymbol{v}$$

4.2.1.1. Case of analysis

Considering the case of analysis discussed in this work, the input data considered are the coordinates of the satellites (x_i, y_i, z_i), the offset of the receiver clock from system time (t_i) and the pseudoranges of the satellites (ρ_i). Assuming the first approximation $u_{start}=[x_u, y_u, z_u]'$ and $t_{start}=t_u$ the first step is the calculation of the *geometric range* for each satellite i:

$$r_i = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2}$$

Then, the approximation is set $\hat{u} = u_{start}$, so the *pseudorange approximation* is:

$$\hat{\rho_i} = \sqrt{(x_i - \hat{x}_u)^2 + (y_i - \hat{y}_u)^2 + (z_i - \hat{z}_u)^2} + c\cdot t_i$$

and the measurement model matrix H is determined:

$$H = \begin{bmatrix}a_{x1} & a_{y1} & a_{z1} & 1\\ \vdots & \vdots & \vdots & \vdots\\ a_{xi} & a_{yi} & a_{zi} & \vdots\\ \vdots & \vdots & \vdots & \vdots\\ a_{xn} & a_{yn} & a_{zn} & 1\end{bmatrix}$$

Where a_{xi}, a_{yi} and a_{zi} denote the direction cosines of the unit vector pointing from the approximate user position to the i^{th} satellite, and they are defined as:

$$a_{xi} = -\frac{x_i - \hat{x}_u}{r_i};\qquad a_{yi} = -\frac{y_i - \hat{y}_u}{r_i};\qquad a_{zi} = -\frac{z_i - \hat{z}_u}{r_i}$$

Finally, considering matrix form, the *displacements* can be computed as:

$$\hat{\rho} - \rho = \Delta\rho = H\cdot\Delta u$$

$$\Delta\boldsymbol{u} = \boldsymbol{H}^{-1}\cdot\Delta\boldsymbol{\rho} = [\Delta x_u, \Delta y_u, \Delta z_u, -c\Delta t_u]'$$

And then the *final result* of interest:

$$\boldsymbol{u} = \boldsymbol{u_{start}} + \Delta\boldsymbol{u}(\mathbf{1}:\mathbf{3})$$

$$t = \Delta u(4)$$

The vector u calculated is considered as the new approximation u_{start}, and the whole procedure is reiterated. This is valid until the difference between the position at the step $k+1$ and k, so that the norm of the vector $\Delta u(1:3)$ is lower than a decided tolerance, which in this analysis is considered to be 10^{-4}. Otherwise, the cycles can be stopped when the number of iterations exceed a certain value (which can be 50), but this is less accurate, since the tolerance method ensures that the position update has reached the desired level of accuracy.
In MATLAB implementation, it is important to remember to clear the values at the end of each iteration and epoch, in order to avoid errors due to the dimension of matrices that can change according to the number of satellites in view in each epoch.

In chapter 3 the difference in the implementation of the positioning algorithms on Earth or Lunar environment have been introduced. Obviously, those corrections need to be taken into account in Least Squares implementation: in terrestrial environment, the formula described before are implemented after the check of the satellites in visibility and the Lagrange interpolation. Moreover, the input data of the satellites in visibility must consider the Sagnac effect, while in the pseudorange approximation equation there will be an additional term given by the Ionospheric correction.
In lunar environment, the check to be done is that all the four satellites are in view, otherwise the algorithm cannot be computed. Pseudorange values are not given in input, so that the pseudorange equation is simply substituted by the Slant Range calculation. Of course, ionospheric correction is not present, but the slant range equation will have an additional term that contains the application of the errors introduced: satellite positioning, clock and multipath/receiver errors.

4.2.2. Kalman Filter

The Kalman Filter is a recursive method, which means that the estimate of the state vector is refined with each new input measurement and without the need to store the past measurements. This algorithm provides an efficient computational means to estimate the state of a dynamical system, in a way that minimizes the mean of the squared error.
Within the Kalman Filter implementation there are two different parts to describe: the process model and the measurement model. The process equations describe how the state is updated, while the measurement equations produce the measurement vector as a matrix times the state vector.
The initializations to be considered in this case regards the state vector $u_{start} = [x_u, y_u, z_u, t_u]^T$ and the error covariance matrix P_{start}. Starting from these values, the predict state is:

$$u_k' = F \cdot u_{k-1} + \nu$$

$$P_k' = F \ P_{k-1} \ F^T + Q$$

Where:

- u_k' is the predicted state vector at the epoch of analysis
- u_{k-1} is the state vector at time $k-1$, i.e., the result obtained at the previous step (at the first epoch it is the initialization u_{start})
- F = State transition matrix (deals with time steps & constant velocity)
- Q = State error autocovariance matrix (deals with uncertainty)
- ν = Noise

The state error autocovariance matrix Q is a n x n matrix, where n is the length of the state vector, while the measurement error auto covariance matrix R is a m x m matrix, where m is the length of the measurement vector (in this case it means the number of satellites in view, since the measurements are the respective pseudoranges). Both matrices are defined as the expected value of the respective noise (state and measurement noise):

$$Q = E[\nu \, \nu^T] \qquad\qquad R = E[\omega \, \omega^T]$$

Given these matrixes, the Kalman Gain can be computed:

$$K = P_k{}' H^T \cdot (H P_k{}' H^T + R)^{-1}$$

This is of course one of the most important parameters of the filter, which defines the relative importance given to the measurement and the current state estimation. Generally, a gain closer to one places more weight to the most recent measurements, resulting in a jumpy estimated trajectory, since the system fits faster. Instead, with a gain close to zero the system pays more attention to the predictions and will smooth out noise but decrease responsiveness.
Once defined the Kalman gain and the measurement model matrix *H*, the measurement vector z_{meas_k} and the measurement update equations can be computed:

$$z_{meas\_k} = H \cdot u_{k-1}$$

$$u_k = u_k{}' + K \cdot (z_{meas\_k} - H\, u_k')$$

$$P_k = P_k{}' - K\, H\, P_k{}'$$

These values are then considered as the new approximation for the subsequent step of analysis, in which the same procedure is repeated.

4.2.2.1. Extended Kalman Filter

The procedure described before considers a linear problem. The Extended Kalman Filter can overcome this issue by extending the functioning of the Kalman filter to the case of nonlinear problems. This method aims to linearize the problem, substituting the state transition matrix and the measurement model matrix with functions of the state vector. This means, in matrix form, that these parameters become Jacobian Matrices.

$$\left.\begin{matrix} F \to f(u) \to F_j \\ H \to h(u) \to H_j \end{matrix}\right\} \to Jacobian\ Matrices$$

Considering the matrix *H*, the linearization is computed with the first order Taylor expansion, which tries to predict the behaviour of a linear function taking into account a starting point, in this case the mean value:

$$h(u) = h(\mu) + \frac{\partial h(u)}{\partial u}(x - \mu)$$

- $h(\mu)$ = evaluation of nonlinear function in the mean value μ
- $\frac{\partial h(u)}{\partial u}$ = extrapolate a line around μ

Stating that *m* is the length of the measurement vector z_k and *n* the length of the state vector u_k, the Jacobian Matrix *H* is defined paying attention to the fact that in this case the partial derivates must be used:

$$H_j = \begin{bmatrix} \frac{\partial h_1}{\partial u_1} & \frac{\partial h_1}{\partial u_2} & \cdots & \frac{\partial h_1}{\partial u_n} \\ \frac{\partial h_2}{\partial u_1} & \frac{\partial h_2}{\partial u_2} & \cdots & \frac{\partial h_2}{\partial u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial u_1} & \frac{\partial h_m}{\partial u_2} & \cdots & \frac{\partial h_m}{\partial u_n} \end{bmatrix}$$

The definition of the State Transition Matrix F can change depending on the model used to analyze user's characteristics. In the following section, a dynamic model will be described: this can be considered as a nearly constant velocity or nearly constant acceleration model for example. Another consideration could be the static model, which is of course a variation of the dynamic model not considering velocity or acceleration values, but only position.

4.2.2.2. Case of analysis

Starting from same data of Least Square application, the Extended Kalman Filter allows to also determine velocity or acceleration components depending on the dynamic model considered.
In this analysis, the model considered is Nearly Constant Velocity, but the same reasonings will be valid for other dynamic models that can be implemented.
The state vector will now also include velocity components in addition to position:

$$\boldsymbol{u} = \left[x \;\; \dot{x} \;\; y \;\; \dot{y} \;\; z \;\; \dot{z} \;\; c\Delta t \;\; c\dot{\Delta}t\right]^T$$

The state equations consider the acceleration as the noise of the model:

$$u' = F \cdot u + \; \nu \rightarrow \begin{cases} x' = x + \dot{x}\Delta t + \nu_x \\ \dot{x}' = \dot{x} + \nu_{\dot{x}} \\ y' = y + \dot{y}\Delta t + \nu_y \\ \dot{y}' = \dot{y} + \nu_{\dot{y}} \\ z' = z + \dot{z}\Delta t + \nu_z \\ \dot{z}' = \dot{z} + \nu_{\dot{z}} \\ c\Delta t' = c\Delta t + c\dot{\Delta}t\Delta t + \nu_{c\Delta t} \\ c\dot{\Delta}t' = c\dot{\Delta}t + \nu_{c\dot{\Delta}t} \end{cases}$$

Where u' is the state vector at the step $k+1$, u is the state vector at the step k defined before, Δt is the time step between the two epochs and ν is the state noise that consider acceleration as perturbation of the model (it will be deepened shortly).
The State Transition Matrix can be easily derived from these equations. To simplify, considering only the x component, the state vector is of two elements $[x \; \dot{x}]^T$:

$$u'_x = F_{sub} \cdot u_x \rightarrow \begin{cases} x' = x + \dot{x}\Delta t \\ \dot{x}' = \dot{x} \end{cases} \rightarrow F_{sub} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

Replicating the same procedure for the other components, the State Transition Matrix F is the 8x8 matrix defined as:

$$F = \begin{bmatrix} F_{sub} & 0 & 0 & 0 \\ 0 & F_{sub} & 0 & 0 \\ 0 & 0 & F_{sub} & 0 \\ 0 & 0 & 0 & F_{sub} \end{bmatrix}$$

As already defined for the Least Squares technique, the measurements are given by the pseudoranges of the satellites (in lunar environment the pseudoranges are substituted by the Slant Ranges, but the reasoning and the result of this analysis is the same):

$$\rho_i = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} - c\, t_i + errors$$

Starting from this equation, the direction cosines of the measurement model matrix H can be computed considering the partial derivates:

$$\frac{\partial \rho_i}{\partial x} = \frac{-(x_i - x_u)}{\sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2}} \qquad \frac{\partial \rho_i}{\partial y} = \frac{-(y_i - y_u)}{\sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2}}$$

$$\frac{\partial \rho_i}{\partial z} = \frac{-(z_i - z_u)}{\sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2}} \qquad \frac{\partial \rho_i}{\partial (cdt)} = -1$$

Hence:

$$H = \begin{bmatrix} \frac{\partial \rho_1}{\partial x} & 0 & \frac{\partial \rho_1}{\partial y} & 0 & \frac{\partial \rho_1}{\partial z} & 0 & -1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \rho_m}{\partial x} & 0 & \frac{\partial \rho_m}{\partial y} & 0 & \frac{\partial \rho_m}{\partial z} & 0 & -1 & 0 \end{bmatrix}$$

The error autocovariance matrices (already introduced) needs to be determined too.
The State Error Autocovariance Matrix Q, as already said, is defined as the expected value of the state noise: $Q = E[v\, v^T]$
Considering the noise vector, it is possible to separate the first six elements regarding the noise on position and velocity, and the last two which relate clock and clock drift.
Taking into account the part that relates to position and velocity, the noise on the model is considered to be the acceleration, so that:

$$\boldsymbol{v} = \begin{bmatrix} v_x \\ v_{\dot{x}} \\ v_y \\ v_{\dot{y}} \\ v_z \\ v_{\dot{z}} \end{bmatrix} = \begin{bmatrix} a_x \cdot \frac{\Delta t^2}{2} \\ a_x \cdot \Delta t \\ a_y \cdot \frac{\Delta t^2}{2} \\ a_y \cdot \Delta t \\ a_z \cdot \frac{\Delta t^2}{2} \\ a_z \cdot \Delta t \end{bmatrix} = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 & 0 \\ \Delta t & 0 & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \frac{\Delta t^2}{2} \\ 0 & 0 & \Delta t \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = G \cdot a$$

Where the Matrix G has been introduced. Therefore, the submatrix Q_{xyz} will be:

$$Q_{xyz} = E[v\ v^T] = E[G\ a\ a^T G^T] = G\ E[a\ a^T]\ G^T$$

$$= \begin{bmatrix} \frac{\Delta t^2}{2} & 0 & 0 \\ \Delta t & 0 & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \frac{\Delta t^2}{2} \\ 0 & 0 & \Delta t \end{bmatrix} \cdot \begin{bmatrix} {\sigma_{ax}}^2 & 0 & 0 \\ 0 & {\sigma_{ay}}^2 & 0 \\ 0 & 0 & {\sigma_{az}}^2 \end{bmatrix} \cdot \begin{bmatrix} \frac{\Delta t^2}{2} & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\Delta t^4}{4}{\sigma_{ax}}^2 & \frac{\Delta t^3}{2}{\sigma_{ax}}^2 & 0 & 0 & 0 & 0 \\ \frac{\Delta t^3}{2}{\sigma_{ax}}^2 & \Delta t^2{\sigma_{ax}}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\Delta t^4}{4}{\sigma_{ay}}^2 & \frac{\Delta t^3}{2}{\sigma_{ay}}^2 & 0 & 0 \\ 0 & 0 & \frac{\Delta t^3}{2}{\sigma_{ay}}^2 & \Delta t^2{\sigma_{ay}}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\Delta t^4}{4}{\sigma_{az}}^2 & \frac{\Delta t^3}{2}{\sigma_{az}}^2 \\ 0 & 0 & 0 & 0 & \frac{\Delta t^3}{2}{\sigma_{az}}^2 & \Delta t^2{\sigma_{az}}^2 \end{bmatrix}$$

As it can be seen, Q_{xyz} is a block matrix and can therefore be written more compactly as follows:

$$Q_{sub\_i} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \cdot {\sigma_{ai}}^2 \quad with\ i = x, y, z \quad \rightarrow \quad Q_{xyz} = \begin{bmatrix} Q_{sub\_x} & 0 & 0 \\ 0 & Q_{sub\_y} & 0 \\ 0 & 0 & Q_{sub\_z} \end{bmatrix}$$

The same result can be obtained with the consequent different method. Recalling the model of the state, considering acceleration as noise, it is possible to define the matrix F_a (that corresponds to the state transition matrix of the constant acceleration model) and the matrix Q_a which include the noise only on the acceleration term:

$$F_a = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \qquad Q_a = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot {\sigma_s}^2$$

And the matrix product is defined:

$$F_a \cdot Q_a \cdot F_a^T = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ \Delta t & 1 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \cdot {\sigma_s}^2$$

$$= \begin{bmatrix} 0 & 0 & \frac{\Delta t^2}{2} \\ 0 & 0 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ \Delta t & 1 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \cdot {\sigma_s}^2 = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \cdot {\sigma_s}^2$$

The submatrix Q_{sub_i} considers only position and velocity components, which are the elements of the first two columns and rows. This leads to the same solution obtained with the previous method.

The previous result is valid considering a Discrete noise model, which means that the noise is different in each time period but constant in that time period. Instead, if it is assumed that the noise changes continuously over time, it is necessary to consider a Continuous noise model, in which the Q matrix is obtained integrating the one obtained in the Discrete model:

$$Q_{xyz\_continuous} = \int_0^{\Delta t} F \cdot Q_{xyz\_discrete} \cdot F^T \, dt = \int_0^{\Delta t} \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \cdot {\sigma_{ai}}^2 \, dt = \begin{bmatrix} \frac{\Delta t^5}{20} & \frac{\Delta t^4}{8} \\ \frac{\Delta t^4}{8} & \frac{\Delta t^3}{3} \end{bmatrix} \cdot {\sigma_{ai}}^2$$

Considering the clock part, the Q_{clock} matrix is defined dividing the noise in two parts: one related to the clock state variable, which is governed by the white noise spectral density leading to random walk velocity error (σ_t); the other related to clock drift, which consider the white noise spectral density leading to random walk clock frequency error plus the white noise clock drift (σ_{dt}) [30]. The latter correlation is defined by a matrix Q_{dt} which is the same calculated in discrete model for the components (Q_{sub_i}), while in the matrix Q_t the only element different from zero is the first one.

$$Q_{clock} = Q_t + Q_{dt} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot {\sigma_t}^2 + \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \cdot {\sigma_{dt}}^2 = \begin{bmatrix} {\sigma_t}^2 + \frac{\Delta t^4}{4} {\sigma_{dt}}^2 & \frac{\Delta t^3}{2} {\sigma_{dt}}^2 \\ \frac{\Delta t^3}{2} {\sigma_{dt}}^2 & \Delta t^2 {\sigma_{dt}}^2 \end{bmatrix}$$

As before, the continuous model is obtained integrating the previous matrix:

$$Q_{clock\_con} = \int_0^{\Delta t} \begin{bmatrix} {\sigma_t}^2 + \frac{\Delta t^4}{4} {\sigma_{dt}}^2 & \frac{\Delta t^3}{2} {\sigma_{dt}}^2 \\ \frac{\Delta t^3}{2} {\sigma_{dt}}^2 & \Delta t^2 {\sigma_{dt}}^2 \end{bmatrix} dt = \begin{bmatrix} \Delta t \, {\sigma_t}^2 + \frac{\Delta t^5}{20} {\sigma_{dt}}^2 & \frac{\Delta t^4}{8} {\sigma_{dt}}^2 \\ \frac{\Delta t^4}{8} {\sigma_{dt}}^2 & \frac{\Delta t^3}{3} {\sigma_{dt}}^2 \end{bmatrix}$$

There is no clear rule that specify the choice of a discrete model over a continuous one, or vice versa. Generally, discrete model is more recommended when Δt is very small, otherwise when Δt is large continuous noise model is more accurate. In this work, both models will be implemented to see the differences that they can provide [20].
Finally, the other matrix to be determined is the Measurement error autocovariance matrix R, which has already been defined as the expected value of the measurement noise.

In this analysis, this matrix is simply assumed to be diagonal with equal variance σ_r^2, which means that all measurements are assumed to be statistically uncorrelated, which is reasonable.

$$R = E[\omega\ \omega^T] = \begin{bmatrix} {\sigma_{R1}}^2 & & \\ & \ddots & \\ & & {\sigma_{Rm}}^2 \end{bmatrix}$$

As already said, the explained procedure can be done analogously for the Nearly Constant Acceleration model, considering the changes necessary to adapt the formulation to that case.
Regardless of the choice of model, one of the most important issues in the implementation of the Extended Kalman Filter is the selection of the right values for the process noise and measurement variances, i.e., the values of σ_{ai}, σ_t, σ_{dt} and σ_r. This process is called Tuning and it is defined to be an art more of a science, because there are not tabular values, but they usually come from a process of trial and error, so it needs engineering practice and experience (see *appendix D*).
Taking for example the radar world, the σ depends on the target characteristics and model completeness. For maneuvering targets, like airplanes, the σ shall be quite large, while for non-maneuvering targets, like rockets, σ can be smaller. Moreover, if the model includes environmental influences like air drag, the degree of the process noise randomness is smaller and vice versa [20].

4.2.3. Sensor Fusion

The Sensor Fusion technique combines data from different sensors in order to reach better performance than can be achieved when each source of information is used alone. In this work, the Extended Kalman Filter is considered with the support of an additional measurement, given by an altimeter. This reasoning is obviously valid only considering a dynamic user scenario, in which the distance from the lunar surface can be determined. In particular, altimeter reading is available when the user is less than 10 km away from the surface of the Moon [31] [32].
In this case, the errors implementation on satellites considers only satellites positioning errors, since clock error is not present anymore because the measurements are two-way. Multipath error is not present of course because the user is orbiting around the Moon (or landing), but a receiver error must be considered (implemented as a random error with gaussian distribution).

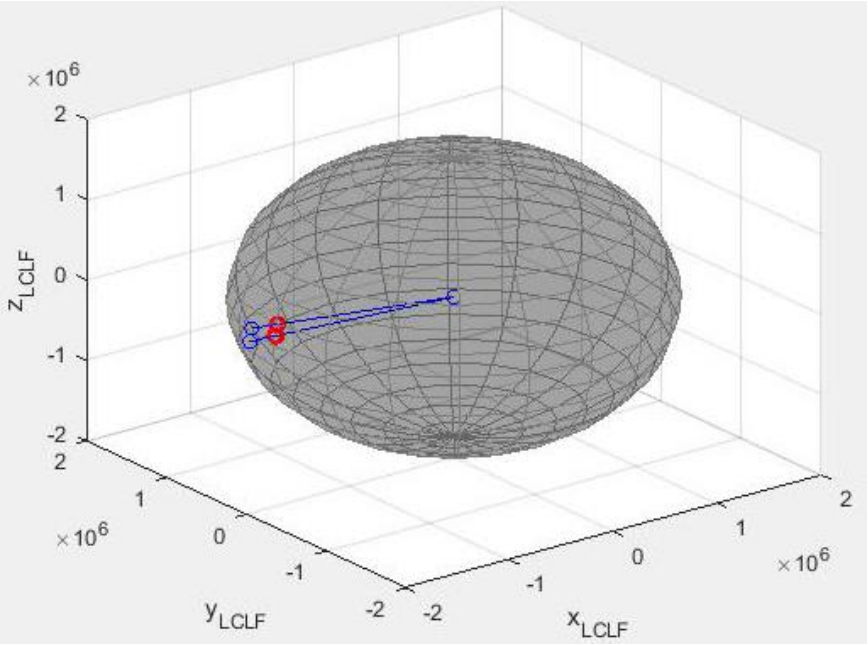

Figure 4.1: 3D animation of the Slant Range of the user (blue) and the Intersection with lunar surface (red) over time

Regarding the altimeter measurement, the first implementation will be made with the hypothesis that the altimeter reading is purely vertical, directed toward the center of the Moon. *Figure 4.1* shows in blue the Slant Range of the user over time, calculated with the coordinates given in input in the observation file. Instead, the red points highlight the intersection between these values of slant ranges and the surface of the Moon. The hypothesis considers that the additional measurement at each epoch is given by the distance between the user position and the intersection with the Moon determined at that epoch, which can be calculated simply by subtracting the radius of the Moon from the slant range of the user. This means that this measurement is directed toward the center of the Moon, i.e., along the Up direction of the ENU reference system. This assumption is of course incorrect, since it does not consider the attitude of the lander: in fact, following its trajectory, the lander does not aim perfectly to the center of the Moon at each epoch, but its attitude over time makes that the measurement is not absolutely vertical.

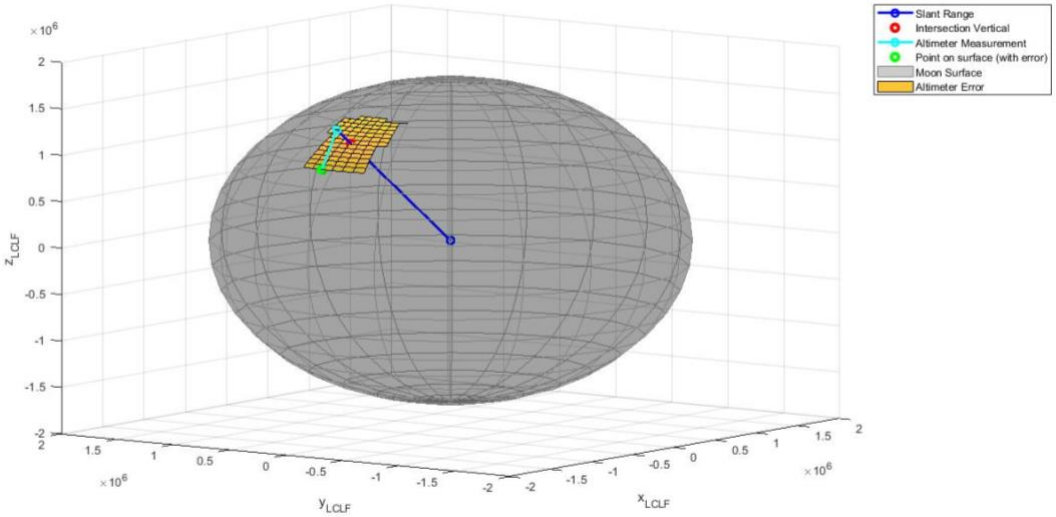

Figure 4.2: Altimeter measurement with and without error

The implementation of this correction is depicted in *Figure 4.2*, considering a single instant. As before, the blue line represents the slant range of the user and the red point the intersection with the lunar surface. The yellow span represents the error of the altimeter reading due to the attitude of the lander: this span is calculated considering an error on the nadir direction comprised in a 1.3° cone [32], as shown in *Figure 4.3*.
Given this cone, the span error in meters on the lunar surface can be easily calculated using the sine theorem and simple geometric definitions.
Once this span error is defined (depicted in yellow in *Figure 4.2*), the lander can aim to any point on the perimeter of this surface (depicted in green), thus the more realistic altimeter reading will be the distance between this point and the user position at that epoch (cyan line in figure).

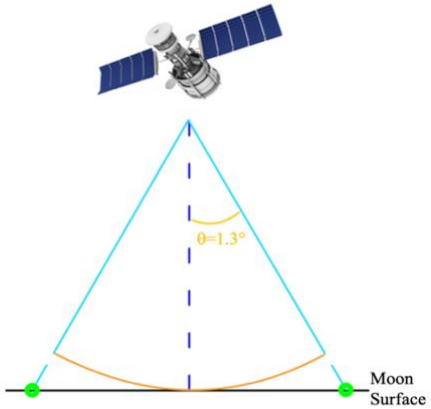

Figure 4.3: Altimeter cone error

4.2.3.1. Algorithm implementation

The first step of the implementation is the determination of the altimeter measurement. Considering the first hypothesis (that the measurement perfectly vertical), this reading is obtained by the difference between the user position and the radius of the Moon. Therefore, for each epoch, the Extended Kalman Filter algorithm is implemented and the user position at that epoch is calculated: after that,

subtracting the radius of the Moon to the position calculated the hypothetical altimeter measurement is determined:

$$EKF \rightarrow u_{app} \rightarrow alt_{meas} = norm(u_{app}) - r_{Moon}$$

If this distance is lower than ten kilometers, the altimeter is effective, so that the Sensor Fusion technique can be considered and then the Extended Kalman Filter is reimplemented with the modification needed.
First, the intersection between the slant range of the user and the lunar surface needs to be determined. This is basically done considering a system of equations that include the equation of the line through the two points $c=[x_c, y_c, z_c]=[0, 0, 0]$ and $u_{app}=[x_{app}, y_{app}, z_{app}]$, and the equation of the sphere with center $c=[x_c, y_c, z_c]=[0, 0, 0]$ and radius equal to r_{Moon}:

$$\begin{cases} \dfrac{x - x_c}{x_{app} - x_c} = \dfrac{y - y_c}{y_{app} - y_c} \\ \dfrac{y - y_c}{y_{app} - y_c} = \dfrac{z - z_c}{z_{app} - z_c} \\ (x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = {r_{moon}}^2 \end{cases}$$

For the first hypothesis, this is the point on the surface where the lander is aiming, so that it will be used in the following calculations. Instead, considering the correction due to the attitude of the user, the error explained before needs to be implemented to produce the more realistic measurement.
Since there is an additional measurement, the only variables that change in the implementation of the Extended Kalman Filter are the ones that depend on the number of measurement m. These variables are the measurement model matrix H (m x n matrix), the measurement error autocovariance matrix R (m x m matrix) and the measurement vector z_{meas} (m x 1 vector).

In order to determine the additional row of the measurement model matrix, the altimeter measurement needs to be expressed in function of the three coordinates, as done for the pseudorange in the previous sections. Stating that the approximation of the filter is u_{app} as before and the calculated point on the Moon surface (where the altimeter aims) is $u_L=[x_L, y_L, z_L]$, the expression of the altimeter reading is:

$$alt_{meas} = \sqrt{(x_{app} - x_L)^2 + (y_{app} - y_L)^2 + (z_{app} - z_L)^2}$$

So that the partial derivates can be computed:

$$\frac{\partial alt}{\partial x} = -\frac{x_{app} - x_L}{alt_{meas}} \qquad \frac{\partial alt}{\partial y} = -\frac{y_{app} - y_L}{alt_{meas}} \qquad \frac{\partial alt}{\partial z} = -\frac{z_{app} - z_L}{alt_{meas}}$$

Hence, the additional row of the measurement model matrix is:

$$H_{alt} = \left[\frac{\partial alt_{meas}}{\partial x}\ 0\ 0\ \ \frac{\partial alt_{meas}}{\partial y}\ 0\ 0\ \ \frac{\partial alt_{meas}}{\partial z}\ 0\ 0\ \ 0\ 0\ 0\right]$$

$$\rightarrow \qquad H = \begin{bmatrix} H_{EKF} \\ H_{alt} \end{bmatrix}$$

There is another possibility to obtain the same result. As already said, the altimeter provides in output a measurement which is referred to an ENU reference system. In the first hypothesis the only component is the Up direction, whilst East and North are null since it has been considered a perfectly vertical measurement. Instead, taking into account the attitude of the lander, the measurement will also have East and North components.

In any case, it is possible to determine the Latitude φ and Longitude λ of the position of the user using Bowring technique *(Appendix C and* [29]*)* and use these parameters to determine the rotation matrix to switch the measurement from ENU coordinates to XYZ reference system with the origin in the center of the Moon. These matrices allow to express the ENU coordinates in function of *x*, *y* and *z*:

$$alt_{meas} = \sqrt{E^2 + N^2 + U^2}$$

$$\begin{bmatrix} E \\ N \\ U \end{bmatrix} = R_1\left[\frac{\pi}{2} - \varphi\right] R_3\left[\frac{\pi}{2} + \lambda\right] \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\cos\lambda\ \sin\varphi & -\sin\lambda\sin\varphi & \cos\varphi \\ \cos\lambda\cos\varphi & \sin\lambda\cos\varphi & \sin\varphi \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}$$

Where:

$$\Delta x = x_{app} - x_L \qquad \Delta y = y_{app} - y_L \qquad \Delta z = z_{app} - z_L$$

Hence, the matrix product yields:

$$\begin{cases} E = -\sin\lambda \cdot \Delta x + \cos\lambda \cdot \Delta y \\ N = -\cos\lambda\ \sin\varphi \cdot \Delta x - \sin\lambda\sin\varphi \cdot \Delta y + \cos\varphi \cdot \Delta z \\ U = \cos\lambda\cos\varphi \cdot \Delta x + \sin\lambda\cos\varphi \cdot \Delta y + \sin\varphi \cdot \Delta z \end{cases}$$

And the partial derivates are computed as:

$$\frac{\partial alt}{\partial x} = \frac{-E \cdot \sin\lambda - N \cdot \cos\lambda\sin\varphi + U \cdot \cos\lambda\cos\varphi}{\sqrt{E^2 + N^2 + U^2}}$$

$$\frac{\partial alt}{\partial y} = \frac{E \cdot \cos\lambda - N \cdot \sin\lambda\sin\varphi + U \cdot \sin\lambda\cos\varphi}{\sqrt{E^2 + N^2 + U^2}}$$

$$\frac{\partial alt}{\partial z} = \frac{N \cdot \cos\varphi + U \cdot \sin\varphi}{\sqrt{E^2 + N^2 + U^2}}$$

And again, the measurement model matrix *H* is defined as before.

The measurement vector z_{meas} also will have an additional row, which will be defined simply as the difference between the altimeter measurement calculated with the approximated position and the measurement calculated with the reference position:

$$z_{meas} = \begin{bmatrix} z_{EKF} \\ z_{alt} \end{bmatrix} = \begin{bmatrix} z_{EKF} \\ alt_u - alt_{meas} \end{bmatrix}$$

Where alt_{meas} is defined as before and alt_u is computed in the same way given the reference position of the user $u_{user}=[x_u, y_u, z_u]$:

$$alt_u = \sqrt{(x_u - x_L)^2 + (y_u - y_L)^2 + (z_u - z_L)^2}$$

Finally, the measurement error autocovariance matrix *R* will have an additional row and column, with the only non-zero element always on the diagonal of the matrix:

$$R = \begin{bmatrix} R_{EKF} & 0 \\ 0 & {\sigma_R}^2 \end{bmatrix}$$

4.3. Performance Analysis Tool

In the previous paragraphs, the different positioning algorithms has been introduced with the explanation of their implementations, formulas and errors applications. In order to validate the scripts produced, in this paragraph they will be tested with the use of Earth input data. This choice has been done because the Earth scenario is known and can be managed with relative ease. In fact, terrestrial results obtainable with the techniques analyzed are known in literature and already applied in other contexts, which means that it can be easier to compare the results obtained in this work and to verify their reliability. Furthermore, as already discussed, in lunar environment the number of satellites in view is reduced and signals are weaker. Considering Earth environment, instead, the number of constellations usable (hence the number of satellites in view) is higher, so that the functioning of the algorithms can be determined more accurately.
These data are referred to a GPS constellation orbiting around the Earth, with the number of satellites in view that can vary at each epoch of observation, up to a maximum of 32 satellites.
The results will be referred to the implementation of the Least Squares algorithm and the Extended Kalman Filter with either a static or a dynamic model. They will show the main outcomes obtained with the implementation of these algorithms, in order to obtain some information about their reliability considering the performance and differences already present in literature regarding these techniques.

4.3.1. Least Squares

First algorithm introduced in paragraph 4.2 is the Least Squares technique, which results will be reported in this section. It is important to underline that the initial approximation of the user position considered in this work is the center of the Earth, $u_{start} = [0, 0, 0]$ (considering ECEF reference system), but it can be demonstrated that the results would not be different considering a different initialization.
The first important outcome to discuss is the Positioning Error, that represents the deviation of the output of the Least Squares from the reference position, giving information about the accuracy of the estimation obtained with this technique. Considering for each observational epoch $u_{ref}=[x_{ref}, y_{ref}, z_{ref}]$ as the reference position retrieved from the observation file and $u_{app}=[x_{app}, y_{app}, z_{app}]$ as the approximated position calculated by the algorithm at that specific epoch, the Positioning Error can be computed as:

$$Positioning\ Error = \sqrt{\left(x_{app} - x_{ref}\right)^2 + \left(y_{app} - y_{ref}\right)^2 + \left(z_{app} - z_{ref}\right)^2}$$

In *Figure 4.4* the Positioning Error is represented with blue circles, while the x axis represents the time of analysis which shows solution every 30 seconds, since the input data provide satellites characteristics every 30 seconds, so that each epoch corresponds to 30 seconds of analysis. In the figure, the number of satellites in view at each epoch is reported in red line, with the respective scale on the right of the graph.

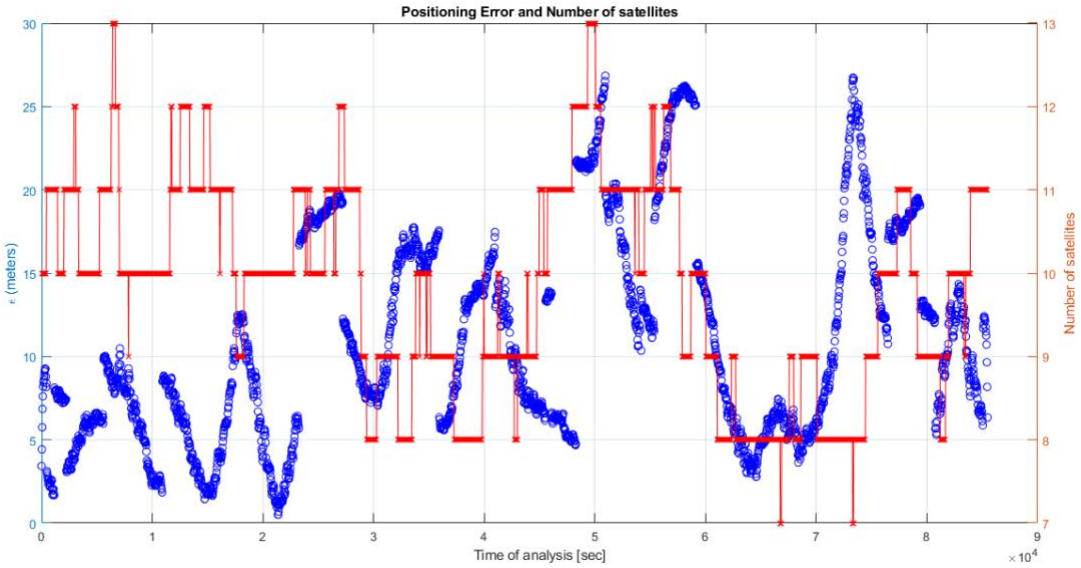

Figure 4.4: Positioning error (blue) and Number of satellites in visibility (red) for LS implementation

The figure shows that the number of satellites in view can change during the analysis between a minimum of 7 satellites and a maximum of 13, which of course gives always the coverage needed to estimate the position of the user. The positioning error demonstrates a trend of the solution a little scattered over time, with levels of accuracy that hardly go over 25 meters or below 5 meters of error. These values are in line with expectations, giving a first confirmation of the reliability of the Least Squares implementation.

Another important result to discuss is the Dilution of Precision (DOP): as shown in *Figure 4.5*, the way the user sees the satellites can affect the positioning estimation. On the left the figure shows that, due to measurement errors, the true range of each satellite is affected by a measurement noise ε, determining an uncertainty region in the position estimation. The size and the shape of this region can vary depending on the relative positions of user and satellite, as highlighted in the 2D illustration on the right. In fact, even with the same measurement error variation, the orange regions highlight that *geometry a)* gives considerably less error than in *geometry b)*, so that the latter will be considered to have a larger Dilution of Precision. This effect will lead, for comparable measurement errors, to larger errors in the computed position for *geometry b).*

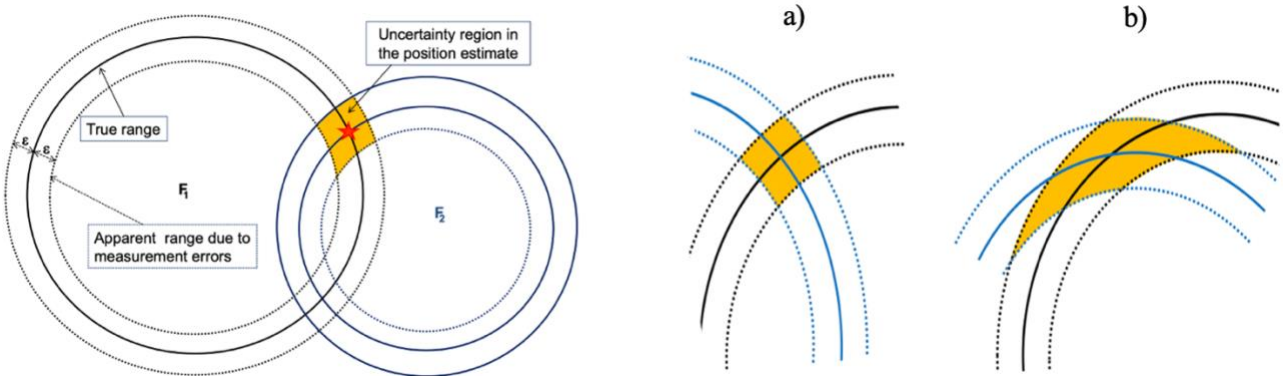

Figure 4.5: DOP (Source: ESA GNSS Book [23])

This effect is called indeed Dilution of Precision, and it is represented by different parameters, reported in *Figure 4.6*. These parameters are defined as geometry factors useful the characterize the accuracy of various components of the position/time solution, since they relate user position and

time bias errors to those of the pseudorange. The determination of these parameters is given from the definition of the matrix D, obtained starting from the Measurement Matrix H, already discussed in section 4.2.1:

$$D = (H^T H)^{-1} = \begin{bmatrix} D_{11} & D_{12} & D_{13} & D_{14} \\ D_{21} & D_{22} & D_{23} & D_{24} \\ D_{31} & D_{32} & D_{33} & D_{34} \\ D_{41} & D_{42} & D_{43} & D_{44} \end{bmatrix}$$

D is a *4 x 4* matrix, since H is *n x 4*, where n is the number of satellites in view. Starting from this matrix, the following parameters are defined:

$$Position\ Dilution\ of\ Precision = PDOP = \sqrt{D_{11} + D_{22} + D_{33}}$$
$$Horizontal\ Dilution\ of\ Precision = HDOP = \sqrt{D_{11} + D_{22}}$$
$$Vertical\ Dilution\ of\ Precision = VDOP = \sqrt{D_{33}}$$
$$Time\ Dilution\ of\ Precision = TDOP = \sqrt{D_{44}}$$

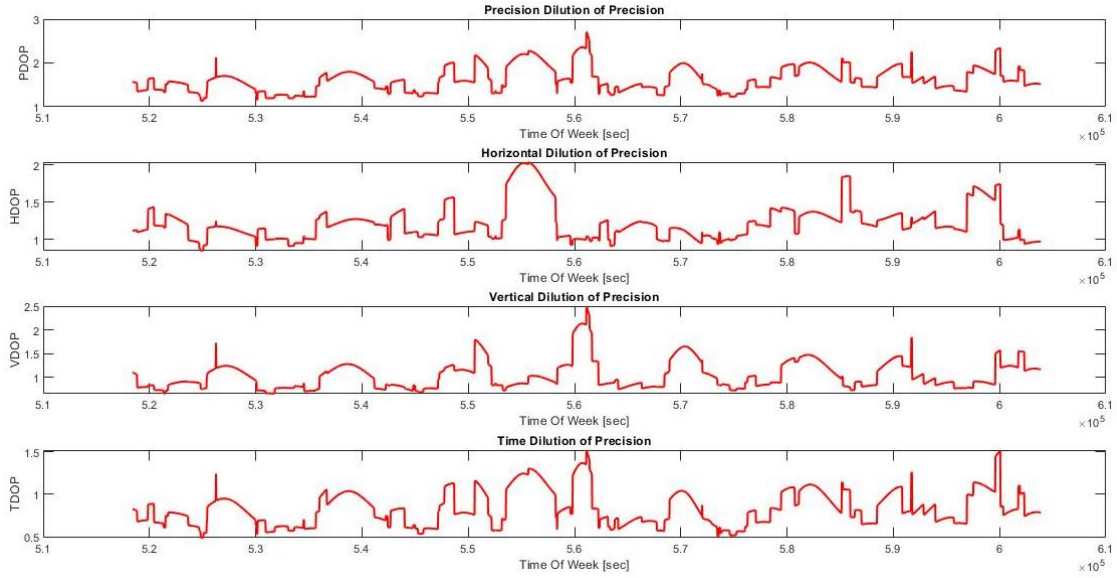

Figure 4.6: Trend of different Dilution of Precision (DOPs)

In *Figure 4.6* on the x axis the Time of Week of each epoch is reported: this value is determined through the conversion from the Gregorian date given in the input file for each observational epoch to GPS time, as explained in *Appendix A.1*. The first observational epoch is *January 1st 2020*, at which corresponds a *TOW* equal to *518400 seconds*: after this value the subsequent ones will be spaced of 30 seconds each as explained before.
The figure shows that the values of DOP parameters vary between 0.5 and 2.5, which can be considered acceptable results, since in literature the thresholds in GPS performance standards are chosen to be lower than 6 [27]. If the DOPs exceed this value, the GNSS could be considered unavailable. Moreover, it can be said that there is a slight correlation between DOPs values and the number of visible satellites: often (but not always) DOPs values are lower (which means better accuracy) when more satellites are visible, as it can be seen for example about halfway through the solution in *Figure 4.6* where DOPs values are higher corresponding to less satellites in view as shown in *Figure 4.4*. This can be easily explained, as more satellites in view can give more measurements to the user and can provide more easily a better geometric configuration, which clearly leads to better accuracy.

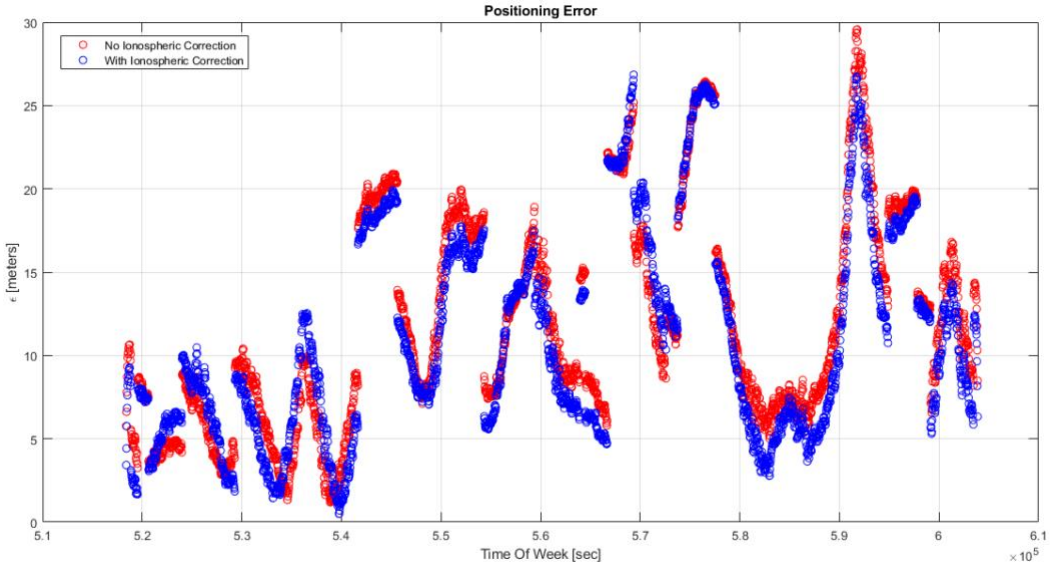

Figure 4.7: Positioning Error for LS implementation with or without Ionospheric correction

Finally, a secondary result concerns the application of the ionospheric effect (introduced in section 4.2.1) on the solution. In *Figure 4.7* the blue line represents the solution with the implementation of the ionospheric correction (same result shown in *Figure 4.4*), while the red line represents the solution that consider the ionospheric effect. As it can be seen, this effect can lead to errors not quite large, with difference of few meters.
This correction will not play an important role on the Moon since this effect is not present in the lunar environment. However, the terrestrial analysis allows to highlight the impact of this effect on the solution and to confirm the reliability of the implemented tool.

4.3.2. Extended Kalman Filter

As explained in section 4.2.2, the Extended Kalman Filter can produce different outcomes depending on the model considered to approximate user position over time. In this section the results will consider the implementation at first of a static positioning model and then a dynamic model, precisely Nearly Constant Velocity model. In both cases, the initial approximation is not the center of the Earth as for the Least Squares, but it is considered to be the outcome obtained after three iterations of the LS itself. The effect of this consideration will be clear observing the results.

4.3.2.1. Static Filter

The static model considers that the user is in a fixed position over time, so that the prediction of the user position and velocity at the instant *k+1* are the same determined at the instant *k*.

$$\text{Static Model} \rightarrow \begin{cases} x' = x \\ y' = y \\ z' = z \\ c\Delta t' = c\Delta t \end{cases} \rightarrow F_{static} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As for the Least Squares, the main result to analyze is the Positioning Error, depicted in *Figure 4.8* as blue circles. The figure also shows in red the number of satellites in view, with the respective scale on the right, while the abscissa axis corresponds to the time of analysis with each epoch separated by 30 seconds from the next one (as already explained for LS).

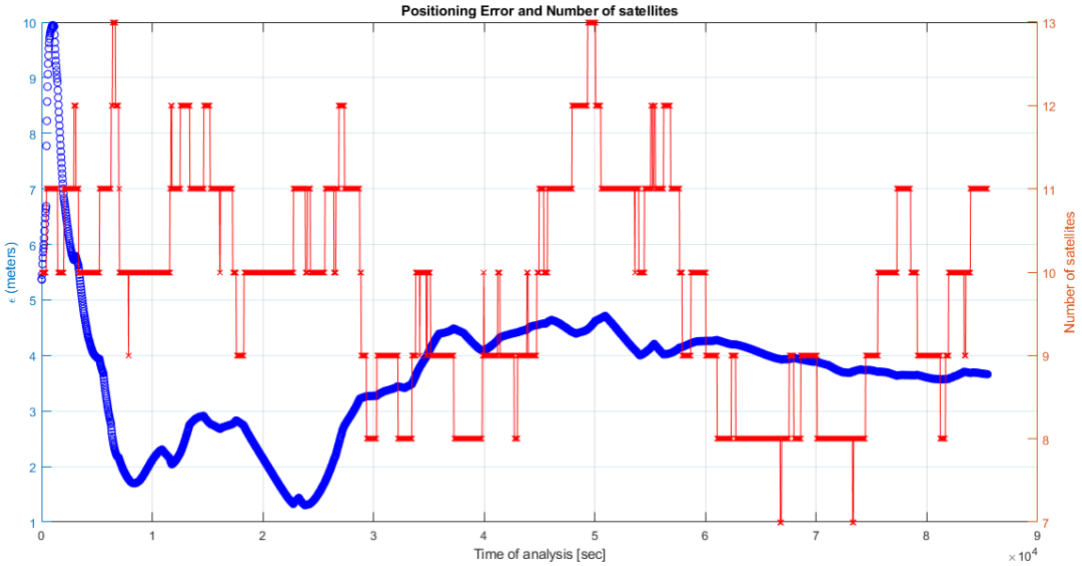

Figure 4.8: Positioning Error (blue) and Number of satellites in visibility (red) for EKF with static model

As before, the number of satellites in visibility varies between a minimum of 7 satellites and a maximum of 13, giving the coverage needed during the analysis.
Regarding the Positioning Error, as expected the figure brings to light the different evolution of the EKF solution, which converges over time to the final results, as opposed to the LS solution which was more scattered during the whole simulation time. In fact, in the first epochs of analysis there is a peak of about 10 meters, but shortly after that the filter converges and settles around values lower than 4 meters. This obviously demonstrates great accuracy of the Extended Kalman Filter compared to the Least Squares results, which is in line with the expectations.
As said before, the filter initialization considers three iterations of LS instead of the center of the Earth as first approximation. In the latter case, the results would be the same shows in *Figure 4.8*, with the only difference that the EKF would take a little longer to converge, since the initial position estimation would be a lot far from the reference one. In any case, this would not affect the analysis that demonstrates the better accuracy of the results obtained.

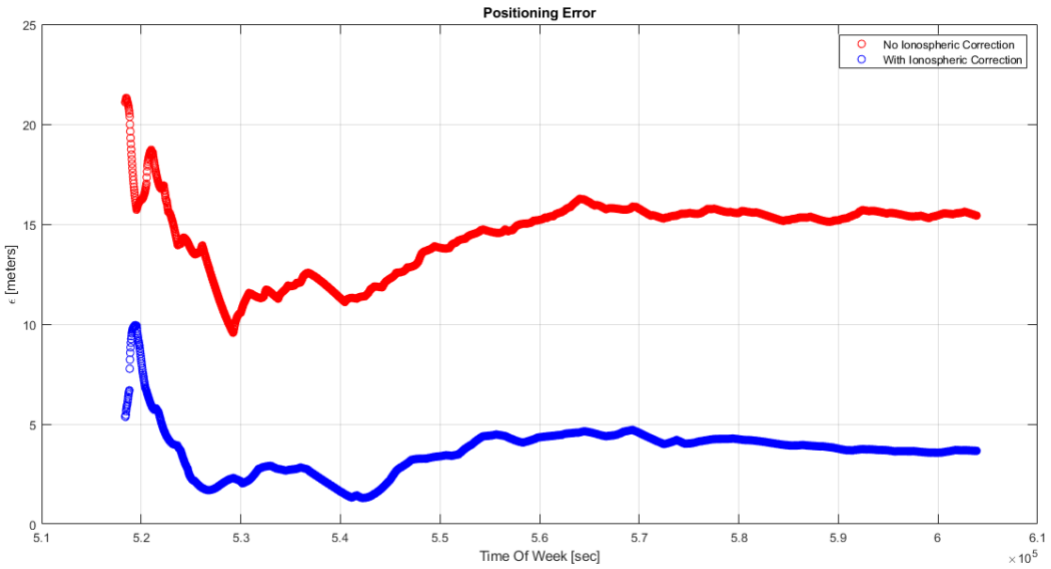

Figure 4.9: Positioning Error for EKF static model implementation with or without Ionospheric correction

As for section 4.3.1, *Figure 4.9* shows the comparison between the results obtained with the implementation of the ionospheric correction (blue circles, same as *Figure 4.8*) and the results considering instead the ionospheric effect (red circles). Once again, the abscissa axis represents the Time of Week of each epoch.
The difference between the two solution is clearer than in the Least Squares analysis. If the correction is not implemented, the errors in the estimation reach values higher than 20 meters in the first epochs and then converges to values around 15 meters, which means almost four times more than the results obtained with the implementation of the correction.
As said before, this will not play an important role for lunar implementation, but it is helpful to verify the reliability of the script produced also for the Extended Kalman Filter.

4.3.2.2. Dynamic Filter

As mentioned before, the results will now be referred to the implementation of the Extended Kalman Filter considering a dynamic model. In particular, it will consider the Nearly Constant Velocity model explained in the section 4.2.2.

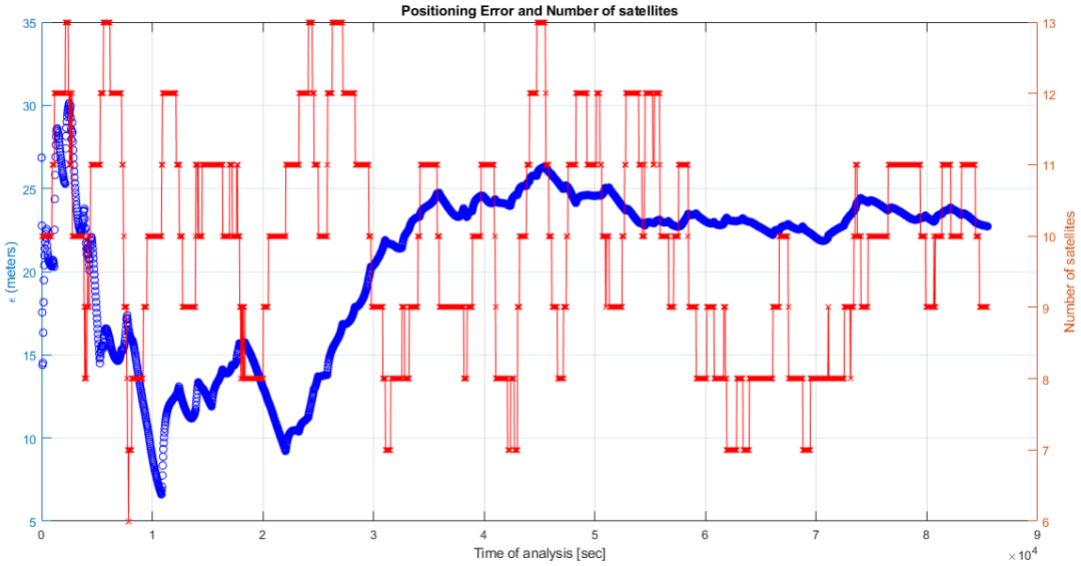

Figure 4.10: Positioning Error (blue) and Number of satellites in visibility (red) for EKF with NCV model

The outcomes analyzed will be the same as before. *Figure 4.10* represents the Positioning Error (blue circles) and the number of satellites in visibility (red line), with the time of analysis on the abscissa.
Once again, during the whole simulation there are at least 6 satellites in visibility, giving the needed coverage. As for the Positioning Error, the figure shows the same trend as for the static model, with peaks at the beginning of the analysis and convergence shortly after few epochs. Despite this, with the NCV model, at the beginning the errors are of almost 30 meters and then the filter converges to values around 22 meters, which is higher compared to the previous results.
Moreover, the filter takes longer to converge with respect to the static model, reaching better values after almost 3 hours, but then increasing back and settles around 22 meters.
This is expectable, since the position to estimate is referred to a static user, which is in a fixed position on Earth that is the reference value reported in the observation file in input. The nearly constant velocity model, instead, considers that the user is in motion over time, so that the filter will need more time to estimate the position, but also the correct velocity of the user which in reality will be static in the same position and without velocity components.

As mentioned in section 4.2.2, the results obtained highly depend on parameters injected in the filter. In fact, the tuning of these parameters is one of the most challenging phase of the validation of the filter, and this will be even more clear in lunar implementation (also demonstrated in *appendix D*). Once again, ionospheric effect will not be considered in lunar implementation, but as for the other cases this implementation allows to verify the reliability of the Extended Kalman Filter also for the dynamic model.

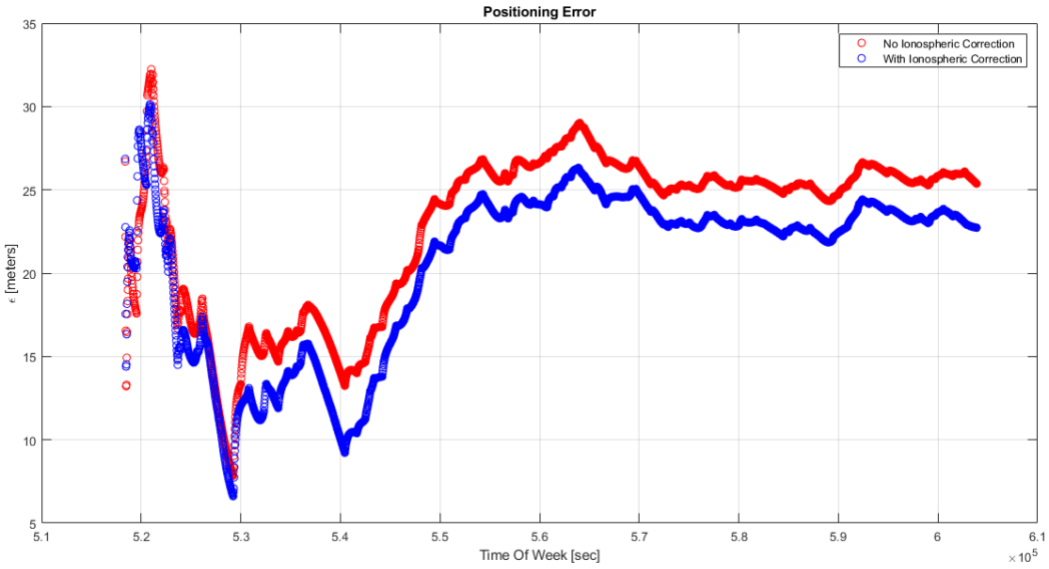

Figure 4.11: Positioning Error for EKF with NCV model implementation with or without Ionospheric correction

Figure 4.11 highlights once again the difference between the implementation of the Extended Kalman Filter with NCV model with (blue circles) and without (red circles) the ionospheric correction, with the Time of Week on the abscissa.
The difference is smaller than the previous case, but as before the presence of this effect leads to error values higher than 30 meters in the beginning of the simulation and then converges to values around 25 meters, which means 3 meters more than with the application of the ionospheric correction.

5. Analysis of Results

5.1. Static User

Results in this paragraph are referred to the implementation of the different algorithms described in the previous chapter considering a static user. As shown in section 3.1, the hypothesis is that the user is inert in a fixed position, that has been considered the South Pole of the Moon. Therefore, this will be the reference position which, considering a coordinate system with its origin in the center of the Moon, will correspond to the coordinates $[0, 0, -r_{Moon}]$. In order to determine the 3D position of the user, at least 4 satellites are needed to be in view: hence, based on the elevation of each satellite, a check has been done to assure that the four satellites of the constellation considered are in visibility during the analysis. Therefore, the results will show some interruptions in the calculation, referred to the epochs in which one or more satellites are no longer in view and the algorithm cannot find a solution, since the number of variables is lower than the unknows to be determined. Therefore, when there is this "blackout" of signals, the algorithm cannot consider anymore the approximation of the previous epoch and it will need to be reinitialized with the initial hypothesis.
The x axis of the figures will indicate the time of analysis: the input data are provided every minute, hence during the simulation each observation epoch corresponds to one minute of analysis.

5.1.1. Least Squares

In this section, the main results obtained with the implementation of the Least Squares technique will be presented. The initialization considered in this analysis is that the first approximation of user position is $[0,0,0]$. The first analysis will be focused on the results obtained on the Horizontal plane, i.e., the x and y coordinates, while in the second part the results will be referred to the Vertical plane, i.e., the z coordinate. The three directions are referred to the reference system defined in section 3.1, with its origin in the center of the Moon. Since the user is a static configuration situated on the South Pole, in this scenario the horizontal and vertical planes with respect to the user position are the same of the reference system considered.
Both analyses will show the comparison between the implementation of the algorithm considering the absence or the presence of the errors described in section 3.2 (satellite positioning, clock and multipath errors).
In *Figure 5.1.1*, the blue circles represent the Horizontal Error, while the green circles represent the Horizontal Dilution of Precision, considering the implementation without any errors in the measurements. HDOP definition has already been analyzed in chapter 4.3, while Horizontal Error is defined as the difference between the user position determined with the algorithm and the reference position considering the horizontal plane, i.e., x and y coordinates. Defining $user=[x_u , y_u , z_u]$ as the reference position (South Pole) and $u_{app}=[x_{app} , y_{app} , z_{app}]$ as the calculated position (approximation) with LS, the Horizontal Error is computed as:

$$Horizontal\ Error = \sqrt{(x_{app} - x_u)^2 + (y_{app} - y_u)^2}$$

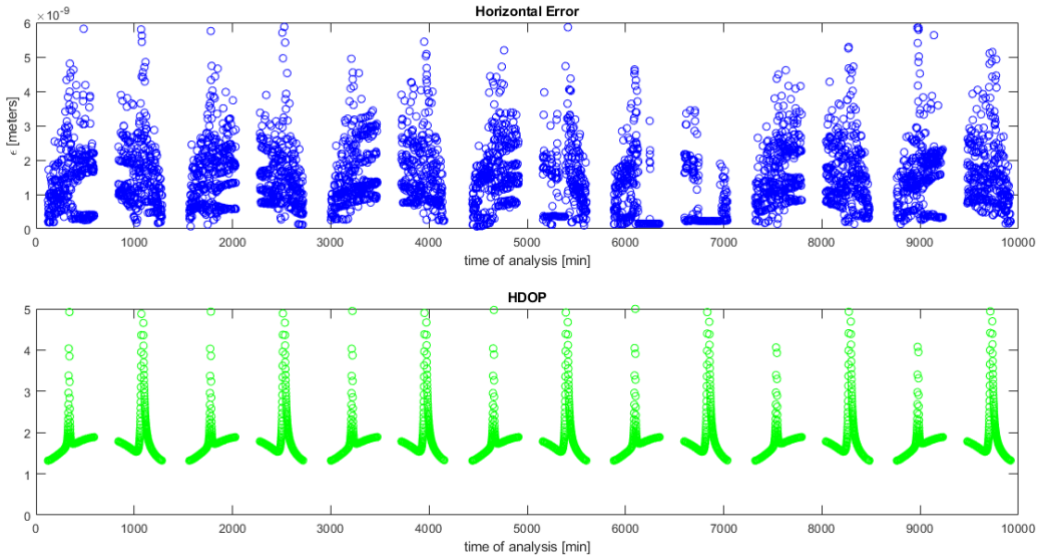

Figure 5.1.1: Horizontal error and HDOP without errors implementation

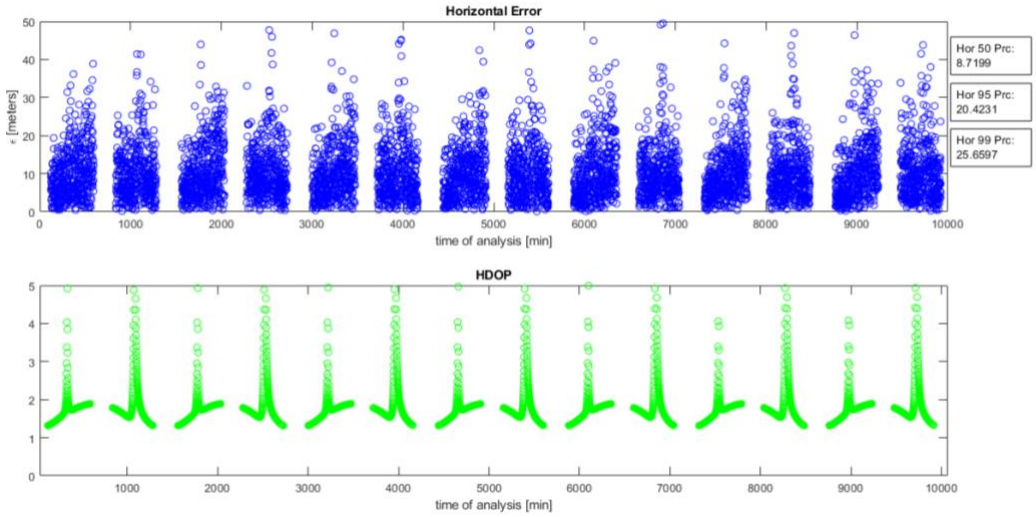

Figure 5.1.2: Horizontal Error and HDOP with errors implementation

Of course, not considering the errors *(Figure 5.1.1)*, the results are much more accurate, since the delta from reference position is on the order of 10^{-9} meters. Instead, it is of interest to observe the values that consider errors implementation *(Figure 5.1.2)*, since they are closer to reality and show a deviation from the reference position that varies from 0 to 30 meters, which gives a first analysis of LS performances on the Horizontal plane. Another value that provides additional information about the performances of the algorithm is the Horizontal 95th percentile: the corresponding value P indicates that the probability that a random value X of the solution is equal or less than this value P is of the 95%. This is valid also for all the other percentiles, like the 50th and 99th percentiles reported in *Figure 5.1.2*, which indicates a value of the 95th percentile of about 20 meters.
As for the HDOP, it can be seen that they reach relatively acceptable values between 1 and 2 and, as before, show absence of solution when there is a "blackout" of signals. Despite this, it is evident that there are peaks throughout the solution which make them reach values over 5: it is plausible that this is due to the constellation of satellites considered, which most likely periodically reaches during the analysis a critical configuration for the determination of user's position. In fact, it can be noticed that even in the Horizontal Error there are peaks of the solution at the same instants of the HDOP ones. Excluding those peaks, the values of the HDOP are similar either with or without errors implementation.

Analogously, *Figure 5.1.3* and *Figure 5.1.4* shows the results obtained considering the Vertical Error (blue circles) and VDOP (green circles). As before, VDOP has already been analyzed, while Vertical Error is determined in the same way as done for the horizontal one, but now considering the vertical plane, i.e., z component:

$$Vertical\ Error = \sqrt{\left(z_{app} - z_u\right)^2}$$

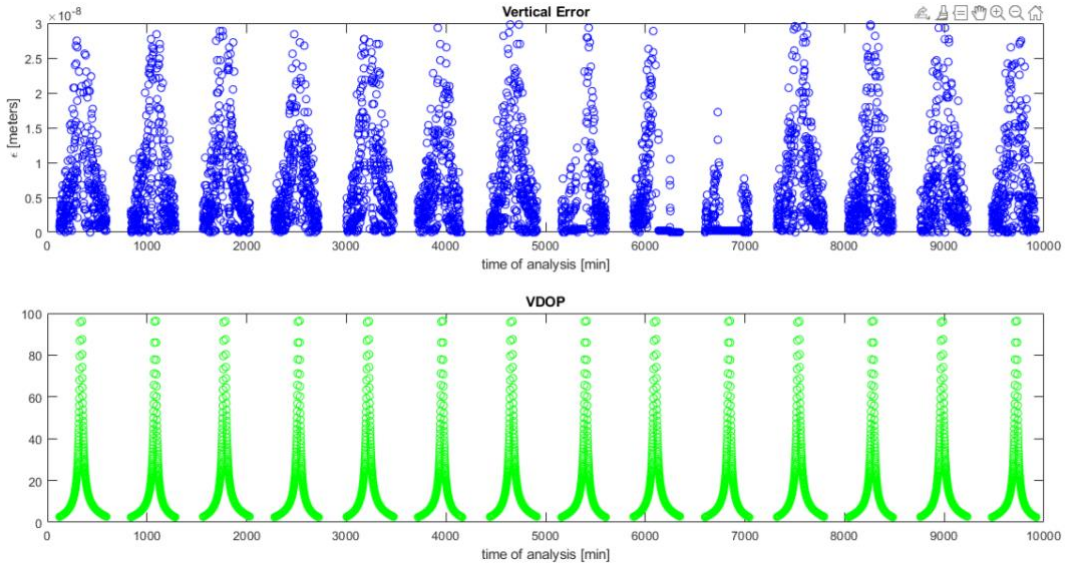

Figure 5.1.3: Vertical error and VDOP without errors implementation

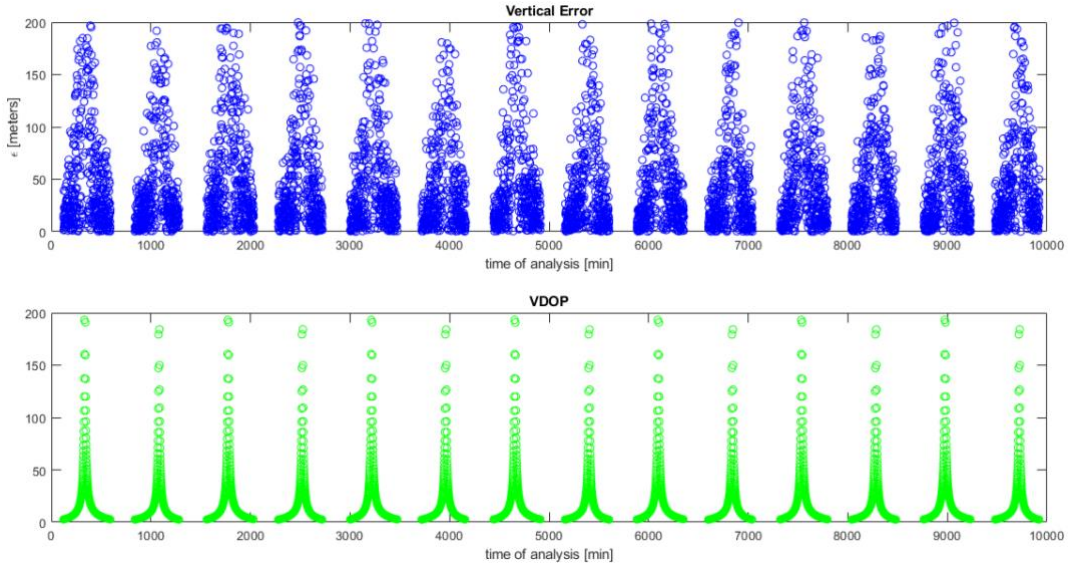

Figure 5.1.4: Vertical error and VDOP with errors implementation

Also for the vertical plane, the values without errors implementation are more accurate, but it can be seen that they are less accurate with respect to the horizontal ones. Without errors *(Figure 5.1.3)*, vertical delta values are about one order higher than horizontal ones, while considering errors application *(Figure 5.1.4)* values of vertical errors are more than four times higher. As well as for the horizontal values, VDOP shows peaks throughout the solution due to the configuration of the satellites in view, but this time the peaks reach values much higher than the HDOP ones.

These results prove the better accuracy of the Least Squares on the horizontal plane compared to the vertical direction, demonstrating also by the values of the Dilution of Precisions, which are worse for the VDOP than the HDOP.

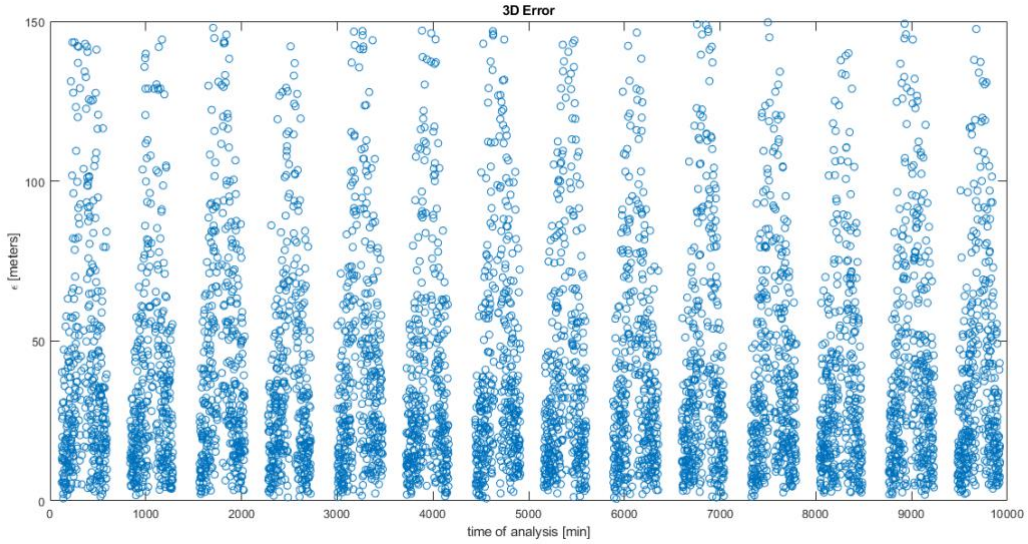

Figure 5.1.5: 3D Position Error for LS implementation over static user

Figure 5.1.5 provide a more general view of the results obtained, depicting the 3D Position Error. It is defined in the same way as done for horizontal and vertical errors, but considering all three components:

$$3D\ Error = \sqrt{\left(x_{app} - x_u\right)^2 + \left(y_{app} - y_u\right)^2 + \left(z_{app} - z_u\right)^2}$$

In this sense, this result can be considered as the combination of horizontal and vertical error. In fact, as the figure shows, the trend of the solution is the same, and the values reached by the errors are in line with the ones observed for the two planes described before.

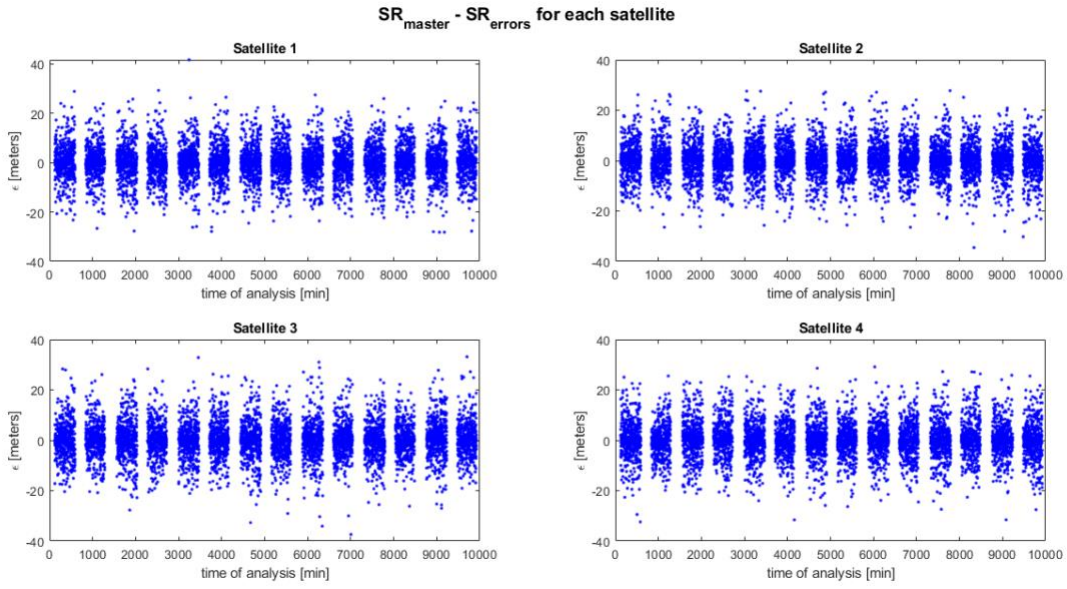

Figure 5.1.6: Effect of the errors implementation on the Slant Range of each satellite

To better understand the effect of the implementation of the errors in the simulation, considering the four satellites in view, *Figure 5.1.6* shows the effect of these errors on the Slant Range measure of each satellite. Since the user is not considered in these measurements, the effect taken into account here are satellite positioning and clock errors. Nevertheless, these calculations are done in the same epochs of the solutions obtained before, so that they will show the same interruptions in the analysis. It can be easily seen that the effect of the errors can have an impact on the results, since it can make the values of Slant Range to draw aside from the real one even of 20 meters, which is in any case a relative value compared to the Slant Range values that are of the order of 10^7 meters.

5.1.2. Static Kalman Filter

The following results will now focus on the implementation of the Extended Kalman Filter technique with a Static Model of motion. Similarly to Least Squares results, the figures will show the comparison of the results obtained with and without errors implementation.
The *Figure 5.1.7* shows the trend of the solution, which is different from the LS results, since the Extended Kalman Filter functioning brings to light the convergence of the solution over time, differently from the LS which at each epoch consider different iterations to converge to the solution at that specific epoch. The initialization in this case is no longer *[0,0,0]*, but it is considered to be the result obtained after two iterations of the Least Squares technique. This is because, as said before, the EKF shows convergence over time, so that in the first phases of the analysis the results will be less accurate, but after very few epochs the filter immediately converges to more accurate values. This is valid every time there is a blackout of signals (i.e., when one or more satellites are not in view for more subsequent epochs), where the filter is again reinitialized with two iterations of LS, but shortly after that immediately converges again. As shown in *Figure 5.1.7*, in the first epochs both horizontal and vertical errors reach values slightly higher than the LS ones, but then after very few epochs (*Figure 5.1.8*) they both converge to errors of order of 10^{-9} meters and oscillate around these values for the time of analysis in which the satellites are in view.

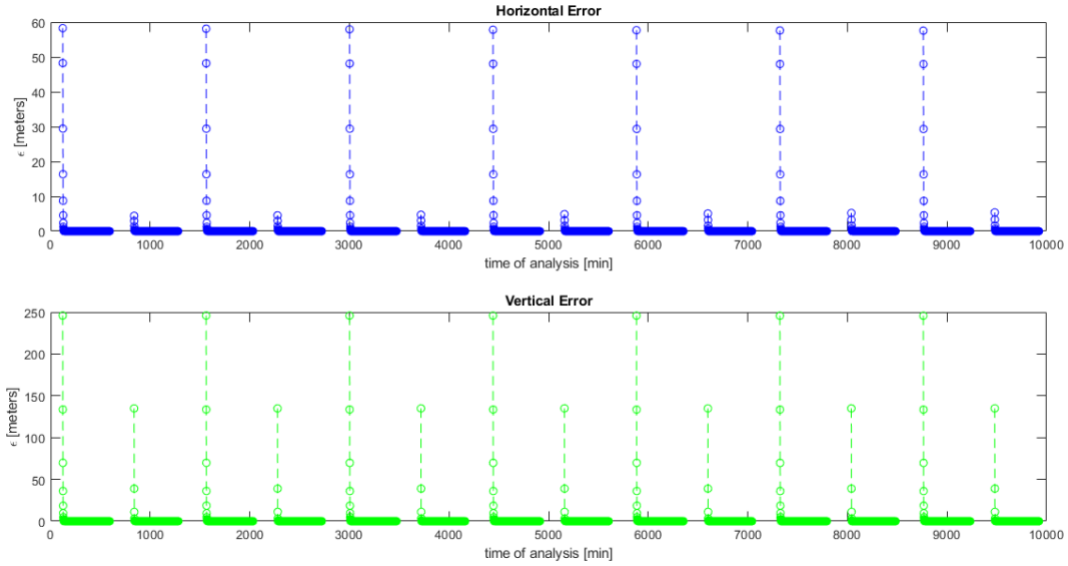

Figure 5.1.7: Trend of the EKF results with static model without errors implementation

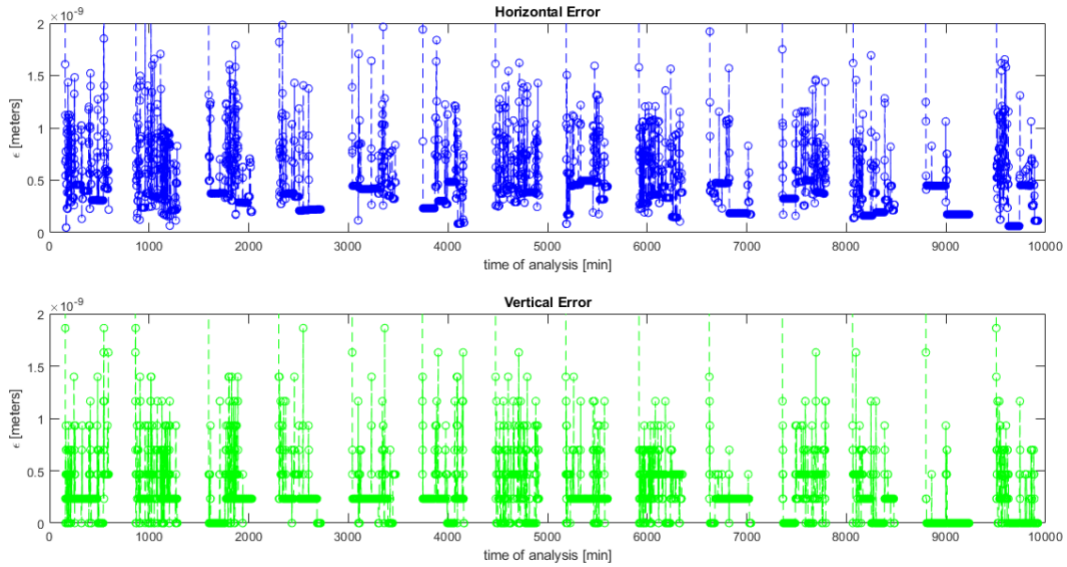

Figure 5.1.8: Zoom of the EKF results with static model without errors implementation

Considering errors implementation, the results of course are worse, but better if compared to Least Squares. In particular, excluding the transitory parts as before and focusing on the phases in which the filter converges, the *Figure 5.1.9* shows that the values of error obtained are less than 15 meters, both for horizontal and vertical error. In the figure the values of the percentiles are also reported: considering the horizontal error, the 95th percentile is about 10 meters, which means half the value obtained with the LS. The other percentile values are also better, demonstrating that the Extended Kalman Filter proves to be much more accurate than the Least Squares technique, both in term of higher convergence rate of the solution and deviation of the results from the reference values.

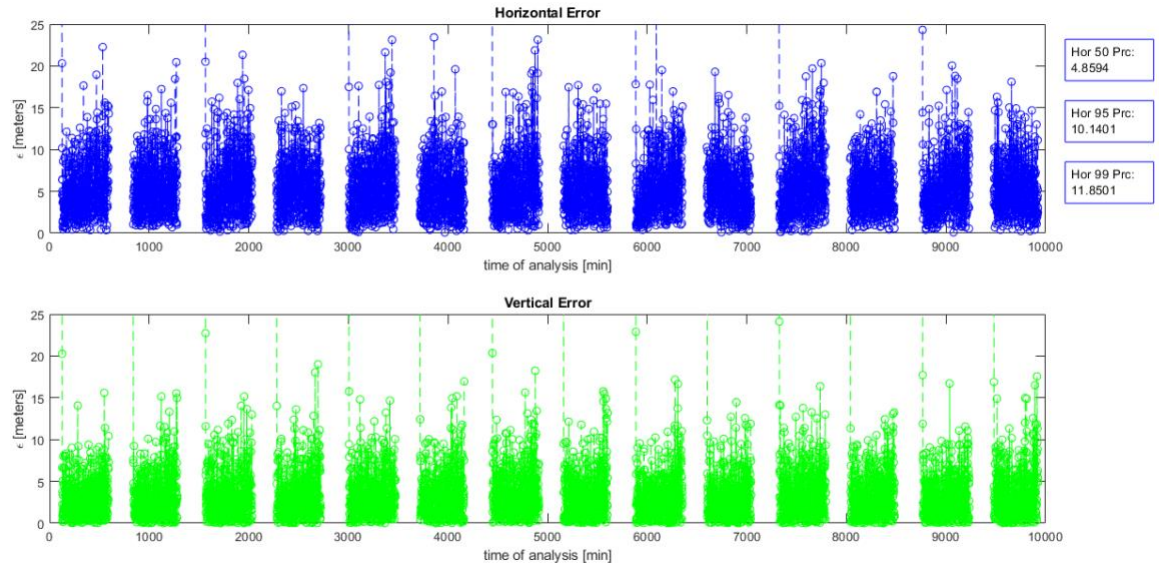

Figure 5.1.9: Zoom of the EKF results with static model with errors implementation

As for the Least Squares, *Figure 5.1.10* shows the 3D Position Errors to give a more general view of the errors obtained with the implementation of the Extended Kalman Filter with static model. As expected, the figure corresponds to a combination of the horizontal and vertical errors, showing the same trend and similar values of error.

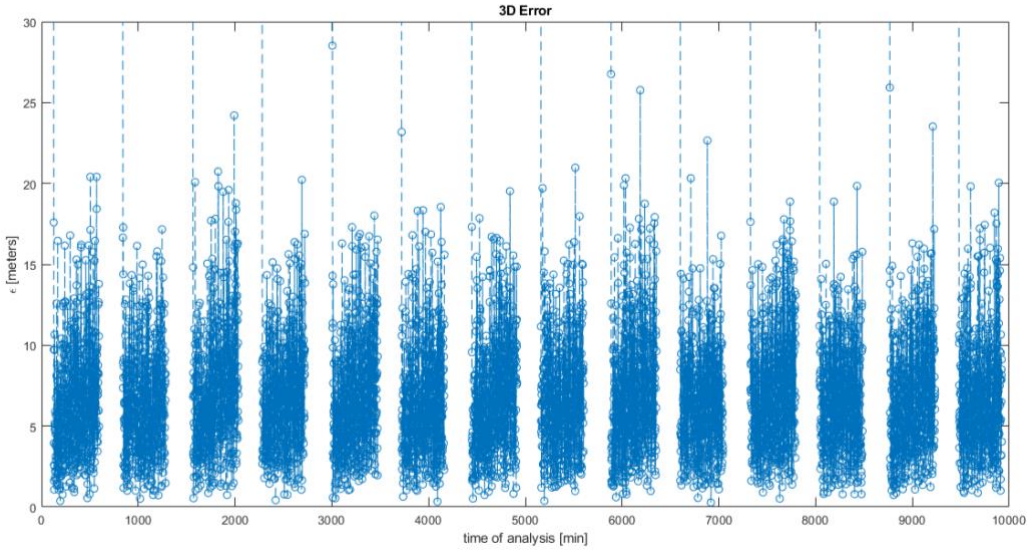

Figure 5.1.10: 3D Position Error for EKF with static model over static user

5.1.3. Dynamic Kalman Filter

As described in chapter 4, the Extended Kalman Filter can be implemented also with a dynamic model of motion. Precisely, in this section the results considering a nearly constant velocity model will be analyzed, again comparing the presence or absence of errors in the implementation.
Regarding the results obtained without errors, the trend (*Figure 5.1.11)* is the same as the static filter: every time there are no sufficient satellites in view for subsequent epochs, the filter is reinitialized with two LS iterations and so there is a transient with higher values, but shortly after that the filter immediately converges to more accurate values.
However, these results show some differences with respect to the ones obtained with the static filter: excluding the transient, *Figure 5.1.12* highlights that the values of both the horizontal and vertical errors oscillate around 0.1 meters and, when the filter converges totally, they reach at most values of 10^{-5} meters (compared to 10^{-9} of the static filter). Moreover, the solution with a dynamic model converges more slowly, and this is reasonable because in this case the filter requires more time to estimate also the correct velocity of the user.

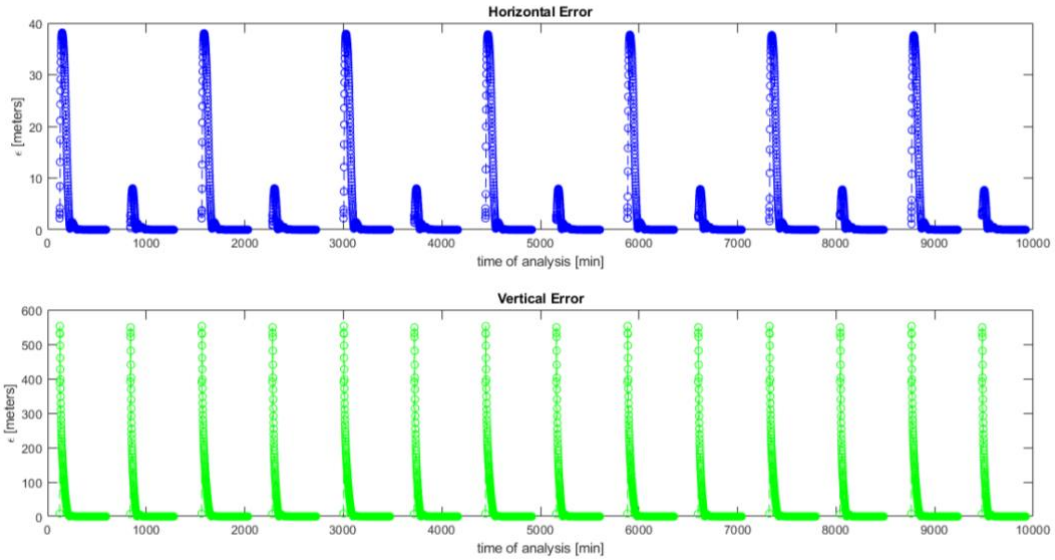

Figure 5.1.11: Trend of EKF results with dynamic model without errors implementation

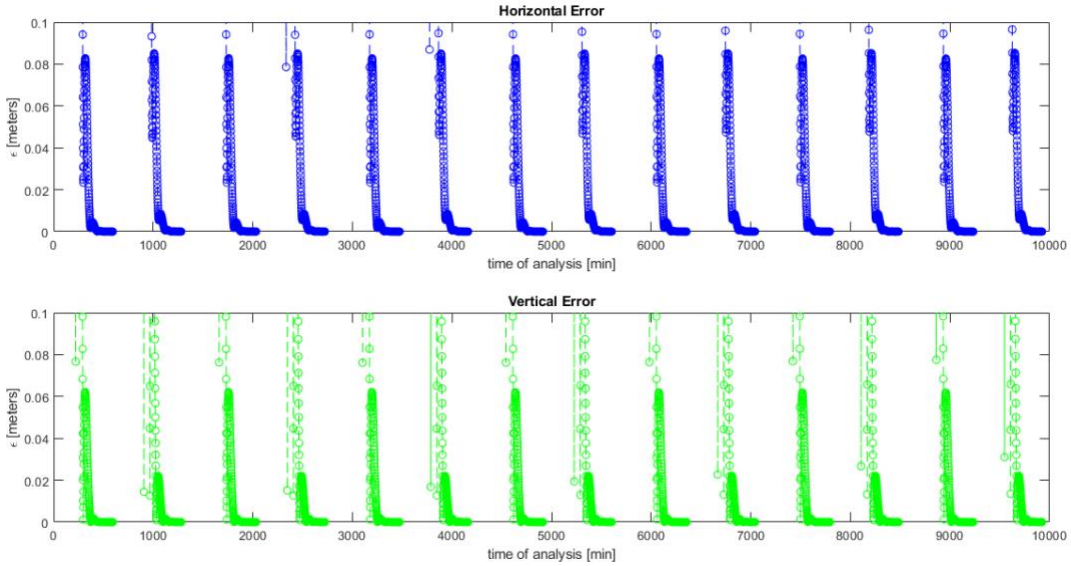

Figure 5.1.12: Zoom of EKF results with dynamic model without errors implementation

Considering errors application (*Figure 5.1.13*), the results are of course worse, but comparable to the ones obtained with the static filter and therefore better than the Least Squares results.
This is clearer observing the percentile values reported in *Figure 5.1.9* for the static filter and *Figure 5.1.13* for the dynamic filter and summarized in *Table 5.1*: in fact, these values show slightly lower 50th and 95th percentiles for the dynamic filter, but higher in the case of the 99th percentile. This outcome confirms what said before, i.e., that the EKF with dynamic model can reach very accurate results as for the static model, but the convergence rate of the solution is slower compared to the EKF implemented with the static model.

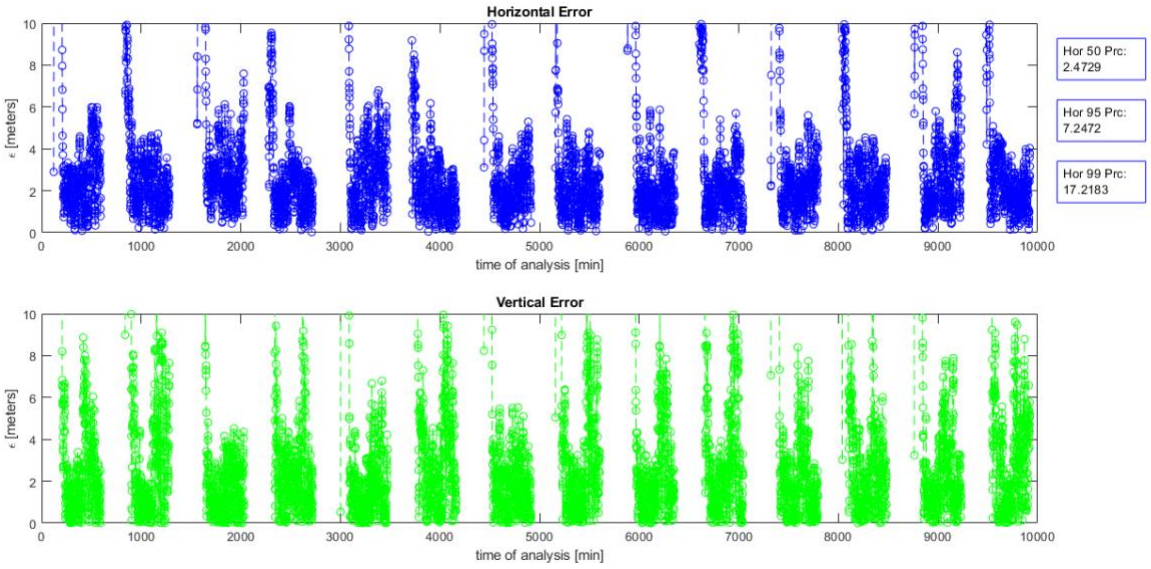

Figure 5.1.13: Zoom of EKF results with dynamic model with errors implementation

Table 5.1: Comparison of the percentile values of Horizontal error for EKF solutions

	Static EKF	***Dynamic EKF***
50^{th} percentile [meters]	4.8594	2.4729
95^{th} percentile [meters]	10.1401	7.2472
99^{th} percentile [meters]	11.8501	17.2183

Finally, also for the dynamic filter, *Figure 5.1.14* shows the 3D Position Error, which again combines horizontal and vertical error.

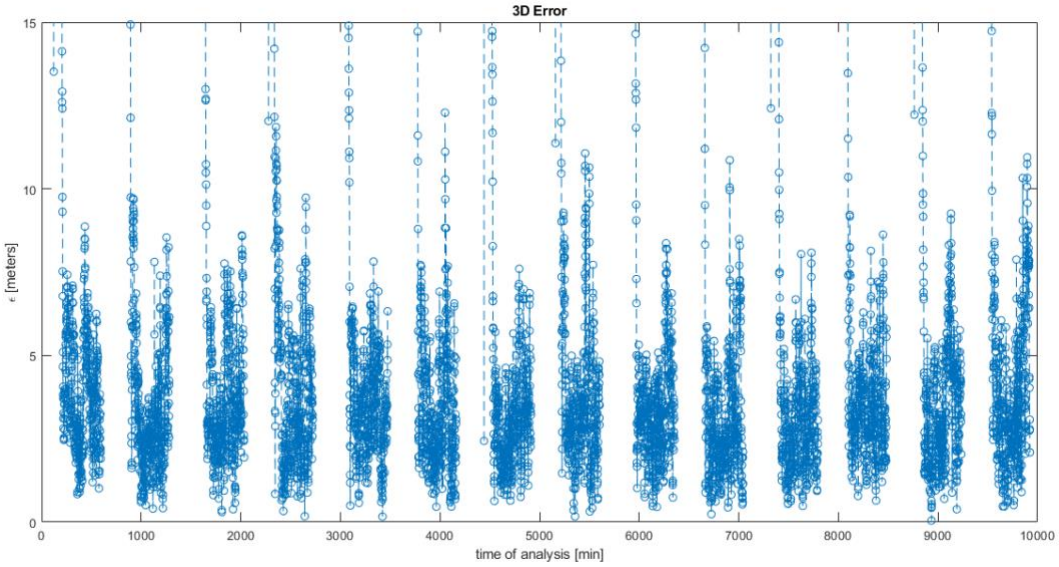

Figure 5.1.14: 3D Position Error for EKF with dynamic model over static user

These results give a first comparison of the two techniques analyzed: it has been demonstrated that the Extended Kalman Filter has better performances with respect to the Least Squares solution in term of accuracy of the position estimation of the user. In the static user scenario, this is true especially if the model applicated to the filter is a static model. This makes sense, since the user is not moving on Moon surface, and so the static model is the most accurate to estimate its position.

5.2. Dynamic User

In this paragraph, the second scenario of analysis presented in section 3.1 will be analyzed, i.e., the implementation of positioning algorithms considering a dynamic user. As already introduced, the user is not in a fixed position, but in motion around the Moon, following a precise trajectory directed to the South Pole, where it will land. In the static user scenario, the consideration of the satellite in visibility were made based on elevation of each one of them with respect to the South Pole. With a dynamic user, this consideration can't be done anymore, so that the check to do to assure the feasibility of the analysis is to verify that the user is at each epoch below all four satellites, so that it can receive the signals necessary for the calculation of the Slant Range.
Unlike the static user scenario, in the case of dynamic user the results without errors implementation will not be considered, since it has already been demonstrated the effect of the errors: obviously, not considering them, the outcomes are better, but they don't reflect reality, so it is more noteworthy to observe more representative results that consider the effects of these errors.
As already mentioned, in this configuration the multipath error will be no longer considered, hence the errors taken into account will be satellite positioning, clock error and receiver error.

The x axis will indicate again the time of analysis, but this time the input data contains measurement with time difference of one second, so that each observation epoch corresponds to one second of the simulation time.
Considering this test case, the implementation of the Extended Kalman Filter with a static model is not really worth mentioning, since the scenario considers a dynamic user, in motion around the Moon and approaching lunar surface. The static model considers a user in a fixed position, so that the results would be practically meaningless.

For this application scenario, an important observation must be made: similar to the static user case, the reference system considered is again centered in the center of the Moon (as also introduced in section 3.1). However, since in the previous case the user is situated on the South Pole, it has been already said that the horizontal and vertical plane of the user correspond to the ones of the reference system. This reasoning is no longer valid considering a dynamic user, since during its orbit around the Moon, the local planes are obviously different from the reference ones. Despite this, given the configuration of the trajectory followed by the user (see figure in section 3.1), in the last phases of the analysis, and in particular in the landing phase, the user is almost aligned with the *z* direction of the reference system, while *x* and *y* components are negligible. This means that in the final phases of the phases the local system again corresponds to the reference one, as for the static user analysis.
These reasonings will be clearer observing the results obtained, especially considering the three components in the landing phase.

5.2.1. Least Squares

Fist results analyzed will be again the ones of Least Square implementation.
One of the most relevant results in this analysis is the 3D Position Error, represented in *Figure 5.2.1*. Its definition has already been explained in section 5.1.1: the reference position $user=[x_u , y_u , z_u]$ now is not a fixed value (as it was the South Pole of the Moon for the static user scenario), but it is an input data that of course change every second, giving the measurement position of the user during the simulation. The approximate position $u_{app}=[x_{app} , y_{app} , z_{app}]$ is again the position computed with the positioning algorithm considered. The 3D error is then computed as:

$$3D\ Error = \sqrt{\left(x_{app} - x_u\right)^2 + \left(y_{app} - y_u\right)^2 + \left(z_{app} - z_u\right)^2}$$

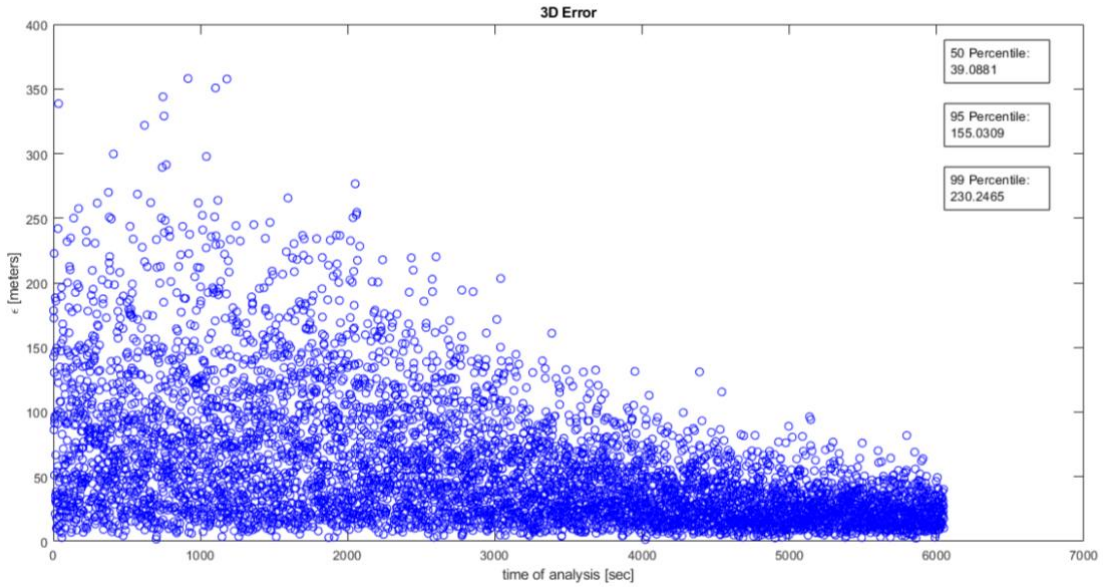

Figure 5.2.1: 3D position errors of LS implementation over dynamic user

Compared to the static user scenario, the trend is now more scattered and shows a slight convergence over time, and there are no interruptions throughout the simulation, showing that the user is always below the four satellites of the constellation given in the input data.
At the beginning of the simulation, the deviation from the reference position is on the order of hundred meters, then converging over time and reaching in the final phases values around 50 meters or less. In order to provide an idea of the distribution of the values, the percentile values are reported in the figure, showing for the 95th percentile a value of about 155 meters. Allegedly, these errors are too much high to provide an accurate estimation, demonstrating the weakness of the Least Squares in this scenario.

As mentioned before, it is of interest to analyze the error in the estimation highlighting the differences among the single components x, y and z. These values can be easily computed as for the 3D error, considering the absolute value of the difference between approximated and reference values:

$$x_{error} = |x_{app} - x_u|$$

$$y_{error} = |y_{app} - y_u|$$

$$z_{error} = |z_{app} - z_u|$$

Figure 5.2.2 displays the same trend of the 3D error for all three components, with a slight difference for the y coordinate, which is more distributed. The figure shows how the values of error on x and y components are overall more than five times less than the ones on z component: for all the simulation time, x and y errors are less than 50 meters reaching values below 20 meters in the final phases (y error is below 20 meters for almost all the time of analysis). Instead, z error reaches values around hundreds meters and then converges to values at most around 35 meters.
This analysis demonstrates that the main problem of the Least Squares estimation is the z component, showing instead better accuracy on the other two directions. This consideration is in part as expected, at least for what concerns the y component: the better accuracy on the y direction can be explained by the fact that the trajectory followed by the user in its approach to the lunar surface is almost exclusively in the x-z plane, while the y component of the motion is always very little compared to the other two. This consideration will be deepened in the next paragraphs.

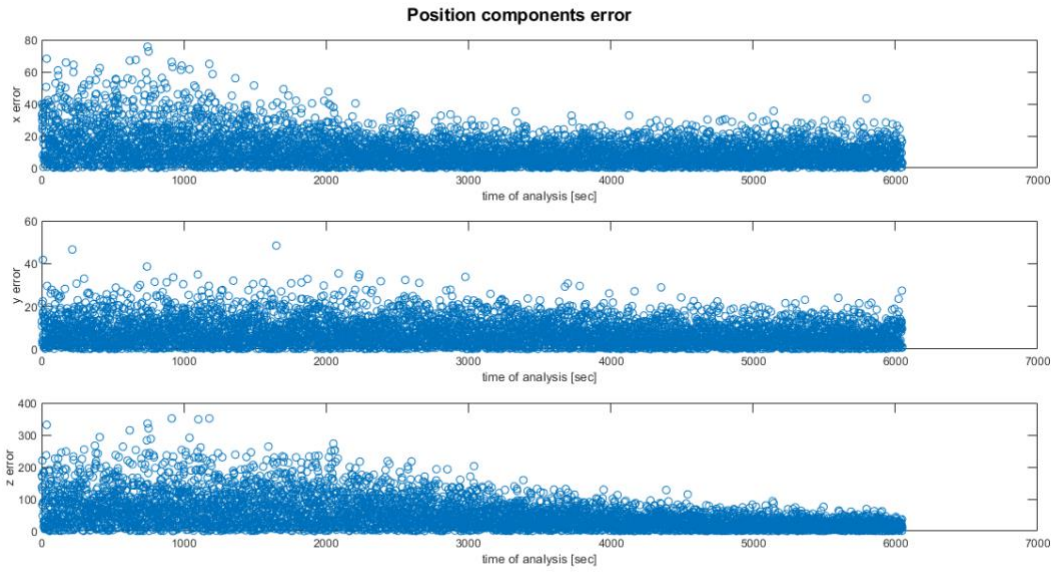

Figure 5.2.2: Position Components Errors for LS implementation with dynamic user

Another important result is the Cumulative Distribution Function of the 3D position error values, depicted in *Figure 5.2.3*. The CDF is a function that describes the probability distribution of a variable (the error values in this case). The CDF of the 3D position error calculated at a value *P* of the solution is the probability that the elements of the error vector will take a value less than or equal to that value *P*. The more the curve is flattened towards the y-axis, the more it has a positive meaning, indicating a better distribution of the values over simulation time.
In this case, the curve is not very flattened because as already seen the values are scattered and the convergence rate of the solution is not very high. This is also confirmed by the values of the 50th, 95th and 99th percentile (red point-lines), which are the same calculated and reported in *Figure 5.2.1*. In fact, the 99th percentile is almost more than 6 times the 50th one, showing again that the distribution of results is very scattered.

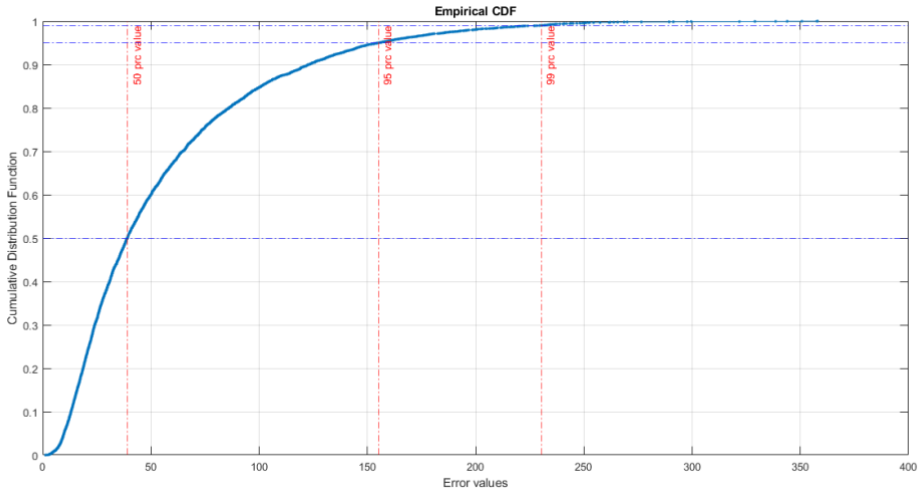

Figure 5.2.3: CDF plot of the 3D position error values for LS solution

Finally, in *Figure 5.2.4*, the velocity and acceleration profiles are reported. Cyan and green circles represent respectively user velocity and acceleration: these are the reference values, given the velocity of the user from the input data and acceleration calculated with the differential of the same velocity values known. Magenta and yellow circles, instead, represent the approximated velocity and acceleration. Since the LS algorithm does not include velocity and acceleration calculation, the approximated values are calculated with the differential of the approximated position determined by the algorithm. Given the user approximated position components at each epoch x_{app} , y_{app} and z_{app}, velocity and acceleration are computed as:

$$vel_{app} = \sqrt{{v_x}^2 + {v_y}^2 + {v_z}^2} \qquad acc_{app} = \frac{\partial vel_{app}}{\partial t}$$

Where $v_x = \frac{\partial x_{app}}{\partial t}$, $v_y = \frac{\partial y_{app}}{\partial t}$ and $v_z = \frac{\partial z_{app}}{\partial t}$.
Reference profiles, instead, are defined starting from the reference velocity $v_{user}=[v_{xu}\ ,\ v_{yu}\ ,\ v_{zu}]$ reported in the input observation file:

$$vel_u = \sqrt{{v_{xu}}^2 + {v_{yu}}^2 + {v_{zu}}^2} \qquad acc_u = \frac{\partial vel_u}{\partial t}$$

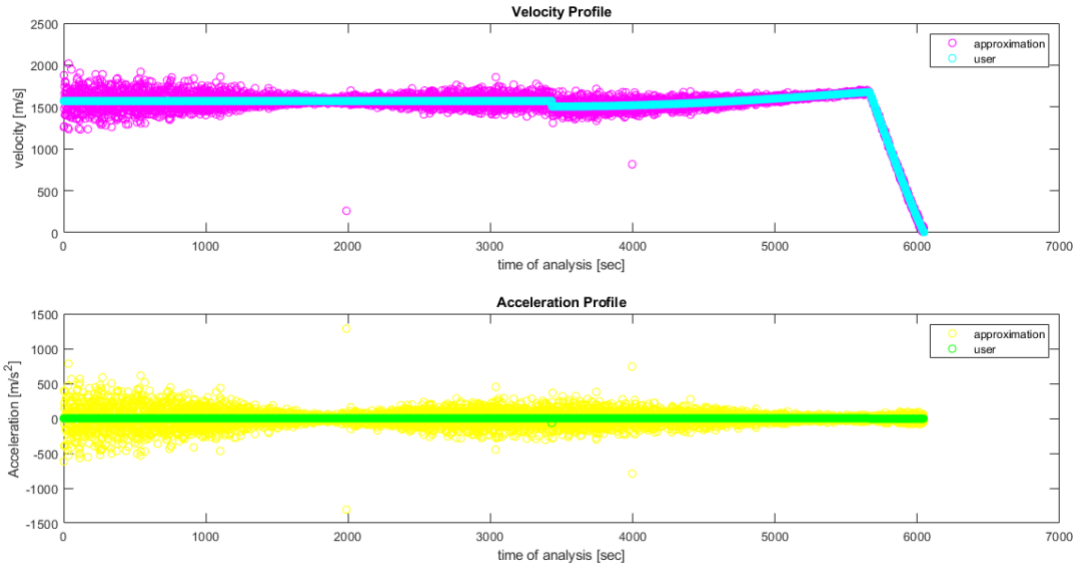

Figure 5.2.4: Velocity and Acceleration profiles for LS analysis

User velocity (cyan circles) is around 1500 m/s for almost all the simulation time, until the final phases in which suddenly decreases toward zero: this descending phase of course corresponds to the landing of the user on the Moon surface. User acceleration (green circles) seems to be null for all the simulation, but it will be clearer in the following paragraphs that in reality there are some variations in acceleration values.
Velocity approximation (magenta circles) demonstrates to reach values that oscillate around the user reference, with fluctuations less than 500 meters per second until the descending phase in which the approximation is more accurate and closer to the user velocity (errors are about 50 meters per seconds). The approximation on acceleration (yellow circles) follows more or less the same reasoning, showing fluctuation of about 500 m/s^2 especially in the first phases of simulation, until the descending phase in which they are reduced to about 100 meters per seconds square.

There is another important consideration: *Figure 5.2.4* shows a step in the velocity profile after about 3400 seconds (precisely at epoch 3431), in which the velocity suddenly decreases of about 70 m/s (from 1570 to 1500 m/s). Parallelly, at this epoch corresponds an anomaly in the acceleration trend, as the value of acceleration is about -65 m/s^2, while the trend is always around zero. This anomaly is present in input data, so that it can be considered as a maneuver to adjust the lander attitude during its trajectory around the Moon. This maneuver causes some modification in the results, which are more evident in the Extended Kalman Filter solution as it will be seen later.

5.2.2. Dynamic Kalman Filter

Implementation of a dynamic model for the Extended Kalman Filter can provide different results based on the model considered. In the following section, two different model will be analyzed: Nearly Constant Velocity Model (NCV) and Nearly Constant Acceleration Model (NCA).

5.2.2.1. Nearly Constant Velocity Model

The first model implemented is the constant velocity model, already introduced in section 4.2.2.
Considering a dynamic model, the 3D position errors (*Figure 5.2.5*) shows that the trend of the solution is similar to what was obtained in the static user case. There is, in fact, a transitory part at the beginning of the analysis in which the solution reaches high values of error, but this is true just for very few epochs after which the filter immediately converges to much better values.

Differently from the static user scenario, in this case there are no blackout of signals during the simulation (as already mentioned for LS), so that the solution continues to converge for all the time of analysis, giving much better results. Since this trend is common for all the results obtained in this analysis, to better analyze the outcomes, the following figures will show a zoom of the solution, focusing on the second part of the analysis, therefore excluding the transient at the beginning.

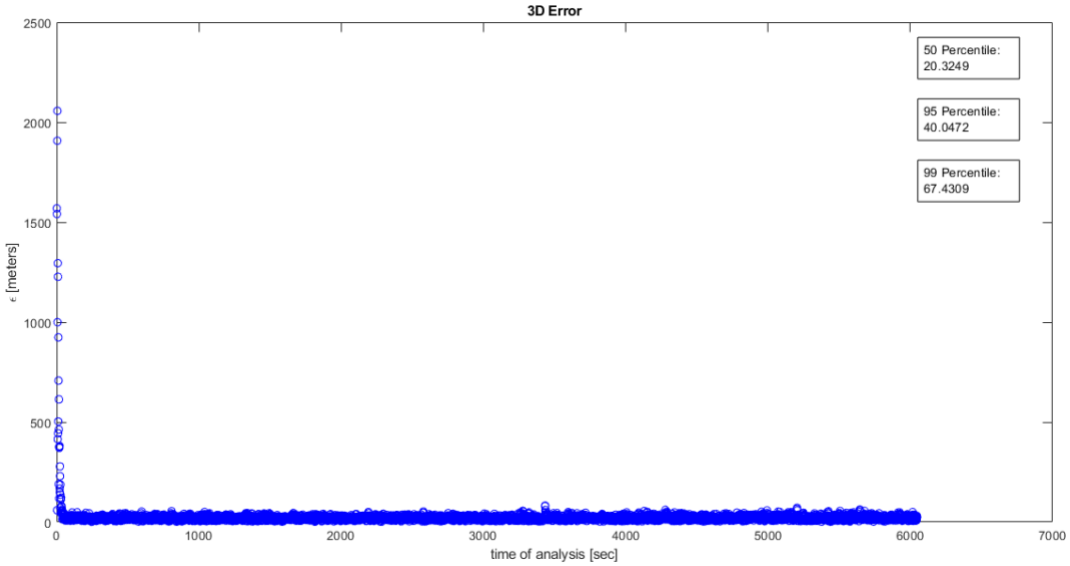

Figure 5.2.5: Trend of 3D position errors for EKF with constant velocity model

As shown in *Figure 5.2.6*, 3D positions errors reach values lower than 50 meters after very few epochs and remain so oscillating more or less between 0 and 30 meters. In LS solution, the results obtained in the final phases were not too much worse (the errors were about 50 meters), but the main difference to be seen is that the Extended Kalman Filter converges much earlier, reaching better accuracy already after very few seconds of simulation. This result is confirmed by observing the value of the 95th percentile, which is now about 39 meters, a value almost four time lower than that obtained with LS (about 155 meters), confirming the higher convergence velocity of the EKF.

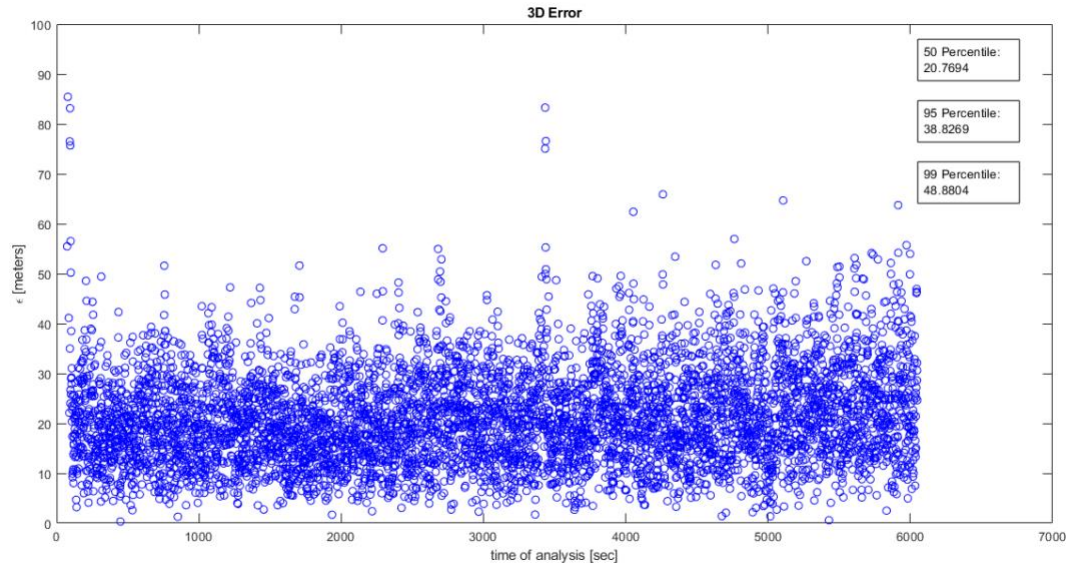

Figure 5.2.6: Zoom of 3D position errors (excluding transient) for EKF with constant velocity model

Although not very clearly, in the figure it can also be seen the effect of the anomaly in the velocity and acceleration profiles, already described in the previous paragraphs. In fact, the sudden change in the acceleration value cause a glitch in the functioning of the filter, which produces a peak of about 80 meters (visible around epoch 3431), but immediately after that, the filter recovers and converges again to the previous values.
The *Table 5.2* summarizes the value of the percentile considering the transitory (as in *Figure 5.2.5*) or not considering the transitory (as in *Figure 5.2.6*). This comparison underlines once again the strength of the dynamic EKF, since both the 50^{th} and 95^{th} percentile are under 40 meters. Of course, the main difference is the 99^{th} percentile, which has higher value considering the transitory, but excluding it the result reach a value around 49 meters, given a much better accuracy with respect to the results seen for the Least Squares.

Table 5.2: Comparison of the percentile values of 3D position error of EKF solution

	WITH Transitory	***WITOUTH Transitory***
50^{th} percentile [meters]	20.3249	20.7694
95^{th} percentile [meters]	40.0472	38.8269
99^{th} percentile [meters]	67.4309	48.8804

Implementing a dynamic model, the Extended Kalman Filter is able to also determine the velocity components of the user, so that a comparison with the reference value can be made. Therefore, analogously to what done for the position, considering $v_{user}=[vx_u , vy_u , vz_u]$ the reference velocity of the user reported in the input observation file and $v_{app}=[vx_{app} , vy_{app} , vz_{app}]$ the approximated velocity calculated with the filter, the 3D velocity error can be computed as:

$$3D\ Velocity\ Error = \sqrt{\left(vx_{app} - vx_u\right)^2 + \left(vy_{app} - vy_u\right)^2 + \left(vz_{app} - vz_u\right)^2}$$

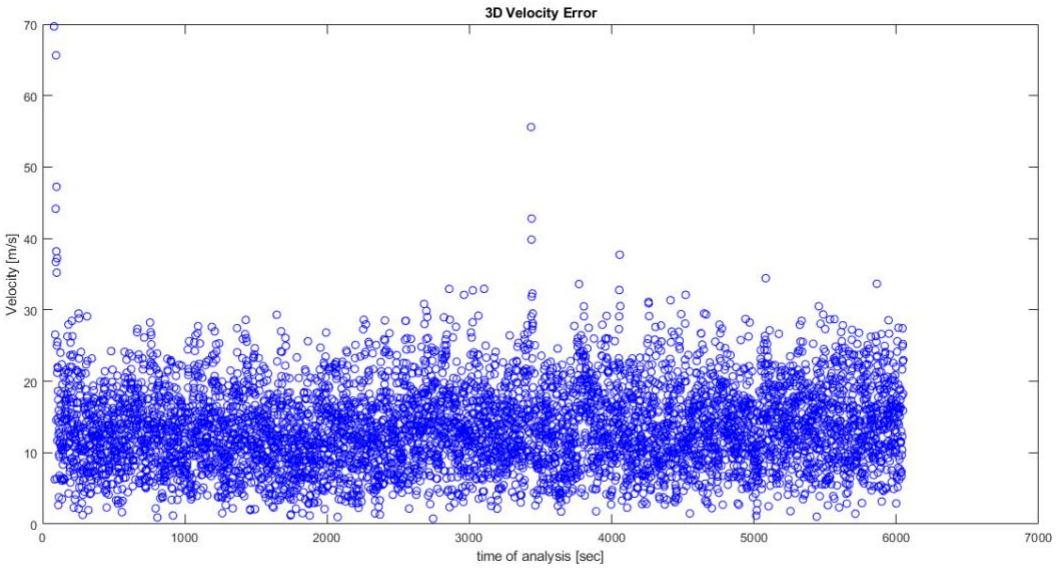

Figure 5.2.7: 3D velocity errors (excluding transient) for EKF with constant velocity model

As already said, the trend of the 3D velocity error (*Figure 5.2.7*) is the similar to the position one, since it reaches better accuracy after the transitory. In line with the result obtained for the position, the errors of the velocity estimation compared to the reference values show an oscillation around values lower than 30 m/s. Also in this graph, the peak at epoch 3431 is visible, and corresponds for the velocity to a glitch that reach a value of about 55 meters per second.

The same considerations are valid for errors on each component: position errors component by component has already been introduced in LS analysis, while the velocity error can be defined in a similar way, considering user velocity $v_{user}=[vx_u , vy_u , vz_u]$ as the reference velocity reported in the input observation file and the approximated velocity $v_{app}=[vx_{app} , vy_{app} , vz_{app}]$ the one calculated with the filter:

$$vx_{error} = \left| vx_{app} - vx_u \right|$$

$$vy_{error} = \left| y_{app} - vy_u \right|$$

$$vz_{error} = \left| vz_{app} - vz_u \right|$$

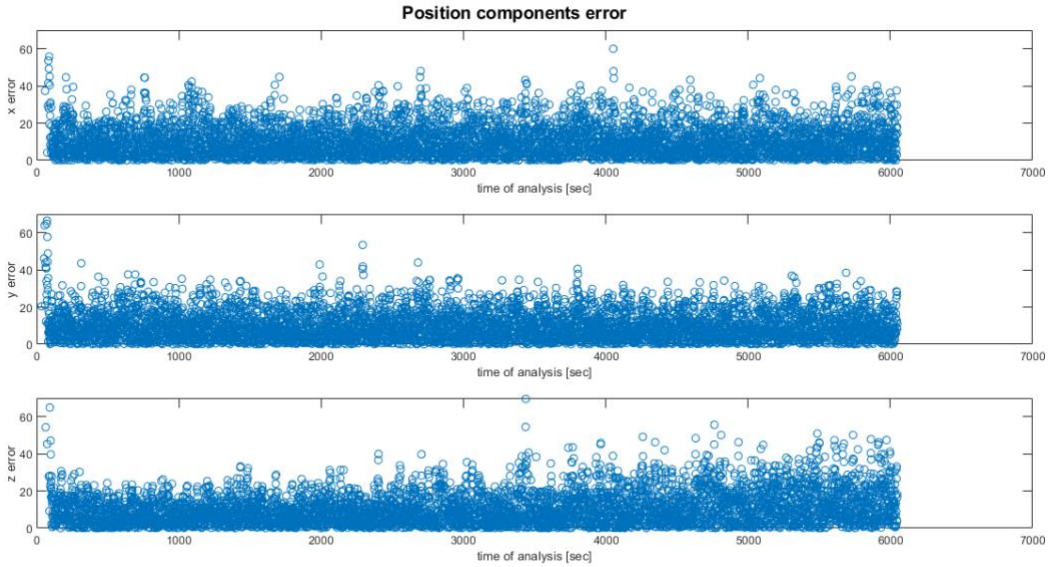

Figure 5.2.8: Position components errors (excluding transient) for EKF with constant velocity model

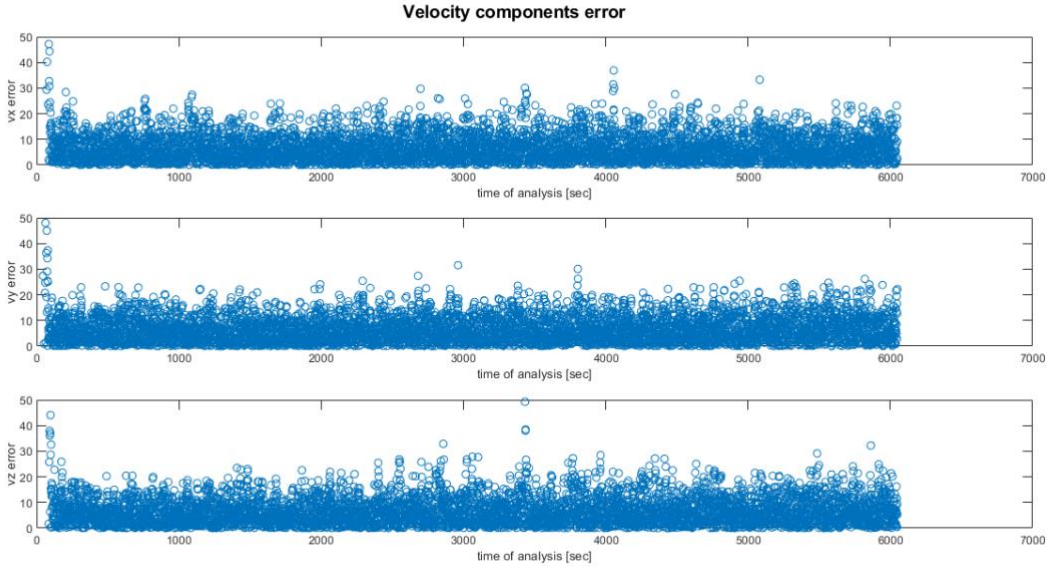

Figure 5.2.9: Velocity components errors (excluding transient) for EKF with constant velocity model

Differently from what already seen previously, implementing the EKF with dynamic model, the difference among the three component is not so evident. Indeed, the oscillations of the position errors on each component (*Figure 5.2.8*) are more or less equivalent, around the same values approximately between 0 and 30 meters, or in any case hardly above 40 meters. As for the velocity components error (*Figure 5.2.9*) the estimations are even more accurate, since deviations from the reference values are hardly above 20 m/s. This means that the dynamic model is able to better manage the position and speed of the user along all three directions of motion, unlike what was observed for the Least Squares implementation, which was more performing on *x* and *y* components with respect to the *z* direction.

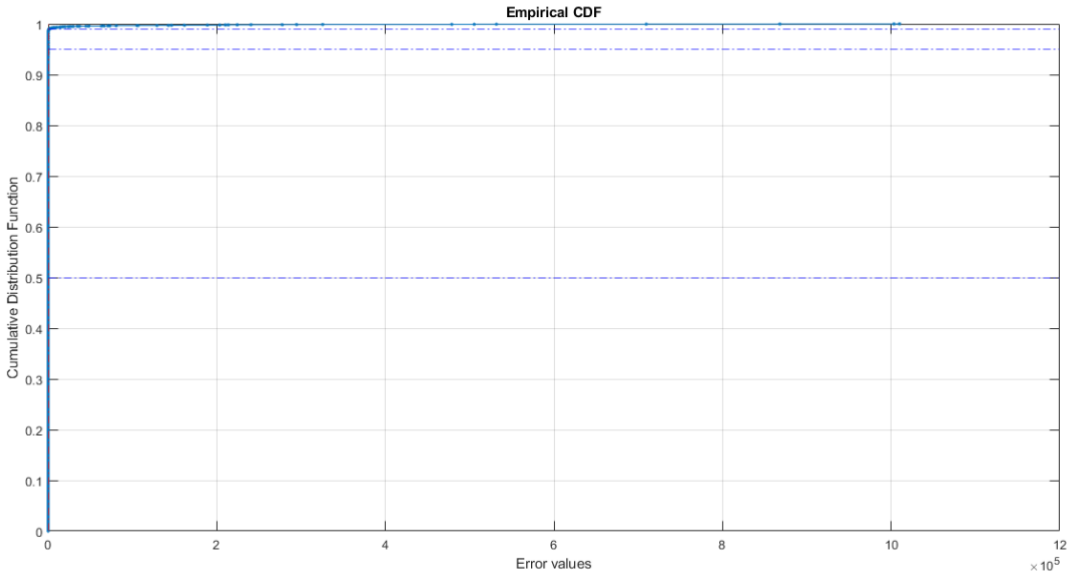

Figure 5.2.10: CDF plot of the 3D position error values for EKF with constant velocity model

Figure 5.2.10 shows the Cumulative Distribution Function of the 3D position errors for the EKF solution with constant velocity model. As it can easily be seen, the graph is much more flattened, which has a positive meaning, since it indicates greater accuracy in the results throughout the analysis time. In fact, the values of the 50th, 95th and 99th percentile are confused with each other, since the vertical lines that intercept the graph at these points (displayed in red point-line strokes in the previous paragraphs) are all crushed close to the y-axis. In any case, as already shown in *Table 5.2*, all these values are below the 50 meters not considering the transitory part at the beginning.

Finally, also for the dynamic model, the velocity and acceleration profile are analyzed. As before, in *Figure 5.2.11* cyan and green circles represent the reference values, while magenta and yellow circles represent the approximated values determined with the Extended Kalman Filter algorithm.
Since the model applicated is nearly constant velocity, in this case velocity approximation are calculated by the algorithm, while acceleration approximations are determined as already introduced for the Least Squares:

$$acc_{app} = \frac{\partial vel_{app}}{\partial t}$$

Excluding the transient at the beginning, the velocity approximation is really accurate compared to the previous results of Least Squares. In fact, the deviation from the reference values is not above 20 m/s for almost all the time of analysis, until the descending phase where the difference is even lower, of just few meters per seconds.

On the contrary, the approximate acceleration is not so accurate, reaching oscillation up to 20 m/s^2 for all the analysis, even though these values are way better than the ones obtained with the previous techniques. This is because the model considered is a nearly constant velocity model, which inevitably leads to more realistic results considering the velocity approximation, but not for the acceleration approximation.
Therefore, the filter analysis still needs another additional step to reach more accurate results, implementing a constant acceleration model.

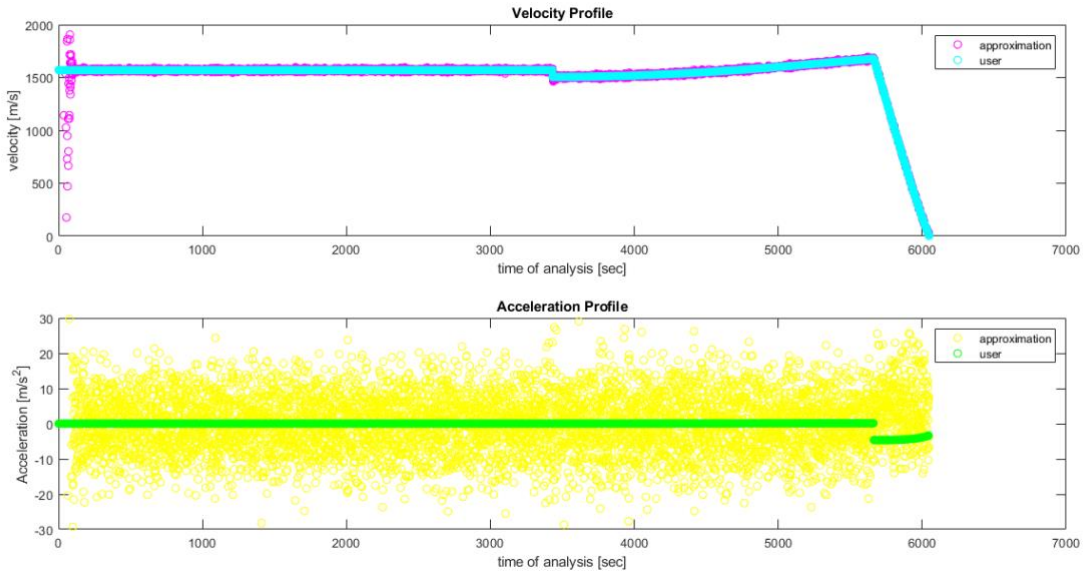

Figure 5.2.11: Velocity and Acceleration profiles for EKF with constant velocity model

The acceleration profile in *Figure 5.2.11* allows to make an observation that the other figures in previous paragraphs made not so clear: in the final phases of the simulation (around epoch 3660) there is a step in the profile (green circles) which brings the acceleration from values near zero to a value about -5 m/s^2 and then growing back to about -3 m/s^2 near the end of the analysis. It is easily deductible that this step corresponds to the maneuver made by the user to land on the Moon surface. In fact, as it can be seen from the velocity profile, starting from the same epoch of the acceleration step (epoch 3660), the velocity of the user suddenly decreases from about 1500 m/s toward 0. Until this maneuver, the user is in motion with a model of motion that is almost the same of the constant velocity model, since the acceleration is almost zero. Actually, the acceleration is not zero, but shows values of the order of 10^{-5} m/s^2 until the first maneuver after 3431 seconds (already discussed in the previous paragraphs), after which acceleration increases to values around 10^{-2} m/s^2. In any case, once the lander starts the descending phase, the values of acceleration are not so negligible and velocity starts to decrease very rapidly, so that the constant velocity model cannot accurately simulate user movement.
This consideration will be even more clear analyzing the results obtained with the implementation of the Extended Kalman Filter with constant acceleration model.

5.2.2.2. Nearly Constant Acceleration Model

The second model implemented in this analysis is the constant acceleration model (NCA).
Similarly to the nearly constant velocity model, being NCA also a dynamic model, the trend is similar to the case of the static user, as shown in *Figure 5.2.12*. Excluding the transient at the beginning, the EKF converges very rapidly to very accurate values, so that, as done for the NCV model, also for this scenario the analysis of results will be carried out without considering the initial transient.

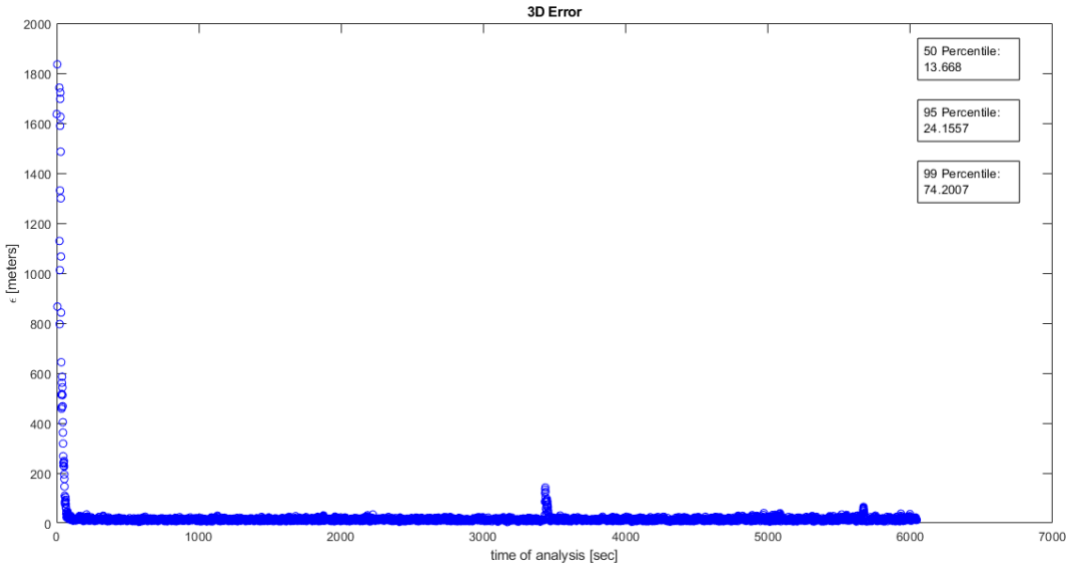

Figure 5.2.12: Trend of 3D position errors for EKF with constant acceleration model

The solution displayed in *Figure 5.2.13* converges again very rapidly to values more accurate than those of the constant velocity model, given that the errors are hardly above 25 meters (in *Figure 5.2.7* errors reached also values of 50 meters). The high convergence rate is confirmed also in this analysis by the percentile values represented in figure, showing a 95th percentile value of about 23 meters, which means almost half the value of the NCV model (40 meters).

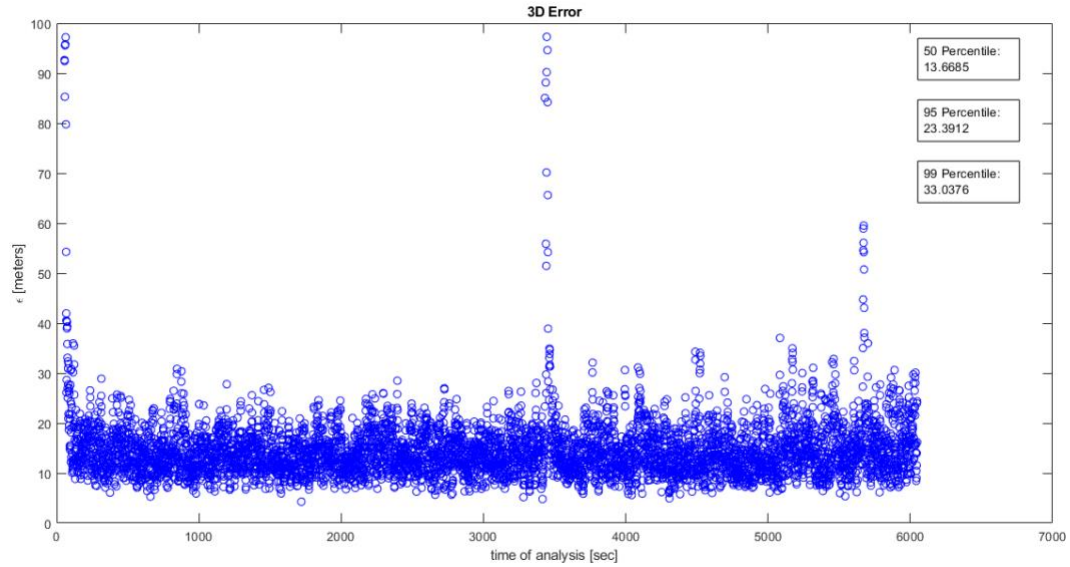

Figure 5.2.13: Zoom of 3D position errors (excluding transient) for EKF with constant acceleration model

In this case, *Figure 5.2.13* shows more clearly the effect of the maneuver made about halfway through the simulation. In fact, the peak of the 3D position error reaches almost values of 100 meters for few seconds, which is just 10 meters more than before, but the difference is clearer since the convergence values of this model are lower than the NCV model. Once again, after this peak, the filter immediately recovers and converges again to the previous values.
Moreover, in this case the glitch related to the maneuver made at the beginning of the descent on lunar surface is also clearly visible, corresponding to a peak of about 60 meters error.

The *Table 5.3* compares the values of the percentiles already reported in *Figure 5.2.12* and *Figure 5.2.13*. First of all, it can be noticed that all the values are lower compared to the respectively values obtained with the constant velocity model (recalled in yellow in the table): excluding the transitory part there is a difference of almost 15 meters both for 95th and 99th percentiles. For the latter, the value obtained in the current test case is about 33 meters (compared to the almost 49 of the previous test case). Moreover, it can be seen that the value of 50th and 95th percentiles change very little whether or not the transient is considered, especially the 50th percentile which is practically the same.

Table 5.3: Comparison of the percentile values of 3D position error of EKF with constant acceleration model

	WITH Transitory	***WITHOUT Transitory***	***NCV without transient***
50th percentile [meters]	13.668	13.6685	20.7694
95th percentile [meters]	24.1557	23.3912	38.8269
99th percentile [meters]	74.2007	33.0376	48.8804

These values demonstrated the greater accuracy of the nearly constant acceleration model, since the convergence rate is higher than the nearly constant velocity model, thus giving overall more accurate results in less time.
These considerations are confirmed also seeing the 3D velocity error (*Figure 5.2.14*), computed as already shown in the NCV model case.

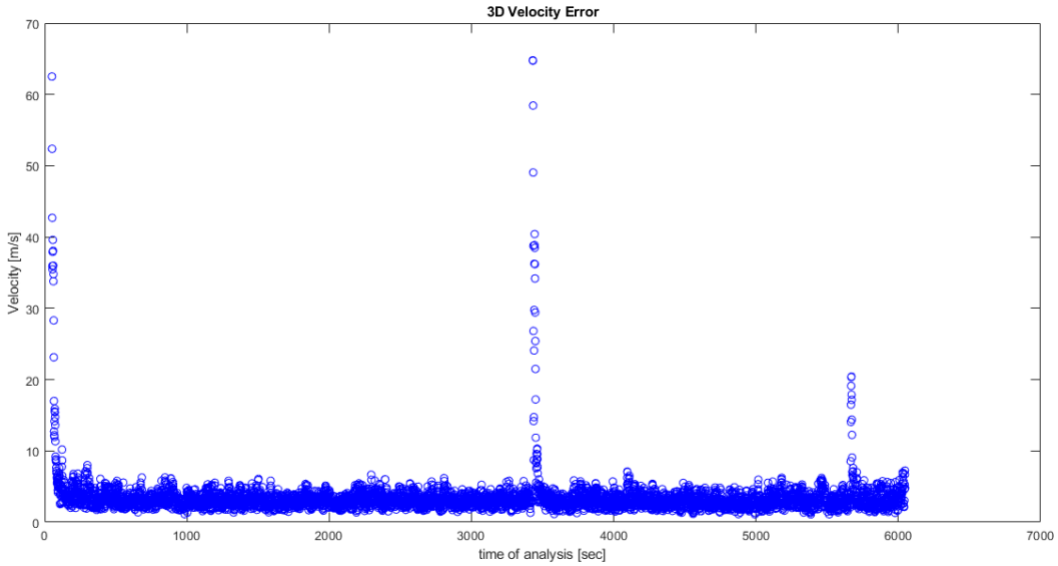

Figure 5.2.14: 3D velocity error (excluding transient) for EKF with constant acceleration model

The 3D velocity error emphasizes the greater accuracy of the nearly constant acceleration model, since the errors in velocity calculation are way lower than obtained before, oscillating between values lower than 8 m/s (compared to the 30 m/s of the constant velocity model). Given these lower values, the peak at epoch 3431, which has again a value of about 65 m/s, is much more visible. Same reasoning is valid for the glitch related to the descending maneuver, corresponding to an error in velocity estimation of about 20 m/s.

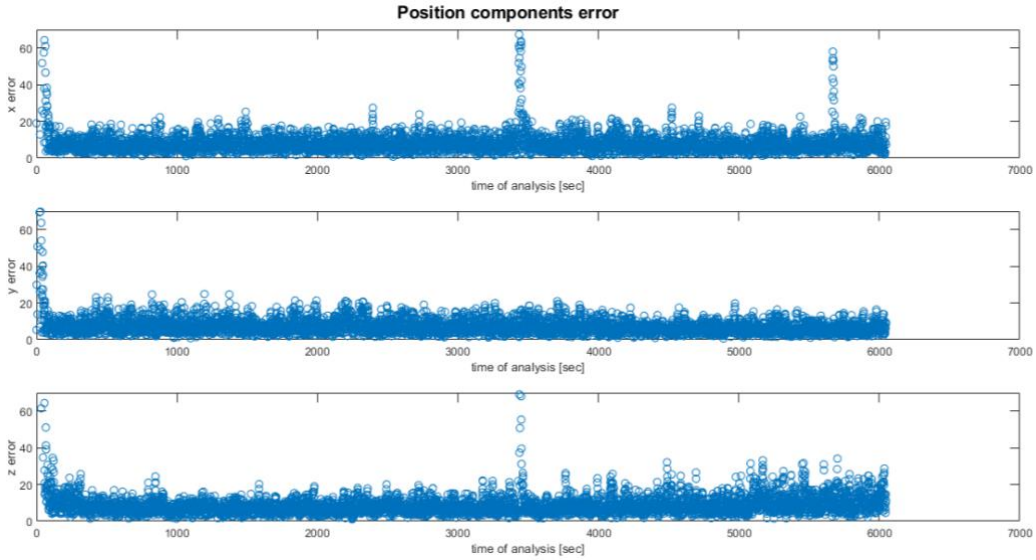

Figure 5.2.15: Position components errors (excluding transient) for EKF with constant velocity model

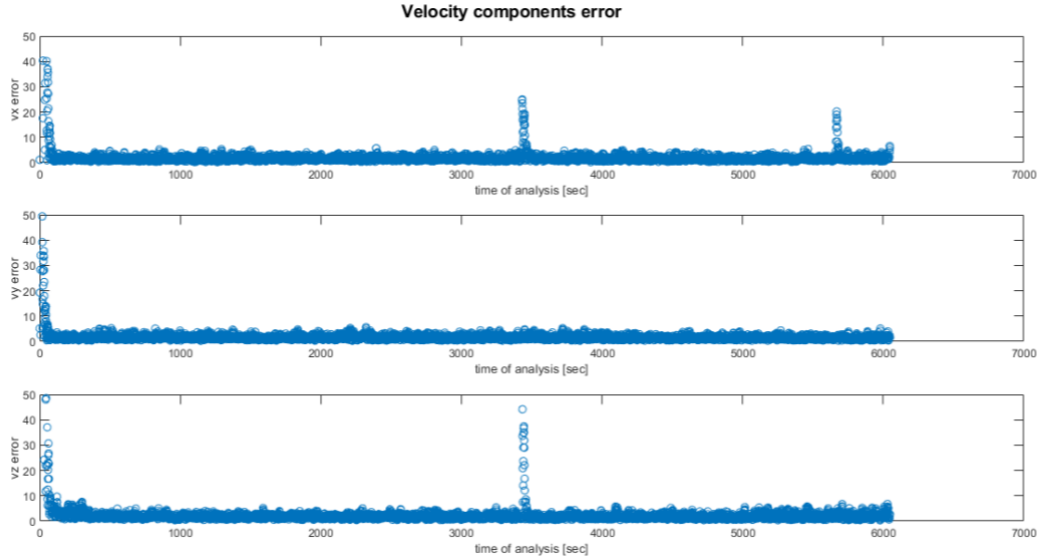

Figure 5.2.16: Velocity components errors (excluding transient) for EKF with constant velocity model

Focusing on the single components, as for the NCV model, the difference among the three direction is negligible, since all three position errors (*Figure 5.2.15*) oscillate around values lower than 20 m, again more accurate than the previous case where the errors often exceed 30 meters.
The same results are pointed out in *Figure 5.2.16*, where velocity components errors do not exceed 5 m/s, four times lower than the nearly constant velocity model errors.
An important consideration can be made observing both figures: the maneuver done halfway through the simulation has an impact on the *x* component (which reaches errors of about 67 m on position and 25 m/s on velocity) and on the *z* component (which reaches errors of even 130 m and 60 m/s). Instead, it has no effect on the *y* component, which does not have a glitch at that epoch. Focusing on the final phase, the maneuver done to start the descent to lunar surface has a major impact only on *x* component, giving errors of about 60 meters on position and 20 m/s on velocity.
This analysis allows to deduce that the maneuver done to settle the attitude of the user at epoch 3431 involves a modification of direction on the *x*-*z* plane, while the maneuver done to start the descend to the Moon surface involves mainly *x* component. This was in part predictable, at least for what concerns the *y* component, since it has already been seen that the orbit is almost entirely on the *x*-*z* plane, while the *y* direction variations are almost negligible compared to the other directions.

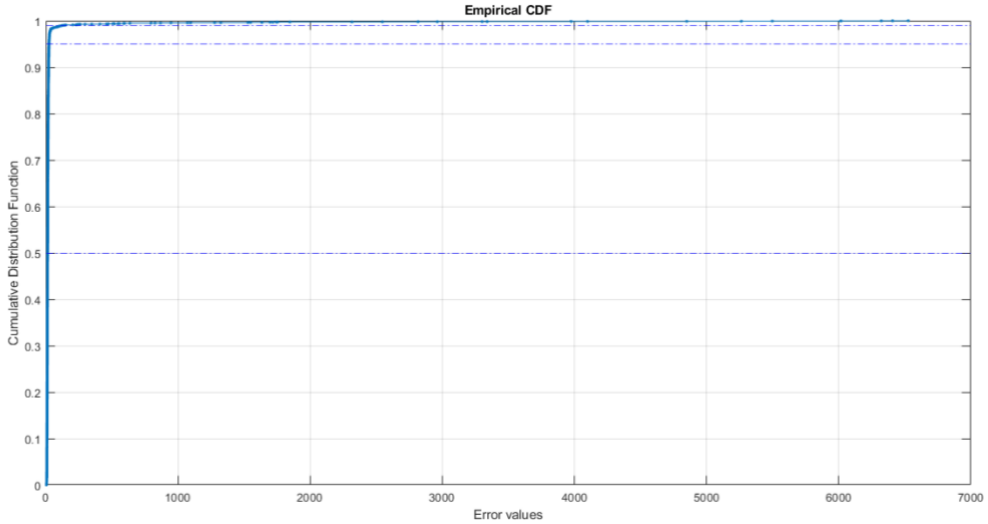

Figure 5.2.17: CDF plot of the 3D position error values for EKF with constant acceleration model

The Cumulative Distribution Function of the 3D position errors (*Figure 5.2.17)*, is also in this case more flattened, giving the greater accuracy of the solution. Also, the three percentile values analyzed can't be represented, since are crashed toward the y axis, having values under 40 meters (as already discussed before).

Finally, *Figure 5.2.18* highlights once again the velocity and acceleration profiles, with the cyan and green circles representing the references and magenta and yellow ones representing the approximations. Excluding the transient, the accuracy in velocity approximation is more accurate than before, giving deviations from the reference lower than 5 m/s, until the descending phase in which the difference is even lower (about 2 or 3 m/s).
As expected, the acceleration approximation is more accurate, since in the nearly constant acceleration model the components of acceleration are calculated in the algorithm. The values of the approximation oscillate between -0.5 and -2 m/s^2 (with respect to the reference values around zero), both before and after the peak in the middle of the solution, that reaches about -6 m/s^2. Focusing on the final phase, corresponding to the beginning of the landing, the approximation follows the same trend as the reference profile: the step in the approximated acceleration profile reaches -5.5 m/s^2 (compared to the -4.5 m/s^2 of the user profile) and then gradually increases together with the reference values, with deviations of only 0.5 m/s^2.

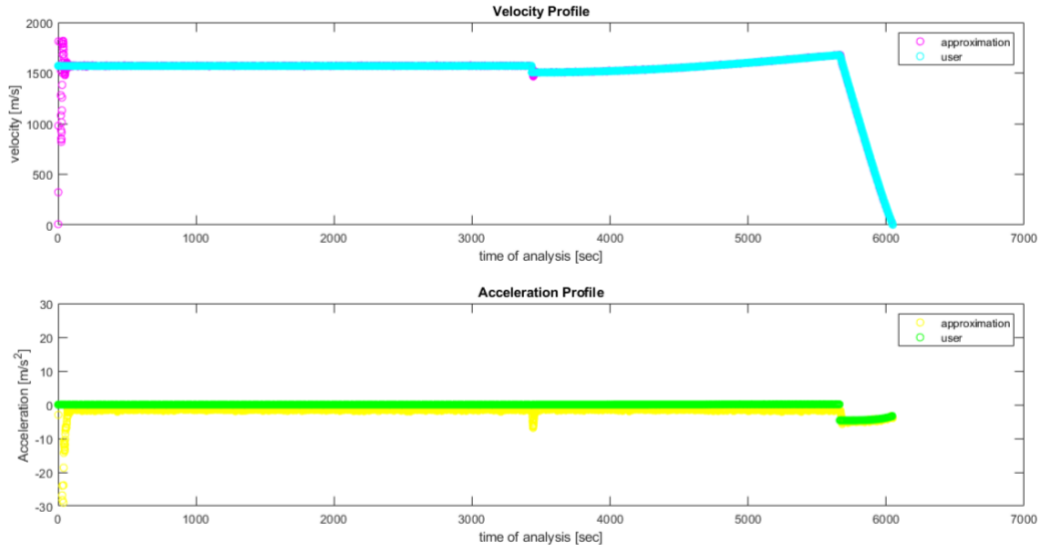

Figure 5.2.18: Velocity and Acceleration profiles for EKF with constant acceleration model

These results demonstrated that the Extended Kalman Filter implemented with a Nearly Constant Acceleration model provides the best performances in terms of accuracy for the scenario of a dynamic user landing to the Moon surface. This has been proven both by the lowest error values obtained comparing the filter calculations to the reference values of position and velocity, but also observing the velocity and acceleration profiles that differ very little from the reference ones.

5.2.3. Sensor Fusion

Since the Extended Kalman Filter with nearly constant acceleration model has demonstrated to be the best performing algorithm, in this section the Sensor Fusion technique will be implemented considering this model with the additional measurement of the altimeter.
This measurement is activated when user position is approximately 10 km far from Moon surface, so the trend of the solution is the same until this distance is reached. For this reason, the analysis will focus on the last part of the results, in order to highlight the differences between the Extended Kalman Filter performances with or without the altimeter measurement.

5.2.3.1. Pure vertical altimeter measurement

As already explained in section 4.2.3, the first approximation is that the measurement given by the altimeter is purely vertical directed toward the center of the Moon, which is not a realistic case, but gives a first information about the performances of the Sensor Fusion technique.

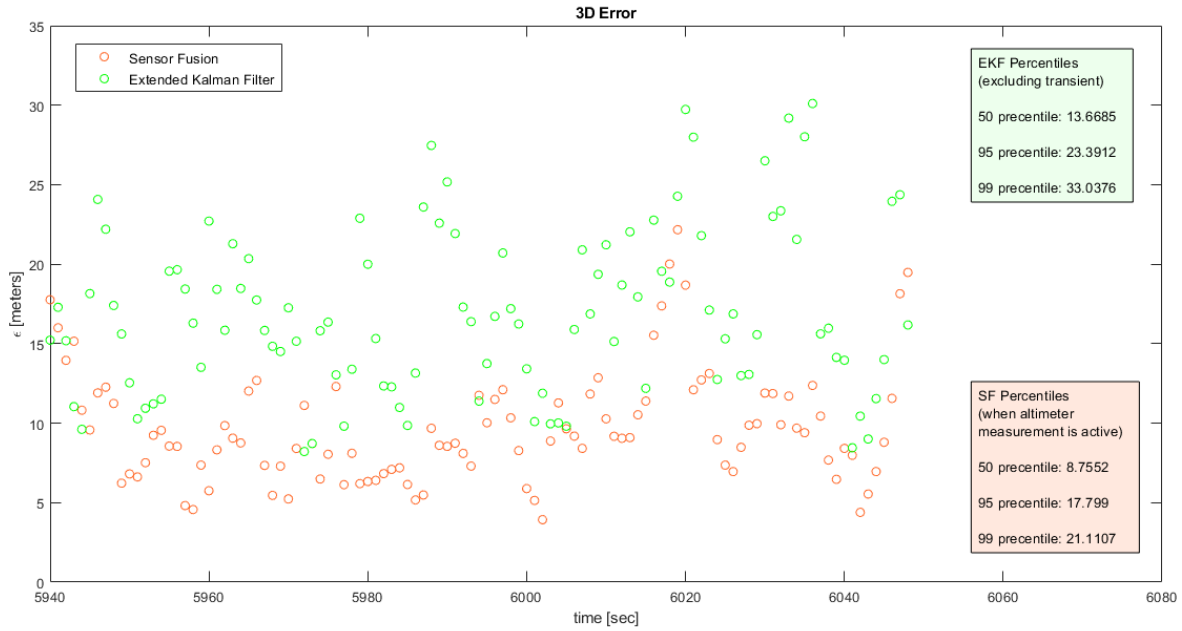

Figure 5.2.19: Comparison of 3D position errors (when altimeter is active) for EKF and SF

Figure 5.2.19 shows the comparison between the values of the 3D position errors obtained by the EKF with constant acceleration model (green circles) and the values obtained by the SF technique with the altimeter measurement (orange circles).
There is a small difference between the two cases, but the SF technique produces slightly better results, given lower values overall with respect to the EKF ones. In fact, the EKF solution also reaches error values above 25 meters and not lower than 10 meters, while with the SF the maximum peak is at 20 meters, but in general the results fluctuate around values less than 10 meters, reaching also values below 5 meters.

The boxes in *Figure 5.2.19* and the *Table 5.4* summarize the percentile values excluding the transient for the EKF solution and considering only the final epochs when the altimeter measurement is activated for the SF solution. These values confirm the previous observation: all three percentile values are lower in SF case, with differences of about 5 or 6 meters, except for the 99th percentile, which is more than 11 meters smaller, showing that the SF values are more flattened and distributed over the time of analysis.

Table 5.4: Comparison of percentile values for 3D position errors of EKF and SF

	EKF	***SF***
50th percentile [meters]	13.6685	8.7552
95th percentile [meters]	23.3912	17.799
99th percentile [meters]	33.0376	21.1107

The 3D velocity error comparison generates similar results, shown in *Figure 5.2.20*. Nonetheless, the difference between the two cases is even less clear considering velocity errors: EKF values oscillates between 2 and 5 m/s with peaks slightly above 7 m/s, while SF values are just few units lower, fluctuating between 1 and 4 m/s with peaks of 6 m/s.

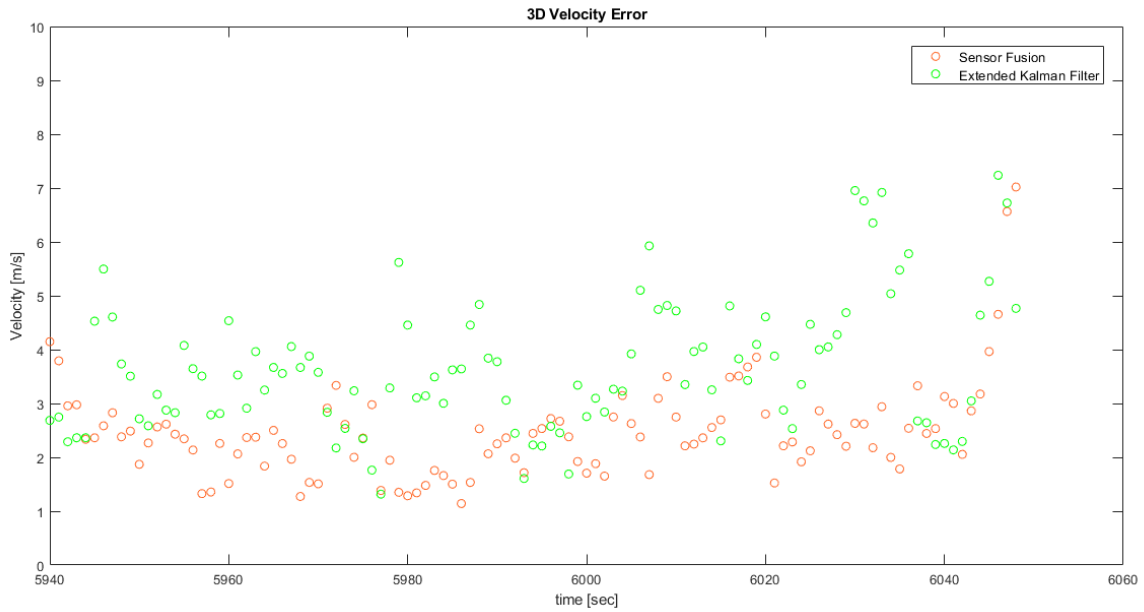

Figure 5.2.20: Comparison of 3D velocity errors (when altimeter is active) for EKF and SF

The *Figures 5.2.21* and *5.2.22* emphasize the previous analysis, showing the errors on each component both for position and velocity. Once again, the difference between the two solutions is more evident by observing position errors rather than velocity errors.
Among the three directions, the *z* component seems to be the only one on which the altimeter has a significant effect. This is partly true, but it is important to underline that the measurement of the altimeter has an impact on all three components: this is because the altimeter reading is a vertical measure pointing on lunar surface and precisely it is directed toward the Up direction considering ENU coordinates (East, North, Up). This means that the measurement will affect not only the *z* component, but also *x* and *y*, since the XYZ reference system origin is the center of the Moon, so that each direction will have an UP direction if considered in the ENU system. Nevertheless, as observed before, the error is more blunted along the *z* component: this is because, in the final phase of the landing, the direction followed by the user is almost perpendicular to the moon surface and the lander is approaching the South Pole, so that Up direction is almost aligned with the *z* direction, and therefore altimeter error will be much more limited (close to zero).

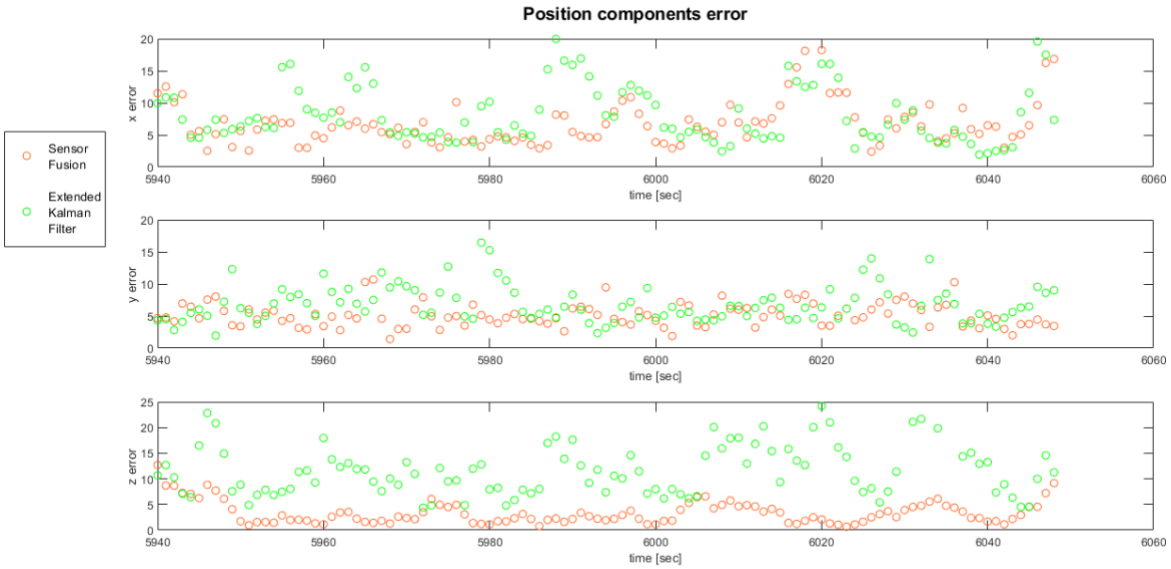

Figure 5.2.21: Comparison of position components errors (when altimeter is active) for EKF and SF

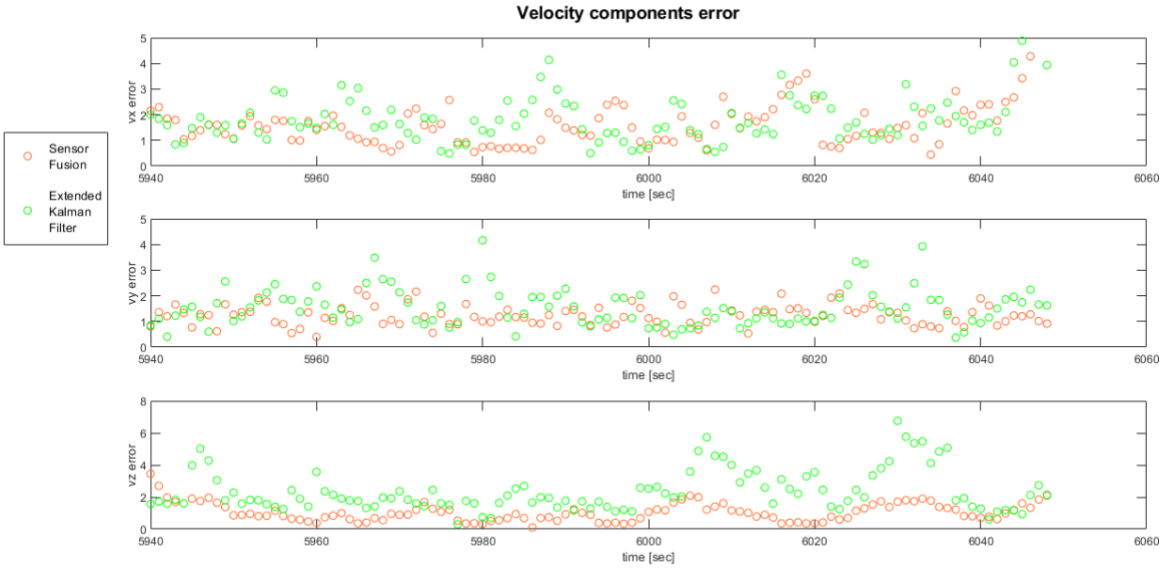

Figure 5.2.22: Comparison of velocity components errors (when altimeter is active) for EKF and SF

5.2.3.2. Error on altimeter measurement

In order to realize a more realistic analysis, it is necessary to consider an error on the altimeter measure. In fact, as already introduced in section 4.2.3, the attitude of the user must be considered so that the direction of the altimeter measurement is not absolutely vertical, i.e., perpendicular directed to the Moon surface along the Up direction of the ENU reference system.

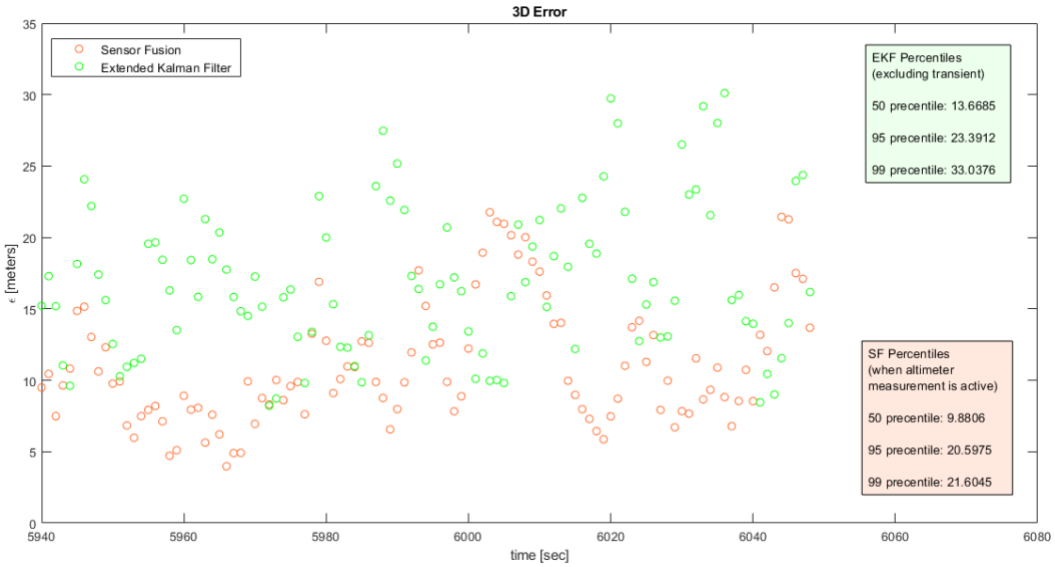

Figure 5.2.23: Comparison of 3D position errors (when altimeter is active) for EKF and SF

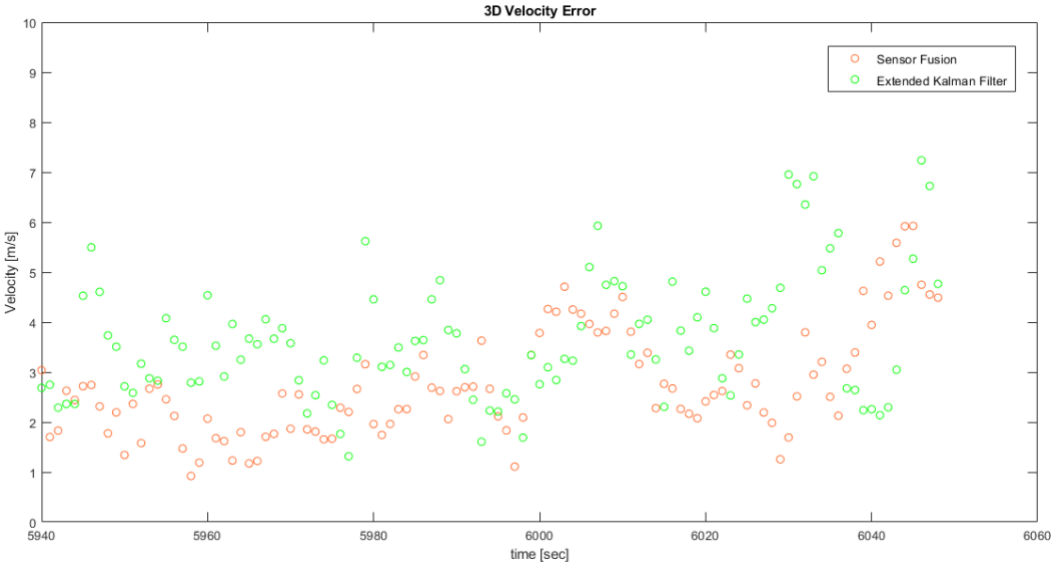

Figure 5.2.24: Comparison of 3D velocity errors (when altimeter is active) for EKF and SF

Observing the 3D position error (*Figure 5.2.23*) and 3D velocity error (*Figure 5.2.24*), it can be noticed that the effect of the error on the altimeter reading does not significantly affect the results. In fact, the errors oscillate between the same values as before and the difference to be seen is almost negligible, both for position and velocity graphs. The values of the percentiles allow to highlight this very slight difference, as shown in *Table 5.5*: implementing the altimeter error, the percentiles are slightly worse (as expected), being just a few meters higher than the values obtained previously. The only exception is the 99th percentile, which are almost the same given a difference of the order of centimeters.

Table 5.5: Comparison of percentile values for 3D position errors of SF with and without altimeter error

	SF without altimeter error	***SF with altimeter error***
50th percentile [meters]	8.7552	9.8806
95th percentile [meters]	17.799	20.5975
99th percentile [meters]	21.1107	21.6045

Figures 5.2.25 and *5.2.26* represent once again the position components errors and velocity components errors respectively. As for the previous case, the effect of the additional altimeter measurement is more evident on the position components errors rather than on the velocity components, and again has a greater impact on the *z* component than on the *x* and *y* directions.
The difference between these errors and those obtained with the measurement of the purely vertical altimeter is minimal, thus demonstrating a good accuracy on single components also in this case.

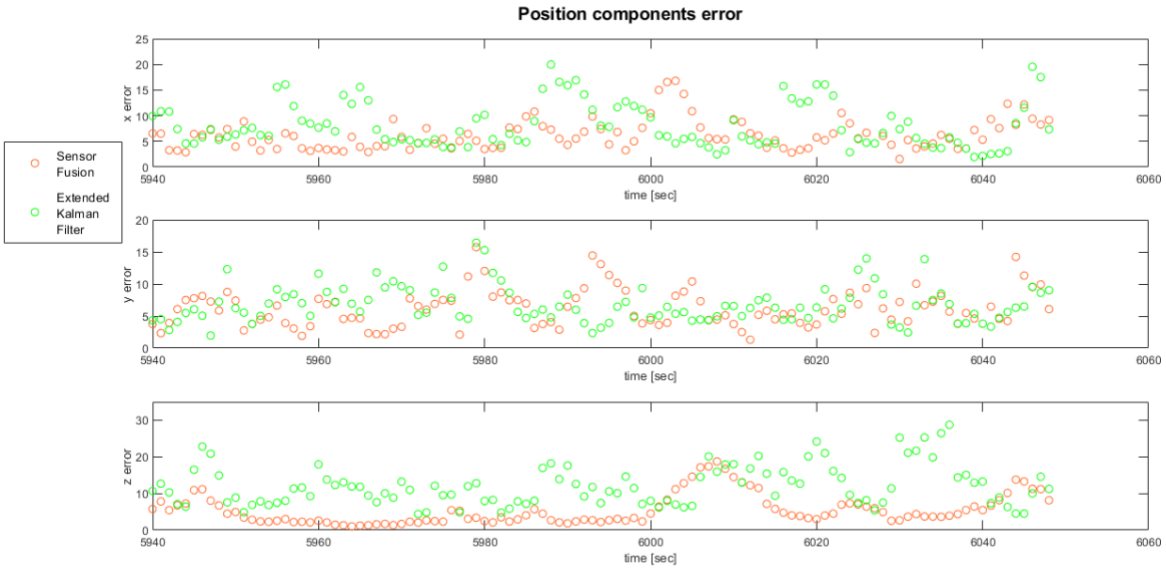

Figure 5.2.25: Comparison of position components errors (when altimeter is active) for EKF and SF

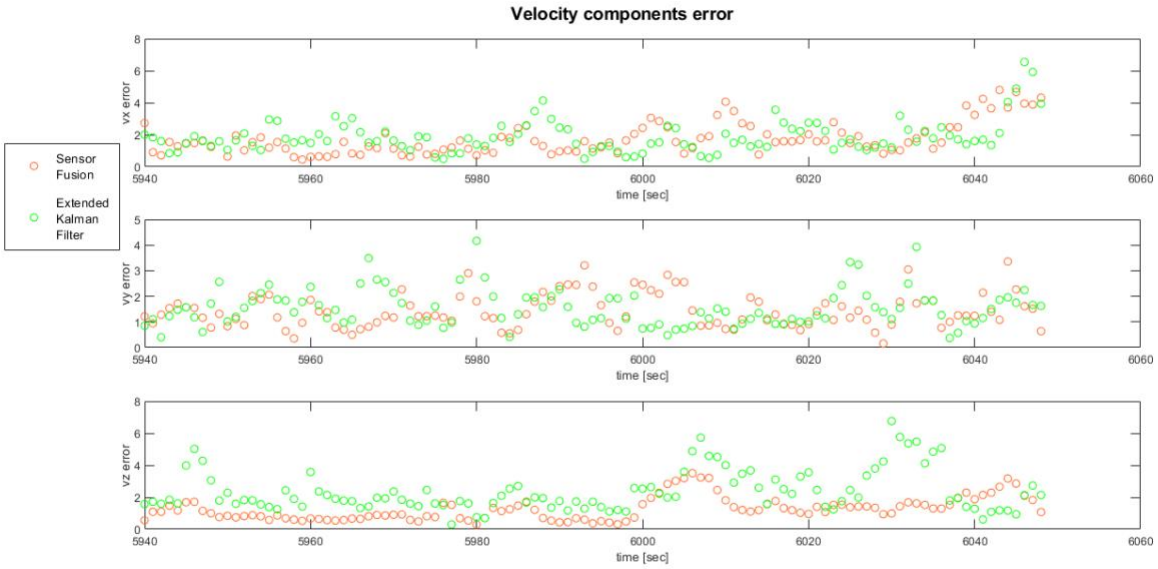

Figure 5.2.26: Comparison of velocity components errors (when altimeter is active) for EKF and SF

In the previous section it has been demonstrated that the additional measurement of the altimeter provides more accurate results, but that measurement was not realistic since it didn't consider the attitude of the lander. In this section, though, it has been proven that, even considering a more realistic measure that takes into account the user's attitude during landing, the results are still optimal and more accurate than those obtained with the Extended Kalman Filter alone.

5.2.3.3. *Exclusion of a satellite*

In this section another an interesting simulation of a possible case in lunar environment will be considered. It has already been said that the user needs at least four satellites to estimate his position (and velocity and acceleration). The altimeter measurement gives an additional reference to the user, which allows the filter to determine more accurately the characteristics needed. To evaluate the effect of this measure and compare it to the information that each satellite gives to the user, it is possible to substitute the altimeter reading to a satellite Slant Range, instead of adding it to the Slant Range measurements of the four satellites in visibility. Therefore, instead of having four satellites plus altimeter reading, now the positioning algorithm will receive input data from three satellites in view and the altimeter measurement in substitution of the fourth. This simulation is of course valid in the last phases of the analysis, since it is only in the landing phase that the altimeter measurement is present (below 10 km from Moon).

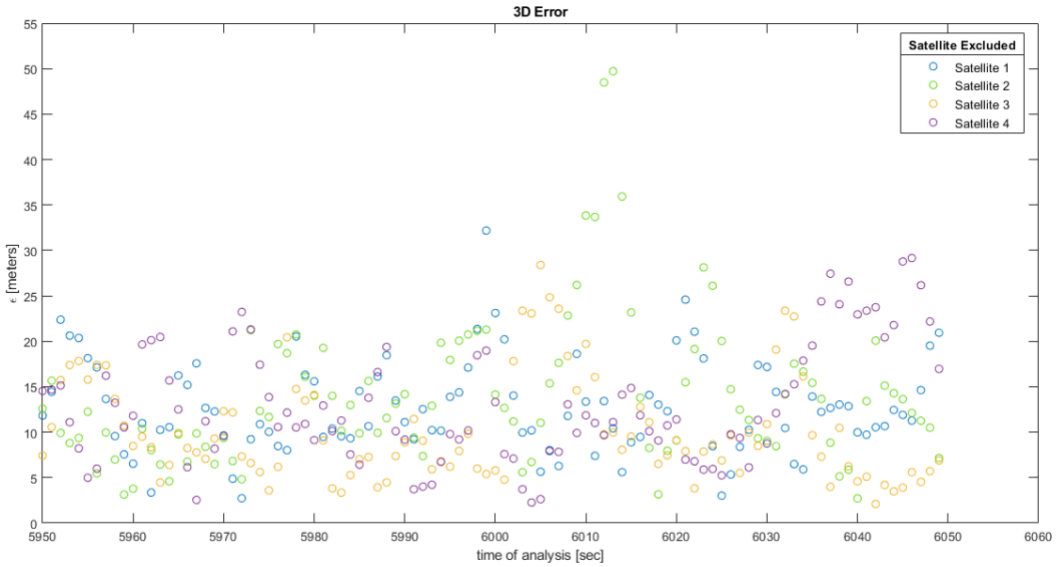

Figure 5.2.27: Comparison of 3D position errors with substitution of each satellite with the altimeter

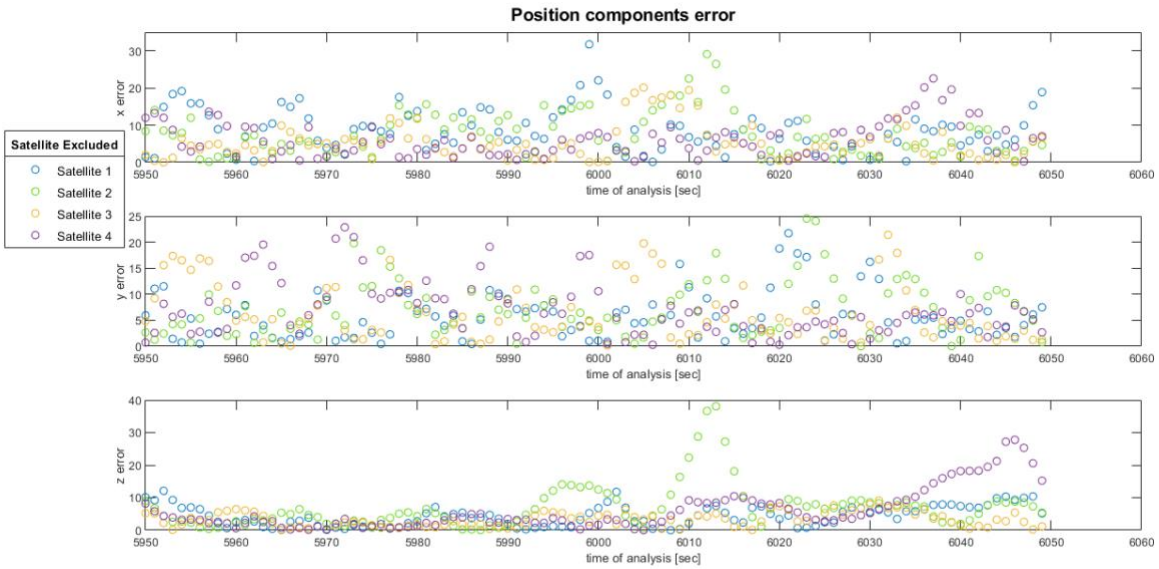

Figure 5.2.28: Comparison of position components errors with substitution of each satellite with the altimeter

The *Figure 5.2.27* and *5.2.28* show the 3D position errors and the position components errors, comparing the results obtained removing each satellite and substituting with the altimeter reading. The outcomes are very similar in each case considered. There is a slight difference regarding the result obtained with the exclusion of satellite 2: the error values obtained in this case show a peak of about 60 meters, when the trend of all errors is hardly exceeds the 25 meters. This is evident in the single components, especially in the *z* direction, where the peak reaches almost 40 meters, compared to the standard values that are below 10 meters.
Moreover, this consideration is confirmed by the percentile values shown in the *Figure 5.2.29,* which highlights the higher values of the percentiles for this second case, especially the 99th percentile that is almost doubled with respect to the other cases.

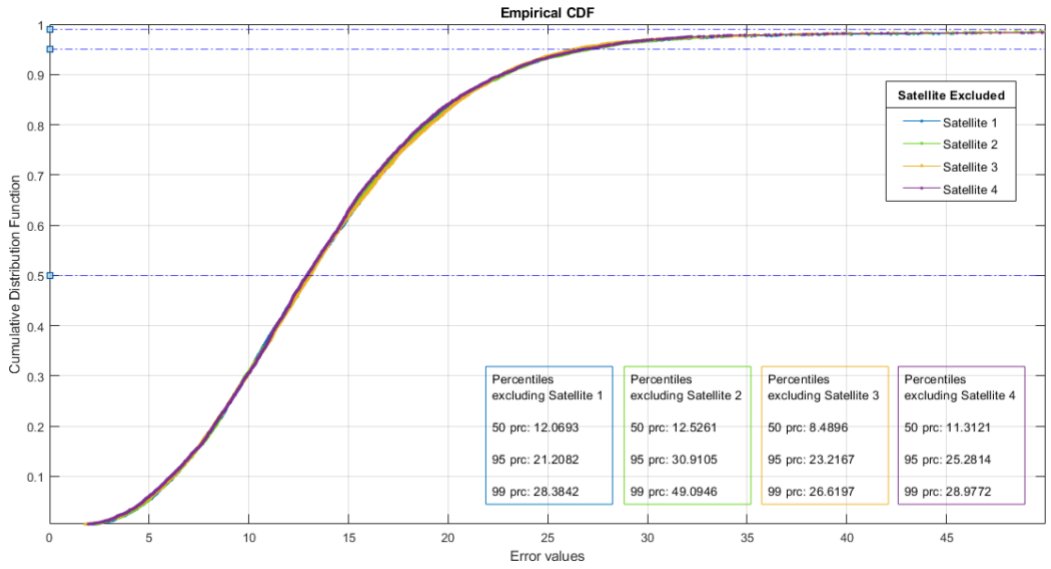

Figure 5.2.29: Comparison of CDF of 3D position errors with substitution of each satellite with the altimeter

Aside from the comments made before, these results determine an important consideration for lunar navigation: the analysis made before has considered that one satellite was unavailable for all the simulation time, and these outcomes demonstrated that the altimeter measurement can successfully substitute a satellite not in view. Taking for the example the results obtained with the static user, it has been seen that there were blackouts in the solution, in which the user cannot determine its position, since there were not enough input data to compute the positioning algorithms. In the scenario just analyzed, this would not have been a problem, since in the epochs in which one satellite is not in view, the altimeter could compensate its absence, increasing the availability of the solution. This result is very important considering lunar environment: considering Earth environment, there are a lot of constellations that can give information to the user even in case of failures. In lunar environment, there are not a lot of signals available, so that in case of a failure or unavailability of a satellite, this can cause serious problem for the determination of the user position and velocity over time. The additional measurement of the altimeter gives a great aid, covering for these problems that could present during an expedition. As for the altimeter, this is obviously true only for the landing phase, when the user is 10 km far from the Moon surface and the measurement is activated. However, this reasoning can be expanded to any sensor that can be part of the user characteristics, giving the possibility to cover satellites failure or unavailability also during orbiting around the Moon.

5.3. Comparison of results

Considering the results shown in the previous chapters, the following figures will highlight the comparison of the performances that each algorithm can achieve under the assumption considered in this work. In order to produce more consistent results, a Montecarlo simulation based on 1000 iterations has been carried out, considering then the root mean squares of the results obtained.

The *Figure 5.3.1* shows the comparison of the 3D position errors obtained with the implementation of the different methodologies analyzed. Similarly as defined in the previous paragraphs, considering the reference position $user=[x_u, y_u, z_u]$ and the approximate calculated position $u_{app}=[x_{app}, y_{app}, z_{app}]$, the 3D error has been computed as:

$$3D\ Error = \sqrt{(x_{app} - x_u)^2 + (y_{app} - y_u)^2 + (z_{app} - z_u)^2}$$

In particular, the blue circles represent the results achieved by the Least Squares, the green ones are referred to the Extended Kalman Filter, and finally the orange markers show the results obtained with the Sensor Fusion technique with the altimeter measurement. Also, the red vertical line indicates the instant of time from which the additional measurement of the altimeter is activated (i.e., when user position is below 10 km from lunar surface).

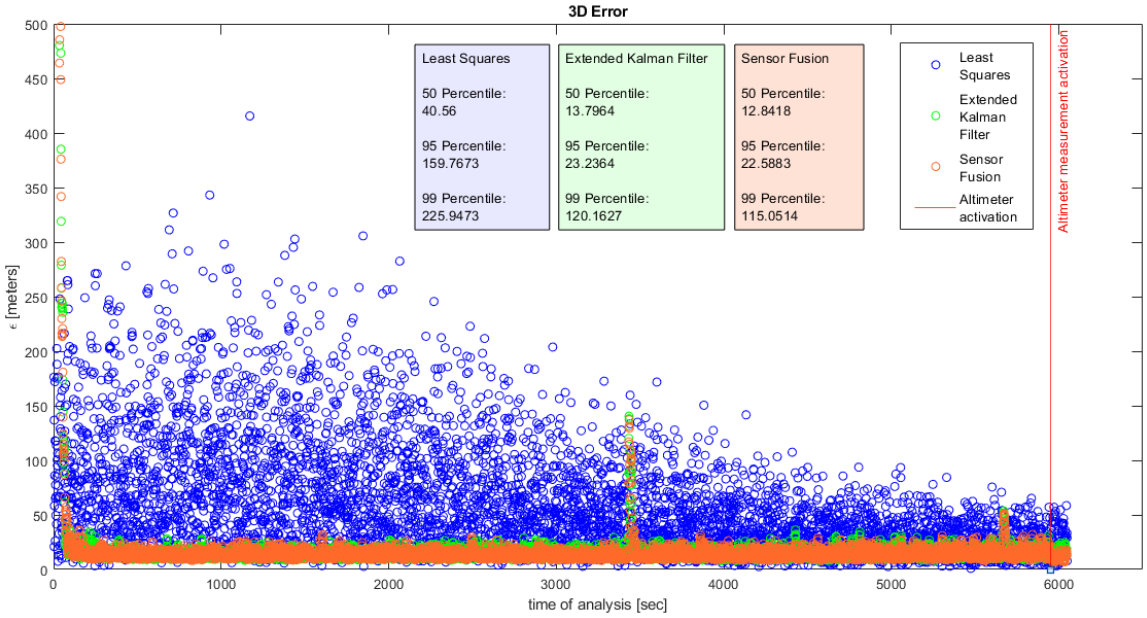

Figure 5.3.1: Comparison of 3D position error trend for the different algorithms

These results confirm the analysis made in the previous paragraphs: the Least Squares technique demonstrate more scattered errors and slightly convergence to less accurate values. Instead, the Extended Kalman Filter results confirm once again better performances, both in term of accuracy since the error values are way lower than those obtained with LS, but also in term of high convergence rate since, after the transient at the beginning, the filter immediately converges after very few seconds. As already mentioned, the input data contain two anomalies, which have been discussed considering the velocity and acceleration profiles of the user: the first one is after about 3430 seconds and it has been consider a maneuver to adjust the attitude of the lander orbiting around the Moon, while the second one is almost at the end of the analysis and corresponds to the initial maneuver done to start the landing to lunar surface. These maneuvers cause a glitch in the estimations, but this is not very evident in the Least Squares analysis given the more scattered and high values of errors. It is quite

evident, instead, considering the Extended Kalman Filter results: since the errors are much lower, the peaks are more visible, reaching values more than 3 times higher due to the first maneuver and almost doubled for the second one.
Obviously, the trend of the 3D position error (*Figure 5.3.1*) is the same for the EKF with or without the altimeter measurement, because the additional measurement is not present in the analysis until the user is approximately 10 km far from Moon surface (red line in the figure). Instead, the difference is evident in *Figure 5.3.2*, which focus on the last part of the analysis, i.e., the final part of the landing phase when the altimeter measurement is activated.

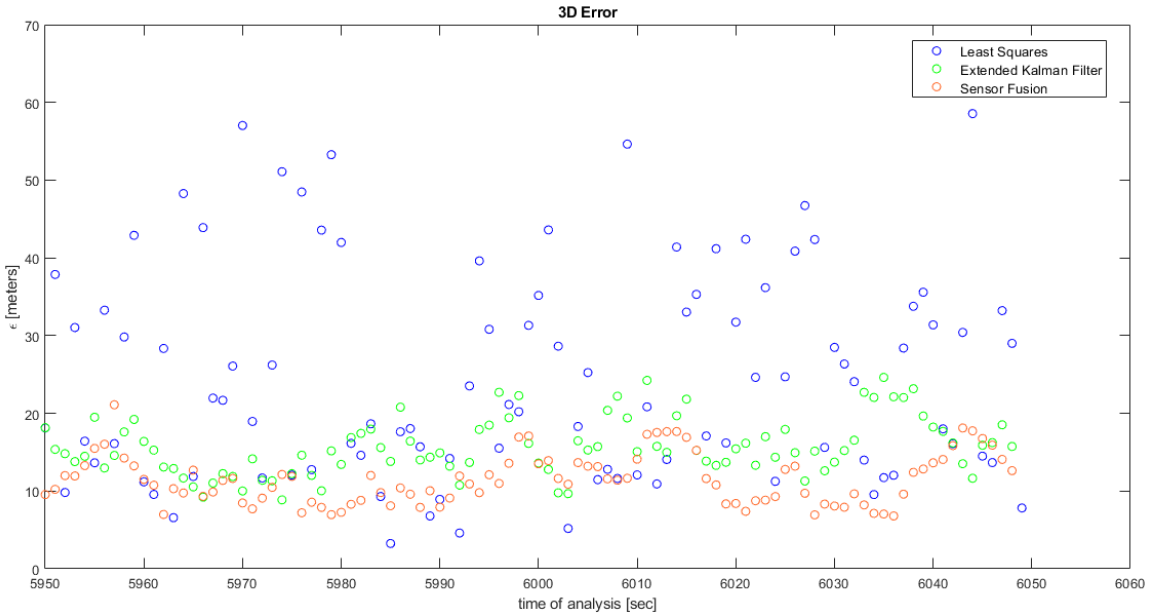

Figure 5.3.2: Comparison of 3D position error considering last phases (when altimeter is active)

This figure confirms the even more accurate results considering the additional measurement of the altimeter, which error values are lower than those of the Extended Kalman Filter alone.
This consideration is even more clear observing the percentile values (50th percentile, 95th percentile and 99th percentile), summarized in the three boxes reported in *Figure 5.3.1* and in *Table 5.6*, where the Sensor Fusion technique (EKF + Altimeter) shows the lowest values.

Table 5.6: Comparison of percentile values for the different algorithms

	Least Squares	***Extended Kalman Filter***	***Extended Kalman Filter + Altimeter***
50 Percentile [meters]	40.56	13.7964	12.8418
95 Percentile [meters]	159.7673	23.2364	22.5883
99 Percentile [meters]	225.9473	120.1627	115.0514

These values are also visible in *Figure 5.3.3*, which represent the comparison of the Cumulative Distribution Function of the 3D position error of each algorithm. As already discussed, the curves are crushed on y-axis and the difference is not very clear, so the figure shows a zoom on the initial values. Once again, the CDF of the Extended Kalman Filter and Sensor Fusion are more flattened, which means higher convergence rate to more accurate values.

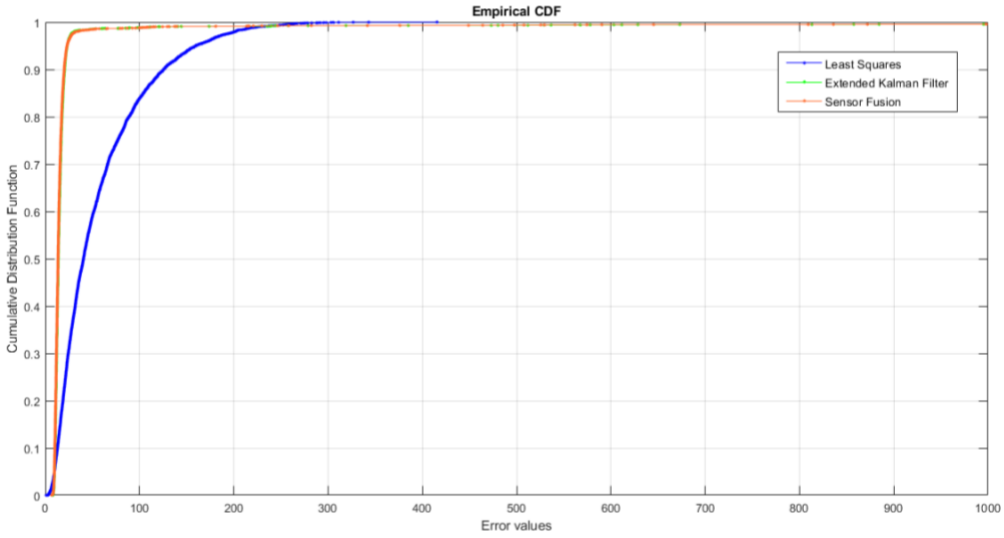

Figure 5.3.3: Comparison of CDF of 3D position errors for each algorithm

To highlight the differences between the solution before and after the activation of the altimeter measurement, the CDF plot and percentile values are analyzed in both cases.
Figure 5.3.4 shows the CDF of the 3D position errors for each technique before the activation of the measurement and *Table 5.7* compares the percentile values as before.

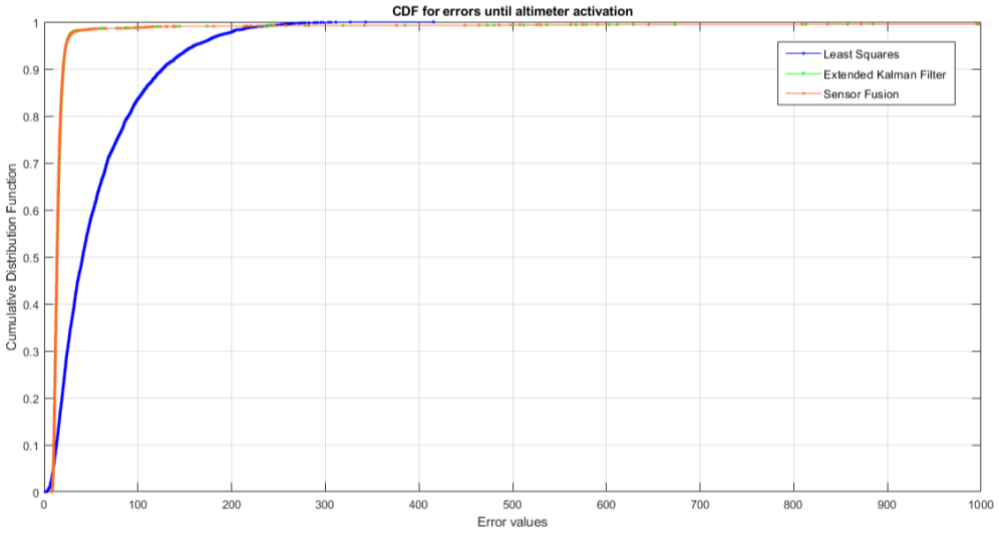

Figure 5.3.4: Comparison of CDF of 3D position errors for each algorithm before altimeter activation

As expected, the results are similar to those obtained considering the whole simulation, since the altimeter is activated only in the last part of the analysis when the user is landing. In fact, EKF and SF percentile values are almost the same (as already said for the trend of the error) and they are lower than the values obtained with the LS technique.

Table 5.7: Comparison of percentile values for each algorithm before altimeter activation

	Least Squares	***Extended Kalman Filter***	***Extended Kalman Filter + Altimeter***
50 Percentile [meters]	41.0715	13.7538	13.1763
95 Percentile [meters]	160.3238	23.2909	22.7518
99 Percentile [meters]	226.1575	121.3540	120.5595

Figure 5.3.5, instead, shows the CDF of the 3D position errors for each technique in the last phases, i.e., when the altimeter measurement is activated, while the *Table 5.8* compares the percentile values as before. The trend is similar, but of course there are much lower values of analysis with respect of the rest of the simulation. These results confirm that in the landing phase the SF technique shows the best performances, since the CDF curve is the nearest to the y-axis and as a consequence the value of the percentiles are lower than the ones obtained with the EKF alone.

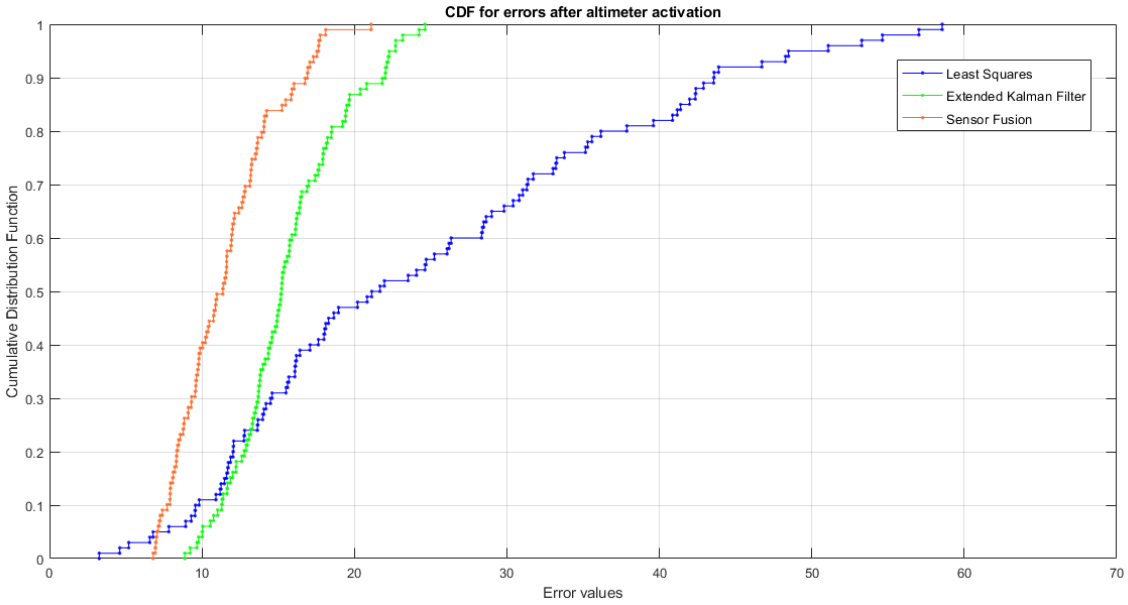

Figure 5.3.5: Comparison of CDF of 3D position errors for each algorithm after altimeter activation

Table 5.8: Comparison of percentile values for each algorithm after altimeter activation

	Least Squares	***Extended Kalman Filter***	***Extended Kalman Filter + Altimeter***
50 Percentile [meters]	21.4062	15.2151	11.3712
95 Percentile [meters]	49.7758	22.5102	17.5984
99 Percentile [meters]	57.7915	24.4353	19.6354

The reasonings made before are also confirmed in the *Figure 5.3.6*, depicting the position errors component by component. Color markers are the same as for the 3D position errors, and as before the difference in the accuracy between EKF and EKF + Altimeter is more evident in *Figure 5.3.7*, depicting the final phase of the landing.
As already discussed, the measurement of the altimeter has an impact on all three components, since the altimeter reading is a vertical measurement directed is directed toward the Up direction, thus affecting not only the z component, but also x and y. Nevertheless, once again the error is more blunted along the z component because during the landing the direction followed by the user is almost perpendicular to the Moon surface, so that Up and z directions are almost aligned.

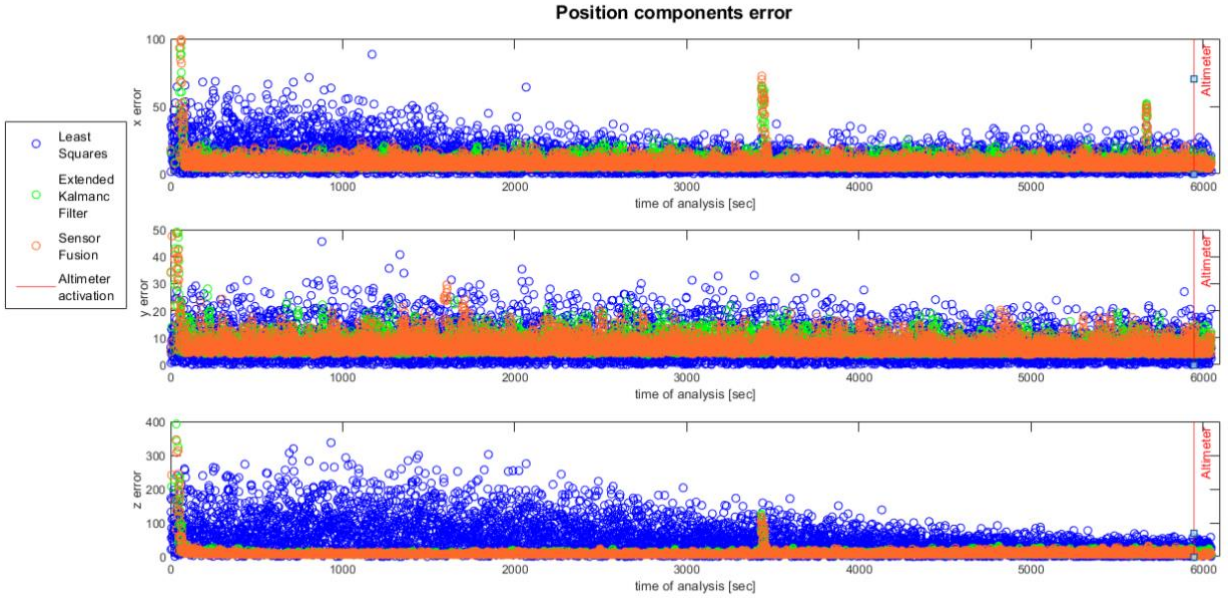

Figure 5.3.6: Position components error trend

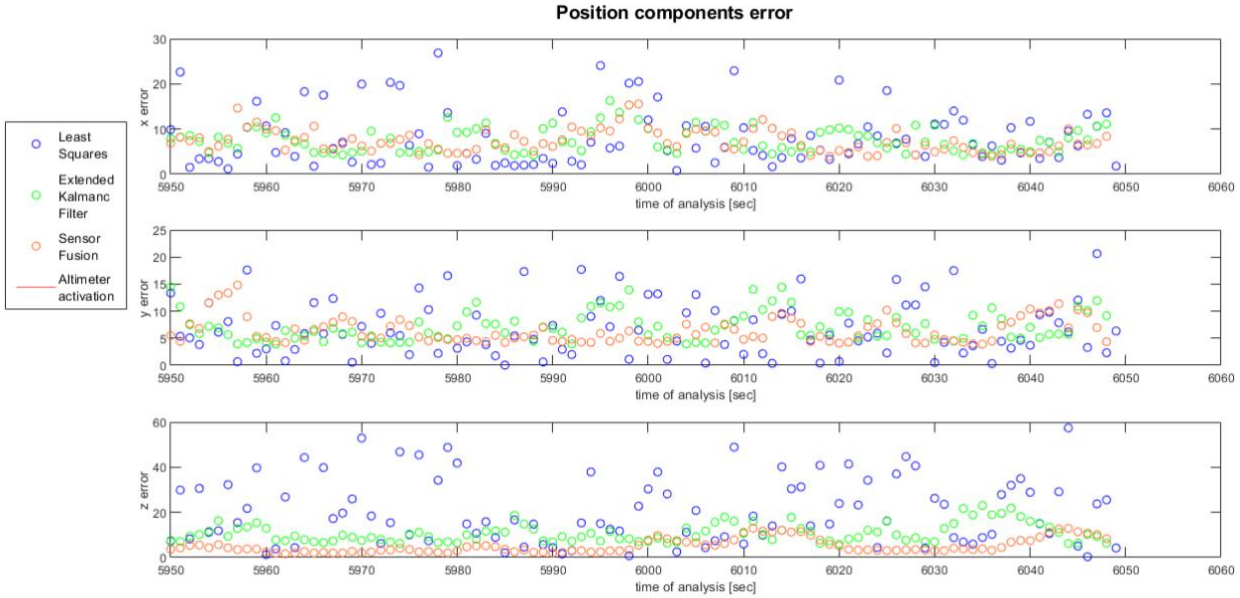

Figure 5.3.7: Position components error considering last phases

To conclude the analysis, *Figure 5.3.8* and *Figure 5.3.9* show the comparison between the true trajectory of the user and the approximation values obtained with the different techniques, on the horizontal and vertical plane respectively. The figures are focused on the last 30 seconds of the simulation to better highlights the differences among the different algorithms.

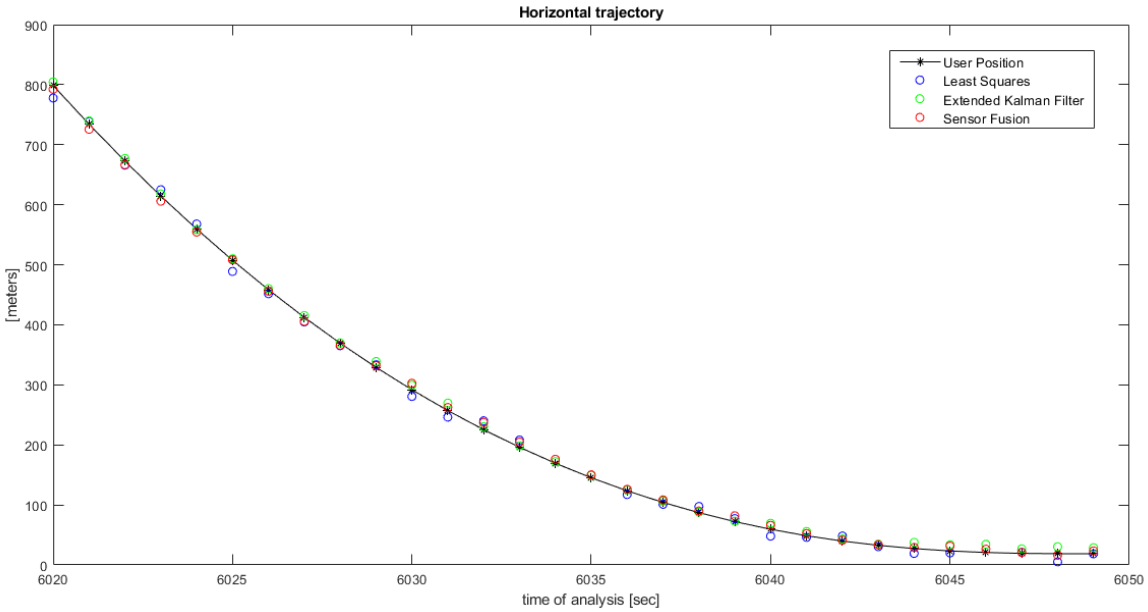

Figure 5.3.8: Comparison between horizontal trajectories of user and the different algorithms

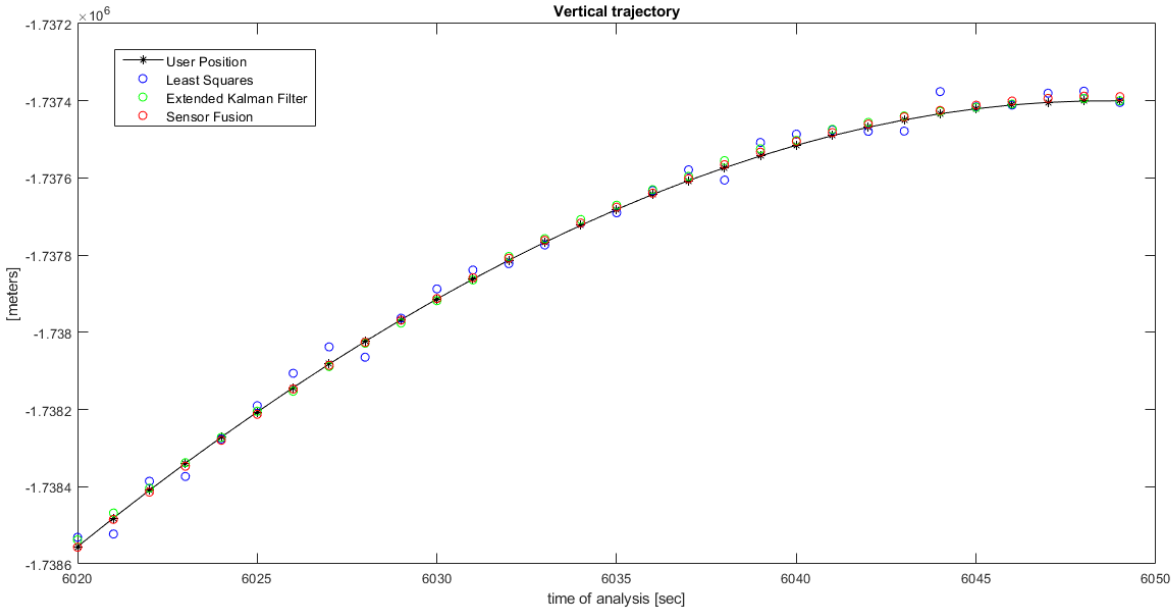

Figure 5.3.9: Comparison between vertical trajectories of user and the different algorithms

These results are consistent, since it has been proven the better performances of the Extended Kalman Filter compared with the Least Squares technique, especially with the additional measurement of the altimeter. This is in line with expectations, as the Kalman Filter analysis that can be found in literature on Earth applications almost always show better results compared to Least Squares. The comparison of the outcomes obtained in this paragraph has demonstrated that this is valid also in Lunar environment, which is an important result. In fact, the Earth-based techniques currently adopted for navigation with satellites around the Moon are not able to cover all the needs for future missions, and this is because the number of satellites foreseen in a lunar service is much smaller, but also due to the weaker signals and limited coverage in lunar environment.

Moreover, the additional altimeter measurement makes the filter even more precise, and that makes perfectly sense since this is the functioning and objective of the Sensor Fusion technique: additional measurement gives more references to the user, which then has the ability to determine its position and velocity more accurately.
Finally, even if errors implementation in measurement readings leads to slightly worse results, the overall consistency is not affected, confirming that the Sensor Fusion technique shows better performances, and the additional measurement of an altimeter provides an added value in the landing phase on lunar surface.

Despite the positive outcomes achieved, both 3D positions and velocity errors has shown that there is still an error at the end of the analysis, giving that the approximated position and velocity of the user calculated with the positioning algorithm are not the expected ones at the end of the landing (position should be *South Pole = [0, 0, -r_{Moon}]* and velocity should be zero).
In fact, the analyses done in this work are not 100% perfect, because there are other errors and considerations to be taken into account. Nevertheless, the results can be considered acceptable for this phase of study, demonstrating the reliability of the positioning algorithms analyzed and giving a first comparison of the performances achievable with each of them.

6. Conclusion & Future Works

This thesis aims to provide an assessment of the performance achievable with different suitable positioning algorithms for Lunar PVT estimation, with the goal of allowing users to perform a positioning over the Moon surface using a limited number of navigation signals broadcasted by a dedicated lunar navigation system. Combining the best features of each technique, it can be defined a unique tool that could perform a PVT estimation with the best possible performance and accuracy.
Firstly, the most used positioning techniques of Least Squares and Extended Kalman Filter have been presented, discussing the current state of the art and their applications. Afterwards, the Sensor Fusion technique has been described, considering the additional measurement of an altimeter.
Focusing on lunar environment, the main differences with Earth applications have been presented, together with the case of studies that consider both a static user in a fixed position on lunar surface and a dynamic user orbiting around the Moon and approaching it to land on a specific point.
Subsequently, the simulated navigation system considered for the implementation has been presented, describing the constellation of satellites and the input data provided for the analysis, but also the formulations behind the implementation of each positioning algorithm.

The analysis of the results shows that generally the Extended Kalman Filter (EKF) provides better performance in comparison with the Least Squares (LS) estimation, in line with expectations, since on Earth application KF almost always performs better than LS. However, the analysis carried out in this work demonstrates that this reasoning is also valid considering a dedicate lunar navigation system. This is an important result, given that the Earth-based techniques currently adopted for navigation with satellites around the Moon are not able to cover all the needs for future missions, because of the smaller number of satellites foreseen in a lunar service, the weaker signals, and the limited coverage in lunar environment. The analysis of this thesis demonstrates that a dedicated lunar service, which do not rely on Earth measurements, can successfully provide information to the user in order to perform the PVT estimation.
Moreover, the implementation of the Sensor Fusion (SF) technique with the aid of the additional measurement of an altimeter shows even better performance, providing an added value especially in the landing phase on lunar surface. This is the basic functioning of the SF technique: additional measurement gives more references to the user, which then has the ability to determine its characteristics more accurately.
Finally, Sensor Fusion technique shows another important advantage: one of the simulations carried out in this work demonstrates that the additional altimeter reading can successfully substitute the measurement of a satellite not in view. This a very important outcome for lunar implementations. Considering Earth applications, several constellations can give a great amount of information to the user. In lunar environment, instead, the coverage is limited and in case of a failure or unavailability of a satellite, this can cause serious problems for the determination of the user PVT over time. The Sensor Fusion technique provide the aid of an additional measurement, so that in the epochs in which one satellite cannot provide information to the user, this additional measurement can compensate its absence, increasing the availability of the solution.

Despite the positive outcomes achieved, the results analyzed still show an error at the end of the analysis, giving that the approximated position and velocity of the user calculated with the positioning algorithms are not the expected ones at the end of the landing (position should be South Pole of the Moon and velocity should be zero).
To improve the analysis proposed, as introduced in the state of the art, there are several variants of positioning techniques that can be implemented for the lunar PVT estimation, so that the results could be compared, and observe if one of these variants can perform better.

In addition, besides from the ones introduced in this thesis, there are other sources of errors that can be taken into account in the implementation of the algorithms.
Moreover, the Sensor Fusion technique has been implemented considering the additional measurement of an altimeter, but of course the functioning of this algorithm is valid also with other sensors that could give more information to the user so that it can reach even better performance.
One of the most interesting sensors that could be analyzed is the Inertial Measurement Unit (IMU), which is a device that typically consist of gyroscopes and accelerometers.

Despite these considerations, the outcomes discussed in this thesis can be considered acceptable for this phase of study. The results demonstrate the reliability of the positioning algorithms analyzed and give a comparison of the performances achievable with each of them, so that the lunar PVT estimation can be performed with the best possible accuracy.

Appendices

Appendix A: Earth Validation

Following data extraction, some issues had to be resolved to make the code more easily interpretable and to achieve consistent and acceptable results.

A.1. GPS Time

A first thing to do is the conversion of the single epochs, provided as dates of the Gregorian calendar, into *GPS time*, which is a continuous time scale defined by the GPS Control Segment based on a set of atomic clocks at the Monitor Stations and onboard the satellites. The system transmits the *number of weeks since January 6th, 1980*, *and the number of seconds since the beginning of the current week*. *Table A.1* shows the steps followed in the conversion (example date: January 2nd , 2021, at 06:22:43):

Table A.1: Conversion Gregorian date to GPS

DATE	$\boldsymbol{dd/mm/aaaa - hh:min:sec}$	$\mathbf{01/02/2021 - 06:22:43}$
1a: Years from 1980	$aaaa - 1980 = \boldsymbol{yy}$	$2021 - 1980 = \mathbf{41}$
1b: Conversion to days	$yy \cdot 365 = \boldsymbol{d_{YEARS}}$	$41 \cdot 365 = \mathbf{14965}$
1c: Add days from January 6th	$(n.\,days\ of\ months\ from\ Jan\ to\ mm) + dd - 6 = \boldsymbol{d_{DAYS}}$	$Months\ from\ Jan\ to\ Jan = 0$ $2 - 6 = \mathbf{-4}$
1d: Add one day for each lap year (year divisible by 4 but not by 100, unless divisible by 400)	$\boldsymbol{d_{LAP}}$	$Lap\ years\ from\ 1980\ to\ 2021 = \mathbf{11}$ ('80 – '84 – '88 – '92 – '96 – '00 – '04 – '08 – '12 – '16 – '20)
1. TOTAL DAYS from January 6th, 1980	$d_{YEARS} + d_{DAYS} + d_{LAP} = \boldsymbol{tot.days}$	$14965 - 4 + 11$ $= \mathbf{14\,972\ days}$
2a: Total number of seconds	$tot.days \cdot 86400 \frac{seconds}{day}$ $= \boldsymbol{tot.sec}$	$14972 \cdot 86400$ $= \mathbf{1\,293\,580\,800\ sec}$
2b: Total number of weeks	$\frac{tot.sec}{604800} = \boldsymbol{tot.weeks}$	$\frac{1\,293\,580\,800}{604800}$ $= \mathbf{2138.85714\ weeks}$
2. The integer part of tot. weeks (which we call WEEKS) represents the first part of the final result, while for the second part it is necessary to work on the decimal part (which we call x_{dec})		$\boldsymbol{WEEKS = 2138}$
3a: Days into weeks	$x_{dec} \cdot 7 \frac{days}{weeks} = \boldsymbol{x_{days}}$	$0.85714 \cdot 7 = \mathbf{6\ days}$
3b: Number of seconds into weeks	$x_{days} \cdot 86400 \frac{seconds}{days} = \boldsymbol{x_{sec}}$	$6 \cdot 86400 = \mathbf{518\,400\ sec}$
3c: Seconds from *00:00:00*	$hh \cdot 3600 + \min \cdot 60 + sec = \boldsymbol{y_{sec}}$	$6 \cdot 3600 + 22 \cdot 60 + 43$ $= \mathbf{22\,963\ sec}$
3: SECONDS = second part of the final result (added to the first part obtained in step 2)	$x_{sec} + y_{sec} = \boldsymbol{SECONDS}$	$518400 + 22963 = 541363\ sec$ $(\boldsymbol{SECONDS = 541\,363})$

FINAL RESULT	WEEKS. SECONDS	2138.541363

The weeks are transmitted with a 10-bit encoding: this means that the number of weeks spent are reset every 1024 weeks (about 19.6 years).

A.2. Lagrange Interpolation

Another very important step that was required to implement the Least Squares algorithm, was the *Lagrange interpolation*.
This process was necessary because the observation periods present in the SP3 file (and therefore also the respective data for each satellite) were reported at intervals of 5 minutes from each other, while in the RINEX file the observation epochs were at intervals of 30 seconds from each other. Therefore, it was necessary to interpolate the data coming from the SP3 file in such a way as to obtain all the data necessary for the implementation of the Least Squares, at *intervals of 30 seconds*. To accurately implement the interpolation, one must consider a certain number of values before and after the current value, in order to have a sample of values around the current value to make the interpolation.
The implementation of the function that performed the Lagrange interpolation has raised several problems to be solved so that the final results were accurate and correct: the most challenging ones were the times alignment and the verification that for each current value there were a chosen number of previous and subsequent values (ten in total), otherwise the interpolation at that point could not have taken place (or it could but with less accuracy since there are less values to interpolate).

Appendix B: Effect of errors

In chapter 3, the different sources of errors that influence the pseudorange/slant range measurements have been discussed, with the implementation of the model and formula associated.
The following figures show the effect of these errors considered alone or in combination with each other, considering the implementation of the Extended Kalman Filter over a static user.

Single Errors effect

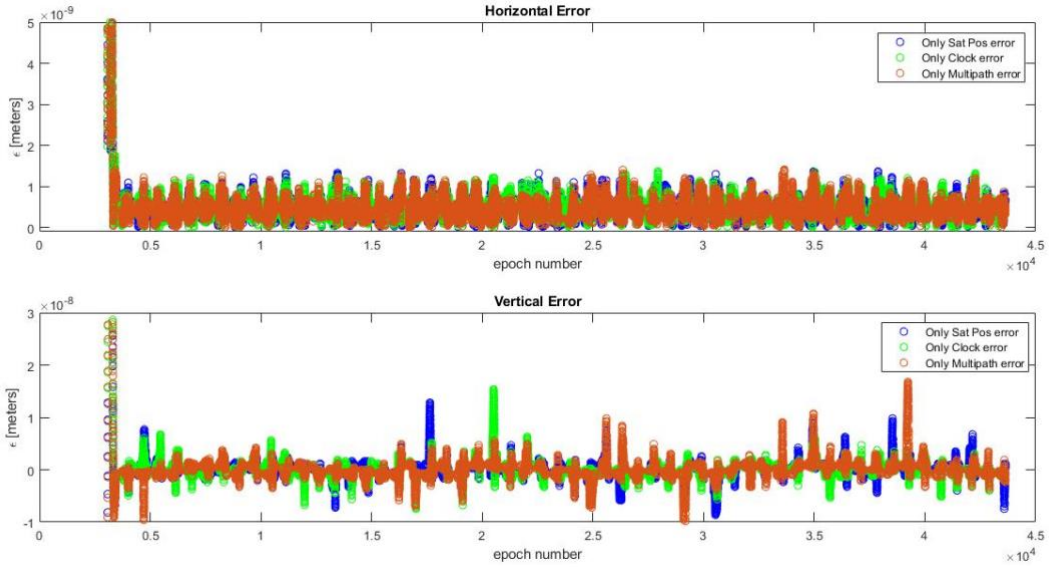

Figure B.1: Effect of the different errors alone

Errors Combinations effect

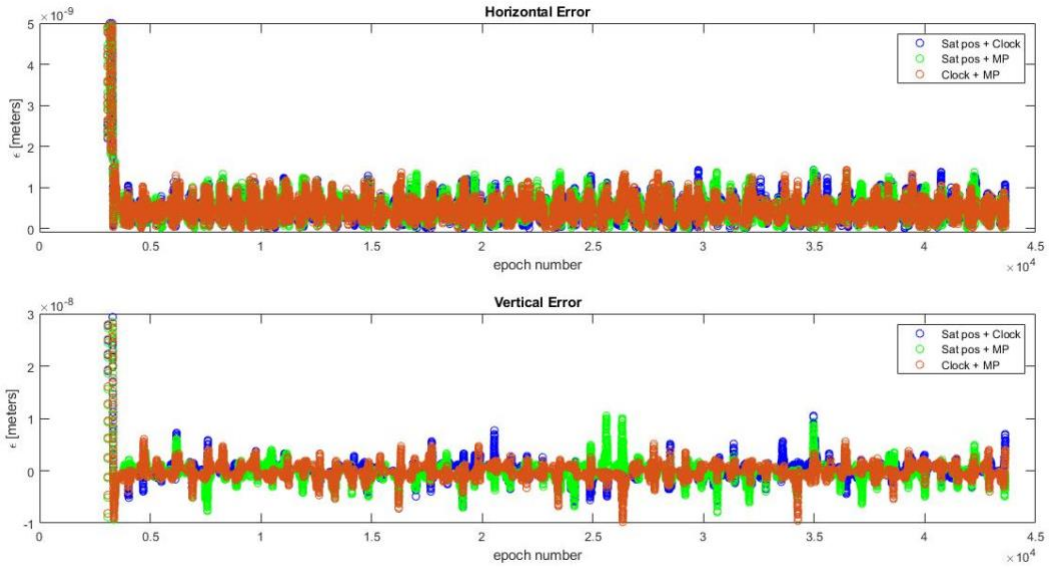

Figure B.2: Effect of the errors in combination with each other

Appendix C: Bowring iterative method

In order to determine the curvilinear coordinates *(Latitude φ, Longitude λ and Height h)* starting from the cartesian coordinates *(x,y,z)*, numerous solution have been devised, both closed-form and iterative. A popular and highly convergent iterative method was introduced by B.R. Bowring in 1976 [29] based on Newton method.
The procedure of this iterative method is described here, showing its fundamental steps.

Input data:

- $[x, y, z]$ are user coordinates
- a is the semimajor axis of the reference ellipsoid
- b is the semiminor axis of the reference ellipsoid
- $e = \sqrt{1 - \frac{b^2}{a^2}}$ is the eccentricity of the reference ellipsoid
- $e' = \sqrt{\frac{a^2}{b^2} - 1} = \frac{a}{b} e$ is second the eccentricity of the reference ellipsoid

Procedure:

$$p = \sqrt{x^2 + y^2}$$

$$\tan u = \left(\frac{z}{p}\right)\left(\frac{a}{b}\right)$$

Iteration Loop

$$\cos^2 u = \frac{1}{1 + \tan^2 u}$$

$$\sin^2 u = 1 - \cos^2 u$$

$$\tan \varphi = \frac{z + e'^2 \ b \ \sin^3 u}{p - e^2 \ a \ \cos^3 u}$$

$$\tan u = \left(\frac{b}{a}\right) \tan \varphi$$

Until *tan u* converges, then

$$N = \frac{a}{\sqrt{1 - e^2 \sin^2 \varphi}}$$

$$h = \begin{cases} \dfrac{p}{\cos \varphi} - N, & \varphi \neq 90° \\ \dfrac{z}{\sin \varphi} - N + e^2 N, & \varphi \neq 0 \end{cases}$$

$$\lambda = \begin{cases} \arctan\left(\dfrac{y}{x}\right), & x \geq 0 \\ 180° + \arctan\left(\dfrac{y}{x}\right), & x < 0 \ and \ y \geq 0 \\ -180° + \arctan\left(\dfrac{y}{x}\right), & x < 0 \ and \ y < 0 \end{cases}$$

Appendix D: Tuning of EKF parameters

In the implementation of the Extended Kalman Filter technique, the tuning of the parameters of the state error autocovariance matrix Q and the measurement error autocovariance matrix R is an issue of fundamental importance.

Depending on the values assigned to the σ_{xyz}, σ_t, σ_{dt} of the matrix Q and σ_R of the matrix R, the solution modified, given that the filter gives more weight to the model considered or the measurement given in input. The more the values assigned to these parameters is lower, the more the filter gives much weight to the matrix associated.

The following figures show the modification of the results considering the application of the errors on the implementation of the Extended Kalman Filter with a dynamic model over a static user scenario. This scenario has been considered to give an impression of the effect of the tuning on the solution, but of course the reasoning is valid for other applications. In some cases, the effect could be even more impacting on the solution, showing that this procedure is of fundamental importance in the implementation of the Extended Kalman Filter algorithm.

Tuning of σ_{xyz}

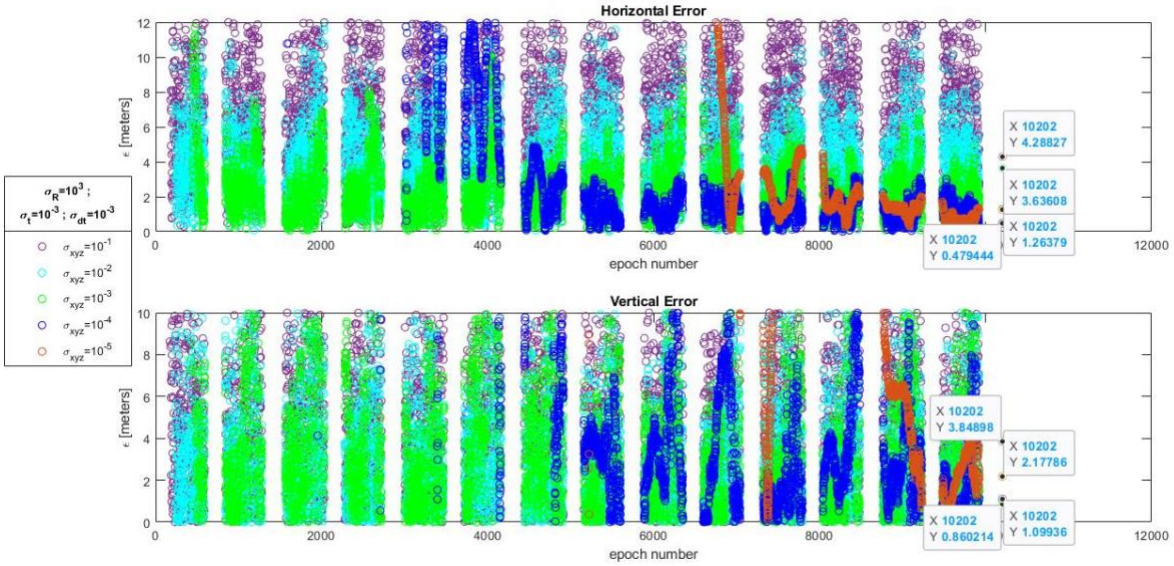

Figure D.1: Tuning of position parameters

Tuning of σ_t and σ_{dt}

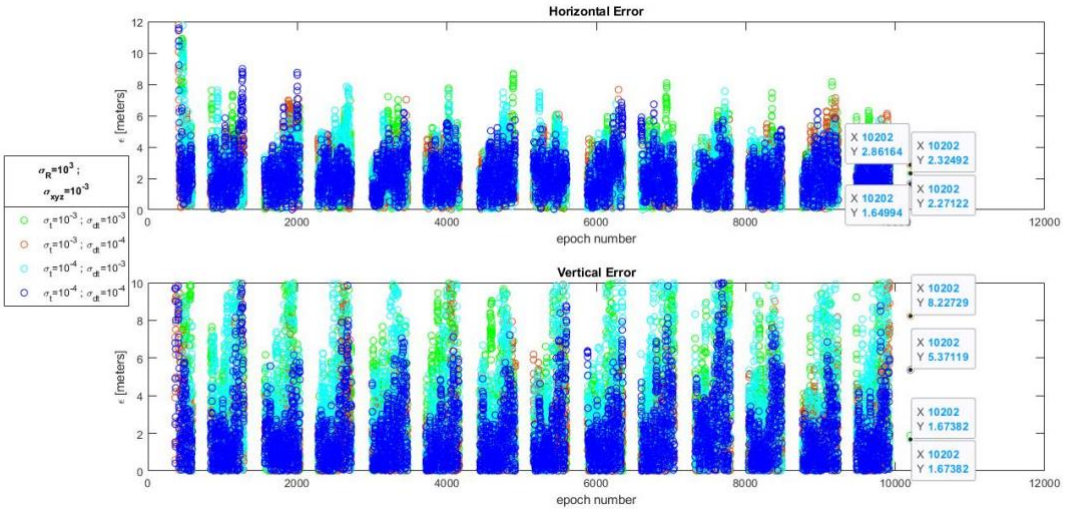

Figure D.2: Tuning of clock parameters

Tuning of σ_R

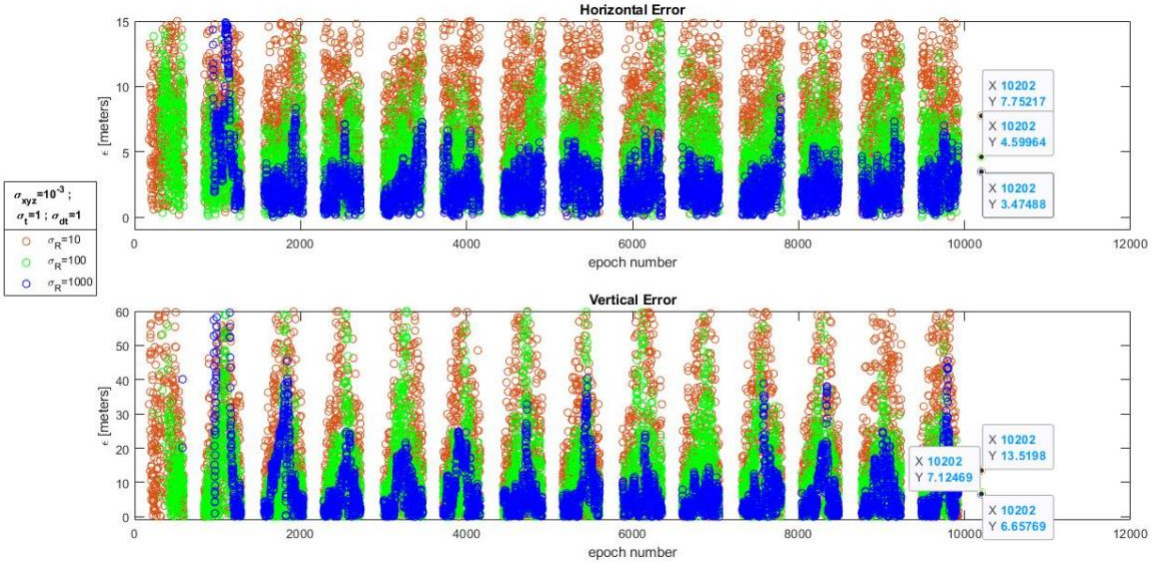

Figure D.3: Tuning of measurement parameters

References

[1] (ISECG), International Space Exploration Coordinate Group, "The Global Exploration Roadmap," NASA, 2018.

[2] A. Grenier, P. Giordano, L. Bucci, A. Cropp, P. Zoccarato, R. Swinden and J. Ventura-Traveset, "Positioning and Velocity Performance Levels for a Lunar Lander using a Dedicated Lunar Communication and Navigation System," Institute of Navigation, 2022.

[3] T. F. Melman, P. Zoccarato, C. Orgel, R. Swinden, P. Giordano and J. Ventura-Traveset, "LCNS Positioning of a Lunar Surface Rover Using a DEM-Based Altitude Constraint," MDPI, Basel, Switzerland, 2022.

[4] ESA, "Navipedia," 12 January 2012. [Online]. Available: https://gssc.esa.int/navipedia/index.php/Main_Page.

[5] J. R. Clynch, "The GLOBAL POSITIONING SYSTEM," 5 February 2003. [Online]. Available: https://www.oc.nps.edu/oc2902w/gps/gpsoview.htm.

[6] GNSS-SDR, "PVT," 2020. [Online]. Available: https://gnss-sdr.org/docs/sp-blocks/pvt/.

[7] D. J. Jwo, M. H. Hsieh and Y. C. Lee, "GPS navigation solution using the iterative least absolute deviation approach," Scientia Iranica, Teheran, Iran, 2015.

[8] Vector Nav, "Least Squares, Weighted Least Squares and NonLinear Least Squares," 2008. [Online]. Available: https://www.vectornav.com/resources/inertial-navigation-primer/math-fundamentals/math-leastsquares.

[9] M. A. Griffioen, "Assessment of Lunar Positioning Accuracy with PECMEO Navigation Satellites," Delft University of Technology, Delft, 2020.

[10] M. F. Abdel-Hafez, "The Autocovariance Least-Squares Technique for GPS Measurement Noise Estimation," IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, 2010.

[11] K. Lemon and B. W. Welch, "Comparison of Nonlinear Filtering Techniques for Lunar Surface Roving Navigation," NASA, 2008.

[12] H. J. Ramos, K. M. Brink and J. E. Hurtado, "Square Root Partial-Update Kalman Filter," in *22nd International Conference on Information Fusion*, Ottawa, Canada, 2019.

[13] D. S. Chiu and K. P. O'Keefe, "Bierman-Thornton UD Filtering for Double-Differenced Carrier Phase Estimation Accounting for Full Mathematical Correlation," in *ION NTM*, San Diego, CA, 2008.

[14] G. H. Born, "Potter Square Root Filter," ASEN 5070, 2002.

[15] C. D'Souza and R. Zanetti, "Information Formulation of the UDU Kalman Filter".

[16] Y. Oshman and I. Y. Bar-Itzhack, "Square Root Filtering via Covariance and Information Eigenfactors," International Federation of Automatic Control, Great Britain, 1986.

[17] M. A. Bashir, F. M. Malik, Z. A. Akbar and M. Uzair, "Kalman Filter Based Sensor Fusion for Altitude Estimation of Aerial Vehicle," IOP Conference Series: Material Science and Engineering, Rawalpindi, Pakistan, 2020.

[18] A. M. Sabatini and V. Genovese, "A Sensor Fusion Method for Tracking Vertical Velocity and Height Based on Inertial and Barometric Altimeter Measurements," *MPDI,* 24 July 2014.

[19] J. R. Bruzzi, K. Strohbehn, B. G. Boone, S. Kerem, R. S. Layman and M. W. Noble, "A Compact Laser Altimeter for Spacecraft Landing Applications," Johns Hopkins APL Technical Digest, 2012.

[20] T. Lacey, *Tutorial: The Kalman Filter,* Massachusetts Institute of Technology.

[21] M. S. Grewal and A. P. Andrews, "Applications of Kalman Filtering in Aerospace 1960 to the Present," 2010.

[22] H. Yong, H. XiaoGong, L. PeiJia, C. JianFeng, J. DongRong, Z. WeiMin and F. Min, "Precise positioning of the Chang'E-3 lunar lander using a kinematic statistical method," Chinese Science Bulletin, Shangai-Beijing, China, 2012.

[23] J. Sanz Subirana, J. M. Juan Zornoza and M. Hernàndez-Pajares, ESA GNSS DATA PROCESSING, Vol.I: Fundamentals and Algorithms, 2013.

[24] Hexagon, "Novatel," 1978. [Online]. Available: https://novatel.com/an-introduction-to-gnss/chapter-4-gnsserror-sources/error-sources.

[25] G. Sirbu and M. Leonardi, *Performance evaluation of a satellite navigation system for lunar exploration,* Roma, 2020.

[26] M. Karaim, M. Elsheikh and A. Noureldin, "GNSS Error Sources," 6 April 2018. [Online]. Available: https://www.intechopen.com/books/6540.

[27] E. D. Kaplan and C. J. Hegarty, Understanding GPS/GNSS: Principles and Applications, Artech House, 2017.

[28] C.-M. Lee and K.-D. Park, "Generation of Klobuchar Ionospheric Error Model Coefficients Using Fourier Series and Accuracy Analysis," Journal of Astronomy and Space Sciences, Incheon, Korea, 2011.

[29] R. M. Toms, "An Improved Algorithm for Geocentric to Geodetic Coordinate Conversion," Lawrence Livermore National Laboratory, Orlando, FL, 1996.

[30] M. Wickert and C. Siddappa, "Exploring the Extended Kalman Filter for GPS Positioning Using Simulated User and Satellite Track Data," 2018.

[31] Corning Museum of Glass, "Reflections on Apollo," 15 October 2019. [Online]. Available: https://blog.cmog.org/2019/10/15/reflections-on-apollo/.

[32] V. Rosmorduc, J. Benveniste, E. Bronner, S. Dinardo, O. Lauret, C. Maheu, M. Milagro, N. Picot, A. Ambrozio, R. Escolà, A. Garcia-Mondejar, E. Schrama, M. Restano and M. Terra-Homem, "Radar Altimetry Tutorial - brat," ESA - CNES, 2018.

[33] G. Sirbu and M. Leonardi, *Analisi preliminare e valutaione delle prestazioni di un sistema di posizionamento per la navigazione lunare,* Roma, 2021.

[34] IGS and RTCM-SC104, *The Receiver Indipendent Exchange Format (RINEX),* 2018.

[35] S. Hilla, N. G. Survey, N. O. Service and NOAA, *The Extended Standard Product 3 Orbit Format (SP3-d),* 2016.

[36] J. Sanz Subirana, J. M. Juan Zornoza and M. Hernàndez-Pajares, ESA GNSS DATA PROCESSING, Vol.II: Laboratory Exercises, 2013.

[37] J. J. Parker, F. Dovis, B. Anderson, L. Ansalone, B. Ashman, F. H. Bauer, G. D'Amore, C. Facchinetti, S. Fantinato, G. Impresario, S. A. McKim, E. Miotti, J. J. Miller and M. Musmeci, "The Lunar GNSS Receiver Experiment (LuGRE)," 2022.

[38] E. S. J. Muller and P. M. Kachmar, "The Apollo rendezvous navigation filter theory, description and performance," in *APOLLO: Guidance, Navigation and Control*, Cambridge, Massachusetts, MIT Charles Stark Draper Laboratory, 1970, p. 70.

[39] Aerospace America, "Honoring a legacy algorithm," September 2016. [Online]. Available: https://aerospaceamerica.aiaa.org/departments/honoring-a-legacy-algorithm/.

[40] P. Lynch, "Kalman filters have applications from moon to motorway," The Irish Times, 22 June 2021. [Online]. Available: https://www.irishtimes.com/news/science/kalman-filters-have-applications-from-moon-to-motorway-1.4600269.

[41] A. Becker, "The Kalman Filter Tutorial," [Online]. Available: https://www.kalmanfilter.net/default.aspx.

[42] Vector Nav, "Getting up to speed with Kalman Filter," 2008. [Online]. Available: https://www.vectornav.com/resources/inertial-navigation-primer/math-fundamentals/math-kalman.

[43] Y. Laamari, B. Athamena and K. Chafaa, "Particle swarm optimization of an extended Kalman filter for speed and rotor flux estimation of an induction motor drive," 2015.

[44] P. W. Sarunic, "Development of GPS Receiver Kalman Filter Algorithms for Stationary, Low-Dynamics, and High-Dynamics Applications," Australian Government, Department of Defence, DTS-Group-TR-3260, Edinburgh, South Australia, 2016.

[45] C. A. Greenhall, R. Boudjemaa and J. Davis, "The Development of a Kalman Filter Clock Predictor," National Physical Laboratory, Pasadena, CA, 2005.

[46] L. A. Breakiron, "A Kalman Filter for Atomic Clocks and Timescales," U.S. Naval Observatory, Washington, DC, 2001.

[47] I. Reid, "Discrete-time Kalman filter," Hilary Term, 2001.

[48] A. K. N., G. Sasibhushana Rao and C. Suresh, "Extended Kalman Filter for GPS ReceiverPosition Estimation," 2018.

[49] R. G. Brown and P. Y. Hwang, "Introduction to Random Signals and Applied Kalman Filtering, IV ed.," John Wiley & Son, Inc., Hoboken, NJ, 2012.

[50] S. Bhattacharyya, D. L. Mute and D. Gebre-Egziabher, "Kalman Filter-Based Reliable GNSS Positioning for Aircraft Navigation," 2019.

[51] M. F. Rodrìguez, "A Kalman Filter application for GNSS error correction in Intelligent Vehicles," Leganés, 2020.

[52] R. Serrano, "Extended Kalman Filters for Dummies," 18 August 2017. [Online]. Available: https://medium.com/@serrano_223/extended-kalman-filters-for-dummies-4168c68e2117.

[53] Aceinna OpenIMU Developer Manual, "EKF Algorithms," Aceinna Inc Revision, 2018. [Online]. Available: https://openimu.readthedocs.io/en/latest/algorithms.html.

[54] L. A. McGee and S. F. Schmidt, "Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry," NASA, California, 1985.

[55] A. Hooshmand, J. V. Mohammadpour, H. Malki and R. S. Provence, "Distributed Extended Kalman Filtering for Reliable Navigation on Lunar Surface," American Institute of Aeronautics and Astronautics, Inc., Portland, Oregon, 2011.

[56] S. University, *Lecture 9 - The Extended Kalman filter,* 2008.

[57] S. J. Julier and J. K. Uhlmann, "A New Extension of the Kalman Filter to Nonlinear Systems," The Robotics Research Group, Department of Engineering Science, Oxford, OX, 1997.

[58] M. S. Grewal and A. P. Andrews, "Kalman Filtering: Theory and Practice Using MATLAB, II Ed.," John Wiley & Sons, Inc., 2001.

[59] M. I. Ribeiro, "Kalman and Extended Kalman Filters: Concept, Derivation and Properties," Instituto Superior Te´cnico, Lisboa, 2004.

[60] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," Department of Computer Science, UNC-Chapel Hill, 2001.

[61] B. Esme, "Kalman Filter For Dummies," Bilgin's Blog, March 2009. [Online]. Available: http://bilgin.esme.org/BitsAndBytes/KalmanFilterforDummies#.

[62] StackExchange, "How to initialize error covariance matrix in Extended Kalman Filter, Q," June 2009. [Online]. Available: https://math.stackexchange.com/questions/3242936/how-to-initialize-error-covariance-matrix-in-extended-kalman-filter-q.

[63] C. Zucca and P. Tavella, "A mathematical model for the atomic clock error in case of jumps," Turin, 2015.

[64] L. Galleani, L. Sacerdote, P. Tavella and C. Zucca, "A mathematical model for the atomic clock error," Institute of Physics Publishing, Metrologia, Turin, IT, 2003.

[65] H. Li, X. Liao, B. Li and L. Yang, "Modeling of the GPS satellite clock error and its performance evaluation in precise point positioning," Advances in Space Research, Shangai, China, 2018.

[66] B. Bidikar, G. S. Rao, L. Ganesh and S. M. Kumar, "Satellite Clock Error and Orbital Solution Error Estimation for Precise Navigation Applications," Department of Science and Technology, New Delhi, India, 2013.

[67] G. Blewitt, "Basics of the GPS Technique: Observation Equations," Department of Geomatics, University of Newcastle, Newcastle, UK, 1997.

[68] BCS, "Inside GNSS (Global Navigation Satellite Systems Engineering, Policy, and Design)," 2018. [Online]. Available: https://insidegnss.com.

[69] UAV Navigation, "Global Navigation Satellite System (GNSS)," [Online]. Available: https://www.uavnavigation.com/support/kb/general/inertial-navigation-system-and-estimation/global-navigation-satellite-system-gnss.

[70] T. A. Ely and A. H. Chau, "Radar Altimetry and Velocimetry for Inertial Navigation: A Lunar Landing Example," Advances in Astronautcal Sciences, 2011.

[71] GISGeography, "How GPS Receivers Work – Trilateration vs Triangulation," 2019. [Online]. Available: https://gisgeography.com/trilateration-triangulation-gps/.

[72] I. Sarras, G. Gerakios, A. Diamandis, A. I. Dounis and G. P. Syrcos, "Static Single Point Positioning Using The Extended Kalman Filter," World Academy of Science, Engineering and Technology 37, 2010.

[73] J. Sheppard, "What sensors do you need to land on the moon?," SensorTips, 6 April 2022. [Online]. Available: https://www.sensortips.com/featured/what-sensors-you-need-to-land-on-moon-faq/.

[74] F. Amzajerdian, G. D. Hines, L. B. Petway, B. W. Barnes, D. F. Pierrottet and J. M. Carson III, "Development of Navigation Doppler Lidar for Future Landing Mission," NASA.gov, 2016.

[75] G. Vingione, "Radar Altimeter General Waveform Model and Its Application to Cassini Mission," Department of Aerospace and Mechanical Engineering, Second University of Naples, Aversa, Italy, 2007.

[76] D. F. Pierrottet, F. Amzajerdian and B. Barnes, "A long distance Laser Altimeter for terrain relative navigation and spacecraft landing," AIAA Guidance, Navitgation, and Control Conference, Toronto, 2014.