POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Aerospaziale

Tesi di Laurea Magistrale

Development of data processing methods for Space Surveillance and Tracking

Supervisors

Prof. Dario PASTRONE

Dr. Jens UTZMANN

MSc. Guido PEDONE

Candidate

Walter CAMPO

December 2020

Abstract

ESA estimated the current number of space debris to be around 129 million [24]; collisions with orbital debris can bring failure to the mission or completely destroy the spacecraft, so space debris need to be tracked and catalogued to plan and perform collision avoidance. SPOOK is a SST software currently in development at Airbus Defence and Space GmbH in Friedrichshafen, Germany, and the objectives of the thesis were to enhance the software by improving preexisting functions and implementing new ones and to test the implemented methods and settings for Orbit Determination aiming to determine good parameter settings. The following goals have been accomplished:

- two functions have been improved and implemented respectively, that return output files useful for optimization of observation strategies;
- a function has been implemented to compute the orbit determination errors with respect to accurate ephemerides;
- SPOOK has been made faster in performing multiple runs of orbit determination by implementing a function to create multiple initial state vectors and perform orbit determination with each one of them and by writing a Python function that launches orbit determination semi-automatically;
- the tests have determined that simple propagation of the state vector should be performed instead of orbit determination when the accuracy of the initial state is very high, that the Weighted Least Square method is insensitive to the accuracy of the initial state, and that Extended Kalman Filter and Unscented Kalman Filter with simulated measurements return very similar outputs if the physical model used for OD is very accurate.

Acknowledgements

I want to show my gratitude to my company and university supervisors, Dr. Jens Utzmann and Prof. Dario Pastrone, for their wise suggestions that gave me ideas on what to test while at Airbus and on how to structure my thesis.

Special thanks to Guido, who guided me through all my six months at Airbus and always solved my problems with Git; I would not have managed to complete the thesis without him.

Many thanks also to Giovanni and David, who gave me insight into some of the trickiest parts of SPOOK.

Andrea, Federico, Joseba, Jacopo and all the other Airbus interns and thesis students deserve my deepest gratitude for having made my staying in Germany happy and funny.

Last but not least, huge thanks to my parents, who have always supported me and believed in me even in the most difficult situations.

Table of Contents

List of Tables

List of Figures

Abbreviations

ART
: Airbus Robotic Telescope

CCSDS
: Consultative Committee for Space Data Systems

ECEF
: Earth Centered Earth Fixed reference frame

ECI
: Earth Centered Inertial reference frame

EKF
: Extended Kalman Filter

EOP
: Earth Orientation Parameters

ESA
: European Space Agency

GCRF
: Geocentric Celestial Reference Frame

GEO
: Geosynchronous Equatorial Orbit

GNSS
: Global Navigation Satellite System

GPS

Global Positioning System

HEO

High Earth Orbit

ICP

Initial Covariance Propagation

IOD

Initial Orbit Determination

ITRF

International Terrestrial Reference Frame

KVN

Key-value notation

LEO

Low Earth Orbit

MEO

Medium Earth Orbit

OD

Orbit Determination

OEM

Orbit Ephemeris Message

OTDF

Orbit Tracking Definition Format

RAAN

Right Ascension of the Ascending Node

RMS

Root Mean Square

RTN

Radial, Tangential, Normal directions

SBLS

Sequential Batch Least Squares

SNR

Signal to Noise Ratio

SPOOK

Special Perturbations Orbit determination and Orbit analysis toolKit

SPOP

Special Perturbation Object Propagator

SST

Space Surveillance and Tracking

TDM

Tracking Data Messages

TLE

Two-Line Element

UKF

Unscented Kalman Filter

USKF

Unscented-Schmidt Kalman Filter

WLS

Weighted Least Squares

WRMS

Weighted Root Mean Square

Chapter 1

Introduction

1.1 The Space Debris problem

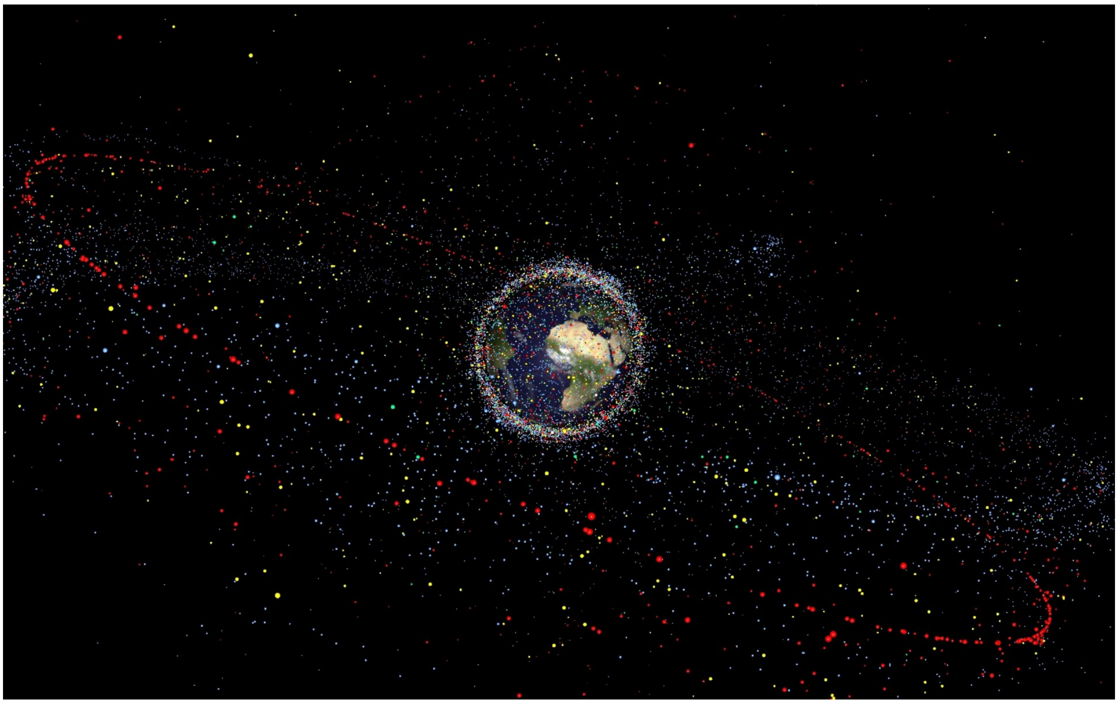

Figure 1.1: Model of the space debris distribution around Earth [23].

Nowadays orbital debris pose a huge threat for the safety and success of space missions. Defined as "all the inactive, man-made objects, including fragments, that are orbiting Earth or reentering the atmosphere" [13], space debris mainly occupy low Earth orbits and geostationary orbits, as shown in Figure 1.1. ESA estimated that by February 2020 there were 34000 debris larger than 10 cm, 900000 debris with size between 1 cm and 10 cm and 128 million debris ranging from 1 mm to 1 cm size [24].

Space debris are dangerous for spacecrafts because of their high kinetic energy: according to ESA [13], collisions with millimetre-size objects could cause local damage on spacecrafts or disable their subsystems; collisions with debris larger than 1 cm can disable or cause the break-up of satellites and rockets; collisions with debris larger than 10 cm can lead to the complete destruction of the spacecrafts and formation of a new debris cloud. If these catastrophic collisions keep happening, eventually they will cause the so-called *Kessler syndrome*, a chain reaction where collisions generate new debris leading to more collisions that generate even more debris and so on. Therefore countermeasures must be taken to avoid collisions with orbital debris and keep their number in check.

One of the countermeasures to the space debris problem is constituted by the Space Surveillance and Tracking (SST) systems, whose objective is monitoring space, detecting space debris, tracking and cataloguing them in order to plan strategies and manoeuvres to avoid collisions between spacecrafts and orbital debris. As shown by Figure 1.2, today the number of catalogued objects amounts to more than 45000, which is still a small fraction of the space debris' number estimated by ESA.

The *Special Perturbations Orbit determination and Orbit analysis toolKit* (SPOOK) is a SST tool for the cataloguing and orbital analysis of Earth orbiting objects; SPOOK is also able to create observation plans for user-defined sensors: in particular it produces observation plans for the *Airbus Robotic Telescope* (ART), whose measurements are used as testbed for most SPOOK modes. SPOOK is currently in development at Airbus Defence and Space GmbH in Friedrichshafen, Germany, and it is the tool used in this thesis, especially for what concerns the Orbit Determination (OD) section of the software.

1.2 Main Goals

These are the objectives of the thesis:

- Enhancement of SPOOK Sensor Simulation and Orbit Determination modes;
- Study of the OD methods that have been implemented in SPOOK;
- Tests on the available OD settings in the tool and comparison between the implemented methods;
- Definition of good parameter settings for orbit determination.

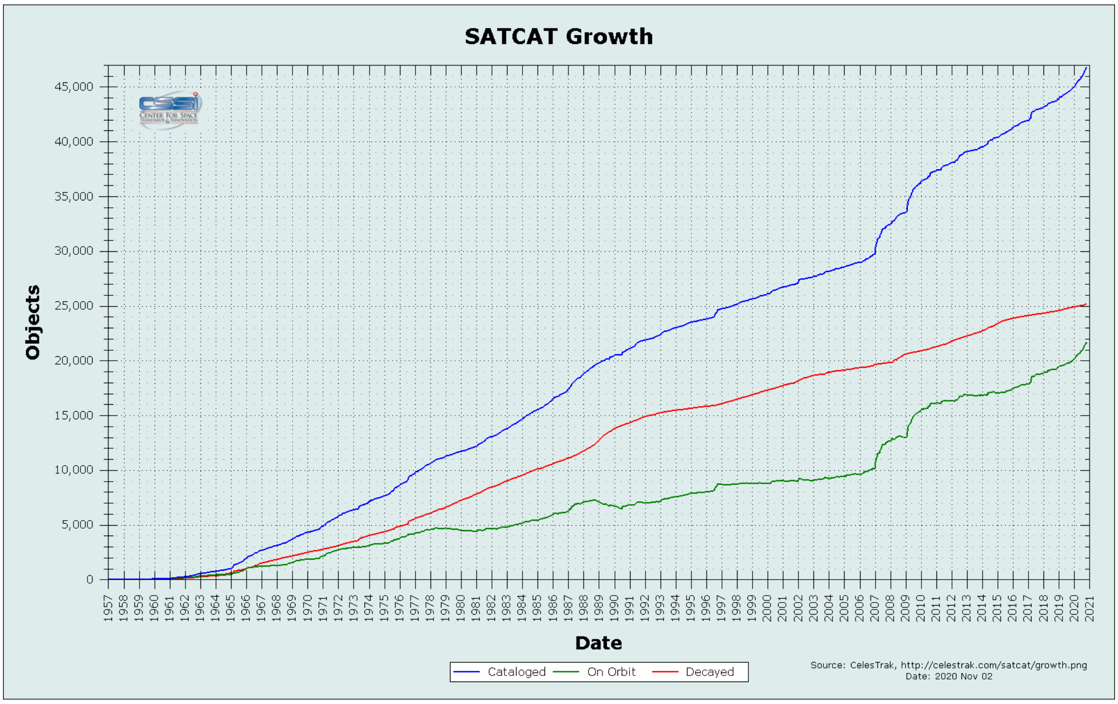

Figure 1.2: Growth of the number of catalogued debris from 1957 to 2020 [22].

1.3 Structure of the Thesis

Here follows an overview of the structure of the thesis:

- the current Chapter 1 briefly describes the orbital debris problem, introduces SPOOK and presents the goals and structure of the thesis;
- Chapter 2 describes SPOOK, its modes and its structure;
- Chapter 3 provides the necessary theoretical background behind the thesis, focusing especially on orbit determination;
- Chapter 4 shows the improvements that have been made on SPOOK in the framework of the thesis;
- Chapter 5 shows and discusses the tests on orbit determination that have been performed;
- Chapter 6 summarizes the conclusions that can be drawn from the previous two chapters;
- Chapter 7 contains some suggestions for who will work on SPOOK in the future.

Chapter 2

SPOOK

The Special Perturbations Orbit determination and Orbit analysis toolKit (SPOOK) is a SST software developed by Airbus Defence and Space in Friedrichshafen. This chapter provides an overview of the SPOOK tool focusing on its structure and modes; also a brief presentation of the ART telescope used to take or simulate measurements is given. A detailed description of SPOOK is available in [21].

2.1 SPOOK structure

The tool consists of a FORTRAN processing core and a Python wrapper: the Fortran part reads the input files, performs the calculations required by the used SPOOK mode and returns the output files; the Python wrapper is used as interface between the user and the core, receiving basic direct inputs from the user, writing the input files and providing them to SPOOK; also the Python wrapper performs the post processing and visualization of the results and manages the objects' database, which would be complex tasks for FORTRAN. SPOOK has its own visualization tool; Figure 2.1 is an example of the pictures it can produce.

SPOOK can be split up into three layers [18]:

- *Simulation layer*: it generates the objects population, performs sensor simulation and propagates the object and the observer (if needed);
- *Analysis layer*: it processes the measurements, performs orbit determination, initial orbit determination, covariance propagation and further analysis on the results;
- *Interface layer*: mostly implemented in the Python wrapper, it writes the input files and manages the outputs and the visualization tool.

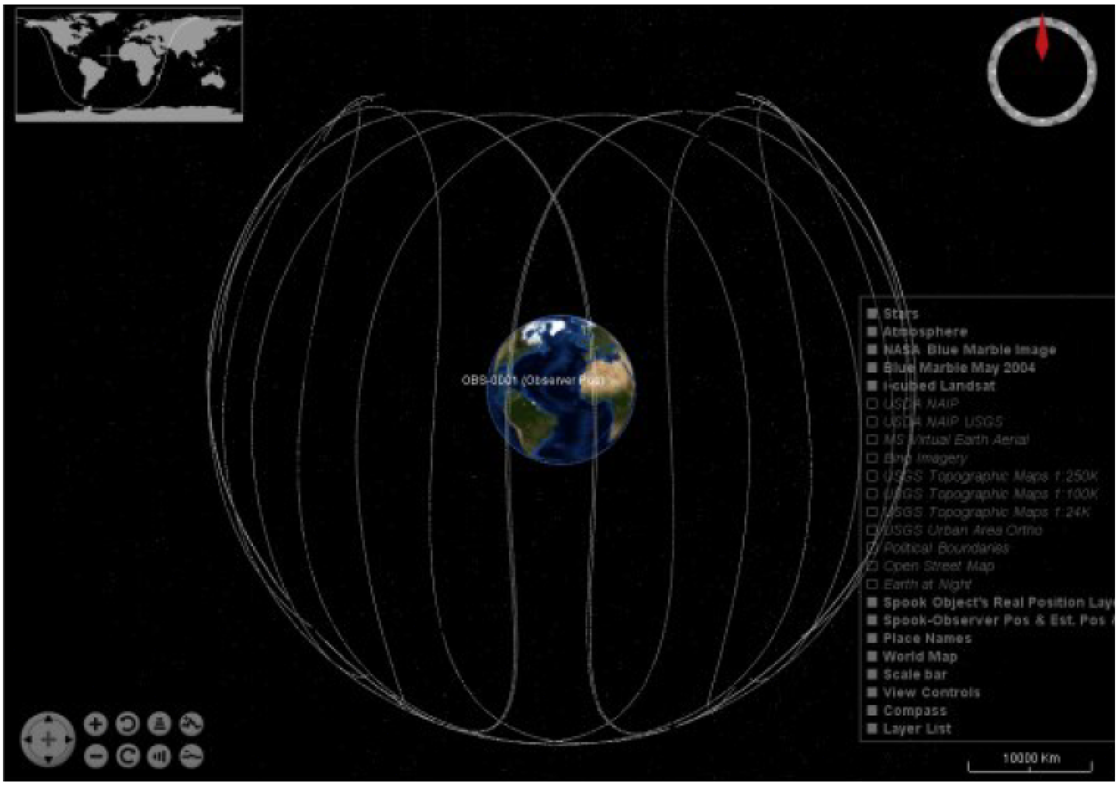

Figure 2.1: Picture produced by SPOOK visualization tool [21].

There are three input files that are always required by SPOOK independently of its mode: the *objects configuration file* to define the characteristics of one or more space objects (see Section 3.3); the *observers configuration file* to define the characteristics of one or more observers and give additional information like weather and illumination conditions, the pointing mode and so on; the *parameters configuration file* to set the options for executing SPOOK in the desired way, e.g. the SPOOK mode, the time step for propagation, the chosen method for orbit determination and so on. These three files are written in Key-Value Notation (KVN), where each option corresponds to a keyword and a value.

Additional input files may be needed basing on the SPOOK mode and the content of the three main files: e.g. measurements file, space weather files and ephemerides.

2.2 SPOOK modes

Figure 2.2 shows the main SPOOK modes, that are:

- *Sensor Simulation*: SPOOK simulates the observer(s) defined in the observers configuration file, estimates the measurements of the object(s) defined in the objects file (by propagating the state vector(s) in the desired time frame and computing the corresponding measurements with Equation 3.4) and applies optical and relativistic corrections to the measurements;

- *Sensor Calibration*: the time bias of the sensor is evaluated by comparing the real or simulated measurements of an object with the measurements estimated from an ephemeris of the same object [20] (usually interpolation is needed to match the measurements' and ephemeris' times);

- *State and Covariance Propagation*: the state vector and covariance matrix of an object are propagated from an initial time (for which the state and covariance have to be provided) to an user-defined final time; the propagation is performed by an indipendent tool, the *Special Perturbation Object Propagator* (SPOP), that is able to take into account several types of perturbations;

- *OEM to ECI conversion*: an ephemeris inside an OEM file is converted into the ECI reference frame;

- *Orbit Determination*: initial orbit determination (if needed) and orbit determination are performed as explained in Sections 3.5 and 3.6;

- *Correlation*: Tracklets, i.e. series of measurements, are correlated to objects or to other tracklets.

2.3 The ART telescope

The *Airbus Robotic Telescope* (ART), located in Extremadura, Spain, is an optical observer that is able to take measurements from every orbital region. It is used to test observation strategies and provide SPOOK with real measurements (there is an interface between ART and SPOOK); thanks to ART, SPOOK is able to perform and manage the complete pipeline from the acquirement of measurements to orbit determination.

In order to take measurements, ART requires an observation plan, i.e. settings and sets of pointings (couples right ascension - declination); if the observation plan is provided, the observation is performed automatically. All the technical information about ART are available in [8], from which the previous description comes.

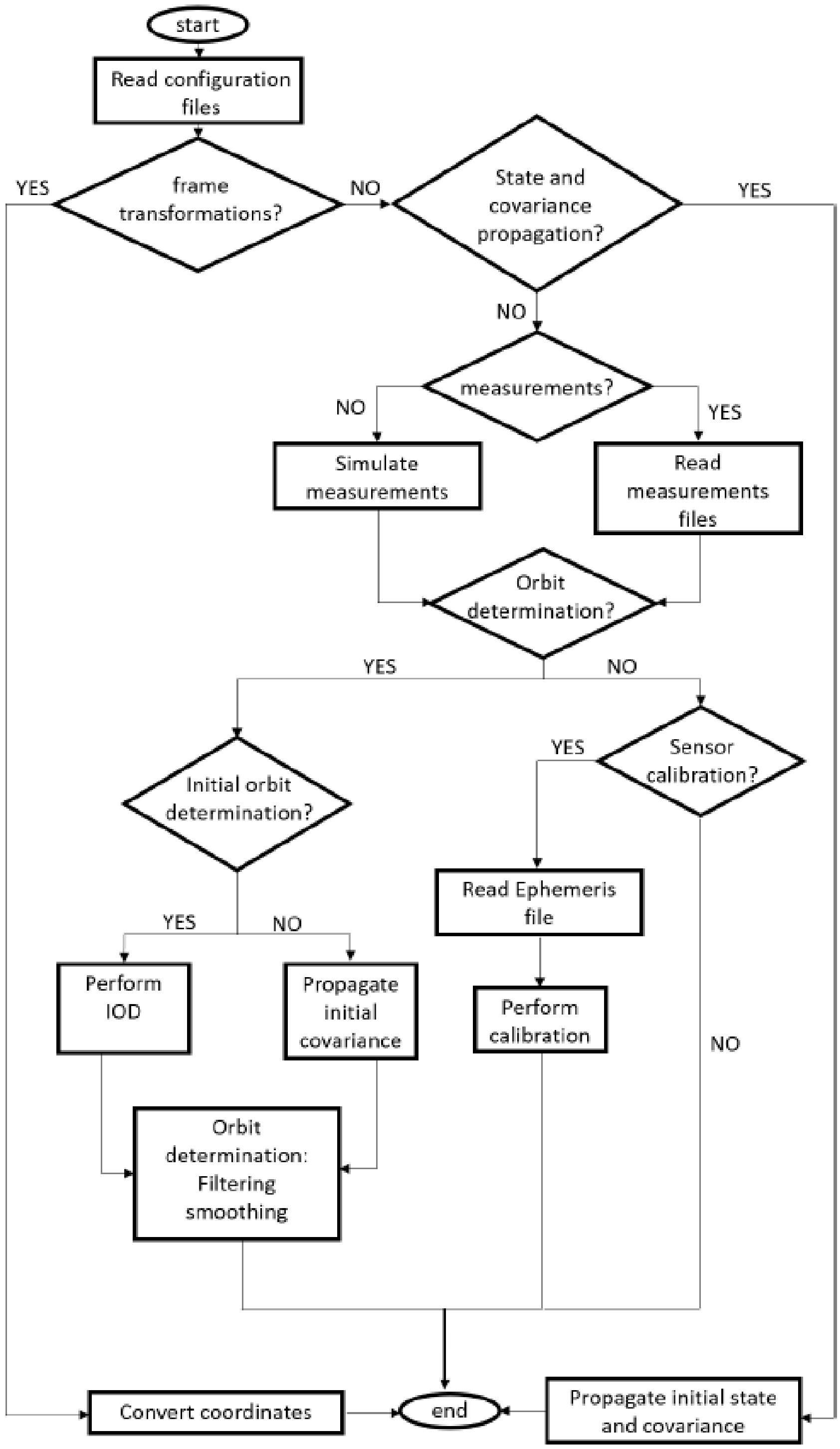

Figure 2.2: Flowchart of the main SPOOK modes [20].

Figure 2.3: Airbus Robotic Telescope [8].

Chapter 3

Theoretical background

This chapter provides the theoretical background behind the work carried out in this thesis. The concept of state vector, the coordinate systems, the files for object definition and measurements input and the different types of measurements are presented; the implemented methods for initial orbit determination and orbit determination are described as well as the OD success criteria; eventually the classification of orbital regions is shown.

3.1 Object state vector

Orbit determination is used to estimate the state vector of a space object, which is a vector whose components determine the position and velocity of the object in space at a specific time t. Several representations are possible for the state vector; in this thesis the state vector $\mathbf{x}$ consists of three components of position, forming the position vector $\mathbf{r}$, and three components of velocity, forming the velocity vector $\mathbf{v}$:

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \end{bmatrix} \tag{3.1}$$

In SPOOK two more parameters can be included in the state vector: the atmospheric drag coefficient C_D and the solar radiation pressure coefficient C_{SP}.

3.2 Reference frames

The state vector of an object can be represented in various reference frames. Here the three reference frames implemented in SPOOK are presented.

The *Earth Centred Inertial* (ECI) coordinate system, also called *Geocentric Celestial Reference Frame* (GCRF), is an inertial frame centred in the Earth's centre of mass. As shown in Figure 3.1, the **I**-axis points towards the Vernal Equinox, the **K**-axis points towards the Earth's north pole and the **J**-axis completes the right-handed system. However, the orientation of the North Pole changes because of Earth's precession and nutation and so should do the orientation of the **K**-axis; to avoid this the axes are defined w.r.t. a particular epoch: in SPOOK the reference date is the 1^{st} January of 2000 at 12:00 [18].

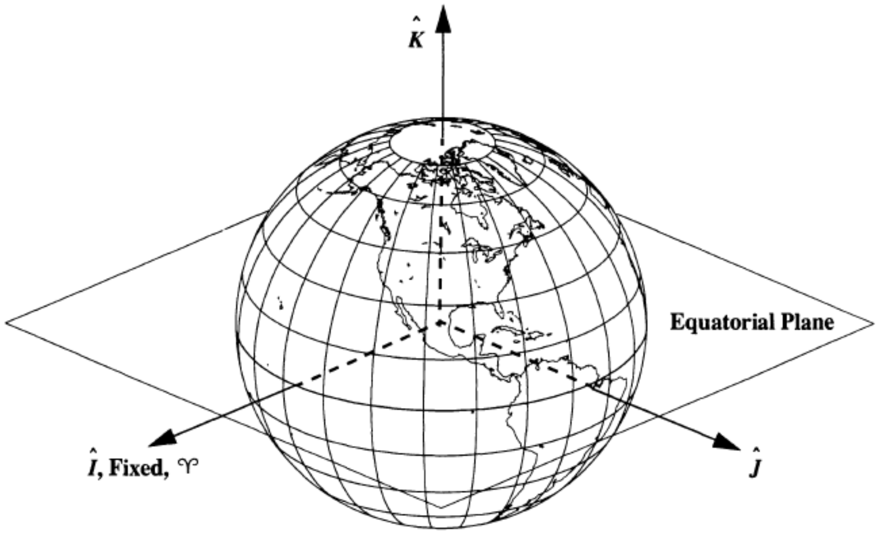

Figure 3.1: ECI reference frame [1].

The *Earth Centered Earth Fixed* (ECEF) coordinate system, also known as *International Terrestrial Reference Frame* (ITRF), is an Earth-centred reference frame that rotates with the Earth. The **X**-axis points towards a reference meridian, the **Z**-axis directs towards the reference pole and the **Y**-axis completes the right handed system [14]. The ECEF coordinate system is shown in Figure 3.2.

The *Radial-Tangential-Normal* (RTN) reference frame is an object-centred coordinate system whose axes are the **R** axis (on the orbital plane, parallel with the object's radius vector), the **T** axis (on the orbital plane, tangential to the orbit) and the **N** axis (perpendicular to the orbital plane); Figure 3.3 shows a representation of the RTN system.

The RTN reference frame has been chosen for the tests performed in Chapters 4 and 5.

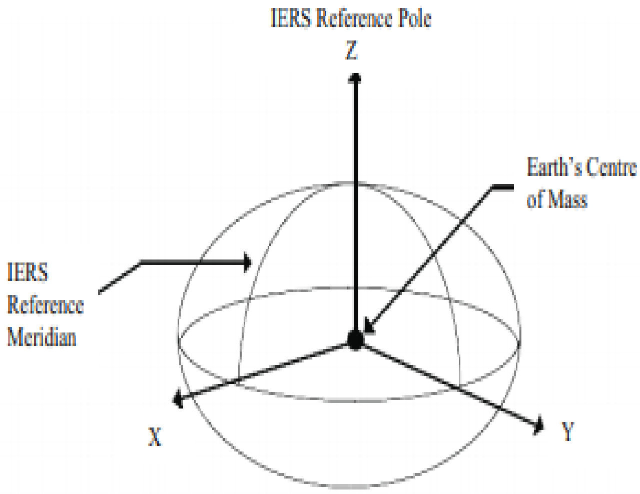

Figure 3.2: ECEF reference frame [3].

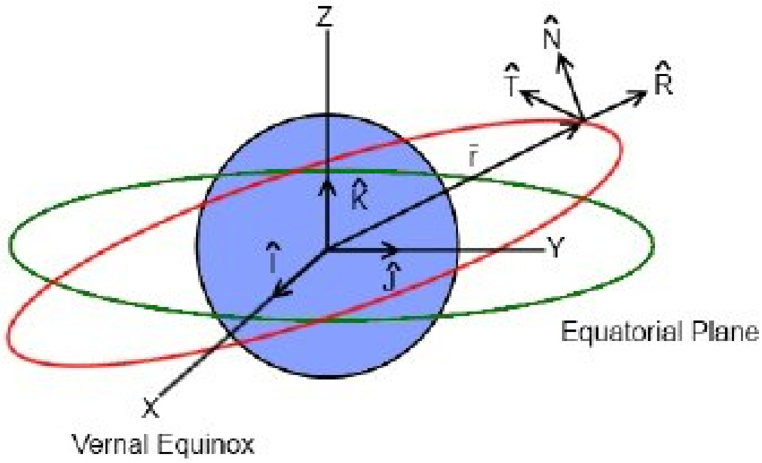

Figure 3.3: RTN reference frame [7].

3.3 Input files for object definition

When SPOOK works in Orbit Determination mode, it needs one or more space objects (also referred to as *target objects*) to be defined so that its or their orbits can be estimated; the definition of one or more objects is also required by the other SPOOK modes, that are not the scope of this thesis.

"Defining an object" means providing SPOOK with data about that specific object, e.g. the name, the size, the mass and so on; different SPOOK modes generally need different types of data about the object(s).

If SPOOK is used in Orbit Determination mode and initial orbit determination is not performed (see Section 3.5), the initial state vector $\mathbf{x_0}$ of the object and the initial covariance matrix $\mathbf{P_0}$, both referred to an initial time t_0 which has to be specified too, are required by the software. SPOOK can be provided with the initial time, state vector and covariance matrix in several ways:

- the user can write them manually in the objects configuration file (see Section 2.1); the initial state vector and covariance matrix can be written in either ECI or ECEF or RTN reference frame;
- real or simulated ephemerides of the object can be provided by passing an *Orbit Ephemeris Message* (OEM) file to SPOOK: OEM files contain the state vector and the covariance matrix of the object at given epoch times;
- a *Two-Line Element* (TLE) file can be provided to SPOOK: this file contains object related information from which the state vector can be derived.

OEM files are written according to the standards set by the *Consultative Committee for Space Data Systems* (CCSDS); the CCSDS recommended standards for the structure, the notation and the contents of OEM files are available in [4]. Subsection 3.3.1 describes the structure of TLE files and the data they contain. Examples of OEM and TLE files can be found in Appendix B.

3.3.1 TLE files

The following description and discussion about TLE files come from [20], where more information on this topic are available.

TLE files contain one or more sets of data arranged in two lines as shown in Figure 3.4; these files can contain data about more than one object but each two-line set refers to a single object and epoch (indeed the object's number, the international designator and the epoch are among the parameters included in a two line set).

The values contained in a two line set include five of the six *Earth Orientation Parameters* (EOP) of the object: inclination i, RAAN Ω, eccentricity e, argument of perigee ω and mean anomaly M; these five values are the "mean" orbital elements, i.e. they have been computed by removing some periodic oscillations [15]. The sixth "mean" orbital element is the semi-major axis $\overline{a}$, which can be computed from the mean angular motion $\overline{n}$ (one of the data in the set) using the equation

$$\overline{n} = \sqrt{\frac{\mu}{\overline{a}^3}} \tag{3.2}$$

where μ is Earth's gravitational parameter ($3.986 \times 10^{14} m^3 s^{-2}$ [6]).

So SPOOK can convert the six orbital elements of a two-line set into a Cartesian state vector in the desired reference frame (RTN for this thesis). It is important to notice that while all the possible methods to provide SPOOK with a state vector introduce uncertainties due to the ways the single components have been computed and how well the measurements have been taken, the TLE format also introduces an additional uncertainty due to the fact that the epoch and the orbital elements are written with a finite number of digits [20].

TLE files do not provide any accuracy information, so it would be impossible to compute the covariance matrix of the object. However, Fiusco [15] implemented a method to extract the covariance matrix from a TLE file in a statistical manner; to use this algorithm the TLE file has to contain several instances of the same object, spread in a time interval between 15 days and one month: the algorithm computes the state vector and covariance matrix at the epoch of the last instance.

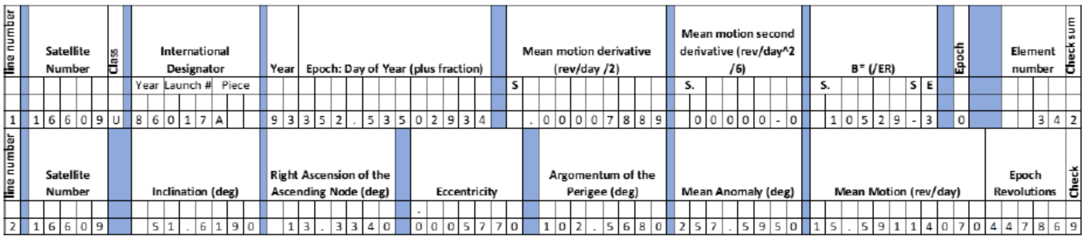

Figure 3.4: Structure of a TLE set [20]. S is the sign of the values, E the exponent.

3.4 Observation Theory

Measurements of the space object are needed to perform orbit determination: SPOOK can either provide the OD algorithms with measurements from external sources or simulate an observer and produce simulated measurements. This subsection presents the different types of real or simulated observers and the kinds of measurements provided by them; it also briefly describes the measurements' files that can be taken as input by SPOOK.

It is worth mentioning that whether they are real or simulated, inside SPOOK the measurements are processed in the form of *tracklets*, i.e. series of consecutive measurements referred to a single object for only a small portion of its orbit. More information on how tracklets are built and associated to an object (because generally the observer takes measurements of more than one object during its observation time) can be found in [19].

3.4.1 Observers types and measurements

If observations are made from Earth, we talk about "ground-based observers"; instead the observer is defined as "space-based" if it is a satellite orbiting around Earth. We can also distinguish between two types of observers basing on their functioning [11]:

- *optical observers* (i.e. telescopes): their sensors receive the light reflected by the observed object; knowing the pointing information of the observer, the direction of the object is derived in the form of two angular measurements: topocentric right ascension and declination (see Figure 3.5).

- *radar observers*: they emit radio waves that are reflected by the object and collected by their sensors; the direction of the object is measured as a couple of angles, i.e. azimuth and elevation (see Figure 3.6). Radar observers can provide two additional measurements: the slant range ρ, by measuring the time needed for the signal to go from the radar to the object and back to the radar, and the slant range rate $\dot{\rho}$, by using the Doppler effect [11].

Both radars and optical observers can be ground-based or space-based.

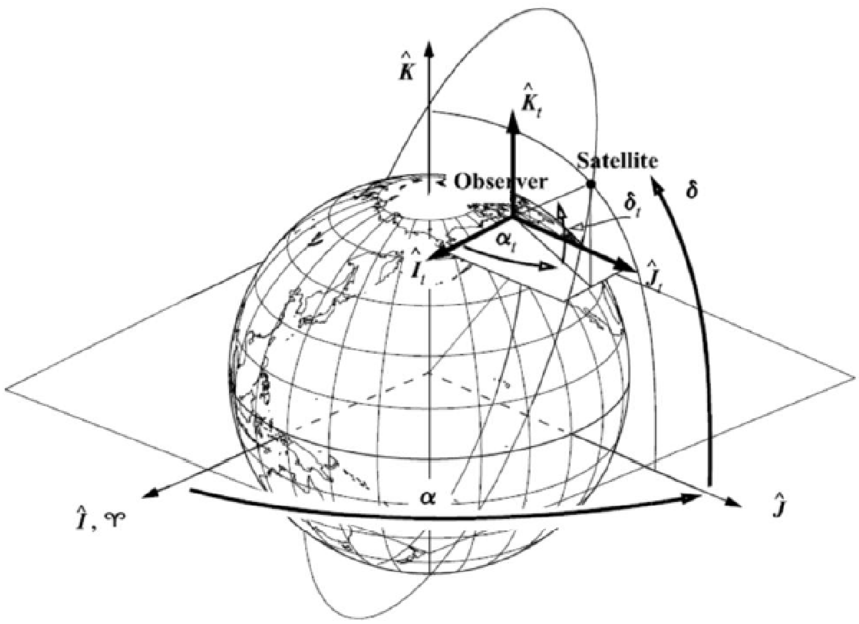

Figure 3.5: Optical measurements: right ascension α_t and declination δ_t [12].

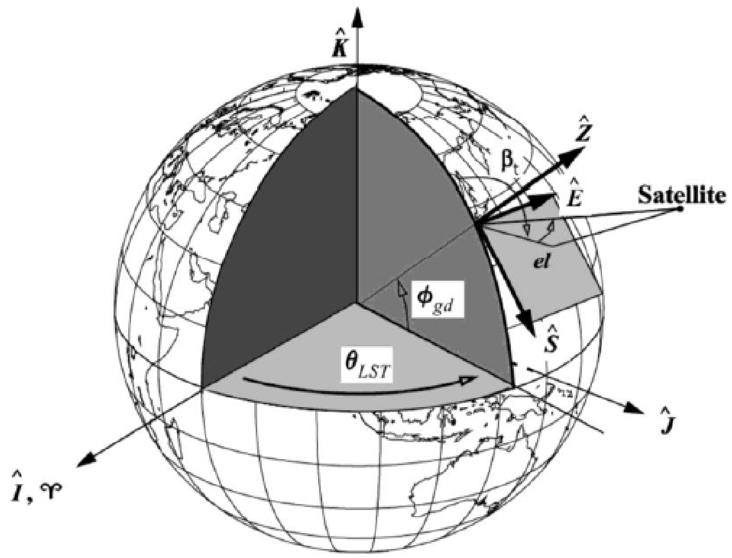

Figure 3.6: Radar measurements: azimuth β_t and elevation el [12].

3.4.2 Measurements' files

Real or simulated measurements have to be written in files that SPOOK receives as input to perform OD or other functions depending on which mode is used. There are two main formats of measurements' files that SPOOK can read and elaborate [16]:

- *Orbit Tracking Definition Format* (OTDF): for each measurement time it shows the measurements mentioned in Subsection 3.4.1 and also the location of the observer and information like pressure, humidity and temperature at observer's location (however these additional information can be shown only in the case of ground-based observers). In this format the measurements are not divided into tracklets;

- *Tracking Data Messages* (TDM): the main difference from OTDF is that TDM format divides the measurements into tracklets; TDM files' structure, content and notation follow the CCSDS recommended standards, that are available in [17].

In this thesis only TDM files have been used. Appendix B provides examples of OTDF and TDM files.

3.5 Initial Orbit Determination

The OD methods of Section 3.6 require an initial state vector $\mathbf{x}_0$ (referred to an initial time t_0) of the target object; filters also require an initial covariance matrix $\mathbf{P_0}$.

In SPOOK two options are available to provide the initial state vector and covariance matrix:

- *Initial Covariance Propagation* (ICP), where the initial state vector and covariance matrix (if needed) are manually written by the user or extracted from the input files that are presented in Section 3.3; then the initial state and covariance matrix are propagated to the time of the first measurement and OD is performed.

- *Initial Orbit Determination* (IOD), where a limited set of measurements is used to estimate the initial state vector.

During the work for this thesis only ICP has been used; however the following subsections briefly describe the implemented methods for IOD. If filters are used to perform OD after initial orbit determination, SPOOK uses the Weighted Least Squares method (Subsection 3.6.1) to improve the estimation of the initial state vector and to estimate the initial covariance matrix.

3.5.1 Gauss' Technique

Gauss' method requires three sets of topocentric angular measurements (couples right ascension - declination or azimuth - elevation) at three different and sequential times ($t_1 < t_2 < t_3$) and it is able to estimate $\mathbf{r}_2$, i.e. the position vector at the middle time.

The implemented algorithm uses the three measurements to define and solve an 8^{th} order equation for $\mathbf{r}_2$, and the estimated $\mathbf{r}_2$ allows the computation of $\mathbf{r}_1$ and $\mathbf{r}_3$; then either Gibbs' Method or Herrick-Gibb's Method estimates the velocity vector at middle time $\mathbf{v}_2$; the whole process is repeated until the slant ranges ρ_1, ρ_2 and ρ_3 stop changing.

Gauss' Technique works well if the angular separation between the three measurements is lower than 60°[18] and even better if the separation is lower than 10°[1]. The full derivation of the method can be found in [1] and [9].

3.5.2 Gibbs' Method

Gibbs' algorithm requires three position vectors and estimates the velocity at the middle time incident: in SPOOK it receives the $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ previously estimated with Gauss' method and returns the velocity $\mathbf{v}_2$. Gibbs' technique uses a geometrical approach and it fails when the angles between the three position vectors are smaller than 1° because of numerical instability [1]. The full derivation of the algorithm can be found in [1].

3.5.3 Herrick-Gibbs' Method

Herrick-Gibbs' method uses Taylor series expansion to estimate $\mathbf{v}_2$ from $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ (derived with Gauss' technique) and their respective times t_1, t_2 and t_3. This algorithm performs best when the angular separation between the three position vectors is smaller than 3° [18]. The full derivation of Herrick-Gibbs' technique can be found in [1].

3.5.4 Lambert Solver

Lambert's problem is a technique to derive the orbit of an object by only knowing two position vectors and the time of flight between them. In SPOOK this problem is applied to IOD and two solvers have been implemented:

- *Lambert-Gauss' solver*, whose derivation can be found in [1], receives two position vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ and the flight time as input and returns estimations of $\mathbf{v}_1$ and $\mathbf{v}_2$.
- *Izzo's solver*, implemented by Fiusco [15], computes $\mathbf{v}_2$ from $\mathbf{r}_1$ and $\mathbf{r}_2$ and the flight time between them; then it computes $\mathbf{v}_2$ again from $\mathbf{r}_2$ and $\mathbf{r}_3$ and their flight time; the final estimate of $\mathbf{v}_2$ is the average of the previous two.

3.6 Orbit Determination methods

The aim of orbit determination is to estimate the state vector and the covariance matrix of an object by using a set of measurements.

The covariance matrix has to be computed too during OD because it is useful to evaluate the errors in the estimation of the state vector. In SPOOK the results of orbit determination are assumed to be Gaussian distributed with the exact state vector as mean (this assumption is applied to every component of the state vector), i.e. the OD errors are Gaussian distributed with null mean and standard deviation equal to the sensor accuracy; the diagonal elements of the covariance matrix are the variances σ^2 of the corresponding state vector's components, so by making the square root we get the standard deviation σ of each component and we can use 3σ as approximation of the real errors (since in Gaussian distribution 99.7% of the errors are smaller than 3σ). More information about the covariance matrix and its meaning are available in Appendix A.

Since 3σ is used as approximation of the real error, the ideal scenario would be that for each component of the state vector the corresponding 3σ is bigger than the real error (at least 99.7% of the time) in order to be conservative, but also that 3σ is very close to the real error: if 3σ were too much bigger, it would mean we expect to get huge errors (since in real applications the real errors are not available) from OD, making it useless.

The following subsections present five methods for orbit determination that are implemented in SPOOK. Also other orbit determination algorithms have been implemented but they are not the focus of this thesis.

3.6.1 Weighted Least Squares

The *Weighted Least Square* (WLS) iterative method aims at correcting the initial state vector $\mathbf{x}_0$ provided by ICP or IOD and improving its accuracy. Depending on the user's needs, at the end of all iterations the final $\mathbf{x}_0$ and its covariance matrix (which is computed by the WLS algorithm) can be propagated either forwards or backwards or in both directions with a chosen step size.

The following WLS algorithm comes from [11] and a more in-depth explanation is available in [1].

Assuming there are n measurements of the object, the initial state vector $\mathbf{x}_0$ is propagated to all measurements' times: so we get predicted state vectors.

$$\mathbf{x}_j = f\left(\mathbf{x}_0, t_j\right) \qquad \forall j = 1, ..., n \tag{3.3}$$

Predicted measurements $\mathbf{z}_{p,j}$ of the same type of the real ones ($\mathbf{z}_{r,j}$) are derived from the predicted state vectors; then the residuals $\mathbf{b}_j$ are computed as the difference between real and predicted measurements:

$$\mathbf{z}_{p,j} = g\left(\mathbf{x}_j\right) \tag{3.4}$$

$$\mathbf{b}_j = \mathbf{z}_{r,j} - \mathbf{z}_{p,j} \tag{3.5}$$

The objective of the WLS method is to correct $\mathbf{x}_0$ in order to minimize the sum of the squared residuals, where each residual is weighted with the inverse accuracy of the sensor: if the observer produces m measurements in an instant of time (e.g. $m = 2$ for optical observers because in an instant they produce a right ascension - declination couple), the weight w_i of the i-th type measurement is

$$w_i = \frac{1}{\sigma_i} \qquad \forall i = 1, ..., m \tag{3.6}$$

where σ_i is the sensor accuracy for that kind of measurement. So the WLS algorithm produces the weighting matrix $\mathbf{W}$:

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_m \end{bmatrix} \tag{3.7}$$

The WLS method uses $\mathbf{W}$ to compute the initial covariance matrix with the following equation:

$$\mathbf{P_0} = \left(\sum^{n} \mathbf{A_j^T W A_j} \right)^{-1} \tag{3.8}$$

where $\mathbf{A_j}$ is the partial derivative matrix of the measurements at time t_j w.r.t. $\mathbf{x}_0$. $\mathbf{A_j}$ can be written as the product of two matrixes: the observation matrix $\mathbf{H_j}$, which shows how the changes in the state vector at time t_j affect the measurements at the same time, and the error state transition matrix $\mathbf{\Phi_j}$, that shows the influence of the initial state on the state at time t_j.

$$\mathbf{A_j} = \frac{\partial \mathbf{z}_j}{\partial \mathbf{x}_0} = \frac{\partial \mathbf{z}_j}{\partial \mathbf{x}_j} \frac{\partial \mathbf{x}_j}{\partial \mathbf{x}_0} = \mathbf{H_j \Phi_j} \tag{3.9}$$

SPOOK computes the error state transition matrix $\mathbf{\Phi_j}$ by solving the following system of differential equations:

$$\dot{\mathbf{\Phi}}_\mathbf{j} = \mathbf{F_j \Phi} \tag{3.10}$$

where $\mathbf{F_j}$ is the matrix of the partial derivatives of the state rates (velocity $\mathbf{v}$ and acceleration $\mathbf{a}$) w.r.t. the state vector.

$$\mathbf{F} = \frac{d\dot{\mathbf{x}}}{d\mathbf{x}} = \begin{bmatrix} \frac{d\mathbf{v}}{d\mathbf{r}} & \frac{d\mathbf{v}}{d\mathbf{v}} \\ \frac{d\mathbf{a}}{d\mathbf{r}} & \frac{d\mathbf{a}}{d\mathbf{v}} \end{bmatrix} \tag{3.11}$$

After solving the system of Equation 3.10, SPOOK computes the correction of the initial state vector and corrects the initial state:

$$\delta \mathbf{x}_0 = \left(\sum^{n} (\mathbf{A_j^T W A_j}) \right)^{-1} \sum^{n} (\mathbf{A_j^T W b}_j) \tag{3.12}$$

$$\mathbf{x}_{0_{new}} = \mathbf{x}_0 + \delta \mathbf{x}_0 \tag{3.13}$$

The improved initial state vector is given as input to the algorithm and the whole process is repeated; this iterative process lasts until the max number of iterations is achieved or the truncation error τ falls below a preset tolerance.

$$\tau = \sqrt{\frac{\sum^{n} (\mathbf{b}_j^T \mathbf{W b}_j)}{nm}} \tag{3.14}$$

3.6.2 Sequential Batch Least Squares

While WLS uses all the available measurements together to improve the initial state vector, the *Sequential Batch Least Squares* (SBLS) method divides the measurements into batches and performs OD with one batch at a time: essentially the WLS algorithm is applied in order to correct $\mathbf{x}_0$ but using only one batch of measurements; at the end of the iterations (the convergence criteria are the same as in Subsection 3.6.1), the improved $\mathbf{x}_0$ becomes the input for a new round of iterations but with the next set of measurements.

The advantage of the SBLS method over WLS is that the former allows the processing of new measurements even if they are received while the previous ones are still being processed [14] and this is very useful especially if a big number of observations are available: the WLS method processes all the observations together and if new ones are received, the WLS algorithm has to restart and process the old measurements again along with the new ones, leading to a waste of time; instead the SBLS algorithm simply finishes processing the old measurements and then starts processing the new ones.

In the SPOOK version of SBLS the measurements' batches correspond to the tracklets (see Section 3.4). When the iterations with one tracklet are over, the algorithm starts the iterations with the next tracklet; however, the information of the previous tracklets should not be lost, so the new round of iterations does not receive only the corrected $\mathbf{x}_0$ as input, but also the matrices $\left(\sum^n \left(\mathbf{A_j^T W A_j}\right)\right)_{old}$ and $\left(\sum^n \left(\mathbf{A_j^T W b}_j\right)\right)_{old}$; so the equations for the computation of $\mathbf{P_0}$ and $\delta\mathbf{x}_0$, i.e. Equations 3.8 and 3.12 respectively, have to be modified [14]:

$$\mathbf{P_0} = \left(\sum^{n_k} \mathbf{A_j^T W A_j} + \left(\sum^{n_{k-1}} \mathbf{A_j^T W A_j}\right)_{old}\right)^{-1} \tag{3.15}$$

$$\delta\mathbf{x}_0 = \left(\sum^{n_k} \mathbf{A_j^T W A_j} + \left(\sum^{n_{k-1}} \mathbf{A_j^T W A_j}\right)_{old}\right)^{-1} \left(\sum^{n_k} \mathbf{A_j^T W b}_j + \left(\sum^{n_{k-1}} \mathbf{A_j^T W b}_j\right)_{old}\right) \tag{3.16}$$

where k is the index of the tracklet that is being processed.

A problem with the SBLS is that when the algorithm starts processing a new batch, it may diverge after a few iterations if the noise in the new measurements is bigger than the error that has been minimized during the iterations with the previous batch. Also the SBLS algorithm implemented in SPOOK turned out to be too inaccurate even before the beginning of the works for this thesis, so no meaningful tests and work have been carried out on SBLS.

3.6.3 Extended Kalman Filter

One problem with the Least Square methods is that they work by processing measurements spread over a certain time interval and if this frame is too long and the forces acting on the object are not perfectly modeled, the orbit determination deteriorates [1]. Also the Least Square methods just estimate the state vector and covariance matrix associated to a specific epoch (t_0 in our case), so the estimated state and covariance need to be propagated in order to estimate the whole orbit of the object in the desired time period. In SPOOK these problems are avoided by using the *Extended Kalman Filter* (EKF), a technique that is able to track the object in real time, i.e. it estimates the state vector and covariance matrix at every measurement time; this also leads to the same advantage SBLS has over WLS: the EKF does not need to reprocess the old measurements when new measurements are provided [18].

For each observation time the EKF algorithm estimates the state vector and covariance matrix by following two steps:

- *prediction step*: the estimates of the state vector and covariance at the previous measurement time are propagated to the time of the current measurement (the predicted state vector and covariance matrix at time t_j are indicated as $\overline{\mathbf{x}}_j$ and $\overline{\mathbf{P}}_j$ respectively);
- *update step*: the predicted state vector and covariance matrix are corrected in order to get better estimates (the estimated state and covariance are shown as $\widehat{\mathbf{x}}_j$ and $\widehat{\mathbf{P}}_j$ respectively).

As shown in Figure 3.7, the ideal outcome of the EKF would be to get closer to the real orbit at every measurement time.

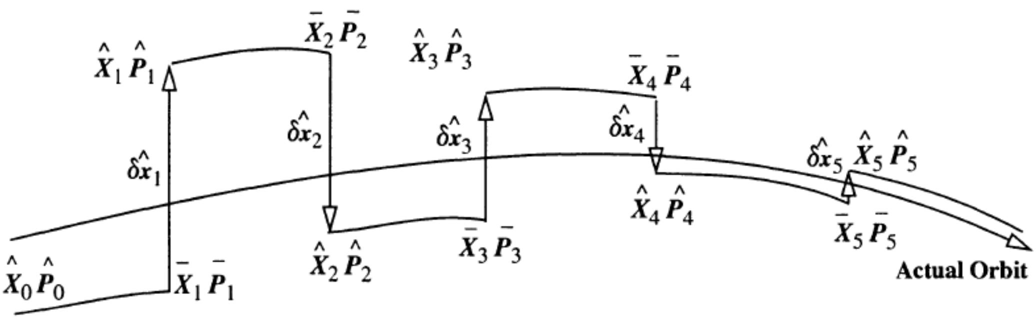

Figure 3.7: Extended Kalman Filter converging over time [1].

A short description of the main steps of the EKF algorithm is provided; the in-depth derivation can be found in [1]; the notation in use is the same as Subsection 3.6.1.

Assuming the state vector and covariance matrix at time t_j have been estimated, these are the steps to estimate state and covariance at time t_{j+1}:

Prediction step

The estimated state $\widehat{\mathbf{x}}_j$ is propagated at time t_{j+1} in order to get the predicted state $\overline{\mathbf{x}}_{j+1}$. To do the same with covariance matrix the state transition matrix Φ has to be computed by integrating Equation 3.10, which requires to derive the matrix

$$\mathbf{F_j} = \frac{\partial \dot{\widehat{\mathbf{x}}}_j}{\partial \widehat{\mathbf{x}}_j} \tag{3.17}$$

So the predicted covariance matrix is

$$\overline{\mathbf{P}}_{j+1} = \mathbf{\Phi} \widehat{\mathbf{P}}_j \mathbf{\Phi}^T + \mathbf{Q} \tag{3.18}$$

where $\mathbf{Q}$ is called "second moment of the process noise"; to compute $\mathbf{Q}$ we need the process noise $\boldsymbol{\nu}$, which is the uncertainty in the propagation process due to uncertainties in the dynamics model [18]:

$$\mathbf{Q} = \mathrm{E}(\boldsymbol{\nu}\boldsymbol{\nu}^T) \tag{3.19}$$

$$\boldsymbol{\nu} = \int_{t_0}^{t} \mathbf{\Phi}\left(t, t_0\right) \mathbf{u}\, dt \tag{3.20}$$

where $\mathbf{u}$ is the statistically measured uncertainty in the force model. So $\mathbf{Q}$ represents the propagation errors due to modelling errors in dynamics.

The prediction step and therefore the whole EKF algorithm is computationally expensive because the integrator has to be run at every observation time.

Update step

The observation matrix $\mathbf{H_{j+1}}$ is computed:

$$\mathbf{H_{j+1}} = \frac{\partial \mathbf{z}_{j+1}}{\partial \overline{\mathbf{x}}_j} \tag{3.21}$$

Then the Kalman Gain matrix $\mathbf{K_{j+1}}$ is calculated:

$$\mathbf{K_{j+1}} = \overline{\mathbf{P}}_{j+1} \mathbf{H^T_{j+1}} \left(\mathbf{H_{j+1}} \overline{\mathbf{P}}_{j+1} \mathbf{H^T_{j+1}} + \mathbf{R} \right)^{-1} \tag{3.22}$$

where $\mathbf{R}$ is the measurement noise matrix, that contains the uncertainties in the measurements:

$$\mathbf{R} = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_m^2 \end{bmatrix} \tag{3.23}$$

Eventually the Kalman Gain is used to correct the predicted state vector:

$$\delta\widehat{\mathbf{x}}_{j+1} = \mathbf{K_{j+1}}\mathbf{b}_{j+1} \tag{3.24}$$

$$\widehat{\mathbf{x}}_{j+1} = \overline{\mathbf{x}}_{j+1} + \delta\widehat{\mathbf{x}}_{j+1} \tag{3.25}$$

where $\mathbf{b}_{j+1}$ is the residual at time t_{j+1}:

$$\mathbf{b}_{j+1} = \mathbf{z}_{j+1} - \mathbf{H_{j+1}}\overline{\mathbf{x}}_{j+1} \tag{3.26}$$

The Kalman Gain is also used to correct the predicted covariance matrix:

$$\widehat{\mathbf{P}}_{j+1} = \overline{\mathbf{P}}_{j+1} - \mathbf{K}_{j+1}\mathbf{H}_{j+1}\overline{\mathbf{P}}_{j+1} \tag{3.27}$$

As it can be seen in Equations 3.24 and 3.27, if the Kalman Gain is small, the corrections of the predicted state and covariance are also small, reducing the contribution of the $j+1$-th measurement; the following explanation on the role of the Kalman Gain comes from [19].

Another way to compute the Kalman Gain is:

$$K = \frac{\epsilon_{est}}{\epsilon_{est} + \epsilon_{meas}} \tag{3.28}$$

where ϵ_{est} is the error estimate and ϵ_{meas} is the uncertainty in the measurements. The value of K ranges between 0 and 1: if the value is close to 1, it means the error in the estimate is far bigger than the measurement uncertainty, so we trust more in the new measurement and apply a big correction to the estimate; if the Kalman Gain is close to 0, it means the uncertainty in the measurement is big compared to the error of the estimate, so we trust the previous estimate more ignoring the new measurement and applying only a small correction (or a null one if the value of K is equal to 0); this second behaviour of the filter is called *saturation* or *smugness* and it becomes a problem when it happens too soon ignoring too many measurements.

One drawback of the EKF is that it does not need only $\mathbf{x}_0$ as input, but also the matrixes $\mathbf{P_0}$, $\mathbf{R}$ and $\mathbf{Q}$, where $\mathbf{Q}$ can be difficult to compute (see [19] for the methods implemented in SPOOK); also if the measurements are very noisy, $\mathbf{R}$ is too big and leads the filter to smugness.

3.6.4 Unscented Kalman Filter

The *Unscented Kalman Filter* (UKF) is a method that works especially well in models where the process noise is not enough to model the uncertainties in the system dynamics [18].

In order to understand why the UKF can be better than the EKF, the nonlinearity of the Orbit Determination problem has to be mentioned: according to [2], the propagation of the state vector $\mathbf{x}$ is influenced by the process input and the model noise, while measurements $\mathbf{z}$ are influenced by measurement noise; so the following nonlinear equations are used to propagate $\mathbf{x}$ and $\mathbf{z}$ to discrete times:

$$\begin{aligned} \mathbf{x}_{j+1} &= f\left(\mathbf{x}_j, \mathbf{u}_j, \mathbf{w}_j\right) \\ \mathbf{z}_j &= h\left(\mathbf{x}_j, \mathbf{v}_j\right) \end{aligned} \tag{3.29}$$

where $\mathbf{u}$ is the process input, $\mathbf{w}$ is the model noise and $\mathbf{v}$ is the measurement noise [2]. During the prediction step the Extended Kalman Filter linearizes Equation 3.29 around the mean of a Gaussian distribution and propagates the state and covariance through this linearized model; instead the Unscented Kalman Filter uses a technique called *unscented transform*, i.e it computes a set of points, called *sigma points*, that completely capture the mean and covariance of the Gaussian distribution, and propagates them through the nonlinear model: then the predicted mean and covariance are calculated from the propagated points. Typically with UKF the accuracy of the estimate increases while the computation time decreases w.r.t. EKF [2].

Here follow the steps of the UKF to estimate the state vector and covariance matrix at time t_j; the equations and comments come from [2].

Prediction step

The state vector $\mathbf{x}$ is augmented in order to include the components of the process noise and measurement noise; the augmented state vector is indicated as $\mathbf{x}^\alpha$ and its size is L, which is equal to the sum of the sizes of the original state vector, model noise vector and measurements noise vector.

$$\mathbf{x}^\alpha_{j-1} = \begin{bmatrix} \mathbf{x}_{j-1} \\ \mathbf{0}_w \\ \mathbf{0}_v \end{bmatrix} \tag{3.30}$$

The advantage of augmenting the state vector is that the sigma points are selected from $\mathbf{x}^\alpha$, so the nonlinear effects due to the two types of noise are captured and represented with the same accuracy as the "original" state. The covariance matrix is augmented to the L^2 size:

$$\begin{aligned} \mathbf{P}^\alpha_{j-1} &= \mathrm{E}\left[\left(\mathbf{x}^\alpha_{j-1} - \widehat{\mathbf{x}}^\alpha_{j-1}\right)\left(\mathbf{x}^\alpha_{j-1} - \widehat{\mathbf{x}}^\alpha_{j-1}\right)^T\right] \\ &= \begin{bmatrix} \mathbf{P}_{j-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{j-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{j-1} \end{bmatrix} \end{aligned} \tag{3.31}$$

Now $2L+1$ sigma points are formed; they are computed in the form of the columns of a matrix $\boldsymbol{\chi}^{\alpha}$:

$$\boldsymbol{\chi}^{\alpha}_{i,j-1} = \mathbf{x}^{\alpha}_{j-1} \qquad i = 0 \tag{3.32}$$

$$\boldsymbol{\chi}^{\alpha}_{i,j-1} = \mathbf{x}^{\alpha}_{j-1} + \left(\alpha\sqrt{L\mathbf{P}^{\alpha}_{j-1}}\right)_i \qquad i = 1, ..., L \tag{3.33}$$

$$\boldsymbol{\chi}^{\alpha}_{i,j-1} = \mathbf{x}^{\alpha}_{j-1} - \left(\alpha\sqrt{L\mathbf{P}^{\alpha}_{j-1}}\right)_{i-L} \qquad i = L+1, ..., 2L \tag{3.34}$$

where i is the index of the columns of $\boldsymbol{\chi}^{\alpha}$. The tuning parameter α belongs to the interval (0,1] and defines the spread of the sigma points; α is usually set to a low value (around 0.001) in order to guarantee positive semidefinite covariance matrices and avoid the sampling of non-local features. The calculation of the square root of $\mathbf{P}^{\alpha}_{j-1}$ is the most coputationally expensive part of the UKF.

The $\boldsymbol{\chi}^{\alpha}_{j-1}$ matrix can be decomposed in $\boldsymbol{\chi}^{x}_{j-1}$ rows, which contain the state, $\boldsymbol{\chi}^{w}_{j-1}$ rows, which contain sampled process noise, and $\boldsymbol{\chi}^{v}_{j-1}$ rows, which contain sampled measurement noise. Each sigma point is also assigned a weight for mean and a weight for covariance, indicated with the apexes (m) and (c) respectively.

$$w_i^{(m)} = 1 - \frac{1}{\alpha^2} \qquad i = 0 \tag{3.35}$$

$$w_i^{(c)} = 4 - \frac{1}{\alpha^2} - \alpha^2 \qquad i = 0 \tag{3.36}$$

$$w_i^{(m)} = w_i^{(c)} = \frac{1}{2\alpha^2 L} \qquad i = 1, ..., 2L \tag{3.37}$$

Eventually the sigma points are propagated through Equations 3.29 and the predicted state, covariance matrix and estimated measurements are computed as weighted averages:

$$\boldsymbol{\chi}^{x}_{j} = f\left(\boldsymbol{\chi}^{x}_{j-1}, \mathbf{u}_j, \boldsymbol{\chi}^{w}_{j-1}\right) \tag{3.38}$$

$$\overline{\mathbf{x}}_j = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\chi}^{x}_{i,j} \tag{3.39}$$

$$\overline{\mathbf{P}}_j = \sum_{i=0}^{2L} w_i^{(c)} \left[\boldsymbol{\chi}^{x}_{j} - \overline{\mathbf{x}}_j\right] \left[\boldsymbol{\chi}^{x}_{j} - \overline{\mathbf{x}}_j\right]^T \tag{3.40}$$

$$\mathbf{Z}_j = h\left(\boldsymbol{\chi}^{x}_{j}, \boldsymbol{\chi}^{v}_{j-1}\right) \tag{3.41}$$

$$\widehat{\mathbf{z}}_j = \sum_{i=0}^{2L} w_i^{(m)} \mathbf{Z}^{x}_{i,j} \tag{3.42}$$

Update step

The state-measurement cross-correlation matrix $\mathbf{P}_{xz}$ and the measurement covariance matrix $\mathbf{P}_{zz}$ are calculated and used to compute the Kalman Gain:

$$\mathbf{P}_{xz} = \sum_{i=0}^{2L} w_i^{(c)} \left[\boldsymbol{\chi}_{i,j}^{x} - \overline{\mathbf{x}}_j\right] \left[\mathbf{Z}_{i,j} - \widehat{\mathbf{z}}_j\right]^T \tag{3.43}$$

$$\mathbf{P}_{zz} = \sum_{i=0}^{2L} w_i^{(c)} \left[\mathbf{Z}_{i,j} - \widehat{\mathbf{z}}_j\right] \left[\mathbf{Z}_{i,j} - \widehat{\mathbf{z}}_j\right]^T \tag{3.44}$$

$$\mathbf{K}_j = \mathbf{P}_{xz}\mathbf{P}_{zz}^{-1} \tag{3.45}$$

Eventually the state vector and covariance matrix are updated:

$$\widehat{\mathbf{x}}_j = \overline{\mathbf{x}}_j + \mathbf{K}_j \left(\mathbf{z}_j - \widehat{\mathbf{z}}_j\right) \tag{3.46}$$

$$\widehat{\mathbf{P}}_j = \overline{\mathbf{P}}_j - \mathbf{K}_j\mathbf{P}_{zz}\mathbf{K}_j^T \tag{3.47}$$

3.6.5 Unscented-Schmidt Kalman Filter

The *Unscented-Schmidt Kalman Filter* (USKF) is very similar to the UKF: it augments the state vector and the covariance matrix to sizes L_{USKF} and L_{USKF}^2 respectively; then the sigma points are formed and propagated to the next timestep and the predicted state and covariance are computed; eventually the state vector and covariance matrix are updated. The difference with the UKF is that in the USKF the state vector is additionally augmented by a number n_c of *consider parameters* c, so $L_{USKF} = L + n_c$. Thus $2n_c$ sigma points more than in the UKF are formed; these additional sigma points are propagated in the same way as the others during the prediction step; however, in the update step the consider parameters are not updated [18]. Although it is similar to the UKF, the USKF probably gives different results because the additional sigma points affect the predicted state and covariance. See [10] for the full algorithm.

USKF gives different results from UKF only if the state vector is augmented to include the consider parameters: in particular the one implemented in SPOOK can consider C_D and C_{SP} as consider parameters and estimate them along with the "regular" components of the state vector; however this thesis focuses on the estimation of the not augmented state vector written in Equation 3.1, so no special test has been performed on the USKF method; the few tests performed on the implemented USKF algorithm just proved that it and the UKF algorithm return exactly the same results if the state vector does not include C_D and C_{SP}.

3.7 Orbit Determination success criteria

Five methods to check the success of orbit determination were implemented in SPOOK by F. Fiusco [15].

After orbit determination SPOOK converts the estimates of the state vector into observable quantities, i.e. measurements of the same type of the real or simulated ones that have been used as input for OD (see Subsection 3.4.1); output measurements and input measurements are compared and the residuals are computed; most of the implemented OD success tests act on the residuals.

Here follows a brief description of the implemented OD success tests:

- χ^2 *test*: this test checks whether the assumption of Gaussian distribution of the OD errors is true or not. Mean and standard deviation are computed for each residual type and a test statistic χ^2 is computed; then the computed statistic is compared with the theoretical χ^2 that would give a confidence level of 95%: if the theoretical χ^2 is smaller than the computed one, the assumption of Gaussian distribution is accepted;

- *Weighted Root Mean Square* (WRMS): it consists in computing

$$WRMS = \sqrt{\frac{\sum_{i=1}^{N} \frac{res(i)^2}{\sigma_i^2}}{N}} \tag{3.48}$$

  which is the RMS of all the N residuals averaged with the proper sensor accuracy σ_i. $WRMS$ should be the smallest possible in order to have a successful OD, but it should never be lower than 1 (otherwise it would mean that the final result is inside the sensor noise [15]);

- *Self consistency test*: this test can be used only for filters and it verifies if the filter's estimate is coherent with the orbit of the target object. Given N measurements, after OD the estimated state vector and covariance matrix at time $N-1$ are propagated to time N and turned into observable quantities; so it is computed how far the real N^{th} measurement is from the propagated one;

- *McReynold's filter-smoother consistency test*: this test can be used only if OD is performed with a filter and a smoother (a technique that improves the filtered state); it computes the difference between filtered and smoothed state vectors and between filtered and smoothed covariance matrices for each time step t_j in order to check if the smoothed state is consistent with the filtered state;

- **3**σ *envelope test*: it checks if at least 99% of the residuals are smaller than 3σ, being σ the sensor's accuracy for the considered observable, in order to verify the assumption of Gaussian distribution.

The described methods have not been used in this thesis because Fiusco himself concluded them to be unreliable and in need of being tested more; see [15] to read the tests he performed and his conclusions about them.

3.8 Orbital regions

Earth-orbiting objects' trajectories fall into orbital regions, whose classification is based on the altitude, i.e. the distance from the Earth surface. An orbital region classification is also implemented in SPOOK in order to perform statistical analysis, simulations or other functions if needed; this classification is a modified version of the one from ESA Space Situational Awareness programme [5] and is shown in Table 3.1.

Name	Perigee [km]		Apogee [km]	
	Min	Max	Min	Max
LEO resident	0	2000	0	2000
LEO transient	0	2000	2000	∞
Low MEO resident	2000	16000	2000	16000
Low MEO transient	2000	16000	16000	∞
High MEO resident (GNSS)	16000	33786	16000	33786
High MEO transient	16000	33786	33786	∞
GEO resident ($i \leq 20°$)	33786	37786	33786	37786
GEO resident ($i > 20°$)	33786	37786	33786	37786
GEO transient	33786	37786	37786	∞
HEO	37786	∞	37786	∞

Table 3.1: Orbital region classification [14].

For the tests of Chapters 4 and 5 GPS satellites have been used as target objects to perform OD. GPS satellites have been chosen because they belong to the class of MEO objects and so they are not as affected by atmospheric drag as LEO objects, leading to less complex results. GPS satellites have been chosen also because it is easy to find their ephemerides and TLE files; in particular Airbus has open access to very accurate ephemerides of GPS satellites, so in the tests of Chapter 5 they have been used as "ground truth" for error evaluation.

Chapter 4

Work carried out

This chapter presents the modifications made and the new functions implemented in SPOOK during the works for this thesis; these extensions were needed to reach the end goal of the thesis, i.e. the definition of the settings to perform OD in the best way. Two improvements have been made on the Sensor Simulation mode in order to obtain better measurements: the augmentation of the accessibility report writing function and the implementation of a function to write optical output files. Also three OD analysis tools have been implemented: a function to evaluate the errors of OD and Covariance Propagation modes w.r.t. ephemerides, a function to perform OD with randomized initial states and a function to run OD faster from the Python interface.

4.1 Expansion of the Accessibility Report

As mentioned in Section 2.2, Sensor Simulation mode makes SPOOK simulate at least one observer and produce simulated measurements of one or more objects. When it works in this mode, SPOOK can produce output files called *accessibility reports* if the user enables the corresponding option in the parameters configuration file.

One accessibility report file is produced for each simulated observer and it consists of three sections:

- the first section is related to the observer and is a list of the constraints applied to it (that are set by the user in the observers configuration file), like the maximum elevation considered for the Sun, the minimum and the maximum elevation considered for target objects, the weather and illumination models and so on;
- the second section provides a list of all the objects (among the ones written

in the objects configuration file) that have crossed the field of view of the observer during a certain time interval, called *crossing time*, and so have been observed, producing measurements;

- the third section is just a list of the objects (from the objects file) that have not been observed at all during the observation time.

An example of accessibility report is shown in Appendix B.

We are interested in the second section of the document, that does not only list the crossing objects, but also provides a set of data for each one. Before the start of this thesis the data associated to each crossing objects were:

- the object's name and its international identification codes, that SPOOK extracts from the objects configuration file;
- the start time, the end time and the duration of the object's crossing time interval;
- the azimuth and the elevation of the object at the start and at the end of the crossing time frame;
- the apogee and the perigee of the object's orbit and the orbital region it belongs to (based on the classification of Section 3.8).

In this thesis the function to write the accessibility report has been modified in order to write additional data for each crossing object. The added values are:

- the altitude range, i.e. the minimum and maximum altitude reached by the object during the crossing interval;
- the minimum angular motion n_{min} and the maximum angular motion n_{MAX}, computed with the equation $n = \sqrt{\frac{\mu}{r^3}}$, where r is the distance between the object and the Earth center;
- the minimum and maximum *shadowing parameter* during the crossing time, i.e. an index that is equal to 1 if the object is fully illuminated by sunlight, is equal to 0 if the object is totally inside the Earth's shadow, and has a value between 0 and 1 if the object is in penumbra [19];
- the minimum and maximum *Signal to Noise Ratio* (SNR), which is the ratio of the power of the signal coming from the object over the noise;
- the minimum and maximum *phase angle*, which is the angle Sun-object-observer, nearly equal to the angle Sun-object-Earth if the observer is ground-based [27].

We can see that all the data (both the old ones and the new ones) in the second section of the accessibility report, except for the objects' names and identifiers, are simulated measurements or are derived from them; the shadowing parameter of the object is computed with a function implemented by Cirillo [19].

The data added to the accessibility report are useful because they enable the user to optimize the generation of observation plans: if the user wants to observe a target object with a specific observer during a certain time frame, first he can perform a test by simulating the observation and reading the corresponding accessibility report; basing on whether the object has crossed the observer's field of view (in the simulation) and on the object related values added to the accessibility report, the user can decide if it is worth planning and performing that observation for real (e.g. it is not convenient if the simulated SNR is too low or if the object is in Earth's shadow). Additionally the values of minimum and maximum altitude can be used to compute the exposure time of the camera.

4.2 The Optical Output file

The work that has been carried out in the thesis includes the implementation of a new function in the frame of Sensor Simulation mode: if the corresponding option in the parameters configuration file is enabled, the new function produces so-called *optical outputs files*; one optical output file is generated for each couple observer-observed object. Appendix B provides an example of this file.

Given a step size that is written in the parameters configuration file, the optical output file provides some object related parameters (that are derived from the simulated measurements) for each time step from the beginning to the end of the crossing time. Here follows the list of the values:

- altitude of the object;
- phase angle;
- elevation;
- range, which is the distance between the object and the observer;
- angular velocity of the object, computed as the 2-norm of the first derivative of the angular measurements vector;
- apparent magnitude of the object;
- SNR.

The SNR is function of the other values contained in the file, so the optical output makes it possible to plot and visualize how the parameters evolve in time and how their evolution affects the value of SNR; thus we can evaluate the performances of the measurements estimation model: for example we can detect errors in the model if we see that certain values of the parameters lead to unexpected values of SNR. We can also use the optical output to optimize observations plans along with the accessibility report.

4.3 Errors evaluation with ephemerides

In the frame of this thesis a function for errors evaluation has been implemented both for the Covariance Propagation mode and for the Orbit Determination mode. If the corresponding option in the parameters configuration file is enabled, at the end of OD or propagation the new function compares the propagated or estimated state vector to an ephemeris from an input OEM file and computes the errors; of course this function should be used only if the ephemeris is accurate enough to be assumed to be the ground truth, and this is the case for the GPS satellites' ephemerides used in Chapter 5.

OD and covariance propagation modes return the state vector and covariance matrix of the object at times set by the user (as he or she sets the start and end times and the size of the time step), which are usually different from the epochs of the ephemeris: so the first step performed by the function is the interpolation of the ephemeris at the times of SPOOK's output; a Lagrange interpolator of ninth order is used, the same as in Sensor Calibration mode. Once the interpolation has ended, the function derives the following values for each time of the OD or covariance propagation output:

- the estimation or propagation error of every component of the state vector, computed as $x_{i,ephemeris} - x_{i,estimate}$;
- the standard deviations and variances (i.e. the diagonal components of the covariance matrix) of the state vector's components, that are simply extracted from the outputs of OD or propagation;
- the total errors in position and velocity, computed as the 2-norm of the position error vector and the 2-norm of the velocity error vector respectively;
- the total standard deviations in position and velocity, i.e. the 2-norms of the corresponding standard deviation vectors.

All these values are computed in the RTN reference frame. Eventually the function writes the computed values in an output file called *ephemeris errors file*, that can be used to plot the behaviour of the errors and standard deviations in time; in particular these files have been used to plot and compare the errors and the 3σs (that in SPOOK are assumed to be estimates of the real errors) in Chapter 5.

Appendix B contains an example of ephemeris errors file.

4.4 Randomization of the initial state vector

As discussed in Section 3.5, the orbit determination process always requires an initial state vector. During this thesis' work a function has been implemented that "randomizes" a given initial state vector $\mathbf{x}_0$ by creating random initial state vectors $\mathbf{x}_{0,rand}$ and performs one OD run for the original state vector and one OD run for each new state; so if N is the number of new initial states, that is set by the user in the parameters configuration file, SPOOK performs $N + 1$ orbit determinations. Performing multiple runs of orbit determination where only the initial state changes can be useful to analyze the sensitivity of the chosen OD algorithm to the initial state itself.

The randomizing function creates the new initial state vectors assuming the original $\mathbf{x}_0$ to be the mean of a Gaussian distribution, i.e. assuming that each of its components $x_{0,i}$ is the mean of a Gaussian distribution with variance σ_i^2 which is the corresponding diagonal element of the initial covariance matrix $\mathbf{P}_0$. So the function uses $\mathbf{P}_0$ to create N state vectors $\mathbf{x}_{0,rand}$ that should follow a Gaussian distribution centred around $\mathbf{x}_0$ for each of their components.

It follows that if the assumption of Gaussian distribution of the new initial state vectors is true, for each OD output time the average of the estimated state vectors for the runs with the random initial states should be equal to the theoretical mean, i.e. the estimated state vector for the run with the original $\mathbf{x}_0$. Whether the assumption is true or false is verified in Section 5.1.

4.5 User interface for orbit determination

One of the contributions given to SPOOK by this thesis is the implementation of a Python function to perform orbit determination without having to write or modify the configuration files manually.

Before this thesis the user had to write or modify the configuration files and execute SPOOK every time he wanted to use the OD mode; this process was very time consuming, especially when the user wanted to run orbit determination several times with different settings. Instead the new Python function writes the configuration files almost completely by itself: of all the entries in the configuration files, only a few are read and used when running SPOOK in Orbit Determination mode;

so the Python function writes the files by assigning default values to the "useless" entries and requiring the user to input only the values of the "useful" entries. The advantage of the Python function is that if the user wants to perform multiple runs of orbit determination with different settings, now he or she can just call the Python function from command line or in a script many times as needed and change only a few input arguments at every call instead of manually writing a great number of files.

The implemented Python function does not set only the "useless" entries of the configuration files to default values, but also some of the entries that affect the orbit determination process; in particular:

- corrections of light time delay and diurnal aberration (more information in [11]) are enabled by default when real measurements are used;
- ART is set as the default observer (because it is the one used by Airbus to perform tests) and so all the entries of the observers configuration file are set to its parameters;
- Initial Covariance Propagation is set as the default method to compute the initial state vector and covariance matrix because the methods for Initial Orbit Determination have not been fully tested yet.

Chapter 5

Tests and comments

This chapter presents and discusses the main tests that have been performed on orbit determination: test on the randomization function, comparison between Orbit Determination and Covariance Propagation modes' results, test on the sensitivity of the WLS method to $\mathbf{x_0}$, comparison of the performances of EKF and UKF. In all the tests the RTN reference frame has been used, so the state vector of the object is

$$\mathbf{x} = \begin{bmatrix} r_R \\ r_T \\ r_N \\ v_R \\ v_T \\ v_N \end{bmatrix} \tag{5.1}$$

5.1 Tests on the randomization function

As explained in Section 4.4, the assumption of Gaussian distribution of the "random" initial state vectors is true if at every OD output time the mean computed from the estimated states of the runs with the new $\mathbf{x}_0$s is equal to the estimated state of the run with the original $\mathbf{x}_0$. The same logic can be applied to the errors: the error of the run with the original $\mathbf{x}_0$ should be the mean of the errors of the runs with the randomized initial states if the distribution is really Gaussian. In this section two tests are described that verify if the true mean error is equal to the error of the run with the original $\mathbf{x}_0$.

In the first test $\mathbf{x}_0$ was one of the entries of an OEM file and the initial covariance matrix $\mathbf{P}_0$ has been manually written in the objects configuration file. The target object was the GPS satellite with NORAD ID 40105 (this is the identification number assigned by US Space Command [28]).

$$\mathbf{P}_0 = \begin{bmatrix} 25\ \text{km}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100\ \text{km}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 25\ \text{km}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10^{-6}\ \frac{\text{km}^2}{\text{s}^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 4*10^{-6}\ \frac{\text{km}^2}{\text{s}^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^{-6}\ \frac{\text{km}^2}{\text{s}^2} \end{bmatrix} \tag{5.2}$$

The randomization function has been used to generate 100 initial state vectors and for each one of them and for the original $\mathbf{x}_0$ an Orbit Determination with EKF has been performed, for a total of 101 runs; then the function presented in Section 4.3 computed the errors of each run with respect to the ephemeris from an OEM file (the same from which the original $\mathbf{x}_0$ has been extracted); eventually the average errors of the 100 "randomized" runs have been computed for each component of the state vector at every output epoch.

The results of this test are shown in Figure 5.1, where for each state's component the true average error (i.e. the average of the 100 errors) is plotted together with the error of the run whit the original state vector, that is the theoretical mean; the two vertical lines in the figure mark the beginning and the end of a tracklet; in the figure the 3σ of the run with the original initial state and the 3σ computed from the errors are different because the former has been estimated by the EKF while the latter has been calculated with Equation A.1.

In Figure 5.1 it can be seen that the lines of the true and the theoretical mean are almost completely overlapped in the time interval inside the tracklet, where OD has been performed, and completely overlapped after the end of the tracklet, where the state vectors have been simply propagated forward. So it can be concluded that the distribution of the 100 random initial state vectors is almost Gaussian (and would be Gaussian if more than 100 initial states were generated) if the original $\mathbf{x}_0$ comes from an ephemeris; the conclusion is reinforced by the fact that for every run the total errors read in the ephemeris errors file are smaller than the respective 3σ, as expected from a Gaussian distribution.

The second test had the same settings as the first one except for the original initial state vector that was extracted from a TLE file. Figure 5.2 shows the results.

It is clearly visible that the true average and the theoretical average are different for most of the time for all the components, so the distribution is not Gaussian; this is also proved by the fact that 88 runs out of 100 have bigger total errors than the respective 3σ.

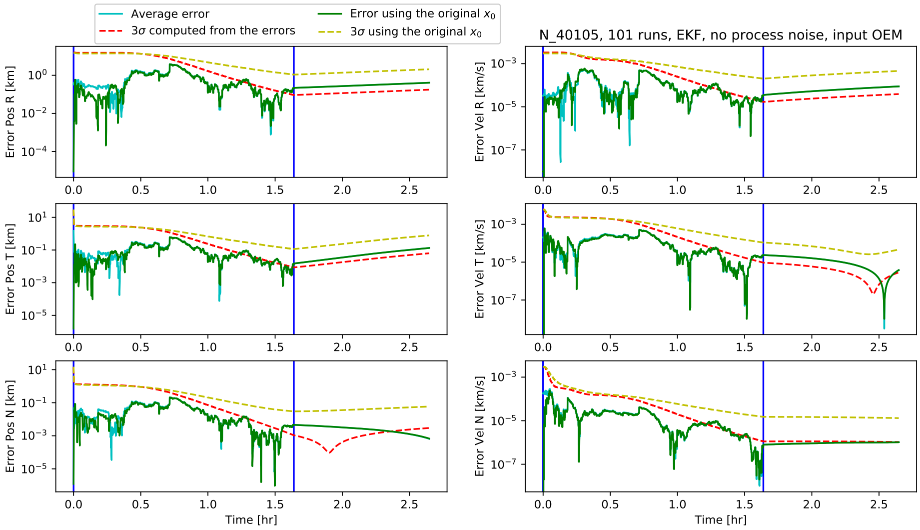

Figure 5.1: Average components' errors of 101 runs; $\mathbf{x}_0$ from OEM file.

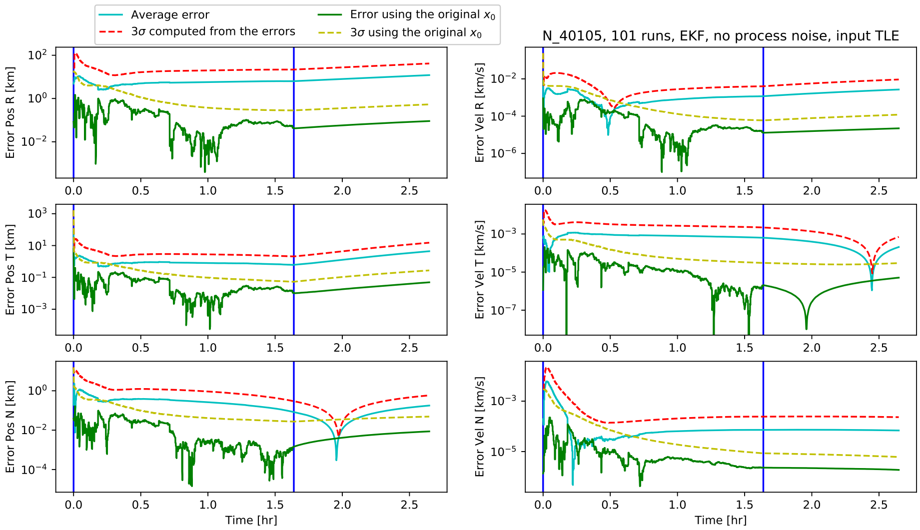

Figure 5.2: Average components' errors of 101 runs; $\mathbf{x}_0$ from TLE file.

While it can be concluded that 100 randomized initial states do not follow a Gaussian distribution if the original $\mathbf{x}_0$ comes from a TLE file, it is also true that if the randomizing function produced more than 100 states, there would be a number of initial states for which their distribution would be Gaussian. However, tests for more than 100 randomized states have not been performed because they would have been too computationally expensive: the reason is that the state vector extracted from the TLE file has to be propagated to the time of the first measurement; in the case of this test the TLE and the first measurement were more than one day apart (and the TLE file was the closest of the available ones), leading to a very long propagation; it took almost two hours to perform just 101 runs.

5.2 Comparison between OD and Propagation

This section discusses the tests performed to compare the accuracies of OD and simple propagation of the space vector and covariance matrix. In general orbit determination methods should be more accurate than propagation because they either improve the initial state and propagate it (Least Squares methods) or correct the state after every propagation step (Kalman Filters); however, the aim of the following tests is to analyze what happens if the initial state vector is already very accurate: in these tests the initial state vector is one of the entries of the ephemeris in the OEM file used as ground truth for the error calculation, so the OD error and the propagation error at time t_0 are null. The target object is the GPS satellite with NORAD ID 25933.

The first test was a comparison between the Extended Kalman Filter and simple propagation. At first four runs of orbit determination have been performed with different initial covariance matrices, whose non-diagonal components were null while the diagonal components (i.e. the variances) are written in Table 5.1. The plots of the errors and 3σ of each run are shown in Figure 5.3.

Run	$\sigma^2_{r_R}$ [km^2]	$\sigma^2_{r_T}$ [km^2]	$\sigma^2_{r_N}$ [km^2]	$\sigma^2_{v_R}$ [km^{2}s^{-2}]	$\sigma^2_{v_T}$ [km^{2}s^{-2}]	$\sigma^2_{v_N}$ [km^{2}s^{-2}]
1	10^{-1}	1	10^{-1}	10^{-6}	10^{-6}	10^{-6}
2	10^{-3}	10^{-2}	10^{-3}	10^{-8}	10^{-8}	10^{-8}
3	10^{-5}	10^{-4}	10^{-5}	10^{-10}	10^{-10}	10^{-10}
4	10^{-7}	10^{-6}	10^{-7}	10^{-12}	10^{-12}	10^{-12}

Table 5.1: Variances of the initial covariance matrix for each run.

Then for each component of the state vector the "best" variance has been chosen from Figure 5.3: what is meant by "best" variance is that the corresponding 3σ is bigger than the real error (because we want 3σ to be a conservative overestimation of the error) for most of the time but also that 3σ is very close to the real error (in order to not overestimate it too much making the OD results seem too inaccurate); when two or more variances satisfied the requirements, the one that returned the smallest error was chosen. So a new initial covariance matrix has been formed with

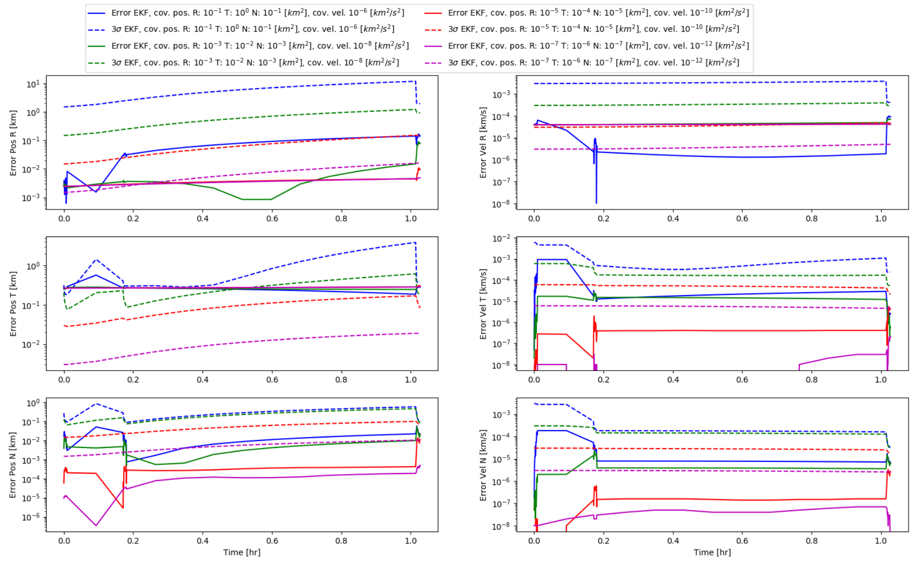

Figure 5.3: EKF errors for different covariances.

the selected best variances in its diagonal:

$$\mathbf{P_0} = \begin{bmatrix} 10^{-7}\ \mathrm{km}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1\ \mathrm{km}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10^{-7}\ \mathrm{km}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10^{-6}\ \frac{\mathrm{km}^2}{\mathrm{s}^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 10^{-12}\ \frac{\mathrm{km}^2}{\mathrm{s}^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^{-12}\ \frac{\mathrm{km}^2}{\mathrm{s}^2} \end{bmatrix} \tag{5.3}$$

Eventually OD and propagation have been run with the $\mathbf{P}_0$ from Equation 5.3 as initial covariance matrix and the errors and 3σ have been plotted in Figure 5.4. The plots of Figure 5.4 show that for every component of the state vector the 3σ from OD is lower than the 3σ from propagation for nearly all the time and both are larger than the respective errors (except for the tangential position in the first 20 minutes of the EKF run): this means that in both cases 3σ overestimates the error (as we want) but the EKF overestimates it less. Regarding the errors, we can see that propagation always returns the smallest errors, except for the normal components of position and velocity where propagation errors are similar to EKF errors; the lower accuracy of the filter may be due to the influence of the non-diagonal elements of the covariance matrix, that are null at t_0 but acquire non-null values in other time steps.

The same test has been repeated for the WLS method, using the same settings and the variances from Table 5.1; the errors of the four runs have been plotted in Figure

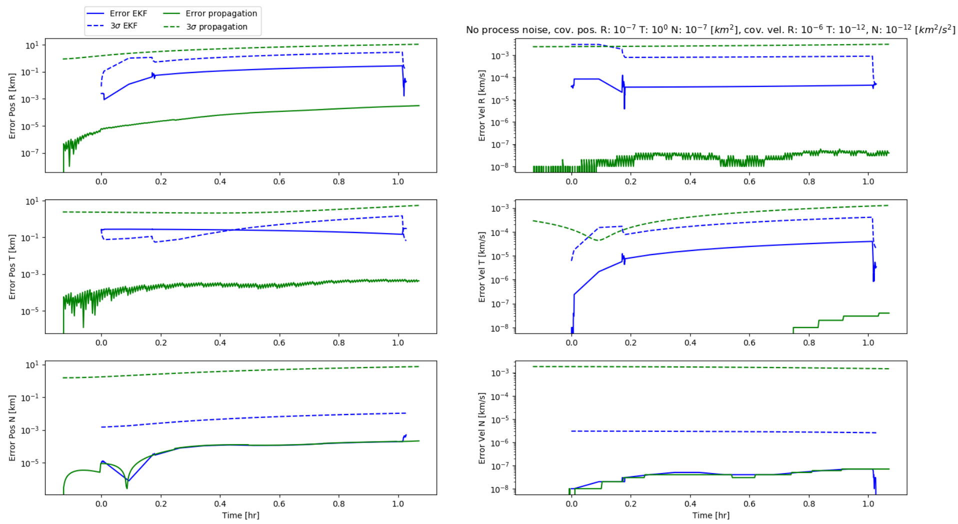

Figure 5.4: EKF vs propagation.

5.5, the "best" variances have been chosen and the new initial covariance matrix has been formed (Equation 5.4). Eventually $\mathbf{P_0}$ has been used to run propagation and orbit determination; Figure 5.6 contains the plots of the errors and 3σs.

$$\mathbf{P_0} = \begin{bmatrix} 10^{-5}\ \mathrm{km}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1\ \mathrm{km}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10^{-7}\ \mathrm{km}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10^{-8}\ \frac{\mathrm{km}^2}{\mathrm{s}^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 10^{-12}\ \frac{\mathrm{km}^2}{\mathrm{s}^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^{-12}\ \frac{\mathrm{km}^2}{\mathrm{s}^2} \end{bmatrix} \tag{5.4}$$

The results are similar to the test with EKF: WLS returns a 3σ that is smaller and closer to the error than propagation's one for every state's component, but the propagation is more accurate because it returns mostly smaller errors than WLS (which also presents bigger errors than 3σ for tangential position and radial velocity). So the same comments as the test with EKF can be repeated.

The conclusion that could be derived from these two tests is that if the initial state is very accurate, it is more convenient to perform propagation rather than OD in order to minimize the error; this would be true if we knew that the accuracy of the initial state is very high, but in real applications the accuracy of the initial state is usually unknown and there is no way of computing the real error, so it is worth doing orbit determination because it gives a smaller 3σ, i.e. a smaller

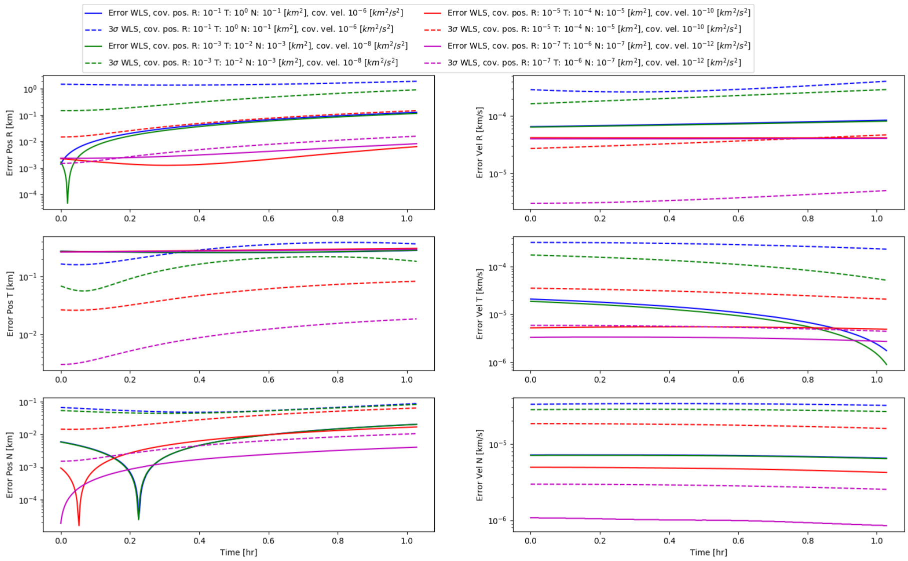

Figure 5.5: WLS errors for different covariances.

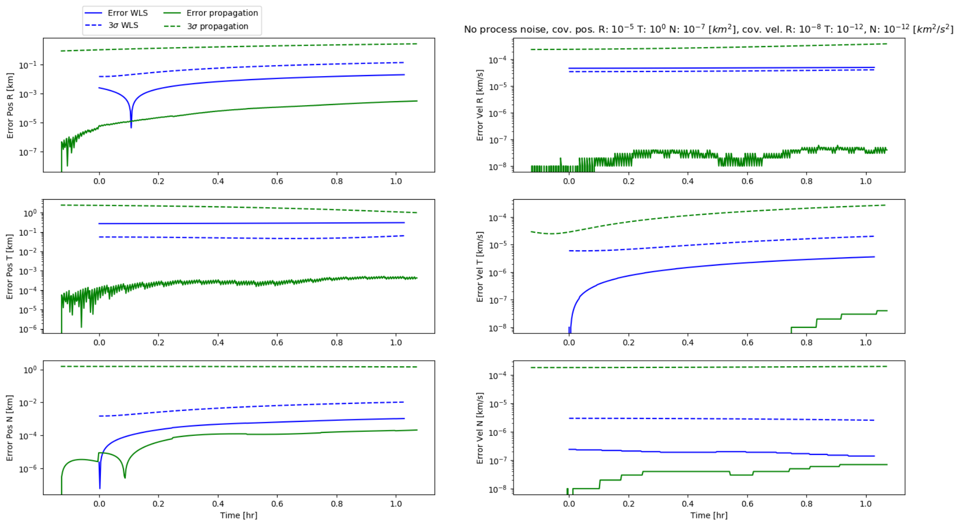

Figure 5.6: WLS vs propagation.

theoretical error, than propagation.

5.3 WLS sensitivity to the initial state

The tests in this section were aimed at analyzing how much sensitive the Weighted Least Square method is to the accuracy and variations of the initial state vector. The target object of the performed orbit determinations was the GPS satellite with NORAD ID 32711. The initial covariance matrix used in the following tests is shown in Equation 5.5; although the WLS method does not need an initial covariance matrix because it computes $\mathbf{P_0}$ at every iteration with Equation 3.8, the algorithm implemented in SPOOK can receive an initial covariance matrix as input, like in this case, and skip the computation of $\mathbf{P_0}$ during the first iteration.

$$\mathbf{P_0} = \begin{bmatrix} 10\ \text{km}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10^2\ \text{km}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10\ \text{km}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10^{-6}\ \frac{\text{km}^2}{\text{s}^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 10^{-6}\ \frac{\text{km}^2}{\text{s}^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^{-6}\ \frac{\text{km}^2}{\text{s}^2} \end{bmatrix} \tag{5.5}$$

The measurements in the TDM file used for these tests were split into 13 tracklets, whose start and end times are marked with vertical lines of the same colour in the plots of Figures 5.7 and 5.8.

In the first test orbit determination with WLS method has been run twice: first using an initial state vector from the OEM file used for error computation, so that the error at time t_0 is null; then extracting the initial state vector from a TLE file, leading to a quite inaccurate $\mathbf{x}_0$. The total errors and the corresponding 3σ of the two runs are plotted in Figure 5.7.

Looking at Figure 5.7 it seems that only the plots of the run with the initial state from the TLE file have been drawn; actually, also the plots of the run with the initial state coming from the OEM file are in the figure, they are just totally overlapped with the corresponding plots of the other run. Looking at SPOOK output files it turns out that the two runs return exactly the same state vectors (and therefore the same errors) and covariance matrices at every output time, except for the initial states, that are not plotted in the picture (time 0.00 in the figure is the relative time of the first measurement).

The second test consisted of the randomization of the initial state vector from the same TLE file as before by using the function of Section 4.4: 100 new initial state vectors were formed and 101 (including the original $\mathbf{x}_0$) OD runs with the WLS algorithm were performed. The total errors and 3σs of all the runs are shown in Figure 5.8.

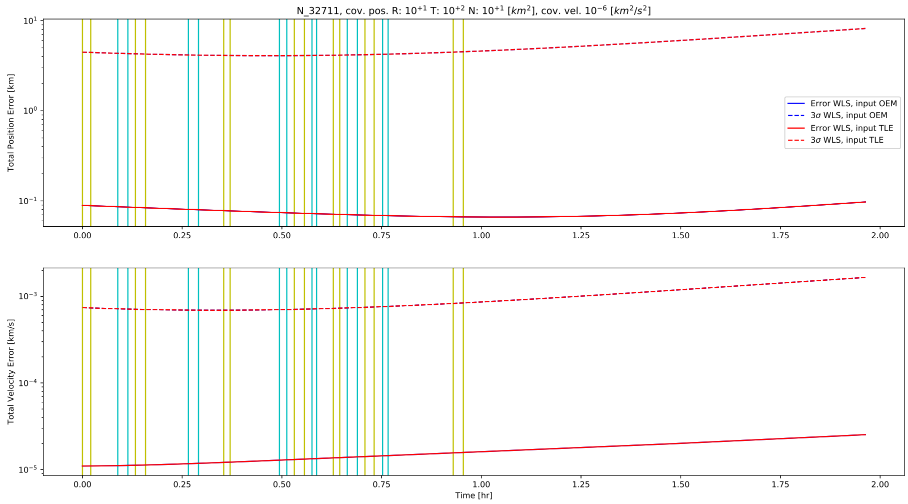

Figure 5.7: Total WLS errors for TLE input and OEM input.

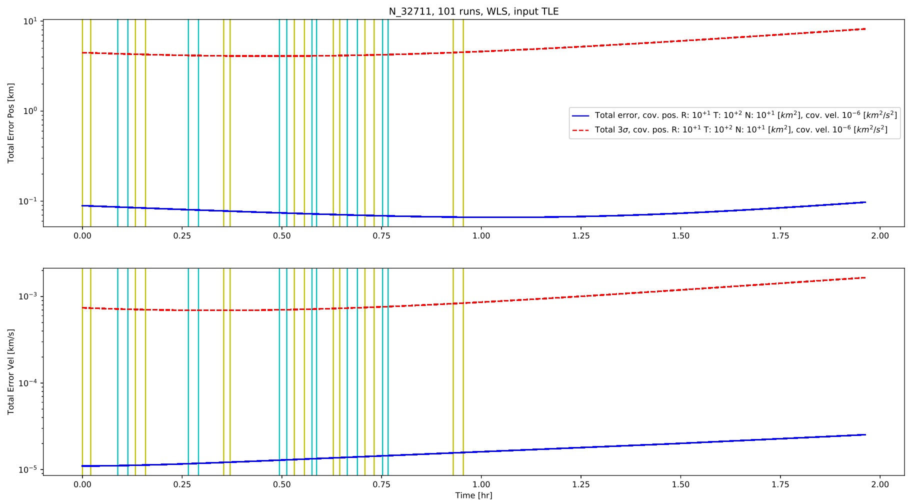

Figure 5.8: Total WLS errors for 101 runs.

From the thickness of the lines we can see that in Figure 5.8 all the plots of the same type are overlapped for all the runs; also the output files show that the estimated states, errors and covariances are the same at every time step for all the 101 runs (except for the initial states, that have not been plotted). Repeating this test using an initial state vector from the OEM file gave the same outcome.

From these tests it can be concluded that the Weighted Least Squares method is insensitive to the accuracy and the changes of the initial state vector: a possible explanation may be that since the implemented WLS algorithm propagates the initial state vector to the time of the first measurement and then improves it at every iteration, it manages to correct the state vector at the point of getting the same improved state whatever the initial one is (and then the outputs of the propagation will be the same). This hypothesis needs to be tested more, maybe by finding a way to reduce the computational costs of the extraction and initial propagation of $\mathbf{x}_0$ and performing more than 100 runs.

Anyway, even if the WLS output were really the same independently of which $\mathbf{x}_0$ is given as input, the choice of the initial state vector would still be important because it affects the number of iterations and the computation time needed to perform the extraction of $\mathbf{x}_0$ from its file and propagate it to the first measurement time.

5.4 Comparison between EKF and UKF

Several comparative tests have been conducted to assess the differences between the outputs of the Extended Kalman Filter and the outputs of the Unscented Kalman Filter: runs of orbit determination with both methods have been performed on four GPS satellites, whose NORAD IDs were 25933, 26360, 32711, 40105, using different settings in terms of source of the initial state vector, values of the elements of the initial covariance matrix, length and number of tracklets in the TDM files, length of propagation time at the end of OD and so on. The outcome of all these tests was that EKF and UKF gave very similar results for both estimated states, errors and estimated covariance matrices.

Figures 5.9 and 5.10 are shown here as examples: they contain the plots of the errors and 3σs of two tests conducted on the GPS satellite 32711 with the same initial covariance matrix of Equation 5.5; in the first test the initial state vector came from an OEM file, in the second test $\mathbf{x}_0$ came from a TLE file; the measurements came from a SPOOK simulated TDM file, like all the tests of this section.

In the pictures it is clearly visible, and it is also confirmed by SPOOK output files, that for both tests the errors (and so the state vectors) resulted from UKF and EKF are very similar, especially in the time intervals inside the tracklets (delimited by vertical lines of the same colors) where the error plots of the two methods are almost overlapped; in particular the 3σs from UKF and EKF are so

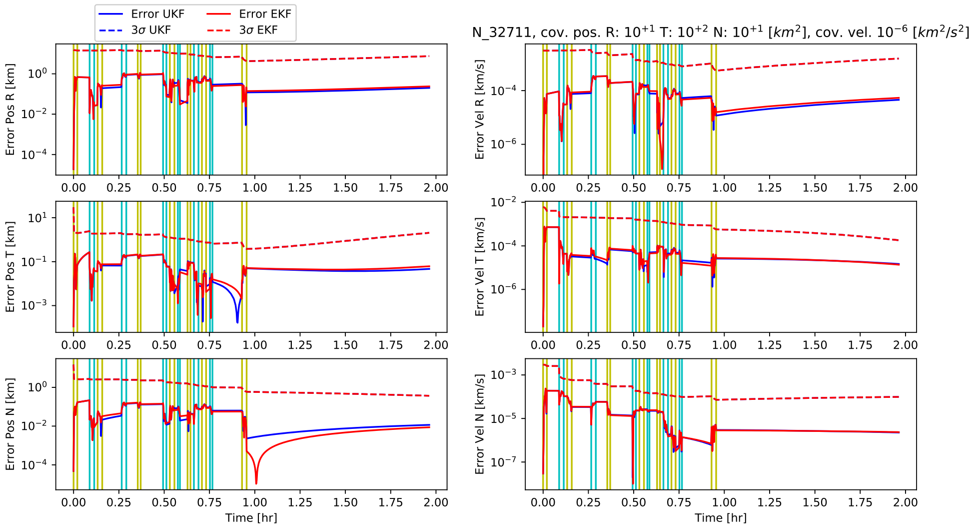

Figure 5.9: UKF and EKF errors; $\mathbf{x}_0$ from OEM file.

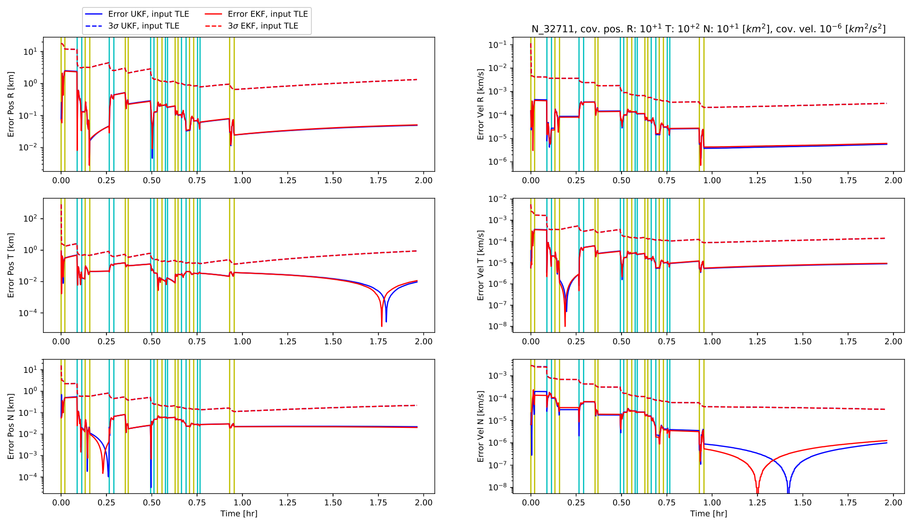

Figure 5.10: UKF and EKF errors; $\mathbf{x}_0$ from TLE file.

similar that their plots are completely overlapped in both figures. More or less the same outcomes have been gotten for all these kinds of tests.

One possible explanation of the similarity of the outputs of EKF and UKF could be the goodness of the physical model used for orbit determination: as explained in Subsection 3.6.4, the UKF works better than the EKF if the physical model used to perform orbit determination is not accurate; when it is run in Orbit Determination mode with simulated measurements, SPOOk uses a physical model to generate measurements and another physical model to perform OD; the two models are set by the user in the parameters configuration file, where he or she can decide to enable or disable certain orbital perturbations and can choose how to model them (for example the user can set the maximum degree and order of Earth's spherical harmonics). In these tests the two physical models had the same settings, so the model used for OD was accurate because it was equal to the model where measurements had been "taken" from and this was why EKF could work as effectively as UKF.

In order to verify the previous explanation, comparative tests should be performed in which the OD model is less accurate than the measurements model (e.g. not considering the third body effects of Venus and Mars or using a less accurate model for the atmosphere, etc.); if EKF results were less accurate than UKF ones with these settings, it would mean the explanation is correct.

Chapter 6

Conclusions

In the frame of this thesis the following improvements have been performed on SPOOK:

- the function to write accessibility reports has been modified in order to include additional information in these output files, making them useful to help planning real future observations;

- a new type of output file has been introduced, i.e. the optical output, whose information can be used to study the influence of certain parameters on the SNR and to optimize observation strategies along with the accessibility report;

- a function to compute the errors of Orbit Determination and Covariance Propagation modes has been implemented, making it possible to test the accuracy of OD results if an accurate ephemeris of the target object is available;

- a randomizing function for the generation of multiple initial state vectors and the performance of multiple OD runs has been implemented and tested: if the original state vector comes from an OEM file, 100 new states are almost enough to have a Gaussian distribution; if the original state vector is extracted from a TLE file, more than 100 new states are needed for their distribution to be Gaussian;

- the Python wrapper has been enriched with a function that speeds up the process of launching an Orbit Determination run.

Also several tests have been performed on the OD section of SPOOK's code; these are the conclusions of the tests:

- if the initial state vector is very accurate, propagation of the state vector returns smaller errors than orbit determination; however, if the accuracy of the state vector is unknown and the real error cannot be computed, it is more convenient to perform orbit determination because it returns a lower 3σ than propagation;

- the accuracy of the Weighted Least Squares method is not affected by the accuracy of the initial state vector; so the main criteria for choosing the initial state should be the minimization of the number of iterations and the minimization of the time needed to extract the initial state and propagate it to the first measurement time;

- Extended Kalman Filter and Uscented Kalman Filter seem to give very similar results when the physical model used for orbit determination is very accurate with respect to the physical model used to simulate measurements.

Chapter 7

Future Projects

Here some suggestions are presented for people who will follow up on this thesis:

- the Python function for launching orbit determination should be improved by automatizing even more settings and adding calls to functions that plot the results automatically;
- the Python function should be used several times in order to create some statistics and define the best OD settings for different scenarios (e.g. what is the best OD method for certain tracklet lengths and certain sensors' accuracy) and orbital classes of the objects;
- the tests to compare EKF and UKF should be performed with less accurate physical OD models than the physical model for measurements simulation in order to prove if UKF really is better when the model is inaccurate;
- it could be possible to improve the number of orbit determinations performed by the randomization function without having a large computation time by implementing code parallelization in the function, so that it could perform multiple runs simultaneously;
- the OD success tests implemented by Fiusco [15] should be accurately tested in order to enable them and use them with confidence;
- tests on the implemented IOD methods should be performed to evaluate how accurately they estimate the initial state vector and covariance matrix and whether they improve the whole OD process;
- the implemented methods for wrong measurements (called *outliers*) removal should be tested and improved.

Appendix A

Fundamentals of Statistics

The aim of this appendix is to explain some of the statistical concepts that have been used in the previous chapters. The following definitions are taken from [15].

Given a random variable x and N samples x_i, the *mean* of x can be computed as $m = \mathrm{E}(x)$ and the *variance* of x is

$$\sigma^2 = \mathrm{E}[(x-m)^2] \approx \frac{1}{N-1}\sum_{i=1}^{N}(x_i - m)^2 \tag{A.1}$$

The variance describes how close the samples are to the mean [29]; the square root of the variance $\sigma = \sqrt{\sigma^2}$ is called *standard deviation*.

If two random variables x and y are considered, their *covariance* can be computed as

$$\mathrm{cov}(x, y) = \mathrm{E}[(x - \mathrm{E}(x))\,(y - \mathrm{E}(y))] \tag{A.2}$$

and represents the tendency of the two variables to be linearly correlated; comparing Equations A.1 and A.2 it is clear that the variance is just the covariance of a variable with itself.

It is worth introducing the *correlation coefficient* μ of two variables [26] to better understand the meaning of covariance; considering the two variables x and y, their correlation coefficient is

$$\mu_{xy} = \frac{\mathrm{cov}(x, y)}{\sigma_x \sigma_y} \tag{A.3}$$

The correlation coefficient can range from -1 to $+1$: if positive, the two variables have a direct correlation (if one increases, the other one increases too); if negative, the variables have an inverse correlation (one increases if the other one decreases and viceversa); if null, the two variables are indipendent.

If we now consider a vector $\mathbf{X} = [X_1, ..., X_N]$ of random variables, we can compute the covariance matrix $\mathbf{P}$:

$$\mathbf{P} = \mathrm{E}\left[(\mathbf{X} - \mathrm{E}(\mathbf{X}))(\mathbf{X} - \mathrm{E}(\mathbf{X}))^T\right] \tag{A.4}$$

where $\mathrm{E}(\mathbf{X})$ is the vector of the expected values of the components of $\mathbf{X}$. $\mathbf{P}$ is a $N \times N$ matrix where every component $P_{ij} = \mathrm{cov}(X_i, X_j)$ is the covariance between two components of $\mathbf{X}$: it results that the matrix is symmetric and the elements on the main diagonal are the variances of the components of $\mathbf{X}$.

So for example, if $\mathbf{X} = [\alpha, \beta, \gamma]$, the covariance matrix is

$$\mathbf{P} = \begin{bmatrix} \sigma_\alpha^2 & \mu_{\alpha\beta}\sigma_\alpha\sigma_\beta & \mu_{\alpha\gamma}\sigma_\alpha\sigma_\gamma \\ \mu_{\alpha\beta}\sigma_\alpha\sigma_\beta & \sigma_\beta^2 & \mu_{\beta\gamma}\sigma_\beta\sigma_\gamma \\ \mu_{\alpha\gamma}\sigma_\alpha\sigma_\gamma & \mu_{\beta\gamma}\sigma_\beta\sigma_\gamma & \sigma_\gamma^2 \end{bmatrix}$$

In this thesis it is assumed the errors in measurements and OD process to be Gaussian distributed for every component of the target object's state vector [11], so that their mean is null (because the mean of each component of the state vector corresponds to its exact value) and their standard deviation is approximately equal to the sensor accuracy [15]. The assumption of Gaussian distribution also means that most (at least 99.7%) errors are lower than 3σ (for each component of the state vector) and that the object's position should be inside a region called *covariance ellipsoid* [11], whose shape and size depend on the covariance matrix.

Appendix B

Examples of files

This appendix provides examples of the files described in the previous chapters.

B.1 Object definition files

B.1.1 OEM file

The following two pictures show two sections of an OEM file: in Figure B.1 the state vector is given for each epoch; Figure B.2 shows the covariance matrix at every epoch.

```
CCSDS_OEM_VERS = 2.0
COMMENT WRITING ORBIT EPHEMERIS MESSAGE V=2.0 SPOOK
CREATION_DATE =2018-01-01T12:00:00
ORIGINATOR = AIRBUS DEFENCE & SPACE

META_START
OBJECT_NAME = DebrisOb
OBJECT_ID = yyyy-nnnA
REF_FRAME = GCRF
CENTER_NAME = EARTH
TIME_SYSTEM = UTC
START_TIME = 2015-01-01T07:12:10.000
USEABLE_START_TIME = 2015-01-01T07:12:10.000
USEABLE_STOP_TIME = 2015-01-04T07:12:10.000
STOP_TIME = 2015-01-04T07:12:10.000
INTERPOLATION = NA
INTERPOLATION_DEGREE = NA
META_STOP
2015-01-01T07:12:10.000  -2840.257742  -3348.296743  -6469.410259  9.069690 -0.449748 -2.117425
2015-01-01T07:12:40.000  -2567.132058  -3360.523366  -6530.485842  9.137659 -0.365419 -1.953646
2015-01-01T07:13:10.000  -2292.058298  -3370.203871  -6586.590405  9.199594 -0.279617 -1.786067
2015-01-01T07:13:40.000  -2015.219743  -3377.287765  -6637.615119  9.255183 -0.192529 -1.614999
2015-01-01T07:14:10.000  -1736.813181  -3381.743497  -6683.459334  9.304135 -0.104333 -1.440789
2015-01-01T07:14:40.000  -1457.040514  -3383.539357  -6724.035106  9.346195 -0.015233 -1.263808
2015-01-01T07:15:09.000  -1176.113083  -3382.650552  -6759.263770  9.381138  0.074568 -1.084457
2015-01-01T07:15:39.000   -894.245988  -3379.061197  -6789.082591  9.408778  0.164850 -0.903151
2015-01-01T07:16:09.000   -611.660434  -3372.758033  -6813.438448  9.428969  0.255387 -0.720328
2015-01-01T07:16:39.000   -328.583323  -3363.736452  -6832.291409  9.441609  0.345959 -0.536447
2015-01-01T07:17:09.000    -45.241046  -3352.000455  -6845.617898  9.446640  0.436332 -0.351965
2015-01-01T07:17:39.000    238.138963  -3337.558929  -6853.407164  9.444046  0.526277 -0.167345
2015-01-01T07:18:09.000    521.325458  -3320.430116  -6855.661968  9.433862  0.615576  0.016937
```

Figure B.1: OEM file part 1 [16].

```
COVARIANCE_START
EPOCH =  2015-01-01T07:12:10.000
COV_REF_FRAME = GCRF
  2.64960E-05
  3.61899E-05   1.52291E-04
 -3.73115E-05  -9.54536E-06   7.04612E-05
  1.33357E-08   1.62344E-08  -2.17914E-08   1.32549E-11
 -8.09778E-08  -2.19120E-07   7.92375E-08  -7.07396E-11   5.44220E-10
  9.78294E-08   6.60059E-08  -1.65378E-07   4.59876E-11  -1.89713E-10   4.20421E-10
EPOCH =  2015-01-01T07:12:10.000
COV_REF_FRAME = RTN
  6.17317E-05
  2.74519E-05   4.95439E-05
  4.56878E-05  -4.16022E-05   1.37973E-04
 -8.74010E-08  -9.00944E-08   1.57707E-08   2.60584E-10
 -2.62348E-08  -2.39228E-08  -2.75226E-09   9.32538E-11   3.97429E-11
 -1.93337E-08   1.33718E-07  -2.59838E-07  -7.98377E-11   1.73282E-11   6.77569E-10
EPOCH =  2015-01-01T07:12:40.000
COV_REF_FRAME = GCRF
  2.63664E-05
  3.32660E-05   1.33554E-04
 -3.38250E-05  -5.66048E-06   5.90597E-05
  1.27219E-08   1.39130E-08  -1.85249E-08   1.19361E-11
 -8.26412E-08  -1.93406E-07   7.53451E-08  -6.80445E-11   5.29565E-10
  9.62465E-08   6.42457E-08  -1.46825E-07   4.17080E-11  -1.98570E-10   4.04839E-10
```

Figure B.2: OEM file part 2 [16].

B.1.2 TLE file

Here follows an example of a two-line set that can be found inside a TLE file. Figure B.3 has been downloaded from *Space-Track.Org* as well as the other TLE files used in this thesis.

```
ISS (ZARYA)
1 25544U 98067A   04236.56031392  .00020137  00000-0  16538-3 0  9993
2 25544  51.6335 344.7760 0007976 126.2523 325.9359 15.70406856328906
```

Figure B.3: TLE element [25].

B.2 Measurements files

B.2.1 OTDF file

Figure B.4 is an example of OTDF file. It can be seen that the observer in the picture is an optical one because there are non-zero values for the columns of right ascension and declination; this file also contains the columns (that are not included in the picture) for the measurements of a radar observer, i.e. azimuth and elevation, that in this case have only null values (it would be the opposite for a radar observer).

```
# OTDF File: Contains the measurement values obtained by one observer for one tracklet (object pass), along with some observer properties.
# Col1     Col2      Col3                     Col4       Col5 Col6   Col7     Col8      Col9      Col10  Col11 Col12       Col13      Col14       Col15
# 1-8      10-13     15-40                    42-43      45-51 53-59 61-67    69-77     79-87     89-97  99    102-112     114-123    126-135     137-146
# Track    Station   UTC                      Reference  T    p      Humidity Lat,WGS84 Lon,WGS84 Alt    nMeas MeasType    MeasValue  MeasType    MeasValue
# ID       Grp ID    Time Tag                 Time  Flag [C] [hPa]   [%]      [deg]     [deg]     [m]
 OTDF                                          ST
DebrisOb SBOP 2014/04/07-02:45:49.000029  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.891302 Declination  -1.079619
DebrisOb SBOP 2014/04/07-02:45:50.500041  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.894661 Declination  -1.094398
DebrisOb SBOP 2014/04/07-02:45:52.000012  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.897107 Declination  -1.110285
DebrisOb SBOP 2014/04/07-02:45:53.500024  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.900577 Declination  -1.125070
DebrisOb SBOP 2014/04/07-02:45:55.000036  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.903425 Declination  -1.140416
DebrisOb SBOP 2014/04/07-02:45:56.500008  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.906315 Declination  -1.155809
DebrisOb SBOP 2014/04/07-02:45:58.000020  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.909613 Declination  -1.170925
DebrisOb SBOP 2014/04/07-02:45:59.500032  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.912341 Declination  -1.186099
DebrisOb SBOP 2014/04/07-02:46:01.000044  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.915307 Declination  -1.201406
DebrisOb SBOP 2014/04/07-02:46:02.500016  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.918291 Declination  -1.216537
DebrisOb SBOP 2014/04/07-02:46:04.000028  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.920918 Declination  -1.231698
DebrisOb SBOP 2014/04/07-02:46:05.500040  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.924051 Declination  -1.247074
DebrisOb SBOP 2014/04/07-02:46:07.000012  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.926861 Declination  -1.262390
DebrisOb SBOP 2014/04/07-02:46:08.500024  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.929656 Declination  -1.277448
DebrisOb SBOP 2014/04/07-02:46:10.000036  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.932511 Declination  -1.292965
DebrisOb SBOP 2014/04/07-02:46:11.500007  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.935007 Declination  -1.307754
DebrisOb SBOP 2014/04/07-02:46:13.000019  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.937679 Declination  -1.322994
DebrisOb SBOP 2014/04/07-02:46:14.500031  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.940449 Declination  -1.338346
DebrisOb SBOP 2014/04/07-02:46:16.000043  1  10.000  850.00  40.000   0.0000   0.0000   0.0000 2  Right_Asc  192.943046 Declination  -1.353059
```

Figure B.4: OTDF file [16].

B.2.2 TDM file

Figure B.5 presents an example of the measurements inside a TDM file; additional measurements like humidity and pressure can be written in these files too.

```
DATA_START
ANGLE_1 = 2015-01-03T16:10:10 138.544765301766
ANGLE_2 = 2015-01-03T16:10:10 49.862023764362

ANGLE_1 = 2015-01-03T16:11:60 139.019279110396
ANGLE_2 = 2015-01-03T16:11:60 49.618492053123

ANGLE_1 = 2015-01-03T16:13:50 139.493395273963
ANGLE_2 = 2015-01-03T16:13:50 49.370589110805
```

Figure B.5: TDM file [16].

B.3 Output files

B.3.1 Accessibility report

Figures B.6 and B.7 show an example of the accessibility report produced by SPOOK. The columns from "ALTITUDE" to "PHASE ANGLES" in Figure B.7 are the ones added during this thesis' works.

```
Accessibility report SPOOK

List of constraints
Use Line of Sight: YES
MaxRange considered: NO , value: [km]      0.0000
MaxPhaseAngle considered: NO , value: [rad]      0.0000
Minimum illumination considered: YES, Type (1 = sunlight, 2 = penumbra):  2
Exclude Sun zone: NO , value: [rad]      0.0000
Exclude Moon zone: NO , value: [rad]      0.0000
Exclude Venus zone: NO , value: [rad]      0.0000
Exclude Jupiter zone: NO , value: [rad]      0.0000
Exclude MilkyWay zone: NO , value: [rad]      0.0000
Earth limb height considered: NO , value: [rad]      0.0000
Maximum Sun elevation considered: YES, value: [deg]     -9.0000
Minimum target elevation considered: YES, value: [deg]     20.0000
Maximum target elevation considered: NO , value: [deg]     90.0000
Local meridian crossings considered: NO , value: [deg]      0.0000
WeatherModel: not yet implemented
Sensor Performance Model used: YES

Accessibility observer name:    2
Observer type:  optical

list of visible objects
SPOOK_NUMBER -    NAME    - COSPAR_ID - NORAD_ID - CAT_ID  -|-       START_TIME        - ELEVATION - AZIMUTH   -|-       END_TIME          - ELEVATION - AZIMUTH -
                                                           -|-          [UTC]          -   [deg]   - [deg]     -|-          [UTC]          -   [deg]   -  [deg]
0001            DebrisOb    1989-062B   20169     N_20169     2019-11-12T20:39:13.500   38.548       6.63 (NW)   2019-11-12T21:20:52.500   20.052     41.93 (NW

list of uncrossed objects
SPOOK_NUMBER -    NAME    - COSPAR_ID - NORAD_ID - CAT_ID
```

Figure B.6: Accessibility report part 1.

```
· DURATION - APOGEE/PERIGEE                  - ALTITUDE      - ANGULAR_MOTION           - SHADOWING                              - SNR          - PHASE_ANGLE
     [s]        [km]                           [km]           [deg/s]           [0: umbra - 1: fully illuminated]                                 [deg]
)    2499    35640/526    (LEO Transient)      6614/15787     0.936E-02/0.386E-01              1/1                                2.520/3.029      10.191/55.479
```

Figure B.7: Accessibility report part 2.

B.3.2 Optical output

An example of optical output file is given in Figure B.8. The function to write this file has been written in the framework of this thesis.

TIME [JD]	ALTITUDE [km]	PHASE ANGLE [deg]	ELEVATION [deg]	RANGE [km]	ANGULAR VELOCITY ["/s]	APPARENT MAGNITUDE [mag]	SNR
2458800.36057291646	15787.647951	55.479	38.548	17615.965747	33.713	14.396	2.520
2458800.36059027771	15782.894004	55.466	38.545	17611.262457	33.732	14.395	2.521
2458800.36060763849	15778.139336	55.452	38.543	17606.558580	33.751	14.394	2.521
2458800.36062499974	15773.383691	55.438	38.540	17601.853862	33.770	14.393	2.522
2458800.36064236099	15768.627196	55.425	38.538	17597.148431	33.789	14.392	2.522
2458800.36065972224	15763.869852	55.411	38.535	17592.442287	33.808	14.392	2.523
2458800.36067708302	15759.111785	55.397	38.533	17587.735555	33.827	14.391	2.523
2458800.36069444427	15754.352741	55.384	38.530	17583.027984	33.846	14.390	2.524
2458800.36071180552	15749.592847	55.370	38.528	17578.319700	33.865	14.389	2.524
2458800.36072916631	15744.832231	55.356	38.525	17573.610829	33.884	14.388	2.525

Figure B.8: Optical output file.

B.3.3 Ephemeris errors

An example of ephemeris errors file can be seen in Figures B.9 and B.10. The first and last columns of the file, corresponding to the time in Julian Days (JD) and to P_{66} respectively, are not included in the pictures.

Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9
time [UTC]	Interpolation status (0: OK)	Tot_r_err [km]	sigma_r_err [Km]	Tot_v_err[km/s]	sigma_v_err[Km/s]	delta_r_R_RTN[km]	delta_r_T_RTN[km]
2019/03/12-01:14:42.0000	0	0.00000E+00	0.12247E+02	0.00000E+00	0.24495E-02	0.00000000	0.00000000
2019/03/12-01:22:26.0570	0	0.82894E-04	0.12286E+02	0.63947E-08	0.24515E-02	-0.00000558	-0.00008222
2019/03/12-01:22:28.6965	0	0.18052E-01	0.51014E+01	0.15446E-06	0.24498E-02	-0.00053774	-0.00456844
2019/03/12-01:22:31.3525	0	0.49775E-01	0.51003E+01	0.33563E-04	0.24484E-02	-0.03923274	0.01852746
2019/03/12-01:22:33.9920	0	0.32790E-01	0.50981E+01	0.73672E-05	0.24444E-02	-0.00088453	0.01294127
2019/03/12-01:22:36.6323	0	0.16050E+00	0.50934E+01	0.13506E-03	0.24365E-02	0.15829582	0.00962543
2019/03/12-01:22:39.2874	0	0.12997E+00	0.50855E+01	0.10962E-03	0.24235E-02	0.12749376	0.00856938
2019/03/12-01:22:41.9287	0	0.25100E-01	0.50740E+01	0.44265E-04	0.24048E-02	0.01955133	0.00394023
2019/03/12-01:22:44.5829	0	0.59942E-01	0.50587E+01	0.46657E-04	0.23800E-02	-0.05750797	-0.00046370
2019/03/12-01:22:47.2388	0	0.19239E+00	0.50397E+01	0.15863E-03	0.23490E-02	-0.19205537	-0.00829481
2019/03/12-01:22:49.8947	0	0.37791E-01	0.50175E+01	0.67744E-04	0.23123E-02	0.03458662	0.00680424
2019/03/12-01:22:52.5343	0	0.21003E+00	0.49929E+01	0.19909E-03	0.22711E-02	0.20859073	0.01832366

Figure B.9: Ephemeris error file part 1.

Col 10	Col 11	Col 12	Col 13	Col 14	Col 15	Col 16	Col 17	Col 18
delta_r_N_RTN[km]	delta_v_R_RTN[km/s]	delta_v_T_RTN[km/s]	delta_v_N_RTN[km/s]	P_RTN (1,1)	P_RTN (2,2)	P_RTN (3,3)	P_RTN (4,4)	P_RTN (5,5)
0.00000000	0.00000000	0.00000000	0.00000000	2.50000E+01	1.00000E+02	2.50000E+01	1.00000E-06	4.00000E-06
0.00000894	-0.00000000	-0.00000000	0.00000000	2.58211E+01	1.00018E+02	2.50956E+01	1.03480E-06	3.97704E-06
0.01745635	-0.00000002	-0.00000004	0.00000015	2.57429E+01	2.59013E-01	2.21680E-02	1.03518E-06	3.96996E-06
0.02439457	-0.00000069	0.00003337	0.00000354	2.57415E+01	2.53715E-01	1.78618E-02	1.03558E-06	3.96342E-06
0.03011531	-0.00000016	0.00000183	0.00000713	2.57226E+01	2.51240E-01	1.64095E-02	1.03597E-06	3.94464E-06
0.02472030	0.00000264	-0.00013440	-0.00001314	2.56778E+01	2.49303E-01	1.56610E-02	1.03636E-06	3.90802E-06
0.02375238	0.00000214	-0.00010881	-0.00001313	2.56001E+01	2.47407E-01	1.51860E-02	1.03675E-06	3.84877E-06
0.01523859	0.00000088	-0.00002374	-0.00003735	2.54857E+01	2.45409E-01	1.48373E-02	1.03713E-06	3.76417E-06
0.01689984	-0.00000068	0.00004437	-0.00001442	2.53328E+01	2.43239E-01	1.45556E-02	1.03751E-06	3.65307E-06
0.00785820	-0.00000196	0.00014779	-0.00005759	2.51434E+01	2.40916E-01	1.43069E-02	1.03788E-06	3.51691E-06
0.01362419	0.00000143	-0.00003939	-0.00005510	2.49228E+01	2.38490E-01	1.40809E-02	1.03824E-06	3.35937E-06
0.01638706	0.00000429	-0.00018617	-0.00007043	2.46789E+01	2.36032E-01	1.38682E-02	1.03860E-06	3.18621E-06

Figure B.10: Ephemeris error file part 2.

Bibliography

[1] D. A. Vallado. *Fundamental of Astrodynamics and Applications 1st Edition*. New York, USA: McGraw-Hill, 1997 (cit. on pp. 12, 18–20, 23).

[2] F. Orderud. «Comparison of Kalman Filter Estimation Approaches for State Space Models with Nonlinear Measurements». In: *SIAM Review* 33.3 (2005) (cit. on p. 26).

[3] A. Greenland and M. Higgins. *FIG Guide on the Development of a Vertical Reference Surface for Hydrography*. Copenaghen, Denmark: International Federation of Surveyors (FIG), Sept. 2006 (cit. on p. 13).

[4] *Orbit Data Messages*. ser. Blue Book. Recommendation for Space Data System Standard. Newport Beach, CA: Consulative Committee for Space Data Systems (CCSDS), Nov. 2009. URL: `https://public.ccsds.org/pubs/502x0b2c1.pdf` (cit. on p. 14).

[5] E. J. Fletcher. *Status and progress in the Space Surveillance and Tracking Segment of ESA's Space Situational Awareness Programme*. Tech. rep. Madrid, Spain: European Space Agency, Sept. 2010 (cit. on p. 30).

[6] J. J. Lissauer and I. de Pater. *Fundamental planetary science: physics, chemistry, and habitability*. Cambridge University Press, 2013 (cit. on p. 14).

[7] J. D. Biggs et al. «Planning Natural Repointing Manoeuvres for Nano-Spacecraft». In: *IEEE Transactions on Aerospace and Electronic Systems* 50.3 (Oct. 2014) (cit. on p. 13).

[8] J. Utzmann et al. «Airbus Robotic Telescope». In: *ESA NEO and Debris Detection Conference*. Darmstadt, Germany, 2014 (cit. on pp. 7, 9).

[9] H. H. Dreyer. «Development of an Orbit Determination Simulation Tool based on an Extended Kalman Filter». MA thesis. Stuttgart, Germany: Universität Stuttgart, Feb. 2015 (cit. on p. 18).

[10] J. Stauch and M. Jah. «On the Unscented Schmidt-Kalman Filter Algorithm». In: *Journal of Guidance Control and Dynamics* 38.1 (Jan. 2015) (cit. on p. 28).

[11] Ó. Rodríguez. «Improvement and Validation of a Space Debris Orbit Determination Tool». MA thesis. Cranfield, United Kingdom: Cranfield University, Sept. 2016 (cit. on pp. 16, 20, 36, 54).

[12] M. Winzen. «Enhancement of an Orbit Determination Simulation Tool for Space Debris». MA thesis. Aachen, Germany: Aachen University, May 2016 (cit. on pp. 16, 17).

[13] K. Fletcher. *Space Debris: the ESA Approach.* Brochure. Mar. 2017. URL: `https://download.esa.int/esoc/downloads/BR-336_Space_Debris_WEB.pdf` (cit. on pp. 1, 2).

[14] I. U. Duendar. «Improvement of a Space Surveillance Tracking Analysis Tool». MA thesis. Würzburg, Germany: Universität Würzburg, Nov. 2018 (cit. on pp. 12, 22, 30).

[15] F. Fiusco. «Improvement of a Space Debris Cataloguing Tool». MA thesis. Stockholm, Sweden: Kungliga Tekniska Högskolan, Oct. 2018 (cit. on pp. 14, 15, 19, 29, 30, 51, 53, 54).

[16] Z. A. Shakil. «Optimization of a Simulation Framework for Space Debris Observation». MA thesis. Berlin, Germany: Technische Universität Berlin, June 2018 (cit. on pp. 17, 55–57).

[17] *Tracking Data Message.* ser. Pink Book. Draft Recommendation for Space Data System Standards. Newport Beach, CA: Consulative Committee for Space Data Systems (CCSDS), May 2018. URL: `https://public.ccsds.org/Lists/CCSDS%205030P11/503x0p11.pdf` (cit. on p. 17).

[18] J. Burgard. «Improvement of Data Processing for the Optical Observation of Space Debris». MA thesis. Stuttgart, Germany: Universität Stuttgart, Nov. 2019 (cit. on pp. 5, 12, 18, 19, 23–25, 28).

[19] G. Cirillo. «Space Surveillance and Tracking Tool: Implementation and Test of New Methods». MA thesis. Turin, Italy: Politecnico di Torino, Mar. 2019 (cit. on pp. 15, 25, 32, 33).

[20] G. Pedone. «Strategies and Data Processing for the Optical Observation of Space Debris». MA thesis. Milan, Italy: Politecnico di Milano, Mar. 2019 (cit. on pp. 7, 8, 14, 15).

[21] Ó. Rodríguez et al. «SPOOK - A comprehensive Space Surveillance and Tracking analysis tool». In: *Acta Astronautica* 158 (May 2019), pp. 178–184 (cit. on pp. 5, 6).

[22] CelesTrak. *SATCAT Growth.* URL: `https://celestrak.com/satcat/growth.png`. (last edited on: 02/11/2020) (cit. on p. 3).

[23] ESA. *Distribution of debris*. URL: https://www.esa.int/ESA_Multimedia/Images/2013/04/Distribution_of_debris. (last edited on: 16/04/2013) (cit. on p. 1).

[24] ESA's Space Debris Office. *Space debris by the numbers*. URL: https://www.esa.int/Safety_Security/Space_Debris/Space_debris_by_the_numbers. (last edited on: 02/2020) (cit. on pp. i, 2).

[25] Space-Track.Org. *Basic Description of the Two Line Element (TLE) Format*. URL: https://www.space-track.org/documentation#tle. (last visited on: 31/10/2020) (cit. on p. 56).

[26] The Free Encyclopedia Wikipedia. *Pearson correlation coefficient*. URL: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient. (last edited on: 15/10/2020) (cit. on p. 53).

[27] The Free Encyclopedia Wikipedia. *Phase angle (astronomy)*. URL: https://en.wikipedia.org/wiki/Phase_angle_(astronomy). (last edited on: 29/05/2020) (cit. on p. 32).

[28] The Free Encyclopedia Wikipedia. *Satellite Catalog Number*. URL: https://en.wikipedia.org/wiki/Satellite_Catalog_Number. (last edited on: 19/10/2020) (cit. on p. 38).

[29] The Free Encyclopedia Wikipedia. *Variance*. URL: https://en.wikipedia.org/wiki/Variance. (last edited on: 13/10/2020) (cit. on p. 53).