Improvement of a Space Surveillance and Tracking Analysis tool

FRANCESCO FIUSCO
**Abstract**

This thesis deals with the improvement of SPOOK (SPace Objects Observation and Kalman filtering), an orbit calculation tool developed by Airbus Defence and Space GmbH. The work described in this thesis aims at improving the architecture and analysis capabilities of the software on different levels:

- Design and build a framework that can use SPOOK as a calculation engine and use its capabilities to build a complete SST system for man-made objects orbiting the Earth, providing commercial services (e.g. collision avoidance, visualization, re-entry analysis, etc.), catalog maintenance and simulations. A complete Python API was designed and implemented, which makes now SPOOK a complete cataloguing system for man-made space objects that can provide services to the end user;

- Estimate covariance information from TLE data published by the US Space Command (available e.g. on Space-track.org);

- Devise and validate metrics that can assess the quality of an orbit determination process automatically, to ensure as small human interaction as possible;

- Preliminarily implement a fast Lambert problem solver.

In addition to this, a variety of miscellaneous activities were performed.
Sammanfattning

Detta examensarbete handlar om förbättringar av SPOOK (observation av rymdobjekt och Kalmanfiltrering), ett beräkningsverktyg för omloppsbanor utvecklat av Airbus Defence och Space GmbH. Detta arbete syftar till att förbättra arkitekturen hos programvaran och dess förmåga att utföra analys på olika nivåer:

- Designa och bygga ett ramverk användes SPOOK som beräkningsmotor och använda dess kapacitet för att bygga ett komplett SST-system för konstgjorda material kretsande runt jorden, tillhandahålla kommersiella tjänster (e.g. undvika kollision, visualisering, analys av återinträde etc.), katalogunderhåll och simuleringar. En komplett Python-API designades och implementerades, som nu gör SPOOK till ett komplett katalogiseringsystem för konstgjorda rymdobjekt som kan tillhandahålla tjänster för slutanvändare;

- Uppskatta kovariansen av TLE data publicerad av US Space Command (tillgängligt via Space-track.org);

- Utforma och validera kvalitetskoefficienter som automatiskt kan bedöma kvaliteten hos uppskattningen av en omloppsbana och därmed minimera interaktionen med användaren;

- Preliminärt implementera en snabb lösare för Lambertproblem.
Acknowledgements

I would like to thank Oscar Rodriguez Fernandez and Jens Utzmann at Airbus Defence and Space in Friedrichshafen for their guidance and constant support in writing this thesis. Your doors were always open for insightful and interesting discussions about this work. I would also like to thank my friends both in Germany and in Sweden for having made this journey memorable and having shared joy and pain during these two years. A special mention goes to my family, without whom I couldn’t have made it, Luigi for always being there for me even thousands of kilometers away, and Paola, who stood with me through everything and taught me to always strive to be the better version of myself.
Contents

1 Introduction ........................................ 7
   1.1 Background ................................... 7
   1.2 SPOOK ......................................... 9

2 SPOOK API ........................................... 10
   2.1 SPOOK calculation core ......................... 10
   2.2 SST architecture ............................... 11
      2.2.1 The databases ............................ 13
      2.2.2 Sensor simulator ......................... 14
      2.2.3 Correlation ............................... 17
      2.2.4 Initial Orbit Determination ............. 17
      2.2.5 Orbit Determination ...................... 17
      2.2.6 Covariance propagation ................. 18
      2.2.7 Python framework logic .................. 18
   2.3 Cataloguing requirements ..................... 21
      2.3.1 Introduction .............................. 21
      2.3.2 Requirements ............................. 21
         2.3.2.1 Catalogue maintenance ............. 21
         2.3.2.2 Collision avoidance ................. 22
   2.4 Test run of the SST system ................. 23

3 Estimation of covariance information from TLEs ... 29
   3.1 Introduction .................................. 29
   3.2 Theoretical background ....................... 30
   3.3 Estimation of a covariance matrix from TLEs .. 31
   3.4 Test cases ................................... 32
   3.5 Results and discussion ....................... 32

4 Orbit Determination success criteria ............... 37
   4.1 Chi squared test .............................. 38
   4.2 Weighted Root Mean Square .................. 38
   4.3 Self consistency test ......................... 39
   4.4 McReynolds filter-smoother consistency test . 40
4.5 $3\sigma$ envelope test .................................................. 40
4.6 Validation and discussion ........................................... 41
  4.6.1 Results of the tests ............................................... 41
  4.6.2 Discussion .......................................................... 43

5 Solver for the Lambert problem ........................................ 44
  5.1 Algorithm implementation ......................................... 44
  5.2 Performance comparison ........................................... 49

6 Conclusions and future development ................................. 51

A Results of the OD success criteria validation tests ................ 55

B NORAD and COSPAR IDs .................................................. 69

C TLE format ............................................................... 70
List of Figures

1.1 Number of objects tracked by USSTRATCOM[17] ........................................ 8
1.2 Debris orbiting the Earth ........................................................................ 9
2.1 SPOOK functional diagram[7] ................................................................. 11
2.2 SPOOK data logic ................................................................................. 12
2.3 Structure of the Objects catalog DB ...................................................... 14
2.4 Python data model ............................................................................. 19
3.1 Galileo 15 residuals ............................................................................ 33
3.2 LAGEOS 1 residuals .......................................................................... 33
3.3 E-st@r-2 residuals ............................................................................ 33
3.4 SPOT 5 residuals ............................................................................... 34
5.2 Time of flight curves on the $\tau,\xi$ plane[10] ...................................... 48

List of Tables

2.1 Objects catalog table fields .................................................................. 15
2.2 Object data table fields ....................................................................... 15
2.3 Observers DB fields ........................................................................... 16
2.4 Measurements pool fields ................................................................... 17
2.5 Accuracy envelope for various orbital regions [1] ................................. 22
2.6 Accuracy envelope for the two categories[6] ........................................ 22
2.7 Performances as estimated in [9] ......................................................... 23
2.8 Starting catalog ................................................................................ 24
2.9 Measurement from the sensor simulator ............................................. 25
2.10 Correlated measurements .................................................................. 26
2.11 New tracklets correlated T2T .............................................................. 26
2.12 Catalog with updated DebrisOb ........................................ 27
2.13 Catalog with new object myDebris .................................... 27
2.14 Observers DB ............................................................. 27

3.1 Test cases for TLE covariance extraction .......................... 32
3.2 E-st@r-2 covariance matrix ............................................. 34
3.3 GALILEO 15 covariance matrix ....................................... 34
3.4 LAGEOS 1 covariance matrix .......................................... 35
3.5 SPOT 5 covariance matrix .............................................. 35

4.1 Summarised results of validation tests ............................. 42

5.1 Performance comparison between different IOD methods ...... 50

C.1 TLE format for first line ............................................... 70
C.2 TLE format for second line ............................................ 71
Glossary

**BC** Ballistic Coefficient.

**CDF** Cumulative Distribution Function.

**CPF** Consolidated Prediction Format.

**EKF** Extended Kalman Filter.

**FoV** Field of View.

**GCRF** Geocentric Celestial Reference Frame.

**IOD** Initial Orbit Determination.

**JSpOC** Joint Space Operations Command.

**OD** Orbit Determination.

**OEM** Orbit Ephemerides Message.

**RMS** Root Mean Square.

**RTN** Radial-Transverse-Normal.

**SBLS** Sequential Batch Least Squares.

**SoR** System of Reference.

**SRIF** Square Root Information Filter.

**SRPC** Solar Radiation Pressure Coefficient.

**SSA** Space Situational Awareness.

**SST** Space Surveillance and Tracking.
**TDM**  Tracking Data Message.

**TEME**  True Equator Mean Equinox.

**TLE**  Two-Line Element sets.

**Tracklet**  Collection of measurements of a single pass of an object through a sensor.

**UKF**  Unscented Kalman Filter.

**USKF**  Unscented Schmidt-Kalman Filter.

**WLS**  Weighted Least squares.

**WRMS**  Weighted Root Mean Square.
Chapter 1

Introduction

1.1 Background

Ever since the first satellite was placed in orbit (Sputnik 1, 1957), space has been an invaluable asset for humans. However, for a long time expended rocket stages and non-operational satellites have been carelessly abandoned in orbit without any end-of-life disposal strategy. With the increasing number of operations and satellites in orbit during the space age, this has become a major problem, as even small pieces of debris (1 mm) can destroy subsystems of a spacecraft [5], not to mention the danger for humans during extra-vehicular activities (EVAs). This is even more problematic for heavily-used strategic orbital regions such as LEO, where if active removal missions will not be performed there is concrete risk of cascade collisions between operational satellites and debris which would make them unusable for many generations (Kessler syndrome[12]).

The mitigation of this phenomenon is made more difficult by the fact that it is extremely hard to locate and track all these pieces; as an example, the number of catalogued objects by USSTRATCOM (US Strategic Command) as of February 2018 is shown in Figure 1.1 and amounts to roughly 19000 objects. However, ESA estimates that there are more than 170 million pieces of debris larger than 1 mm (Figure 1.2).

In order to tackle this phenomenon, both effective tracking and removal need to be performed (active control), as well as good end-of-life management of expended satellites and rocket stages (in this regard, reusable launchers can play a major role).

This calls for the development and implementation of a Space Surveillance and Tracking (SST) system, which can monitor the region of space around Earth and prevent catastrophic collisions. As of today, the American Joint Space Operations Center (JSpOC) is the only entity that manages a comprehensive SST system called SSN (Space Surveillance Network), with 24 sensors scattered around the globe. Part of the collected data is publicly available (e.g. on http://space-track.org); however, as shown in Sections 2.3.2.2 and 3, it cannot be realistically used for operations.

An operational SST system could be used to provide a variety of services, such as:

- Collision avoidance;
Figure 1.1: Number of objects tracked by USSTRATCOM[17]

- Pass prediction;
- Maneuver detection;
- Visualization and plotting;
- Re-entry analysis;
- Fragmentation detection.

In this framework, SPOOK is introduced as Airbus’ in-house software developed to provide the aforementioned services.
1.2 SPOOK

SPOOK (SPace Objects Observations and Kalman filtering) is a tool developed by Airbus Defence and Space GmbH that can estimate the orbit of objects around the Earth (Orbit Determination, OD) using a variety of techniques (sequential filtering, batch WLS and others, cfr. Section 2).

At the beginning of this work, SPOOK had a variety of calculation capabilities, such as Orbit Determination and Initial Orbit Determination (cfr. Section 2.2) with both Space- and Earth-based observers, Sensor simulation to generate synthetic measurements, generation of an observation plan to track a certain object, propagation of orbit and covariance information \[7\]. However, as explained before, the objective is to build a comprehensive system that can generate and maintain a catalog of objects and provide services to the end user. The following chapters deal with the development of the aforementioned data logic, as well as a number of other improvements implemented in the software, such as:

- Estimation of covariance information from TLEs to use the publicly available JSpOC data;
- Introduction of pass/fail tests that can be run automatically to evaluate the results of an OD process;
- Implementation of a fast solver for the Lambert problem.
Chapter 2

SPOOK API

2.1 SPOOK calculation core

SPOOK was born as Airbus Defence and Space’s orbit determination tool using Kalman Filters (SPace Objects Observation and Kalman filtering). During the years, it evolved to perform a variety of additional tasks, such as:

- Initial Orbit Determination and Orbit Determination;
- Sensor simulation;
- Measurement analysis
- Sensor calibration;
- Correlation;
- State and covariance propagation.

The tool is written in FORTRAN; in the last few years, it was improved in many ways, e.g. by allowing the process of multiple objects in parallel using OpenMP, the analysis of entire statistical populations, and others. The interested reader can refer to [7] for a wider description of the tool. What is relevant for the scope of this thesis is that, at the beginning of this work, SPOOK’s structure was the one showed in Figure 2.1. SPOOK interacts with the external world mainly using three configuration files:

- parameter.ini, which defines all the parameters needed for the SPOOK run: the mode (OD, correlation, sensor simulator, etc.), OD and IOD methods to use, source of the measurements (simulated or from external file), type of output, gravity and atmospheric drag models to use, etc.;

- objects.dat, which contains information about all the object to analyse (initial state vector/orbital elements/TLE, epoch, initial covariance, reference orbit if available, etc.);
• **observers.dat**, which contains information about the observers to use, such as the type (radar or optical), position (ground-based or space-based, with respectively altitude, longitude and latitude, or state vector), accuracy of the sensors, etc.).

After the run, SPOOK outputs a variety of files depending on the mode, which span from coverage analysis and observation plan for measurement analysis, to simulated measurements for sensor simulator, to state vectors and respective covariances in different systems of reference for OD and state/covariance propagation.

However, the objective is to develop a fully-fledged SST system, i.e. a complete framework to of both hardware and software that can be used to track, monitor and catalogue all the objects of interest around Earth, and provide important services both to satellite operators and academia. A complete data logic and workflow has been developed during this thesis, and is presented in the next chapter.

### 2.2 SST architecture

The complete data flow developed during this work is shown in Figure 2.2.

It was decided to build the cataloguing data logic in Python and have it interact with the FORTRAN processing core, as FORTRAN is not really suited to perform complex I/O operations and has almost non-existent database interface libraries. The cataloguing process has no major performance requirements, thus the interpreted nature of Python does not represent a show-stopper; moreover, the possibility to use Object Oriented Programming at
Figure 2.2: SPOOK data logic
this level of the program structure represented a very useful asset, because, as shown later, it hides the internal working of the system ”under the hood”. In Figure 2.2 the blue modules are part of FORTRAN SPOOK code, the yellow blocks are databases and green ones are Python objects implemented during this thesis.

The business data logic represented in the picture can be summarized in the following steps:

- The whole cycle starts when new tracklets (i.e. collections of measurements of a single pass of an object through the field of view of a sensor) are added to the Measurements Pool. These measurements can be either simulated with the sensor simulator for research purposes or obtained by external sources (telescopes, radars, etc.). At the moment of insertion, the tracklets are marked as ”Unprocessed”;

- Now, for each tracklet we want to check if it was originated by one of the objects that are already present in the catalog. This process will be referred to as ”Tracklet-to-catalog correlation” (T2C correlation). If a suitable candidate is found, the tracklet will be marked as ”Correlated to object”, otherwise the system will consider it as belonging to a new, unknown object;

- The system will go through the uncorrelated tracklets and will try to associate the ones belonging to the same (unknown) object; this process is referred to as ”Tracklet-to-tracklet correlation” (T2T correlation). The tracklets that are correlated to one another are marked as ”Correlated T2T”, while the ones that have not been correlated to neither objects or other tracklets are marked as ”Uncorrelated”;

- At this stage, the measurements are actually used: the tracklets that have been correlated to objects are used to perform OD and improve the estimate of state vector and covariance of that object that was previously recorded in the Objects Catalog (the result of the OD is evaluated using the criteria developed in Chapter 4). The tracklets that have been correlated to one another will be used to perform IOD (if there are at least 3) and add a new object to the catalog (if more than 3 tracklets are present, an OD can be already performed right after the IOD);

- After every modification to the catalog, all the objects are propagated for a certain amount of time and their state vectors are saved in a prediction table specific to each object (as shown in Figure 2.3). The prediction table is used to provide the actual services to the user.

All the operations explained above will be detailed in the following subsections, then the Python framework that connects all the blocks together will be illustrated.

2.2.1 The databases

The core serialization structure is of course represented by the databases, which are implemented in PostgreSQL as detailed in 21:
Figure 2.3: Structure of the Objects catalog DB

- **Objects DB**: contains all the objects that have been catalogued. Each object is one line in a table, with the fields detailed in Table 2.1. Every object has also a specific table containing all of its recorded instances (i.e. past state vectors with their covariances) which is referred to as the "data table" (Table 2.2), and a table containing propagated states for a user-defined time which is referred to as "prediction table" (shown in Figure 2.3);

- **Observers DB**: it has a single table containing data about every observer: position (space-based or ground-based), type of sensors, accuracy, bias, etc. The detailed structure is reported in Table 2.3;

- **Measurements Pool**: it is where all the measurements are stored; each line (whose structure is shown in Figure 2.4) is a tracklet, and in a similar fashion to the Objects catalog, every tracklet has a specific table in which all of its measurements are contained, i.e. all the observables of a particular measurement, together with their rates of change, biases and accuracies. Also the state vector of the observer and environmental variables such as temperature, pressure and humidity are stored here for comfort (if available).

### 2.2.2 Sensor simulator

It is basically a research tool; given a set of observers and objects, it generates the measurements that each observer would produce of the objects in a certain time frame.
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object ID</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>NORAD ID</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>COSPAR ID</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Last observed by</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Last state vector</td>
<td>float[6]</td>
<td></td>
</tr>
<tr>
<td>Last BC</td>
<td>float</td>
<td>Ballistic Coefficient</td>
</tr>
<tr>
<td>Last SRPC</td>
<td>float</td>
<td>Solar Radiation Pressure Coefficient</td>
</tr>
<tr>
<td>SoR of the state vector</td>
<td>enum</td>
<td></td>
</tr>
<tr>
<td>Epoch</td>
<td>timestamp</td>
<td></td>
</tr>
<tr>
<td>Simulated</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>WRMS</td>
<td>float</td>
<td>Weighted RMS (see Chapter 4)</td>
</tr>
<tr>
<td>Total number of instances</td>
<td>int</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Objects catalog table fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object ID</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>NORAD ID</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>COSPAR ID</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Observed by</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>State vector</td>
<td>float[6]</td>
<td>Ballistic Coefficient</td>
</tr>
<tr>
<td>BC</td>
<td>float</td>
<td>Solar radiation pressure Coefficient</td>
</tr>
<tr>
<td>SRPC</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>Epoch</td>
<td>timestamp</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>float[8x8]</td>
<td>Covariance matrix</td>
</tr>
<tr>
<td>SoR of the state vector</td>
<td>enum</td>
<td>Weighted RMS (see Chapter 4)</td>
</tr>
<tr>
<td>WRMS</td>
<td>float</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: Object data table fields
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>string</td>
<td>One of 7 kinds: optical or radar with different observables</td>
</tr>
<tr>
<td>Type of observer</td>
<td>enum</td>
<td>Either space-based or ground-based</td>
</tr>
<tr>
<td>Epoch</td>
<td>timestamp</td>
<td>SoR of the observer</td>
</tr>
<tr>
<td>Observer base</td>
<td>enum</td>
<td>Either accessibility, tracking or surveillance</td>
</tr>
<tr>
<td>Reference frame</td>
<td>enum</td>
<td>Filled only for ground-based observers</td>
</tr>
<tr>
<td>Pointing mode</td>
<td>enum</td>
<td>Filled only for space-based observers</td>
</tr>
<tr>
<td>Latitude, longitude, altitude</td>
<td>float</td>
<td>Bias and sensor accuracy for each observable measurable by the sensor (one or more among declination, right ascension, azimuth, elevation, range, range-rate)</td>
</tr>
<tr>
<td>State vector</td>
<td>float[6]</td>
<td></td>
</tr>
<tr>
<td>Sensor properties</td>
<td>float[]</td>
<td></td>
</tr>
<tr>
<td>Shape of View</td>
<td>string</td>
<td>Type of sensor field (rectangular, circular, File-defined, azimuth-elevation fence)</td>
</tr>
<tr>
<td>Field of View</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>Orbital elements</td>
<td>float[6]</td>
<td>Only for space-based observers</td>
</tr>
<tr>
<td>Simulated</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>Total number of instances</td>
<td>int</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.3: Observers DB fields
### Table 2.4: Measurements pool fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracklet ID</td>
<td>string</td>
<td>ID of the observer (Table 2.3 that generated the tracklet)</td>
</tr>
<tr>
<td>Observer</td>
<td>string</td>
<td>ID of the observer (Table 2.3 that generated the tracklet)</td>
</tr>
<tr>
<td>Correlated</td>
<td>enum</td>
<td>Can be either ”Unprocessed”, ”Correlated to object”, ”Correlated T2T” as detailed before</td>
</tr>
<tr>
<td>Object ID</td>
<td>string</td>
<td>If the tracklet is correlated to an object in the catalog, it will contain the ID of the object</td>
</tr>
<tr>
<td>Simulated</td>
<td>boolean</td>
<td>True if the tracklet is synthetic, false otherwise</td>
</tr>
<tr>
<td>Origin</td>
<td>string</td>
<td>If the tracklet is synthetic, contains the object that generated it</td>
</tr>
<tr>
<td>Correlated tracklets</td>
<td>string</td>
<td>Contains the ID of all the tracklets that have been correlated to it if it is marked ”Correlated T2T”</td>
</tr>
<tr>
<td>Total number of measurements</td>
<td>integer</td>
<td>Number of measurements contained in the tracklet</td>
</tr>
</tbody>
</table>

#### 2.2.3 Correlation

As detailed before, correlation is the process of associating a tracklet to an object or to another tracklet. At the time of writing, Tracklet-to-catalog correlation is implemented and integrated, while Tracklet-to-tracklet is still work in progress.

#### 2.2.4 Initial Orbit Determination

The Initial Orbit Determination (IOD) is performed when a new object is added to the catalog; it provides a first, coarse estimation of the state vector of an object. Various algorithms are available in SPOOK to perform IOD, as detailed in Chapter 5, of which one was added during this thesis; at least 3 tracklets are needed to perform IOD (even though the last one can work with only two). FORTRAN SPOOK returns a txt file containing the generated state vector, which will then be processed and added to the database.

#### 2.2.5 Orbit Determination

Orbit Determination (OD) consists in improving an estimate of the state vector of an object when a set of measurements is provided. Various algorithms are available, which can be either based on least squares method (which concentrate information from all the measurements in one point) or filters (which sequentially use each measurement to improve the state vector).
The available algorithms are the following:

- Extended Kalman Filter;
- Unscented Kalman Filter;
- Unscented Schmidt-Kalman Filter;
- Square root information Filter;
- Weighted Least Squares;
- Sequential Batch Least Squares.

They were not a central part of the present work, so they will not be further discussed. The interested reader can find more information in [16]. What is important for the scope of this thesis is that at the end of an OD SPOOK returns a JSON file containing the improved state vector, some metrics of the quality of the process (see Chapter 4) and ID information of the object. This JSON file is processed (see 2.2.7) to modify the catalog accordingly.

### 2.2.6 Covariance propagation

Once a state vector and covariance of an object are available, they are propagated for a certain amount of time, i.e. their values for a certain period of the future are predicted and used to provide services. The time for which state vector and covariance can be propagated depends on the services that will be provided; generally speaking, the covariance must be inside certain limits in order to be used to provide services, thus a literature research was carried out to assess what this threshold is for various applications. The results of this research are summarized in Section 2.3.

### 2.2.7 Python framework logic

The basic catalog maintenance operations listed in the paragraphs above (sensor simulator, IOD, OD, Correlation, Covariance propagation) are referred to as use cases. Performing and serializing the results of these operations used to be a very tedious task, as all of the involved entities (objects, observers, tracklets, etc.) can be expressed in many different formats. Moreover, the end user would need to be familiar with all these formats and be aware of the source of the data in order to effectively use the system. This calls for the creation of a middle layer of abstraction that takes care of the interaction between FORTRAN SPOOK, the databases and the external world and can become the basic interface library upon which the services are implemented. The implementation and validation of this abstraction layer was a major part of this thesis work and represent the cornerstone of the integrated SST system that uses SPOOK as main calculation engine.

This integration layer was implemented in Python and is summarized in Figure 2.4.
Figure 2.4: Python data model
The central block represents the very foundation of the library: the structures that are communicated between the different parts of the system are instances of those classes (as shown also in the main business logic in Figure 2.2). The main advantage of this structure is that the services can be built upon these classes, without caring about the actual origin of the contained (sensors, files, databases, etc.). The relationship between the types is the following:

- A Space object contains a series of Time instances, i.e. a collection of all the recorded instances of the object. Each instance is basically a state vector/orbital elements with its covariance and time. If the Time instances are generated from TLE files, it is also possible to estimate a covariance matrix as shown in Chapter 3;

- A similar relationship occurs between Measurement-Tracklet-Measurements pool: a measurement pool is a collection of sorted tracklets, which are collection of measurements. The measurements usually contain data about the entity that generated them, i.e. the state vector of the observer if it is space based or its geographical coordinates (latitude-longitude-altitude) if it is ground based. In the latter case, some environmental variables such as humidity and temperature (which can affect the performances of some sensors and are modelled in SPOOK) are also recorded;

- The Observer type contains information about the type of observer (radar or optical), characteristics of the sensors (bias and accuracy), location, etc.

- The basic numerical quantities are coded in the State_vector, Orbital_elements and Covariance classes, which also contain some useful methods to switch between different Systems of reference.

The left and right blocks of the diagram are the I/O structures used to interact with the external world and FORTRAN SPOOK (via FileIO) and the databases (with DatabaseIO). DatabaseIO contains three classes to interact with the three databases; each class implements CRUD operations (Create, Read, Update, Delete) to/from the databases which use the core types.

The classes of the FileIO package take care of the I/O from/to files; the main formats used in the SST community are supported, together with the format of the files specified in Section 2.4 used in FORTRAN SPOOK, namely:

- Objects:
  - OEM (Orbit Ephemerides Message);
  - SPOOK JSON;
  - CPF (Consolidated Prediction Format), used by Satellite Laser Ranging applications;
  - SPOOK output;
- SPOOK objects.dat (Section 2.1);
- STK output;
- TLE files (see Appendix C).

- Observers:
  - SPOOK observers.dat.

- Tracklets:
  - OTDF, mainly used with ground-based observers;
  - TDM (Tracking Data Message), which is a CCSDS standard.

### 2.3 Cataloguing requirements

#### 2.3.1 Introduction

Below are summarized the requirements (in terms of covariance matrix) that an orbit determination must have in order to be used for the various services provided by the cataloguing system, e.g.:

- Basic catalogue maintenance;
- Collision avoidance;
- Sensor tasking.

The requirements for cataloguing maintenance are mainly found in the unclassified ESA document that contains the requirement matrix for an integrated European SST system in the framework of ESA’s activities in SSA (Space Situational Awareness). Sensor tasking policies are to be determined, but some interesting studies about dynamic sensor tasking using Markov decision processes have been performed. However, they go beyond the scope of this thesis and will not be further examined.

As for the collision avoidance service, the accuracy needed is of course dependent on two objects, so an accuracy requirement for any space debris alone cannot be defined (and it would not make much sense anyway). Some techniques that are used in practice are explained in the following sections.

#### 2.3.2 Requirements

##### 2.3.2.1 Catalogue maintenance

The main requirements, as stated before, come from ESA. The SSA/SST team introduced the concept of *accuracy envelope*, i.e. the "size" that the covariance ellipsoid should not
exceed to be inserted into the catalogue. The thresholds come mostly from ESA operational experience\[1\]. The cataloguing system should warn the user 48 hours before the computed covariance matrix at any given epoch time violates this accuracy envelope. The size of the accuracy envelope for each orbital region is given in Table 2.5.

Despite the fact that the SST segment must provide the full covariance matrix (in the OCRF SoR), as per requirement SST-SRD-108, the velocity components of the matrix will not be part of the accuracy envelope.

### 2.3.2.2 Collision avoidance

A network of optical and radar sensors to monitor the upper LEO region has been proposed to ESA SSA segment by a consortium of Italian entities: company Carlo Gavazzi Space (formerly CGS, now OHB Italy) and the Italian Institute for Astrophysics (INAF) with the Italian Research Council (CNR)\[6\]. Regarding Orbit Determination, the following accuracy envelope has been considered, following ”collision avoidance requirements”\[6\]. Different requirements are considered for objects resident or transient in the upper LEO region (perigee between 1100 km-2000 km). The sizes are given in the object-centered frame \{u, v, w\}, where:

- **u**: direction from Earth center to object center;
- **w**: direction of the angular momentum vector \( h \);
- **v = w \times u**

<table>
<thead>
<tr>
<th>Orbital region</th>
<th>Radial (m)</th>
<th>Along-track (m)</th>
<th>Cross-track (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEO</td>
<td>40</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>MEO</td>
<td>600</td>
<td>3000</td>
<td>1500</td>
</tr>
<tr>
<td>GEO</td>
<td>2500</td>
<td>2500</td>
<td>2500</td>
</tr>
</tbody>
</table>

Table 2.5: Accuracy envelope for various orbital regions \[1\]

As an explanatory real case, it is possible to analyze the the CSM (Conjunction Summary Message) issued by JSpOC to EUMETSAT regarding MetOp-A (a meteorological satellite in Sun-synchronous orbit at an altitude of 800 km). A high-risk conjunction happened on May 1st 2011; the first CSM was issued on 26th April, reporting a covariance for the satellite
of (5, 160, 3) m in the radial, along-track and cross track directions respectively, whereas the offending object (a piece of COSMOS 2251, NORAD ID #34451) was known with a covariance ellipsoid of (17, 2375, 10)[20]. The last report was received late on 30th April 2011, which contained positions with a covariance of (9,205,8) m for the debris and (3, 15, 2) m for the object, in the radial, along-track and cross track directions respectively.

**AAS/AIAA Space Flight Mechanics Meeting**

From the academic world, R. Gottlieb et al. showed that if the orbit determination quality is not "exceptionally good"[9], collision avoidance maneuvers are not advisable, as the probability of collision does not decrease by an appreciable amount if "reasonable maneuvers" are considered. The researchers considered the performances in Table 2.7 for various OD systems and proved that radar tracking data has too big of an uncertainty to be trusted with regard to the planning of an avoidance maneuver. The semi-major axis sigma can be calculated by the UVW covariance.

<table>
<thead>
<tr>
<th>OD Source</th>
<th>3σ position error (m)</th>
<th>Error growth rate (m/day)</th>
<th>Semi-major axis σ (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal ephemeris</td>
<td>30</td>
<td>200</td>
<td>1.5</td>
</tr>
<tr>
<td>Improved internal</td>
<td>20</td>
<td>20</td>
<td>0.15</td>
</tr>
<tr>
<td>ephemeris</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Laser tracking</td>
<td>10</td>
<td>700</td>
<td>5</td>
</tr>
<tr>
<td>Radar tracking</td>
<td>500-1000</td>
<td>5000-15000</td>
<td>37</td>
</tr>
</tbody>
</table>

Table 2.7: Performances as estimated in [9]

According to him, only laser tracking and improved internal ephemeris data can provide enough quality to be advised in the planning of an avoidance maneuver.

### 2.4 Test run of the SST system

Now that all the pieces of the system have been illustrated, a test run of the cycle represented in Figure 2.2 will be shown, in order to prove its feasibility. Mostly synthetic data are used. The starting point will be a catalog of objects read from TLE files downloaded from [http://space-track.org](http://space-track.org) from different orbital regions, plus a simulated object, as shown in Table 2.8.
<table>
<thead>
<tr>
<th>Object ID</th>
<th>Object Name</th>
<th>NORAD ID</th>
<th>COSPAR ID</th>
<th>Last Observed By</th>
<th>Last Observed Time</th>
<th>Last Epoch Time</th>
<th>[r, l]</th>
<th>[r, j]</th>
<th>[r, k]</th>
<th>[v, l]</th>
<th>[v, j]</th>
<th>[v, k]</th>
<th>BC</th>
<th>SRC</th>
<th>Simulated</th>
</tr>
</thead>
<tbody>
<tr>
<td>spook_01</td>
<td>DebrisOb</td>
<td>0</td>
<td>0</td>
<td>8/5/15 3:44</td>
<td>97330.02</td>
<td>-9083.1</td>
<td>-655.065</td>
<td>3.938562</td>
<td>-1.89112</td>
<td>0.033211</td>
<td>0.044</td>
<td>0.022</td>
<td>TRUE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spook_06</td>
<td>DELTA 1 R/B</td>
<td>50</td>
<td>60009B</td>
<td>3/21/16 13:48</td>
<td>43:29.8</td>
<td>-2905.53</td>
<td>-7337.35</td>
<td>-4.0589</td>
<td>4.524833</td>
<td>1.74646</td>
<td>5.245717</td>
<td>-0.00289</td>
<td>0</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>spook_07</td>
<td>ECHO 1 DB</td>
<td>51</td>
<td>60009C</td>
<td>3/21/16 20:40</td>
<td>40:49.3</td>
<td>8020.491</td>
<td>-799.692</td>
<td>-1.69545</td>
<td>0.473981</td>
<td>4.724381</td>
<td>5.144274</td>
<td>0</td>
<td>FALSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spook_01</td>
<td>EXPLORER 7</td>
<td>22</td>
<td>59009A</td>
<td>3/22/16 20:20</td>
<td>20:07.5</td>
<td>-6371.13</td>
<td>2907.63</td>
<td>-3.10623</td>
<td>-1.90102</td>
<td>-4.41621</td>
<td>5.759129</td>
<td>0.001169</td>
<td>0</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>spook_04</td>
<td>SOLRAD 1 G</td>
<td>46</td>
<td>60007B</td>
<td>3/22/16 21:14</td>
<td>14:37.1</td>
<td>-396.144</td>
<td>-7235.04</td>
<td>-0.89344</td>
<td>2.898105</td>
<td>-0.22288</td>
<td>6.748351</td>
<td>0.001012</td>
<td>0</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>spook_05</td>
<td>THOR ABLEST</td>
<td>47</td>
<td>60007C</td>
<td>3/21/16 20:54</td>
<td>54:28.4</td>
<td>-2228.72</td>
<td>-7146.98</td>
<td>-1.302</td>
<td>6.080061</td>
<td>-2.0284</td>
<td>3.485145</td>
<td>0.000577</td>
<td>0</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>spook_02</td>
<td>TIROS 1</td>
<td>29</td>
<td>60002B</td>
<td>3/23/16 2:30</td>
<td>30:07.1</td>
<td>4561.879</td>
<td>5345.932</td>
<td>-0.67602</td>
<td>3.81991</td>
<td>2.23907</td>
<td>5.63362</td>
<td>0.000475</td>
<td>0</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>spook_03</td>
<td>TRANSIT 2A</td>
<td>45</td>
<td>60007A</td>
<td>3/22/16 13:47</td>
<td>47:00.7</td>
<td>-5298.36</td>
<td>4539.272</td>
<td>-5.2554</td>
<td>-1.95703</td>
<td>-2.30178</td>
<td>7.032873</td>
<td>0.000415</td>
<td>0</td>
<td>FALSE</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.8: Starting catalog
Table 2.9: Measurement from the sensor simulator

<table>
<thead>
<tr>
<th>Tracklet ID</th>
<th>Observer ID</th>
<th>Correlated</th>
<th>Object ID</th>
<th>Simulated</th>
<th>Origin</th>
<th>Total Number of Measurements</th>
</tr>
</thead>
<tbody>
<tr>
<td>newtrk_01</td>
<td>eEyE</td>
<td>Unprocessed</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>newtrk_010</td>
<td>eEyE</td>
<td>Unprocessed</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>newtrk_011</td>
<td>eEyE</td>
<td>Unprocessed</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>newtrk_02</td>
<td>eEyE</td>
<td>Unprocessed</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>newtrk_03</td>
<td>eEyE</td>
<td>Unprocessed</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>newtrk_04</td>
<td>eEyE</td>
<td>Unprocessed</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>newtrk_05</td>
<td>eEyE</td>
<td>Unprocessed</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>newtrk_06</td>
<td>eEyE</td>
<td>Unprocessed</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>newtrk_07</td>
<td>eEyE</td>
<td>Unprocessed</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>newtrk_08</td>
<td>eEyE</td>
<td>Unprocessed</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>newtrk_09</td>
<td>eEyE</td>
<td>Unprocessed</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>23</td>
<td></td>
</tr>
</tbody>
</table>

As shown in the Table, all of the objects have only one instance as they are read from only one source; the simulated object (DebrisOb) does not have NORAD or COSPAR IDs and has a flag to mark the fact that it is simulated.

At this stage, a sensor simulation is run with the network of sensors shown in Figure 2.14 in order to use the synthetic measurements to improve the state vector of the simulated object via an OD. Of course in a real world scenario the measurements will come from an actual sensor, but this is only to show the capabilities of the system: SPOOk will output some measurements in TDM format which will be added to the measurement pool (shown in Table 2.9) as "Unprocessed", then a Tracklet-to-catalog correlation will be attempted against all the objects in the catalog, resulting in a positive correlation with DebrisOb (as shown in Table 2.10) and thus they will be used to improve the state vector of DebrisOb, which will now contain another instance and its epoch time will be updated accordingly (Table 2.12). Then, given that Tracklet-to-tracklet correlation is not implemented yet, some tracklets belonging to an object called "myDebris" were manually added to the database and marked as correlated with each other, as shown in Table 2.11. These tracklets are used to generate a new object called "myDebris" in the database, so that the catalog is updated as in Table 2.13.
### Table 2.10: Correlated measurements

<table>
<thead>
<tr>
<th>Tracklet ID</th>
<th>Observer ID</th>
<th>Correlated</th>
<th>Object ID</th>
<th>Simulated</th>
<th>Origin</th>
<th>Total Number of Measurements</th>
</tr>
</thead>
<tbody>
<tr>
<td>newtrk_01</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>23</td>
</tr>
<tr>
<td>newtrk_010</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_011</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>23</td>
</tr>
<tr>
<td>newtrk_02</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_03</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_04</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>23</td>
</tr>
<tr>
<td>newtrk_05</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_06</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_07</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_08</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_09</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td>DebrisOb</td>
<td>23</td>
</tr>
</tbody>
</table>

### Table 2.11: New tracklets correlated T2T

<table>
<thead>
<tr>
<th>Tracklet ID</th>
<th>Observer ID</th>
<th>Correlated</th>
<th>Object ID</th>
<th>Simulated</th>
<th>Correlated tracklets</th>
<th>Origin</th>
<th>Total Number of Measurements</th>
</tr>
</thead>
<tbody>
<tr>
<td>newtrk_01</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td></td>
<td>DebrisOb</td>
<td>23</td>
</tr>
<tr>
<td>newtrk_010</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td></td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_011</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td></td>
<td>DebrisOb</td>
<td>23</td>
</tr>
<tr>
<td>newtrk_012</td>
<td>eEyE</td>
<td>Correlated T2T</td>
<td>myDebris</td>
<td>FALSE</td>
<td>newtrk_012</td>
<td>myDebris</td>
<td>23</td>
</tr>
<tr>
<td>newtrk_013</td>
<td>eEyE</td>
<td>Correlated T2T</td>
<td>myDebris</td>
<td>FALSE</td>
<td>newtrk_012</td>
<td>myDebris</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_014</td>
<td>eEyE</td>
<td>Correlated T2T</td>
<td>myDebris</td>
<td>FALSE</td>
<td>newtrk_012</td>
<td>myDebris</td>
<td>23</td>
</tr>
<tr>
<td>newtrk_015</td>
<td>eEyE</td>
<td>Correlated T2T</td>
<td>myDebris</td>
<td>FALSE</td>
<td>newtrk_012</td>
<td>myDebris</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_016</td>
<td>eEyE</td>
<td>Correlated T2T</td>
<td>myDebris</td>
<td>FALSE</td>
<td>newtrk_012</td>
<td>myDebris</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_017</td>
<td>eEyE</td>
<td>Correlated T2T</td>
<td>myDebris</td>
<td>FALSE</td>
<td>newtrk_012</td>
<td>myDebris</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_018</td>
<td>eEyE</td>
<td>Correlated T2T</td>
<td>myDebris</td>
<td>FALSE</td>
<td>newtrk_012</td>
<td>myDebris</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_019</td>
<td>eEyE</td>
<td>Correlated T2T</td>
<td>myDebris</td>
<td>FALSE</td>
<td></td>
<td>myDebris</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_02</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td></td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_020</td>
<td>eEyE</td>
<td>Correlated T2T</td>
<td>myDebris</td>
<td>FALSE</td>
<td>newtrk_012</td>
<td>myDebris</td>
<td>23</td>
</tr>
<tr>
<td>newtrk_021</td>
<td>eEyE</td>
<td>Correlated T2T</td>
<td>myDebris</td>
<td>FALSE</td>
<td>newtrk_012</td>
<td>myDebris</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_022</td>
<td>eEyE</td>
<td>Correlated T2T</td>
<td>myDebris</td>
<td>FALSE</td>
<td>newtrk_012</td>
<td>myDebris</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_03</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td></td>
<td>DebrisOb</td>
<td>23</td>
</tr>
<tr>
<td>newtrk_04</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td></td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_05</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td></td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_06</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td></td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_07</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td></td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_08</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td></td>
<td>DebrisOb</td>
<td>24</td>
</tr>
<tr>
<td>newtrk_09</td>
<td>eEyE</td>
<td>Correlated</td>
<td>DebrisOb</td>
<td>TRUE</td>
<td></td>
<td>DebrisOb</td>
<td>23</td>
</tr>
</tbody>
</table>
Table 2.12: Catalog with updated DebrisOb

| Object ID | Object Name | NORAD ID | COSPAR ID | Last Observed By | Last Observed Time | Last Observed Epoch | (r)  | (r)  | (r)  | (r)  | (r)  | BC | SRC | Simulated WRMS | No. Instances | Sub
|-----------|-------------|----------|-----------|------------------|--------------------|---------------------|------|------|------|------|------|----|-----|--------------|--------------|-----
| sPock_09  | COURIER 18  | 58       | 40010A    | 3/12/16 6:55    | 55188.8             | 4889.372           | 2062.097 | -2.63868 | -2.52372 | 5.812533 | 3.472837 | 3.978-03 | 0  | FALSE | 1 | TEME
| sPock_01  | DebrisOb    | 6        | 0         | 0                | 0                  | 0                   | 0      | 0     | 0    | 0    | 0    |    |     |              |              |     |
| sPock_02  | DELTA 1 KB  | 56       | 40000B    | 3/12/16 13:43   | 401610059           | 316160938           | -18397 | -555115 | -7434192 | 3.834167 | 0.071938 | 0.044 | 0  | FALSE | 1 | TEME
| sPock_07  | ECHO 1 DEB  | 51       | 60000C    | 3/11/16 10:40   | 408003354           | 796992915           | -69545  | 0.479988 | 4.724381 | 5.144274 | 0  | FALSE | 2 | TEME
| sPock_27  | EXPLORER 7  | 27       | 50000D    | 3/12/16 20:28   | 967751307           | 437313312           | -36823  | -810002 | -4.106184 | 2.797015 | 0.011109 | 0  | FALSE | 1 | TEME
| sPock_04  | GIBSUB 2 (B) | 46       | 40000B    | 3/12/16 21:14   | 147317           | 394941625           | -85538  | 1.344683 | 5.245717 | -0.002981 | 0  | FALSE | 1 | TEME
| sPock_05  | THOR ABLEST | 47       | 60000C    | 3/12/16 20:54   | 54288           | 222677 .1         | -174698  | 0.3202 | 0.090061 | 0.20284 | 3.445146 | 0.000777 | 0  | FALSE | 2 | TEME
| sPock_02  | TRANSIT 2A  | 29       | 60000B    | 3/12/16 2:30    | 30171           | 465187530          | -679016  | 0.83991 | 0.239527 | 0.638262 | 0.000875 | 0  | FALSE | 2 | TEME
| sPock_00  | TRANSIT 2A  | 45       | 600010A   | 3/12/16 13:47   | 479071          | -5298.365          | 4153.272 | -5.2554 | -1.95703 | 2.00716 | 7.002875 | 0.000415 | 0  | FALSE | 1 | TEME

Table 2.13: Catalog with new object myDebris

Table 2.14: Observers DB
This dry run of the system shows that already at this stage a working prototype of a complete SST system is available. The Python abstraction layer makes it completely transparent to the user, whom is able to run the system without having to care about the underlying structure. A point worth highlighting is that running this demo did not require any intervention from the user (apart from the insertion of "fake" T2T correlated tracklets, given that this capability is not yet available), but the whole cycle in Figure 2.2 was carried out automatically by the software; the Python framework takes care of selecting objects and observers and writing the appropriate configuration files for FORTRAN SPOOK, running it and collecting the outputs. Also the data and prediction tables for each object were automatically generated, but they were not reported in this section for clarity. This shows that this basic implementation is already a solid starting point for the development of a commercially usable SST system.
Chapter 3

Estimation of covariance information from TLEs

3.1 Introduction

The US military force (NORAD department) shares orbit information about all the non-classified object in the JSpOC catalogue (ca. 10000 man-made object with size \( \geq 10 \) cm. This information is publicly available on a few websites (http://space-track.org and http://celestrak.com above all).

The orbit information is essentially constituted by the mean orbital elements, first and second derivative of the mean motion and a peculiar drag term called \( B^* \), whose meaning is explained later. These orbital elements are disclosed in the so-called TLE format (Two-Line Element sets), which dates back to the punch cards era and whose full specifications are reported in Appendix C. TLE data sets are issued daily and their usage, as pointed out in [23], is strictly linked to SGP4, the propagator used for their generation (Simplified General Perturbations 4). Despite the equations of the model were known for more than ten years (see for example [13]), it became popular only with TLEs. The reason for this lies in the fact that the SGP4 model calculates ”mean” orbital elements, i.e. it removes some periodic oscillations. In order to generate meaningful predictions from TLEs, these variations must be reconstructed by the prediction model in the same way they were removed, explaining the dependency of TLEs on SGP4. As mentioned in Section [2.3.2.2] these publicly available TLEs cannot be used to provide services (e.g. collision avoidance), which is why JSpOC provides more accurate data to the entities that request it; this is generated with a more reliable (yet computationally expensive) propagator which takes into account special perturbations (SP).

There are a variety of issues connected to the use of TLEs for sensitive applications, such as:

- It is not exactly clear which version of SGP4 is being used to generate current TLEs [23]; Vallado and others published versions of the source code specifically designed to comply with the ’official’ TLE predictions, but other than that little is known about the internal workings of the JSpOC procedures. It is worthy pointing out that this situation is the
result of years of uncoordinated development, to the point that some departments of the US military ended up paying to buy parts of their own code from third party sources[23];

- When a TLE is propagated and a state vector is generated, it is expressed in a ECI (Earth Centered Inertial) frame called TEME (True Equator Mean Equinox). While it is very hard to find an operational definition of this reference frame[23], some algorithms are available to convert it to more commonly used frames such as GCRF (even at the cost of losing some accuracy);

- The same problem applies to the date propagation: there is a ”TEME of date” and a ”TEME of epoch”. In the latter, the epoch of the TEME frame is held constant, so ”Subsequent rotation matrices must therefore account for the change in precession and nutation from the epoch of the TEME frame to the epoch of the transformation”[23], whereas in the former the TEME frame epoch is always the same as the epoch of the associated ephemeris. While the majority of researchers believe that the ”of date” formulation is correct, the lack of official confirmation has given rise to confusion on the matter[23];

- No accuracy information is provided with TLEs.

The last is a very important point, as sensitive decisions in collision avoidance situations depend greatly on the accuracy of the orbit information, as shown in section 2.3.2.2. In this section of the work, an algorithm to extract covariance information from a certain number of TLEs is explained and the validity of the results is assessed. The following is assembled starting from [18].

3.2 Theoretical background

The covariance matrix can be described intuitively as an extension of the concept of variance to multidimensional variables. Let us consider a random Gaussian 1D variable $x$; it can be shown that, given a large number of unbiased measurements $N$ with true random error $e_i$, the following properties hold:

- $E(x) = x_0$, where $E$ is the expectation operator and $x_0$ is the true value of the variable;
- The error is truly random, so $E(e) = 0$;
- The average of the error squared is known as variance, i.e. $E(e^2) = \sigma^2$

These concepts can be extended to a set of jointly-distributed random variables, defining the covariance:

$$
cov(X, Y) = E[(X - E(X))(Y - E(Y))]$$  \hspace{1cm} (3.1)
In this context, the covariance expresses the tendency of variables $X$ and $Y$ to be linearly correlated; the variance can then be considered as the covariance of a variable with itself:

$$\sigma^2 = \text{cov}(X, X) \quad (3.2)$$

Considering now a vector $X = [X_1, ..., X_N]$ of random variables with finite variance, the **covariance matrix** $P$ can be defined as:

$$P = E[(X - m)(X - m)^T] \quad (3.3)$$

where $m = E(X)$, remembering that the expected value of a vector is the vector of the expected values of its components. The similarity between 3.2 and 3.3 is evident. The $N \times N$ matrix $P$ expresses the covariance between the $N$ variables of $X$:

$$P_{ij} = \text{cov}(X_i, X_j) = E[(X_i - E(X_i))(X_j - E(X_j))] \quad (3.4)$$

From this definition, the elements on the main diagonal are the variances of the components of $X$. Thus, if we consider the covariance matrix of the 3D position of an object in space, it is clear that it is strictly linked with the probability of finding the object in a region called the **covariance ellipsoid**, whose size and size and shape are related to the eigenvalues and eigenvectors of the matrix.

### 3.3 Estimation of a covariance matrix from TLEs

The following algorithm is a variation from [18].

Given that TLEs do not include any kind of information about accuracy, covariance information is extracted in a purely statistical manner. TLEs for a period of time between 15 days and one month are downloaded from the sources mentioned above for the considered object; the last TLE is the one at whose epoch the covariance will be calculated and will be called ”prime”. Then, using SGP4, all the older TLEs are propagated to the epoch of the last TLE, so that there is a statistical pool which is assumed to be Gaussian distributed. The TLE at the latest epoch is assumed to express the ”observed” state vector (i.e. the most accurate); this is a major assumption, but nonetheless it is the most logical one. As mentioned before, the state vectors derived from TLEs are expressed in the TEME reference frame. For validation purposes, it is useful to have it also in the RTN frame, in which the R component is directed from the planet to the object, the N component is perpendicular to the orbital plane (i.e. in the direction of the angular momentum) and T completes the right-handed triad. So, the conversion for the prime is performed and the rotation matrix $T_{RTN}$ is saved. Then, the ”residual” between each propagated TLE and the prime is calculated:

$$\delta \vec{r} = \vec{r}_{\text{observed}} - \vec{r}_{\text{calculated}}$$
$$\delta \vec{v} = \vec{v}_{\text{observed}} - \vec{v}_{\text{calculated}} \quad (3.5)$$
Then, the residuals are converted to the RTN reference frame:
\[
\delta \vec{r}_{RTN} = T_{RTN} \delta \vec{r}_{TEME} \\
\delta \vec{v}_{RTN} = T_{RTN} \delta \vec{v}_{TEME}
\]

Recalling equation (3.3), for the state vector \( \vec{x} = [\vec{r}, \vec{v}] \) the formulation becomes:
\[
P_{epoch} = E[(\delta \vec{x} - \vec{m})(\delta \vec{x} - \vec{m})^T]
\]

where \( \vec{m} \) is the mean of the residuals:
\[
\vec{m} = \frac{1}{N-1} \sum_{i=1}^{N-1} (\delta \vec{x})_i
\]

From (3.7) follows that:
\[
P_{epoch} = \frac{1}{N-1} \sum_{i=1}^{N-1} (\delta \vec{x} - \vec{m})_i(\delta \vec{x} - \vec{m})_i^T
\]

The RTN covariance matrix can be later expressed in the ECI (TEME) frame using the rotation matrix calculated before:
\[
P_{ECI} = T^T P_{RTN} T
\]

### 3.4 Test cases

The algorithm explained above was validated using objects in different orbital regimes. For each of them, one month of TLEs was used and state vector residuals and covariance matrix were generated. The following objects were analyzed:

<table>
<thead>
<tr>
<th>Name</th>
<th>NORAD ID</th>
<th>Semimajor axis (km)</th>
<th>Inclination (deg)</th>
<th>Eccentricity</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAGEOS 1</td>
<td>8820</td>
<td>12271</td>
<td>109.83</td>
<td>0.0044560</td>
</tr>
<tr>
<td>GALILEO 15</td>
<td>41859</td>
<td>29600</td>
<td>54.6</td>
<td>0.006531</td>
</tr>
<tr>
<td>E-st@r-2</td>
<td>41459</td>
<td>6928</td>
<td>1.71</td>
<td>0.0171534</td>
</tr>
<tr>
<td>SPOT 5</td>
<td>27421</td>
<td>7210</td>
<td>98.7</td>
<td>0.0135193</td>
</tr>
</tbody>
</table>

Table 3.1: Test cases for TLE covariance extraction

There are objects in LEO and MEO, both in polar and equatorial orbits. This allows to study the influence of atmospheric drag on the accuracy of TLEs, as well as the role that inclination plays. All of the objects were propagated for one month, between 15/05/2018 and 15/06/2018.

### 3.5 Results and discussion

The residuals plot trend for each object is shown in Figures 3.1a to 3.4b.
Figure 3.1: Galileo 15 residuals

(a) Galileo 15 position residuals  
(b) Galileo 15 velocity residuals

Figure 3.2: LAGEOS 1 residuals

(a) Lageos 1 position residuals  
(b) Lageos 1 velocity residuals

Figure 3.3: E-str@r-2 residuals

(a) E-str 2 position residuals  
(b) E-str 2 velocity residuals
The covariance matrices for each object are reported in Tables 3.2 to 3.5

Table 3.2: E-st@r-2 covariance matrix

<table>
<thead>
<tr>
<th></th>
<th>R (km)</th>
<th>T (km)</th>
<th>N (km)</th>
<th>R (km/s)</th>
<th>T (km/s)</th>
<th>N (km/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R (km)</td>
<td>0.0664</td>
<td>8.1273</td>
<td>0.0055</td>
<td>-0.0087</td>
<td>-3.5740e-05</td>
<td>-7.7982e-06</td>
</tr>
<tr>
<td>T (km)</td>
<td>8.1273</td>
<td>1171.8851</td>
<td>1.8187</td>
<td>-1.2581</td>
<td>-0.0026</td>
<td>-0.0023</td>
</tr>
<tr>
<td>N (km)</td>
<td>0.0055</td>
<td>1.8187</td>
<td>0.0209</td>
<td>-0.0019</td>
<td>2.411e-05</td>
<td>-2.019e-05</td>
</tr>
<tr>
<td>R (km/s)</td>
<td>-0.0087</td>
<td>-1.2581</td>
<td>-0.0019</td>
<td>0.0013</td>
<td>2.7957e-06</td>
<td>2.5249e-06</td>
</tr>
<tr>
<td>T (km/s)</td>
<td>-3.5740e-05</td>
<td>-0.0026</td>
<td>2.4117e-05</td>
<td>2.7957e-06</td>
<td>5.9870e-08</td>
<td>-2.3241e-08</td>
</tr>
</tbody>
</table>

Table 3.3: GALILEO 15 covariance matrix

<table>
<thead>
<tr>
<th></th>
<th>R (km)</th>
<th>T (km)</th>
<th>N (km)</th>
<th>R (km/s)</th>
<th>T (km/s)</th>
<th>N (km/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R (km)</td>
<td>0.0362</td>
<td>-0.1329</td>
<td>0.01050</td>
<td>8.2282e-06</td>
<td>-4.5375e-06</td>
<td>-6.2709e-06</td>
</tr>
<tr>
<td>T (km)</td>
<td>-0.1329</td>
<td>2.8444</td>
<td>-0.0307</td>
<td>-0.0003</td>
<td>1.6168e-05</td>
<td>2.5288e-05</td>
</tr>
<tr>
<td>N (km)</td>
<td>0.01050</td>
<td>-0.03076</td>
<td>0.0046</td>
<td>1.3138e-06</td>
<td>-1.3166e-06</td>
<td>-1.8823e-06</td>
</tr>
<tr>
<td>R (km/s)</td>
<td>8.2282e-06</td>
<td>-0.0003</td>
<td>1.3138e-06</td>
<td>3.7503e-08</td>
<td>-9.7149e-10</td>
<td>-1.6890e-09</td>
</tr>
<tr>
<td>T (km/s)</td>
<td>-4.5375e-06</td>
<td>1.6168e-05</td>
<td>-1.3166e-06</td>
<td>-9.7149e-10</td>
<td>5.6878e-10</td>
<td>7.8516e-10</td>
</tr>
</tbody>
</table>
A few observations can be drawn by looking at the plots and tables:

- The trend of the residuals is rather steady and immediately approaching zero, apart from the T component; the T component is the one that notoriously has the largest variance and the very irregular shape of the plot confirms it. This suggests that probably less than one month of TLEs could have been used, but given that SGP4 is not that computationally expensive, using one month for a relatively small number of objects is still feasible;

- For the same reason stated above, the largest component of the variance is the along-track (T). This serves also as an empirical validation of the algorithm that has been employed;

- The covariance values that have been derived are not even comparable with the requirements for collision avoidance (cfr. Section 2.3.2.2), but the results for Galileo 15 (and other MEO/GEO satellites) is quite good considering the starting data: an uncertainty of 1.6 km over an height of over 23000 km represents a relative error ≤ 0.01%;

- The influence of atmospheric drag is very strong: the covariances for LEO satellites, especially lower LEO, are very big (e.g. SPOT 5 that is now being deorbited). This serves as proof of the fact that the $B^*$ term of TLEs should be treated with extreme caution and the atmospheric force model is very inaccurate.

Ultimately, this method to extract covariance information from TLEs proves useful during the initialization phase of the SST system, as it allows to use the largest publicly available...

Table 3.4: LAGEOS 1 covariance matrix

<table>
<thead>
<tr>
<th></th>
<th>R (km)</th>
<th>T (km)</th>
<th>N (km)</th>
<th>R (km/s)</th>
<th>T (km/s)</th>
<th>N (km/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R (km)</td>
<td>3.0652e-05</td>
<td>0.0003</td>
<td>0.0001</td>
<td>-1.4483e-07</td>
<td>-1.4067e-08</td>
<td>3.9911e-07</td>
</tr>
<tr>
<td>T (km)</td>
<td>0.0004</td>
<td>0.0047</td>
<td>0.0014</td>
<td>-1.9298e-06</td>
<td>-1.6396e-07</td>
<td>4.7253e-06</td>
</tr>
<tr>
<td>N (km)</td>
<td>0.0001</td>
<td>0.0014</td>
<td>0.0006</td>
<td>-5.4319e-07</td>
<td>-5.4569e-08</td>
<td>1.6870e-06</td>
</tr>
<tr>
<td>R (km/s)</td>
<td>-1.4484e-07</td>
<td>-1.9298e-06</td>
<td>-5.4320e-07</td>
<td>7.9153e-10</td>
<td>6.6102e-11</td>
<td>-1.8785e-09</td>
</tr>
<tr>
<td>T (km/s)</td>
<td>-1.4068e-08</td>
<td>-1.6396e-07</td>
<td>-5.4569e-08</td>
<td>6.6102e-11</td>
<td>6.4579e-12</td>
<td>-1.8367e-10</td>
</tr>
</tbody>
</table>

Table 3.5: SPOT 5 covariance matrix

<table>
<thead>
<tr>
<th></th>
<th>R (km)</th>
<th>T (km)</th>
<th>N (km)</th>
<th>R (km/s)</th>
<th>T (km/s)</th>
<th>N (km/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R (km)</td>
<td>0.05957</td>
<td>-0.7591</td>
<td>-0.0550</td>
<td>0.0007</td>
<td>-6.8968e-05</td>
<td>-1.8033e-05</td>
</tr>
<tr>
<td>T (km)</td>
<td>-0.75913</td>
<td>72.3226</td>
<td>0.2445</td>
<td>-0.07418</td>
<td>0.0015</td>
<td>0.0007</td>
</tr>
<tr>
<td>N (km)</td>
<td>-0.0550</td>
<td>0.2445</td>
<td>0.0603</td>
<td>-0.0002</td>
<td>5.9987e-05</td>
<td>1.1422e-05</td>
</tr>
<tr>
<td>R (km/s)</td>
<td>0.0007</td>
<td>-0.0742</td>
<td>-0.0002</td>
<td>7.6127e-05</td>
<td>-1.4478e-06</td>
<td>-6.9374e-07</td>
</tr>
<tr>
<td>T (km/s)</td>
<td>-6.8968e-05</td>
<td>0.0015</td>
<td>5.9987e-05</td>
<td>-1.4478e-06</td>
<td>8.5289e-08</td>
<td>2.4985e-08</td>
</tr>
</tbody>
</table>
catalogue (JSpOC). That said, any of the services that SPOOK could provide need additional observations in order to improve the accuracy of the orbital information.

It should be noted that in the SPOOK framework, this information is only used in the initialization time to have a starting catalogue; it is also worth mentioning that a starting value for covariance is only needed when performing OD using some kind of filter, while WLS methods (which JSpOC uses, for example) do not need an initial covariance.
Chapter 4

Orbit Determination success criteria

As part of the work done to automatize SPOOK’s workflow, one big step was to implement some “computer-friendly” tests to evaluate if an orbit determination process is reliable or not. So far, the evaluation of the results needed human intervention, as an operator had to manually check the residuals along with other parameters and assess them relying only on his/her experience.

Moreover, in the general case there is no ground truth (i.e. an external source of orbital data to compare against), so the tests need to rely exclusively on the residuals of the OD process and/or the consistency of the obtained data. After each OD, the residuals of the process are also returned in the observable space, i.e. in terms of the quantities that the observer whose data is used can measure (so one or more among angular data, range and range rate). The following checks were implemented:

- SPOOK works in the assumption that the errors in the OD process and the measurements are Gaussian distributed, with null mean and standard deviation equal to the sensor accuracy. For this reason, a $\chi^2$ test to check the gaussianity of the residuals was implemented;
- All the residuals should be located, within numerical errors, within the $3\sigma$ envelope;
- For each OD process, the Weighted Mean Root Square of all the residuals is calculated; because of the way it is defined, for a good orbit determination it should be as close as possible to 1 (cfr. Section 4.2);
- If a smoother is used, the McReynold’s consistency can be applied [24];
- A self-consistency test has also been devised (cfr. Section 4.3).

In the next sections, each test is described and validated using available data.
4.1 Chi squared test

The orbit determination methods available in SPOOK work in the assumption that all the processes and errors involved are Gaussian. Thus, it is natural to check if the calculated residuals follow a Gaussian distribution; this check was performed using the chi squared test in the following way:

- For each residual type, the mean $\mu$ and standard deviation $\sigma$ are calculated;
- The residuals are ordered in ascending order and then binned (i.e. classified into groups called ”bins”); while there is no ”optimal” bin width, there are various rules of thumb that can be used to calculate it. In particular, in this case the bin number has been calculated using Sturge’s formula\[22:\]

$$[k] = \log_2 n + 1$$  \hspace{1cm} (4.1)

where $k$ is the number of bins, $n$ is the number of data points and $[\ ]$ is the ceiling function. From that, it is possible to calculate the bin width as:

$$h = \frac{\max(\text{residual}) - \min(\text{residual})}{k}$$  \hspace{1cm} (4.2)

- For each bin $j$, the observed number of occurrences $O_j$ is counted and the expected number of occurrences is computed from the normal Cumulative Distribution Function:

$$E_j = n \left( \text{CDF}(\mu, \sigma, b) - \text{CFD}(\mu, \sigma, a) \right)$$  \hspace{1cm} (4.3)

where $b$ and $a$ are respectively the upper and lower bounds of bin $j$;
- For each bin, this coefficient is calculated:

$$S_i = \frac{(O_i - E_i)^2}{E_i}$$  \hspace{1cm} (4.4)

So that then the test statistic can be calculated as $\chi^2 = \sum S_i$
- At this point, the actual goodness-of-fit test can be performed: given a certain confidence level $p$ (chosen to be 95% at this stage), the critical $\chi^2_{k-1}$ distribution value that gives a $p$ credibility level is compared to the test statistic. If it is smaller, the null hypothesis of normally-distributed data is accepted, otherwise it is refused.

4.2 Weighted Root Mean Square

For each OD process, the Weighted Root Mean Square of the residuals can be calculated as follows:

$$\text{WRMS} = \sqrt{\frac{\sum_{i=1}^{N} \frac{\text{res}(i)^2}{\sigma_i^2}}{N}}$$  \hspace{1cm} (4.5)
In (4.5) \( N \) is the number of all the residuals for each quantity of each measurement and \( \sigma_i \) is the observer accuracy for that quantity. For example, having 30 measurements of an optical observer (so that each contains two angles), \( N = 60 \) and the WRMS will be the RMS of each residual averaged with its accuracy.

The WRMS as introduced in 4.5 is a single number that conveys information about the whole OD process, thus being a very synthetic source of information about its quality. It is straightforward that its value should be as low as possible; however, it should never be lower than 1 (in the limit of numerical errors), because given that each residual is averaged with its sensor accuracy, it would mean that the final result is inside the sensor’s noise. A very low value of WRMS is thus a sign of divergence of the filter.

4.3 Self consistency test

This test can only be performed if a sequential estimator (such as a filter) is used. The general idea behind it is to check if the filter’s forecast is coherent with the actual path of the object in space.

Let us suppose that an Orbit Determination is being performed using a filter and there are \( N \) measurements available; for each measurement, the filter provides an updated state vector and covariance. The purpose of this test is to check if the filter at the time of the \((N-1)\)-th measurement can predict the \( N \)th measurement, thus proving that it is correctly "following" the object. The way the test is implemented and executed is the following:

- Given a batch of \( N \) measurements belonging to a certain observer, the OD is performed and the state vectors with their covariances at the epochs of the \( N \) and \( (N-1) \) measurements are considered;

- The state vector and covariance at the \( (N-1) \) epoch are propagated to the epoch of the \( N \)th measurement and then transformed to the observable space using an Unscented Transformation[11];

- Now we test if the actual \( N \)th measurement is inside this propagated covariance; to do so, the \( n \)-dimensional covariance matrix \( P \) (where \( n \) is the number of observables) is expressed in its principal inertia axes by diagonalizing it:

\[
P = T^{-1} DT
\]

(4.6)

\( T \) is the rotation matrix with the eigenvectors on the column and \( D \) is the diagonal matrix containing the eigenvalues;

- Now the covariance is an ellipsoid with axes length related to the eigenvalues \( (a_i = \lambda_i^{-1/2}) \). The actual \( N \)th observation is also rotated to the ellipsoid’s principal axes:

\[
o = To
\]

(4.7)
Then the equation of the ellipsoid is used to check if the observation point is inside the covariance:

\[ \sum_{i=1}^{n} \frac{(o_i - o_0)^2 \lambda_i}{3^2} \leq 1 \]  

(4.8)

where \( o_i \) is the \( i \)th component of the propagated \( N - 1 \)th observable vector, \( o_0 \) is the center of the covariance ellipsoid and \( \lambda_i \) is the \( i \)th eigenvalue. The 3 factor is added to consider the 3σ confidence level.

### 4.4 McReynold’s filter-smoother consistency test

This test assumes that a smoother is used in conjunction with the filter and verifies if the smoothed state is consistent with the filtered state, assuming multi-dimensional Normal distributions; the following is derived from [24]. First, for each time \( t_k \), the difference matrix between the filtered state covariance matrix \( P_{k|f} \) and the smoothed state covariance matrix \( P_{k|s} \) is calculated:

\[ \bar{P}_k = P_{k|f} - P_{k|s} \]  

(4.9)

This matrix should not have any negative eigenvalues; let us denote the square root of the \( i \)th diagonal element of this matrix as \( \sigma_i^k \). The same procedure can be applied to the smoothed and filtered state vectors, by computing:

\[ \bar{X}_k = X_{k|f} - X_{k|s} \]  

(4.10)

As before, the \( i \)th element of this vector will be denoted as \( \bar{X}_i^k \). It is now possible to calculate the following ratio:

\[ R_i^k = \frac{\bar{X}_i^k}{\sigma_i^k} \]  

(4.11)

The McReynold’s filter-smoother consistency test is satisfied if for all \( i, k \) the following inequality is valid:

\[ |R_i^k| \leq 3 \]  

(4.12)

### 4.5 3σ envelope test

As stated before, all the OD methods in SPOOK work in the assumption of Gaussian processes; for each observer whose measurements have been used, a sensor accuracy is provided. The hypothesis, both for filters and WLS methods, is that every measurement smaller than its accuracy is noise, as it could not be resolved by the instrument. This means that, for a large number of observations, the standard deviation of the residuals should be close to the sensor accuracy for that type of measurement; moreover, the hypothesis of Gaussian distribution implies that at least 99% of the residuals should fall inside the 3σ envelope; thus, a natural test is to check if more than 99% of the residuals are inside the 3σ envelope, with \( \sigma \) being the sensor accuracy for the considered observable.
4.6 Validation and discussion

The metrics introduced above are validated using a set of synthetic data; 17 cases are considered, for each reporting residuals plot for each observable, value of the WRMS, results of the $\chi^2$ test, result of the consistency test (written as the distance from the covariance ellipsoid, which should be $\leq 1$) and the result of the $3\sigma$ envelope test given as fraction of outsiders. The 17 cases are heterogeneous with respect to orbital region, OD method used, length of observation (represented in Julian Date on the x axis of each plot), number of measurements, type of observer.

4.6.1 Results of the tests

The main results of the validation tests are summarised in Table 4.1. The residuals plots for each case, along with the observers types, are reported in Appendix A for clarity reasons and should be looked at in order to characterise each observation situation.
<table>
<thead>
<tr>
<th>N</th>
<th>OD Method</th>
<th>Orbital region</th>
<th>WRMS</th>
<th>$\chi^2$ results</th>
<th>% outsiders</th>
<th>Consistency test distance</th>
<th>Real error (km)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EKF</td>
<td>MEO</td>
<td>7.17</td>
<td>T F</td>
<td>37% 13%</td>
<td>2.5E-23</td>
<td>27.18</td>
</tr>
<tr>
<td>2</td>
<td>EKF</td>
<td>MEO</td>
<td>0.98</td>
<td>T T</td>
<td>0.18% 0.26%</td>
<td>3E-18</td>
<td>0.016</td>
</tr>
<tr>
<td>3</td>
<td>EKF</td>
<td>LEO</td>
<td>0.99</td>
<td>T T</td>
<td>0.23% 0.19%</td>
<td>4.9E-25</td>
<td>0.015</td>
</tr>
<tr>
<td>4</td>
<td>EKF</td>
<td>LEO</td>
<td>1.00</td>
<td>T T T</td>
<td>0.17% 0.5% 0.17%</td>
<td>1.87E-15</td>
<td>0.05</td>
</tr>
<tr>
<td>5</td>
<td>EKF</td>
<td>MEO</td>
<td>1.07</td>
<td>T T</td>
<td>0.3% 0.3%</td>
<td>4.91E-19</td>
<td>0.03</td>
</tr>
<tr>
<td>6</td>
<td>SBLS</td>
<td>MEO</td>
<td>1.02</td>
<td>T T</td>
<td>0.03% 0.03%</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SBLS with BC est.</td>
<td>LEO</td>
<td>0.94</td>
<td>T T T</td>
<td>0% 0% 0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>WLS with SRPC est.</td>
<td>LEO</td>
<td>1.03</td>
<td>T T</td>
<td>0.03% 0.04%</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>WLS with BC and SRPC est.</td>
<td>LEO</td>
<td>1.05</td>
<td>T T</td>
<td>0% 0%</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>EKF</td>
<td>LEO</td>
<td>5.45</td>
<td>T T T</td>
<td>58% 2.8% 0.3%</td>
<td>1.9E-12</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>SBLS</td>
<td>MEO</td>
<td>1.02</td>
<td>T T T T</td>
<td>0.3% 0.08% 0.3% 0.5%</td>
<td></td>
<td>43.8</td>
</tr>
<tr>
<td>12</td>
<td>WLS</td>
<td>LEO</td>
<td>18</td>
<td>T</td>
<td>48%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>UKF</td>
<td>MEO</td>
<td>1.04</td>
<td>T T T T</td>
<td>0% 0% 0% 1.4%</td>
<td>3.37E-9</td>
<td>0.27</td>
</tr>
<tr>
<td>14</td>
<td>EKF</td>
<td>MEO</td>
<td>9.2</td>
<td>F T</td>
<td>71% 25%</td>
<td>1.15E-16</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>WLS</td>
<td>MEO</td>
<td>6.4</td>
<td>T T</td>
<td>7% 55%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>WLS</td>
<td>MEO</td>
<td>0.99</td>
<td>T T</td>
<td>0.2% 0.2%</td>
<td></td>
<td>0.01</td>
</tr>
<tr>
<td>17</td>
<td>WLS</td>
<td>MEO</td>
<td>2.4</td>
<td>T T</td>
<td>18% 29%</td>
<td></td>
<td>3.2</td>
</tr>
</tbody>
</table>

Table 4.1: Summarised results of validation tests
4.6.2 Discussion

Considering the theoretical discussion in the previous sections, the proposed data seems to prove the goodness of the developed metrics: within numerical limits, when the residuals seems to be converging, all of the metrics agree on the success of the OD process, with a few remarks:

- The $\chi^2$ test cannot be used as conclusive proof, as it seems to return a positive result also in some of the cases that should be rejected. Maybe a stricter $p$ value could be used;

- The consistency test should be further investigated, as in all of the cases proposed above it never fails. One possible reason might lie in the fact that the epochs of the $N$th and the $N-1$th measurements are too close in time, thus making it extremely likely to be able to predict the last measurement from the second-last. This effect could be proved by considering state vectors from “older” epochs to run it;

- The WRMS and outlier percentage seem to agree on each case; values of WRMS bigger than 2 coincide with percentage of outliers of at least 15%-20%, while values of WRMs around 1 are related to less than 1% of outliers; thus, this gives them a primary importance on the overall evaluation of the OD process.

However, these results cannot be used to “tune” the acceptance/rejection process; a threshold for the considered quantities should be set (e.g. on the percentage of allowed outliers, or on the range of accepted WRMS values), which would require a larger taxonomy of tests using different orbital regions, types of observers, length of observation and many other parameters, which does not fit in the scope of this thesis. Also, these metrics should be tested against real data, which will likely inflate these thresholds. Once a larger and more representative set of tests is run, a similar procedure to the one exposed above can be performed to find suitable threshold values for these metrics; it might be a good idea to summarize all those results in a single flag that averages the outcome of all the tests, giving more weight to the ones that are found to be more representative of the truth and less to the ones that are less sensible (for example the chi squared test, according to the data above).
Chapter 5

Solver for the Lambert problem

As mentioned in Section 2 in various parts of the code the so-called Lambert problem (sometimes referred to as orbital boundary value problem) is solved. It is one of the fundamental problems of astrodynamics, together with Euler’s problem of propagation. Its formulation is rather simple: given two position vectors of a body subject to a central force and a time of flight between them, the target is to find an interpolating orbit. The geometry of the problem is shown in Figure 5.1.

The Lambert problem needs to be solved in various parts of the code, such as IOD and correlation (where performance is critical, hence the need for a faster algorithm). Three algorithms were available in SPOOK at the beginning of this thesis:

• Gibbs, which uses a purely geometrical approach;

• Herrick-Gibbs, which is an extension of the Gibbs approach capable of working when the two position vectors are very close together that uses a Taylor expansion;

• Lambert-Gauss, which has the peculiarity to be able to generate a state vector using only angular (i.e. optical) measurements.

The aforementioned algorithms have been discussed in a number of works (most notably [16]) and thus will not be further examined.

In order to improve the performances of this process, another algorithm has been implemented, which is able to converge (on average) in only 2 iterations for the single revolution case and 3 for the multiple revolutions case [10]. This algorithm was firstly derived by Dario Izzo in 2014 and thus came to be referred to in literature as ”Izzo’s Lambert problem solver”. In the next subsections, the algorithm is briefly explained by summarizing the original derivation in [10], then a small comparison test against the available procedures in SPOOK is carried out in order to highlight the gain in computation time.

5.1 Algorithm implementation

The following is summarized from [10].
The algorithm builds upon a procedure firstly developed by Lancaster and Blanchard \cite{14}, which uses the so-called \textit{universal variable formulation} of the Lambert problem. Its main features are the following:

- The algorithm iterates on the universal Lancaster-Blanchard variable \( x \), which will be defined shortly;
- The iterations are carried out using an Householder scheme;
- The initial guess will make use of new analytical results that are presented during this paragraph.

According to Lambert’s theorem, the solutions to the Lambert problem depend only on the orbit semi-major axis \( a \), the sum \( r_1 + r_2 \) and the chord \( c \) of the triangle that has \( r_1 \) and \( r_2 \) as sides. Following a derivation by Battin\cite{3}, two angles \( \alpha \) and \( \beta \) are introduced, such that \( s \) is the semiperimeter of the \( r_1, r_2, c \) triangle:

\[
\frac{s}{2a} = \begin{cases} 
\sin^2 \frac{\alpha}{2} \\
-\sinh^2 \frac{\alpha}{2}
\end{cases} \tag{5.1}
\]

\[
\frac{s - c}{2a} = \begin{cases} 
\sin^2 \frac{\beta}{2} \\
-\sinh^2 \frac{\beta}{2}
\end{cases} \tag{5.2}
\]

The two cases hold respectively for the elliptic and hyperbolic case. Using the parameter \( \lambda \) defined as \( \lambda^2 = \frac{s - c}{c} \), one can derive the relation:

\[
\sin \frac{\alpha}{2} = \lambda \sin \frac{\beta}{2} \tag{5.3}
\]

\( v_1, v_2 \) are obtained after solving the Lambert problem.
It follows immediately that $\lambda \in [-1, 1]$ and $\lambda$ is positive for $\theta \in [0, \pi]$ and negative for $\theta \in [\pi, 2\pi]$. Following [14], an adimensionalised time of flight is introduced:

$$T = \sqrt{\frac{2 L}{s^3}} (t_2 - t_1) \quad (5.4)$$

In (5.4) $(t_2 - t_1)$ is the actual time of flight. Lancaster then introduces two ”universal variables” $x$ and $y$:

$$x = \begin{cases} \cos \frac{\alpha}{2} \\ \cosh \frac{\alpha}{2} \end{cases} \quad y = \begin{cases} \cos \frac{\beta}{2} \\ \cosh \frac{\beta}{2} \end{cases} \quad (5.5)$$

The variable $x$ is said to be a *Lambert invariant parameter*, as different problems with same $\lambda$ (i.e. same c/s) result in the same $x$. Using these variables, the non-dimensional time of flight is:

$$T = \frac{1}{1 - x^2} \left( \frac{\psi + M\pi}{\sqrt{|1 - x^2|}} - x + \lambda y \right) \quad (5.6)$$

where $M$ is the number of revolutions and $\psi$ is an auxiliary angle defined as:

$$\cos \psi = xy + \lambda(1 - x^2)$$

$$\cosh \psi = xy - \lambda(x^2 - 1)$$

where as usual the first case is elliptic and the second is hyperbolic. Computing (5.6) in $x = 0$ we get:

$$T(x = 0) = T_{0M} = \arccos \lambda + \lambda\sqrt{1 - \lambda^2} + M\pi = T_{00} + M\pi \quad (5.7)$$

where $T_{00}$ is the value in the single revolution case $(x = 0, M = 0)$. In this case, however, a loss of precision is encountered because when $x \approx 1$ both $1 - x^2$ and $\psi$ tend to 0. Thus, in this case the time of flight is computed using a series expansion derived from [3]:

$$\eta = y - \lambda x$$

$$S_1 = \frac{1}{2} (1 - \lambda - x\eta)$$

$$Q = \frac{4}{3} \, _1F_2 \left( 3, 1, \frac{5}{2}, S_1 \right)$$

$$2T = \eta^3Q + 4\lambda \eta \quad (5.8)$$

where $_1F_2(a, b, c, d)$ is the ordinary Gaussian or hypergeometric function that can be calculated from the corresponding hypergeometric series.

A few useful expressions can be computed from the equations above, including the derivatives of the time of flight with respect to $x$:

$$T(x = 1, y = 1) = T_1 = \frac{2}{3}(1 - \lambda^3) \quad (5.9)$$
\[(1 - x^2) \frac{dT}{dx} = 3Tx - 2 + 2\lambda^3 \frac{x}{y} \]
\[(1 - x^2) \frac{d^2T}{dx^2} = 3T + 5x \frac{dT}{dx} + 2(1 - \lambda^2) \frac{\lambda^3}{y^3} \]
\[(1 - x^2) \frac{d^3T}{dx^3} = 7x \frac{d^2T}{dx^2} + 8 \frac{dT}{dx} - 6(1 - \lambda^2) \frac{\lambda^5 x}{y^5} \]
\[\left. \frac{dT}{dx} \right|_{x=1,M=0} = 2 \frac{5(\lambda^5 - 1)}{5} \]
\[\left. \frac{dT}{dx} \right|_{x=0} = -2 \]

The last two expressions are useful as pre-computing them saves computation time. The general solution procedure is the following:

- Given the input data, compute \(\lambda\) and \(T^*\), which is the non dimensional time of flight;
- Starting from an initial guess that will be introduced in the next paragraph, use an Householder iteration scheme to find the roots of the equation \(T(x) - T^* = 0\) in terms of \(x\) and \(y\);
- For each \((x, y)\) couple, reconstruct \(v_1\) and \(v_2\) using the procedure detailed in [8].

The Householder scheme is the following [10]:
\[x_{n+1} = x_n - \frac{f(x_n) - f(x_n) f''(x_n) / 2}{f'(x_n) (f'^2(x_n) - f(x_n) f'''(x_n)) + f''(x_n) f'^2(x_n) / 6} \]

where \(f\) is \(T(x) - T^*\) and \(f', f'', f'''\) are its derivatives. The reconstruction procedure is the following (\(r, t\) represent respectively the radial and tangential components):
\[\gamma = \sqrt{\frac{\mu s}{2}} \quad \rho = \frac{r_1 - r_2}{c} \quad \sigma = \sqrt{1 - \rho^2} \]
\[v_{r1} = \gamma [(\lambda y - x) - \rho (\lambda y + x)] / r_1 \]
\[v_{r2} = -\gamma [(\lambda y - x) + \rho (\lambda y + x)] / r_2 \]
\[v_{t1} = \gamma \sigma (y + \lambda x) / r_1 \]
\[v_{t2} = \gamma \sigma (y + \lambda x) / r_2 \]

A new way to estimate an initial guess without using heuristics can be found by considering the following new variables:
\[\xi = \begin{cases} \log(1 + x), & M = 0 \\ \log \left( \frac{1 + x}{1 - x} \right), & M > 0 \end{cases} \quad \tau = \log T \]
With these new variables, the domain of the time of flight curve is $(-\infty, +\infty)$; these curves are shown in Figure 5.2.

The newly introduced variable $\xi$ is still Lambert-invariant, as it is basically a transformation of $x$. Studying the derivatives of those curves leads to some interesting results, which can be derived with basic calculus:

$$M = 0 \implies \begin{cases} \lim_{\xi \to +\infty} \tau = -\xi + \log(1 - \lambda |\lambda|) \\ \lim_{\xi \to -\infty} \tau = -\frac{3}{2} \xi + \log \left(\frac{\pi}{4} \sqrt{2}\right) \end{cases} \tag{5.14}$$

$$M > 0 \implies \begin{cases} \lim_{\xi \to -\infty} \tau = \log \left(\frac{\pi + M\pi}{8}\right) - \frac{3}{2} \xi \\ \lim_{\xi \to +\infty} \tau = \log \left(\frac{M\pi}{8}\right) + \frac{3}{2} \xi \end{cases} \tag{5.15}$$

i.e. the curves in Figure 5.2 have two symmetric asymptotes of inclination $\pm 3/2$. This is the result that will be exploited to provide a good initial guess to the Householder iterations. The idea is to piecewise approximate the $\xi - \tau$ curve with lines that have the asymptotic behaviour of (5.14) and (5.15). For $M = 0$ we get:

$$\tau = -\frac{3}{2} \xi + \tau_0 \quad T \geq T_0$$

$$\tau = -\xi + \log 2 + \tau_1 \quad T < T_1$$

$$\tau = \tau_0 + \frac{\xi}{\log 2}(\tau_1 - \tau_0) \quad T_1 < T < T_0 \tag{5.16}$$
Transformed to the $x - T$ plane, these lines become:

$$x_0 = \left( \frac{T_0}{T} \right)^{\frac{2}{3}} - 1 \quad T \geq T_0$$

$$x_0 = \frac{5}{2} \frac{T_1(T_1 - T)}{T(1 - \lambda^5)} + 1 \quad T < T_1$$

$$x_0 = \left( \frac{T_0}{T} \right)^{\log_2 \left( \frac{T_1}{T_0} \right)} - 1 \quad T_1 < T < T_0$$

(5.17)

Whereas for the multiple revolution case there are two possible values of $x$, thus two starters are needed (left and right asymptotes), which can be calculated as:

$$\xi_{0l} = 2 \frac{2}{3} \left( \log \frac{\pi + M\pi}{8} - \tau \right)$$

$$\xi_{0r} = 2 \frac{2}{3} \left( \tau - \log \frac{M\pi}{8} \right)$$

(5.18)

And then be transformed back to the $x - T$ plane, resulting in:

$$x_{0l} = \frac{\left( \frac{M\pi + \pi}{8T} \right)^{\frac{2}{3}} - 1}{\left( \frac{M\pi + \pi}{8T} \right)^{\frac{2}{3}} + 1}$$

$$x_{0r} = \frac{\frac{8T}{M\pi} - 1}{\frac{8T}{M\pi} + 1}$$

(5.19)

### 5.2 Performance comparison

A simple benchmark test was run in order to characterize the performance of the new Lambert problem solver. In order to do so, 100 IODs with the same (simulated) data were run with the three available solvers and the execution times compared. The data were simulated using an optical observer (as the Lambert-Gauss algorithm can only handle angular data); a point to be kept in mind is that the (Herrick-)Gibbs and Lambert-Gauss algorithms use three position vectors and calculate the velocity vector in the middle, whereas the Izzo solver needs only two position vectors and calculates the velocity for both. Thus, in order to not alter the original structure of SPOOK and provide better accuracy, the Izzo solver has been implemented so that it will run twice using the couples $(r_1, r_2)$ and $(r_2, r_3)$ and then average the two values of $v_2$. For this benchmark test, the average of the two computation times ("left" and "right") has been considered. The results are shown in Table 5.1.

No appreciable difference in the computed velocity vectors was spotted.
Table 5.1: Performance comparison between different IOD methods

<table>
<thead>
<tr>
<th>IOD Method</th>
<th>Runtime for 100 IODs (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Herrick-)Gibbs</td>
<td>6.31 s</td>
</tr>
<tr>
<td>Izzo</td>
<td>6.21 s</td>
</tr>
<tr>
<td>Lambert-Gauss</td>
<td>6.25 s</td>
</tr>
</tbody>
</table>
Chapter 6
Conclusions and future development

A number of improvements has been implemented in SPOOK in the scope of this thesis; the software is now a working prototype of a complete SST system and a sustainable business data logic is defined. A complete test run was executed with satisfying results, which ultimately shows the feasibility of the proposed structure for the cataloguing system. Of course a number of core features still need to be implemented, validated and integrated (most notably T2T correlation), but once those are taken care of, SPOOK will be virtually ready for final testing and deployment. However, a few points should be highlighted:

• Given the size of the catalog that SPOOK should be ready to handle, there still is work to do to make it run with as little human intervention as possible. That applies to both configuration parameters (IOD and OD methods, propagation duration, selection of tracklets that should be used, etc.) and evaluation of results (cfr. Chapter 4).

  The metrics proposed in Chapter 4 while tested and validated ”theoretically” with synthetic data, should be calculated also on a wide set of OD scenarios in order to assess their variability with respect to orbital region, number of measurements, sparseness of data, observation duration, etc. This corresponds to a huge taxonomy of tests whose results should be analysed in order to set appropriate thresholds for each OD scenario. Also, when real data will be used, those thresholds will most likely be inflated.

  On a similar note, an analogous taxonomy of tests should be run to assess the influence of different configuration parameters on the accuracy of the final results. While being a long shot, some kind of AI approach could be worth investigating to select the best parameters for each scenario, using those tests as training data; while some peculiar situations will require human intervention, it might be a powerful solution for ordinary catalog maintenance;

• It is time to start implementing some of the services: most of the capabilities are already available. For example, pass prediction and covariance monitoring as per the requirements detailed in Section 2.3 can be implemented without any other prerequisites;
• An interface to measurement sources needs to be implemented. So far, the measurements provided by external sources were already given in suitable formats, but if Airbus is to use "in-house" sensors (telescopes, radars, etc.), a way to reconstruct meaningful measurements from them must be implemented;

• So far the database was queried using external tools (primarily pgAdmin), but eventually a proprietary interface to the catalog will need to be implemented;

• The new Lambert solver seems to have better performances than the other algorithms previously available (100 IODs show a gain in the order of 0.1 s and the computation time does not scale linearly with the number of data points, thus a much bigger gain with millions of objects can be expected), but it needs further testing and scalability analyses.

A major issue in the implementation and integration of these new features is that the SST system, given the number of objects that it is supposed to handle, must work as autonomously as possible. This means that error handling and reporting will be the most time consuming part of the development, which should focus on continuous regression checking and exception catching.

However, this working prototype shows the feasibility of such a project and now that the "bigger picture" is clearly defined SPOOK is closer to a complete and integrated SST environment than just a research calculation tool. The possibility to use JSpOC data effectively will also represent a great asset, as it will give the chance to already integrate the biggest space debris database publicly available from the start.
Bibliography


[5] ESA. *How many space debris objects are currently in orbit?* 2013. URL: [http://www.esa.int/Our_Activities/Space_Engineering_Technology/Clean_Space/How_many_space_debris_objects_are_currently_in_orbit](http://www.esa.int/Our_Activities/Space_Engineering_Technology/Clean_Space/How_many_space_debris_objects_are_currently_in_orbit) (visited on 08/12/2018).


Appendix A

Results of the OD success criteria validation tests

Below are reported the complete results of the tests used to validate the OD success criteria.

Case 1

OD Method: Extended Kalman Filter
State vector: -37143.6706 -19964.694300 391.315982 1.455882 -2.707204 -0.045592 0.000000 0.000000
Orbital elements: (21085.5069, 0.9999, 0.0174, 5.7239, 1.0523, -3.1415)
Mean of Right_asc residual: 0.00086809.
Mean of Declination residual: 1.90158e-05.
WRMS: 7.17536891
Chi squared test results: T F
Fraction of outsiders from 3σ envelope: 0.3763 0.1381
Consistency test distance: 2.5017E-23
Real error: 27.18 km

Case 2

OD Method: Extended Kalman Filter
State vector: 3994.515210 5309.697050 203.401940 -6.161683 4.618632 -0.146431 0.000000 0.000000
Orbital elements: (3323.7932, 0.9999, 0.0359, 5.2666, -1.1995, -3.1415)
Mean of Right Asc residual: 1.21679e-07.
Mean of Declination residual: 1.05364e-06.
WRMS: 0.985084789
Chi squared test results: TT
Fraction of outsiders from 3σ envelope: 0.0018 0.0026
Consistency test distance: 3.0345E-18
Real error: 0.016 km

Case 03

OD Method: Extended Kalman Filter
State vector: 15413.819600 -13488.325700 -21354.169200 0.852823 3.265255 -1.444294 0.000000 0.000000
Orbital elements: (14794.6046, 0.9999, 0.9651, 1.0712, 1.6161, -3.1415)
Mean of Right Asc residual: -5.89402e-07.
Mean of Declination residual: -3.85299e-07.
WRMS: 0.99245791
Chi squared test results: TT
Fraction of outsiders from 3σ envelope: 0.0023 0.0019
Consistency test distance: 4.9019242169980498E-25
Real error: 0.015 km

Case 04
OD Method: Extended Kalman Filter
State vector: -615.390591 -3244.185270 6168.527010 -0.903769 6.653529 3.460666 0.000000
Orbital elements: (3498.3628, 0.9999, 1.7041, -2.0452, -1.5049, 3.1415)
Mean of Range residual: 0.0399275.
Mean of Azimuth residual: -4.97983e-05.
Mean of Elevation residual: 0.000121053.
WRMS: 1.00030232
Chi squared test results: T T T
Fraction of outsiders from 3σ envelope: 0.0017 0.0053 0.0017
Consistency test distance: 1.8716E-15
Real error: 0.05 km

Case 05
OD Method: Extended Kalman Filter
State vector: 9720.707100 18985.445900 38031.822900 -1.445034 0.435246 1.013741 0.000000
Orbital elements: (21802.2869, 0.9999, 1.1166, -1.813, 0.0415, 3.1415)
Mean of Right_Asc residual: 1.72095e-06.
Mean of Declination residual: -2.24325e-07.
WRMS: 1.06853511
Chi squared test results: T T
Fraction of outsiders from 3σ envelope: 0.0033 0.0033
Consistency test distance: 4.9105E-19
Real error: 0.03 km
Case 06

OD Method: Sequential Batch Least Squares
State vector: -21085.105000 -36526.637000 -0.048535 2.661748 -1.536502 -0.053647 0.000000 0.000000
Orbital elements: (21087.7740, 0.9999, 0.0174, -6.2831, 1.047, 3.1415)
Mean of Right Asc residual: 1.67264e-09.
Mean of Declination residual: 1.47779e-07.
WRMS: 1.0185963
Chi squared test results: T T
Fraction of outsiders from 3σ envelope: 0.0026 0.0026
Real error: 2.88 km

Case 07 (BC estimation)

OD Method: Sequential Batch Least Squares
State vector: 6637.819200 1066.919720 1447.706010 -1.968916 5.261519 5.138155 0.043538 0.000000
Orbital elements: (3438.5612, 0.9999, 0.7852, 3.4439, -0.0577, -3.1415)
Mean of Range residual: 0.00900917.
Mean of Azimuth residual: -3.20806e-05.
Mean of Elevation residual: 2.51602e-05.
WRMS: 0.948511313
Chi squared test results: T T T
Fraction of outsiders from 3σ envelope: 0.0000 0.0000 0.0000
Real error: 0.10 km
Case 08 (SRPC estimation)
OD Method: Weighted Least Squares
State vector: -42109.780500 -2173.168520 619.048277 0.157214 -3.070057 -0.029172 0.000000 0.022001
Orbital elements: (21085.1813, 0.9999, 0.0174, -0.9967, 1.0482, 3.1415)
Mean of Right_Asc residual: -1.93489e-08.
Mean of Declination residual: 2.12481e-07.
WRMS: 1.02906726
Chi squared test results: T T
Fraction of outsiders from 3σ envelope: 0.0026 0.0044
Real error: 0.01 km

Case 09 (both BC and SRPC estimated)

![Graphs showing Right_Asc residual and Declination residual for DebrisOb]

OD Method: Weighted Least Squares
State vector: -9590.292000 -1160.681100 -2277.024650 5.824974 -2.500139 -5.022407 0.043919 0.022160
Orbital elements: (4962.5024, 0.9999, 1.1068, 0.2594, 0.0022, -3.1415)
Mean of Right_Asc residual: -8.4342e-06.
Mean of Declination residual: 2.25206e-06.
WRMS: 1.04841746
Chi squared test results: T T
Fraction of outsiders from 3σ envelope: 0.0000 0.0000
Real error: 0.99 km

Case 10

OD Method: Extended Kalman Filter
State vector: -615.692243 -3241.716590 6169.839480 -0.903561 6.654894 3.457985 0.000000 0.0026
0.000000
Orbital elements: (3498.3825, 0.9999, 1.7041, -2.0448, -1.5049, 3.1415)
Mean of Range residual: 2.07821.
Mean of Azimuth residual: -0.000243776.
Mean of Elevation residual: 6.7285e-05.
WRMS: 5.45195237
Chi squared test results: T T T
Fraction of outsiders from 3σ envelope: 0.5833 0.0283 0.0035 Consistency test distance: 1.9392E-12

**Case 11**

OD Method: Sequential Batch Least Squares
State vector: -42094.901100 -2446.345900 616.434330 0.177131 -3.068964 -0.029464 0.000000 0.000000
Orbital elements: (21085.2157, 0.9999, 0.0174, -0.9902, 1.0482, 3.1415)
Mean of Range residual: -7.80633e-05.
Mean of Azimuth residual: 1.8299e-06.
Mean of Elevation residual: 7.46957e-05.
Mean of Range_rate residual: -0.0148095.
WRMS: 1.01867621
Chi squared test results: T T T T
Fraction of outsiders from 3σ envelope: 0.0026 0.0008 0.0026 0.0052
Real error: 43.7677 km
OD Method: Weighted Least Squares
State vector: 2738.105580 -655.965628 6483.692660 5.620568 -4.126136 -2.785119 0.000000 0.000000
Orbital elements: (3534.3239, 0.9999, 1.7146, -1.1858, 2.566, 3.1415)
Mean of Range residual: 3.30169.
WRMS: 18.4432031
Chi squared test results: T
Fraction of outsiders from 3σ envelope: 0.4768
Case 13

OD Method: Unscented Kalman Filter
Mean of Range residual: -0.0557123.
Mean of Azimuth residual: 0.167918.
Mean of Elevation residual: 0.0667832.
Mean of Range_rate residual: -0.100942. WRMS: 1.03771028
Chi squared test results: T T T T
Fraction of outsiders from $3\sigma$ envelope: 0.0000 0.0000 0.0000 0.0138
Consistency test distance: 3.3684E-09
Real error: 0.27 km

Case 14

OD Method: Extended Kalman Filter
State vector: 9719.573830 18985.677000 38030.824000 -1.445111 0.435190 1.013857 0.000000
0.000000
Orbital elements: (21801.7752, 0.9999, 1.1166, -1.8140, 0.0416, 3.1415)
Mean of Right_Asc residual: 4.1351e-05.
Mean of Declination residual: 5.20124e-05.
WRMS: 9.16500224
Chi squared test results: F T Fraction of outsiders from 3σ envelope: 0.7097 0.2520
Consistency test distance: 1.1486E-16

Case 15

OD Method: Weighted Least Squares
State vector: 8161.743770 -41273.093800 -70.983312 3.019342 0.595959 -0.004125 0.000000
Orbital elements: (21036.2031, 0.9999, 0.0021, -5.3835, 0.8663, 3.1415)
Mean of Right_Asc residual: 2.2528e-05.
Mean of Declination residual: 3.60235e-05.
WRMS: 6.37998201
Chi squared test results: T T
Fraction of outsiders from 3σ envelope: 0.0689 0.5517

Case 16

OD Method: Weighted Least Squares
Mean of Right_asc residual: 2.22541e-07.
Mean of Declination residual: 1.74432e-07.
State vector: 19161.156700 -37463.258900 -88.772832 2.739480 1.401727 -0.002167 0.000000 0.000000
Orbital elements: (21039.5673, 0.9999, 0.0022, 1.2484, 0.7950, -3.1415)
WRMS: 0.994356955
Chi squared test results: T T
Fraction of outsiders from 3σ envelope: 0.0021 0.0021
Real error: 0.01 km

Case 17

OD Method: Weighted Least Squares
State vector: 19164.736900 -37462.350200 -88.777585 2.739367 1.401813 -0.002168 0.000000 0.000000
Orbital elements: (21039.9780, 0.9999, 0.0022, 1.2483, 0.7953, -3.1415)
Mean of Right_asc residual: 1.51455e-05.
Mean of Declination residual: 9.88165e-05.
WRMS: 2.40329353
Chi squared test results: T T
Fraction of outsiders from $3\sigma$ envelope: 0.1888 0.2871
Real error: 3.21 km
Appendix B

NORAD and COSPAR IDs

Both are means of identification of objects in orbit. The NORAD ID is a 5-digit identification number assigned progressively by NORAD to each object. The COSPAR (Committee on Space Research) ID (sometimes referred to as international designator) is a more descriptive form of identification that has the following form:

(XX)XX-NNNC

where:

• XX is the year of the launch (optionally including the century);
• NNN is the number of launch performed in that year;
• C is the piece letter of the launch (e.g. A can be the first stage of the rocket, etc.)
Appendix C

TLE format

TLE (Two-Line Element sets) are a synthetic way to relay orbital information about an object at a certain time. They were developed by NORAD. The format is the following [19]:

The first line (‘0’) is optional and contains the common name of the object (TLEs that contain this line are sometimes referred to as 3le). The fields in the main two lines represent the following quantities:

<table>
<thead>
<tr>
<th>Cell</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Line 1</td>
<td></td>
</tr>
<tr>
<td>03-</td>
<td>NORAD ID</td>
<td>The 'U' is compulsory, derives from 'Unclassified'</td>
</tr>
<tr>
<td>08-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-</td>
<td>COSPAR ID</td>
<td>Usually only the last two digits of the year are used</td>
</tr>
<tr>
<td>17-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19-</td>
<td>Epoch</td>
<td>XX are the last two digits of the year, Ns are the number of Julian days since the beginning of the year</td>
</tr>
<tr>
<td>32-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34-</td>
<td>First derivative of mean motion / 2</td>
<td>( \text{rev/day}^2 )</td>
</tr>
<tr>
<td>43-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45-</td>
<td>Second derivative of mean motion / 6</td>
<td>Decimal point assumed; units are ( \text{rev/day}^3 )</td>
</tr>
<tr>
<td>52-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>54-</td>
<td>( B^* )</td>
<td>Decimal point assumed</td>
</tr>
<tr>
<td>61-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>63-</td>
<td>'0'</td>
<td>Originally should have been ephemeris type</td>
</tr>
<tr>
<td>65-</td>
<td>Element set number</td>
<td></td>
</tr>
<tr>
<td>68-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>69-</td>
<td>Checksum</td>
<td>Modulo 10</td>
</tr>
</tbody>
</table>

Table C.1: TLE format for first line
<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Line 2</td>
<td></td>
</tr>
<tr>
<td>03-07</td>
<td>NORAD ID</td>
<td></td>
</tr>
<tr>
<td>09-16</td>
<td>Inclination</td>
<td>Degrees</td>
</tr>
<tr>
<td>18-25</td>
<td>RAAN</td>
<td>Degrees</td>
</tr>
<tr>
<td>27-33</td>
<td>Eccentricity</td>
<td>Decimal point assumed</td>
</tr>
<tr>
<td>35-42</td>
<td>Argument of perigee</td>
<td>Degrees</td>
</tr>
<tr>
<td>44-51</td>
<td>Mean anomaly</td>
<td>Degrees</td>
</tr>
<tr>
<td>53-63</td>
<td>Mean motion</td>
<td>rev/day</td>
</tr>
<tr>
<td>64-68</td>
<td>Revolution number at epoch</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>Checksum</td>
<td>Modulo 10</td>
</tr>
</tbody>
</table>

Table C.2: TLE format for second line