Flight dynamics multi-mission software development for optical link planning and execution

Giacomo Dal Toso
Author

Giacomo Dal Toso gidt@kth.se
Master’s Programme, Aerospace Engineering, 120 credits
KTH Royal Institute of Technology

Supervisors

Dr. Gunnar Tibert
KTH Royal Institute of Technology

Frederic Schoutetens
DLR GSOC Flight Dynamics

Date: 21 August 2023
Abstract—The Generic Planning Tool (GPT) is a new software package being developed by the Flight Dynamics team at DLR. In an era where laser communications are becoming more and more relevant to data transmission for space missions, the GPT’s purpose is to compute highly accurate visibility windows and provide a wide variety of support information for both satellite-to-ground and inter-satellite links. What sets the GPT apart from previous products, is its shift from mission-specific to multi-mission and being able to accept various orbit and attitude data formats, thus enabling the support of multiple missions from DLR and external clients with flight dynamics information for mission planning applications. Its two main components are the core libraries written in Fortran, which serve as the powerhouse for the orbital mechanic’s computations, and the microservice architecture, enabled by JSON input/output files and Python scripts, which implement an automatic request-response service accessible over the network. This thesis will present why, how, and which GPT software functionalities were developed and tested during the internship at the German Space Operation Center.

I. Introduction

The Deutsches Zentrum für Luft- und Raumfahrt (DLR) is the German Center for Aviation and Space Research and Development in the fields of aeronautics, space, energy, transport, security, and digitalization with its 55 institutes and more than 10,000 employees. The German Space Agency is part of DLR with the role of managing and developing the German Space Program. Its industrial policy mandate is to promote the commercialization of space technologies to “benefit people on Earth and improve the quality of life in Germany, Europe, and the rest of the world”. [1] DLR collaborates with the European Space Agency (ESA), the National Aeronautics and Space Administration (NASA), and many more national and international industries and space agencies. Some of the most famous missions that DLR runs or collaborates with are Mars 2020, Insight, Ariane 6, Galileo, International Space Station, Mars Express, and TerraSAR-X.


Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEM</td>
<td>Attitude Ephemeris Message</td>
</tr>
<tr>
<td>AOCS</td>
<td>Attitude and Orbit Control System</td>
</tr>
<tr>
<td>AOL</td>
<td>Argument of Latitude</td>
</tr>
<tr>
<td>CCSDS</td>
<td>Consultative Committee for Space Data Systems</td>
</tr>
<tr>
<td>CPF</td>
<td>Consolidated Prediction Format</td>
</tr>
<tr>
<td>CRF</td>
<td>Celestial Reference Frame</td>
</tr>
<tr>
<td>CRS</td>
<td>Conventional Celestial Reference System</td>
</tr>
<tr>
<td>DLR</td>
<td>Deutsches Zentrum für Luft- und Raumfahrt</td>
</tr>
<tr>
<td>ECEF</td>
<td>Earth-Centered Earth-Fixed</td>
</tr>
<tr>
<td>ECI</td>
<td>Earth-Centered Inertial</td>
</tr>
<tr>
<td>G2G</td>
<td>Galileo Second Generation</td>
</tr>
<tr>
<td>GNSS</td>
<td>Global Navigation Satellite System</td>
</tr>
<tr>
<td>GPT</td>
<td>Generic Planning Tool</td>
</tr>
<tr>
<td>GSL</td>
<td>Ground-to-Satellite Link</td>
</tr>
<tr>
<td>GSOC</td>
<td>German Space Operations Center</td>
</tr>
<tr>
<td>ICRF</td>
<td>International Celestial Reference Frame</td>
</tr>
<tr>
<td>ICRS</td>
<td>International Celestial Reference System</td>
</tr>
<tr>
<td>IERS</td>
<td>International Earth Rotation Service</td>
</tr>
<tr>
<td>ISS</td>
<td>International Space Station</td>
</tr>
<tr>
<td>ITRF</td>
<td>International Terrestrial Reference Frame</td>
</tr>
<tr>
<td>LCT</td>
<td>Laser Communication Terminal</td>
</tr>
<tr>
<td>LEOP</td>
<td>Launch and Early Orbit Phase</td>
</tr>
<tr>
<td>OEM</td>
<td>Orbit Ephemeris Message</td>
</tr>
<tr>
<td>PINTA</td>
<td>Program for INteractive Timeline Analysis</td>
</tr>
<tr>
<td>TAI</td>
<td>International Atomic Time</td>
</tr>
<tr>
<td>TCB</td>
<td>Barycentric Coordinate Time</td>
</tr>
<tr>
<td>TDB</td>
<td>Barycentric Dynamical Time</td>
</tr>
<tr>
<td>TDT</td>
<td>Terrestrial Dynamical Time</td>
</tr>
<tr>
<td>TLE</td>
<td>Two Line Element</td>
</tr>
<tr>
<td>TOD</td>
<td>True Of Date</td>
</tr>
<tr>
<td>TRF</td>
<td>Terrestrial Reference Frame</td>
</tr>
<tr>
<td>TRS</td>
<td>Conventional Terrestrial Reference System</td>
</tr>
<tr>
<td>UTC</td>
<td>Universal Time Coordinated</td>
</tr>
</tbody>
</table>
This Master’s thesis was linked to an internship in the DLR site of Oberpfaffenhofen, that together with the site at Köln, is one of Germany’s largest research centers with more than 1,800 employees in 13 scientific facilities. [2] The work was carried out in the Space Flight Technology department, part of the German Space Operations Center (GSOC). The department consists of four groups. “Global Navigation Satellite System (GNSS) Technology and Navigation” focuses on GNSS receiver technology, formation flying of spacecraft, and high-precision navigation with GNSS. “On-orbit Servicing and Autonomy (OSS)” focuses on satellite service missions with intelligent and autonomous onboard systems. “Space Situational Awareness (SSA)” focuses on space debris assessment and collision avoidance. These three groups are mainly oriented toward Research and Development. The group of Flight Dynamics, in which the five-month-long work was conducted, mainly supports both DLR’s and external client’s missions, ground station antennas, and mission planning systems in the areas of orbit and attitude dynamics and Orbit Determination (OD). [3]

With its strong knowledge of astrodynamics, guidance navigation and control, and mission analysis, the Flight Dynamics team provides services regarding orbit and attitude determination, orbit prediction, orbit control, collision avoidance, and re-entry. The Flight Dynamics team develops in-house and operates many flight-proven software tools for the two main services: mission preparation and operations support. For the former, the team provides feasibility studies on orbit or formation control and runs mission analysis on Launch and Early Orbit Phase (LEOP) operations. For the latter, the team works on orbit determination, in-flight Attitude and Orbit Control System (AOCS) and GNSS characterization, helps in orbit maneuver planning in Low Earth Orbit (LEO) and Medium Earth Orbit (MEO), supports attitude operations and collision avoidance analysis. [4] It is in the framework of the development of a new software tool for this Operations Support service that the internship took place.

A. Problem definition

GSOC at DLR is shifting from mission-specific software tools that were designed and developed tailored to a specific spacecraft or mission, to multi-mission software able to handle different missions orbiting around the Earth. The long-term goal at DLR is to have both the Flight Dynamics team and the entire Ground Stations Network of DLR move towards generic and highly automated software architectures. The main problem found in developing the Generic Planning Tool was to adapt previous versions of software (designed for a single mission) to accept different orbits or attitude formats, and a multitude of constraints.

The mathematical challenges were to be able to accept a heterogeneous group of orbit and attitude data. These can vary from analytical orbit data of old heritage as the Two Line Element (TLE), to international Consultative Committee for Space Data Systems (CCSDS) standards as the Orbit Ephemeris Message or newly defined NASA Consolidated Prediction Format specifically designed for the highly-precise optical communications. [6] This was possible thanks to extensive libraries of the Flight Dynamics team that enable to convert those formats between each other or into the internal proprietary format of “database orbit record”. The feature of being able to work with both satellite-to-ground and intersatellite link planning options was inherited from existing software. Both spacecraft and ground stations are treated like orbiting objects. For the latter, the orbit lies along the Earth’s surface.

The main customer of the Generic Planning Tool will be the DLR GSOC Mission Planning team, responsible for “design, development, and maintenance of Mission Planning and Scheduling software systems”. [7] Its main focus is on highly-automated planning systems for Earth-Orbiting satellites for communication, observation, and human spaceflight. Mission Planning works at the core of the Space and Ground Segment interacting with the Ground Station for the links with the
Spacecraft, with Flight Dynamics for orbital information and for visibility windows lists, and with the customer to execute its requests, as shown in Figure 1.

At the center of the Mission Planning is the operator’s Graphical User Interface called Program for INteractive Timeline Analysis (PINTA). Its key purpose is to serve as an anchor tool for a mission planning engineer when working with a mission timeline. [8] In the last years, at DLR, the focus has been to increase the portability and multi-mission capabilities of PINTA, integrating it into a web viewer called PintaOnWeb which will use the Generic Planning Tool (GPT). The user can trigger the calculation of the visibilities on demand. The process takes place using the latest orbit and attitude information of the satellites. In the case of a ground station, coordinates can be given in both geodetic and Cartesian coordinates, as shown for Almeria optical ground station in Spain in Figure 2. After receiving the request, the GPT’s Visibility Service calculates all possible visibilities between the two Laser Communication Terminal (LCT)s. [5] These visibilities are then loaded into PINTA and displayed on the Mission Planning timeline, as shown in Figure 3.

The second service of the GPT is called Link Support Service, and as of June 2023 currently under its design phase. Once completed, its goal will be to provide the Coarse Pointing Assembly for LCT and compute the Chebyshev coefficients of the satellites’ orbits so that they can be sent later to the satellites. The Visibility Service and the Link Support Service communicate with PINTA environment through a Message Queue Telemetry Transport (MQTT) based microservice. This will be explored in detail in section III-B.

B. Project drivers and future applications

In the next few years, it is reasonable to assume that Laser-optical technology will have an increasing role in spacecraft operation. This is thanks to many new technology demonstrators that are being developed in Europe and in collaboration between DLR and private companies. Some examples of future missions that will see an increase in Laser Optical applications are discussed below.

Europe’s first-generation Galileo constellation is the world’s most precise satellite navigation system. Galileo Second Generation (G2G) will enable better performance and expand the range of services. To improve the estimation of clocks and orbits, G2G’s new capabilities will include laser range measurements, which deliver a ranging accuracy down to under a centimeter, significantly better than the half meter available from radio ranging. [9] NASA’s Orion spacecraft during the mission of Artemis II will be equipped with a Laser Communication System, known as O2O, with the purpose of transmitting high-resolution images and videos, sending data back to Earth with a downlink rate of up to 260 megabits per second. This will enable live, 4K ultra-high-definition video to and from the Moon. [10] The involvement of Laser Communications in the Artemis Missions is an indicator of the increasing importance that this technology will have in the future.

The Generic Planning Tool is currently being developed at DLR GSOC by the Flight Dynamics team to satisfy the need for planning systems that work with optical communications. This work on a narrower beam width with respect to traditional bands (like K- and Ku-bands) thus requiring higher accuracy in predicting satellite positions.
The purpose of the GPT is to assist in the planning of laser communications between two terminals that can either be a ground station and a satellite, or both satellites. The near-future applications and projects where the GPT will be tested are a CubeSat orbiting in LEO, equipped with an optical laser terminal to transfer high-resolution pictures, and commercial payload installed outside the European Module on the International Space Station (ISS). Test cases demonstrating how the GPT supports those missions will be discussed in section VIII.

C. Aim
There were several objectives defined before the beginning of the internship. They included:

i) further expanding the prototype’s orbit and attitude input data set;

ii) testing and validating those new features as well as the old ones, to have a fully functioning software ready to be released;

iii) starting the design of the Link Support Service following the requirements of the Mission Planning team.

Once the literature review was concluded, those were redefined to include specific goals and tasks. This will be discussed in detail in section III-D.

D. Benefits, ethics, and sustainability
The main benefit of the Generic Planning Tool will be to enable a new level of automatization for operation planning. With the automatic request-response over the MQTT protocol. As the Microservice architecture, Mission Planning in GSOC will have the visibility windows and support information automatically generated by the software of Flight Dynamics, removing the necessity of time-consuming direct interaction between engineers and analysts.

Being as generic as possible, the software can accept a multitude of inputs as orbit or attitude data, as well as multiple types of constraints. The shift of approach from mission-specific software to multi-mission software will enable Flight Dynamics to support multiple missions and offer services to external companies and potential customers that use different data formats.

Also, the constraints that can limit the LCT visibility now include fixed exclusion zones around the sphere of the LCT, link-blocking fixed and flexible masks defined as polygons, and solar panels. Many items such as parabolic radio-antennas or others can be defined as polygon masks and other mission-specific constraints can be easily added if required.

E. Delimitations
One of the main delimitations of the Generic Planning Tool as envisioned as of 2023, is that it was designed and developed for high precision orbit propagation for satellites orbiting around the Earth, both for low and high orbit. Interplanetary missions were not considered. The main goal for the GPT is to be a multipurpose and as generic as possible tool, though mission-specific requirements could be developed in additional packages once necessary.

F. Outline
Section II will cover the project management aspects of the thesis together with its timeline. Section III will discuss the
II. Internship as Project: Management & Planning

The thesis work was characterized since the beginning by a project-like structure, with goals, requirements, and external validation, together with strong project planning. For project management, a combination of Agile and Waterfall (or Traditional) project management techniques was used. The Waterfall method was used to define an approximately flexible schedule that could encompass all five months. The final deadline was known, as the end of the internship, and the resources were represented only by my working hours. The stakeholders could be seen as the Flight Dynamics team and the academic supervisor, and the customer as the company supervisor, while both team and project manager responsibilities were covered by me. Waterfall management was key in defining the internal deadlines and scope month by month. Inside each monthly group, the workload was defined weekly with the use of the Agile method, usually defining subgroups of weeks. Finally, the workload organized day by day was decided using the Kanban technique. Figure 5 shows how the Waterfall and Agile methods (in green and blue, respectively) were combined in the project.

A. Agile method

The Agile method was useful when scheduling single packets of work that corresponded to enclosed new features to be added to the Generic Planning Tool. Over the extended period of time of the internship, the deliverables could be considered numerous and with quick and consecutive due dates. Some of the Agile Principles were made key pillars of the thesis work method. Firstly was to satisfy the customer, represented by the company supervisor. Secondly, delivering working software quickly was implemented trying to divide the new features to implement in increments, followed by a sort of informal delivery represented by the successful end of a debugging period where the software was checked.

The same can be said about the principle of focusing on working software, though in this case a great deal of attention was given to the documentation, the key to the transfer of know-how after the end of the internship. Also, simplicity was always kept in mind with the goal of “maximizing the amount of work not done” to keep the code clean and essential as long as it worked correctly. In the framework of the Agile method, the Scrum process was adopted to have increments of a somewhat variable length of two to three weeks. Inside those sprints, the workload was distributed to achieve some sort of delivery at the end of each of those. Examples would be the design and implementation of the subroutine that could integrate the Attitude Ephemeris Message that lasted for two weeks, or the three weeks of time spent on the Python Microservice development: one for studying the existing code, one for implementing and one for running the tests. As one of the most common risks with the Scrum method is to incur into a stall, due to poorly defined tasks or specific deadlines, the Waterfall planning helped in drawing an overview path for the five months as a whole.

In project management, the Kanban board can help divide the tasks into three categories: (i) to-do, (ii) in progress, and (iii) done. This usually helps in limiting the work in progress, thus increasing speed. Flexibility could also be improved as, if late, tasks could be postponed on less busy days, and if ahead, it was possible to start the tasks of the following day since they would already have been defined. Kanban’s concepts were adopted and used inside an Excel sheet to define the workload of every single day. At the end of the week, the to-do list was defined day-by-day, and labeled as things to do for the Generic Planning Tool, DLR meetings, and thesis writing process work.
The product backlog is a common characteristic of the Agile method that groups all tasks and sub-tasks that will need to be implemented. It is created at the beginning of the project, and it is useful as it provides a pool of tasks from which the team can choose some to start working on a specific increment, following a priority list agreed upon with the customer. This was created at the beginning of the internship, after the in-depth study of the Fortran code. Thus, it was possible to identify features that were only partially implemented and the sub-tasks necessary for completion. The product backlog was proposed to me and prioritized together with the company supervisor. An example of a task was the subroutine that generates ephemeris from CPF data. Its sub-tasks were creating a READ CPF function, considering the reference frames of UTC and GPS, and finally testing and debugging. Also, Sprint (or Increment) backlogs are used for the highest priority tasks inside the increment itself, and roughly estimated for the following increment. Following the example of CPF, they were later defined as studying the CPF documentation, exploring the possibility of converting the CPF to the OEM and later using the already existing ephemeris conversion tool from OEM, adding an orbit frame converter from ITRF2000 to EME2000, and finally test the CPF with test cases both in CPF and OEM.

### B. Timeline and organization

The first period of the internship, up to approximately one month, consisted mainly of the literature review including both operation papers and code review. It was focused on both internal manuals for software development and testing and general documentation of the previous software tools from which the GPT was derived. It also included the study of the existing and relevant Fortran code, its structure, and all its functions and subroutines.

After that, the task of debugging a problem raised in the previous months by Mission Planning was considered a good milestone to prove good mastery of the code. Then debugging of the existing code was the key task with respect to those subroutines responsible for implementing the TLE, Nadir Pointing, and OEM. These are discussed in section VI-A. Then CPF and AEM were integrated into the GPT code. The latter is described in detail in V-C. Also, the LCT displacement (see section V-A) was implemented together with the debugging and validation of OPM and the Reference Orbit.

The official code tests were carried out with the goal of reaching a code coverage of up to 90% of all the Fortran core code. It was followed by the study and review of the Python scripts of the microservice interface. After the official testing of the Python code and the microservice infrastructure, the Link Support Service architecture design phase was started. This was the last task of the internship.

The internship started on the 1st of February 2023 and finished on the 30th of June 2023 with a total of 22 weeks. Around halfway through, in the middle of April, a debrief over the first part of the internship and its milestones was presented to the

### III. Software architecture and state of the art

This chapter presents the Generic Planning Tool description and design characteristics. The first part contains an overview of the software as a whole, including its framework, as discussed in section III-A. What a microservice is and how it is useful for the GPT is explored in section III-B and the GPT design is introduced in section III-C. In section III-E, the state of the art of the software is described, which clearly defines what was already implemented and what were contributions made during this thesis.

#### A. Overview

The overall architecture of the Link Planning System can be seen in Figure 7. In this case, it is assumed that the link is to be established between one ground station and a spacecraft. In the Mission Operation Center, the Mission Planning (MPS) is connected with the Flight Dynamics services, i.e. the GPT. The user can interact with the Mission Operation Center through one integrated interface with the link planning system. At the center of any space mission, the two main components are the space segment, with its main protagonist, the spacecraft, and the ground segment, with the Optical Ground Station, the Mission Operation Center, and the Spacecraft Control Center, elements that are highlighted in blue in Figure 7.

When the need for planning or executing links arises, the user can interact with Mission Planning software (called PINTA), which will show all feasible visibility windows between the spacecraft and the ground station. To have the ground station coordinates, the user can manually input those through its specific interface (see Figure 2) and get the orbit and attitude data of the spacecraft through the Spacecraft Control Center and the Flight Dynamics Orbit Service. When the data are in PINTA, the software forwards those with a MQTT base microservice, discussed below in section III-B, to the Generic Planning Tool which in turn, after the visibility windows are computed, forwards back the results to PINTA.

#### B. Microservice architecture

The Generic Planning Tool aims to support Mission Planning with a Message Queue Telemetry Transport (MQTT) based microservice.

Microservices are an architectural style for building applications. Inspired by Unix’s philosophy “Do one thing and do it well”, what sets a microservice’s architecture apart from more traditional, monolithic approaches is how it breaks an application down into its core functions. Each function is
called a service and can be built and deployed independently, meaning individual services can function (and fail) without negatively affecting others. Microservices can communicate with each other through network protocols. Applications built in this way can be more fault-tolerant, less reliant on a single-service bus. Microservices can communicate through application programming interfaces (APIs) thus letting each microservice be built in its own programming language. [11]

The benefits of using microservices are fast and reliable product generation and delivery, scalable software, an increase in repeatability and an increase in transparency through logging and error handling. In the Flight Dynamics group, microservices provide on-demand, real-time services to the customer that can be accessed over the network and can serve different clients for different tasks with no compromise on performance. [12] One of those microservices that is being developed is the Generic Planning Tool itself.

The microservice core of the flight dynamics libraries, upon which all calculations are performed, is written in Fortran, in order to use its heritage source code and powerful characteristics. The modern interfaces which make the connection with MPS possible require modern languages, thus Python is used. Another key element of the GPT is its containerization using Docker. Its use allows creation of highly-scalable containers with all the necessary libraries of code that allows the tool to work in different operating systems. [12]

C. Visibility and Link Support Services

The Generic Planning Tool’s two main services’ flowcharts are shown in Figure 8. [12] The Visibility Service with its three main blocks is highlighted in blue. The request is done through a JSON file which contains configuration data, orbit and attitude information as well as constraints for both objects. It is followed by the Visibility Calculation segment, where the visibility windows are found, and the results are checked against the constraints. Finally, the Response (JSON format) is created with a list of all the feasible visibilities, ready to be sent back to PINTA thanks to the microservice. The green rectangle represents the first draft of the design of the Link Support Service. The request is similar to the visibility service with the addition of the Link List, a list of links from Mission Planning that are selected to be executed in the near future. During the Visibility Calculation, the Coarse Pointing Assembly is computed. The Orbit Generator creates Ephemeris Data or the Chebyshev coefficients representing the object orbits. Similarly to the Visibility Service, the Response (JSON format) is created but this time it will contain Chebyshev data, CPA, and the Ephemeris File depending on what the user has requested.

The first service, the Visibility Service, provides primarily visibility windows. The user requests visibility opportunities for a specified time window and provides several inputs such as orbit, attitude, configuration, and constraints data for both LCTs. After the computation is finished by the Visibility Service, the response is sent back through a MQTT protocol with a JSON file containing the visibility windows list. Each window is provided with the start and stop times of the link, and optionally with the Coarse Pointing Assembly angles and support information. [5]

The second service, which with the work of this thesis entered its design phase, is called Link Support Service. The user, with the same input data as for the Visibility Service, can request orbit data for a specific visibility window. The response will provide this for both LCTs as ephemeris or Chebyshev polynomials, together with an update of the CPA angles.
D. Internship objectives

The aim of the internship was to implement and integrate successfully new features that could expand the capabilities of the existing prototype of the Generic Planning Tool. The implementations included new orbit data formats such as the Consolidated Prediction Format (CPF), attitude data as the AEM, displacement of the LCT from the center of mass, and the initial design of the Link Support Service. The debugging and official testing, in addition to those above, included Orbit Ephemeris Message (OEM), internal Database Orbit Record for the orbit data, Euler Angles offset, Look-up-tables and Nadir-Pointing for the attitude data, and the microservice interface in Python that could convert both ways the JSON files to a simple text file. The project success requisite was set to have those new features developed, integrated, and tested by the 1st of June 2023 and a first iteration of the Link Support Service design ready by the end of June 2023 and thus by the end of the internship itself. This report will focus on documenting the software development connected to the Visibility Service.

E. State of the art

In this section, the state of the art of the software describes and clearly defines what was already implemented and what were the contributions of this thesis.

The existing prototype of the GPT (v1.0.2) as of January 2023 was released for the Visibility Service. It proved successfully the request-response exchange between the Mission Planning service and the Flight Dynamics service. The work of this thesis was focused on introducing full functionality to the Visibility Service with additional orbit and attitude data options as input and on testing the existing and newly developed features.

What was already implemented and tested was the Microservice sequence, one satellite’s orbit data in the form of a TLE and the default Nadir-Pointing attitude, and a ground station as a second object.

The software was already able to work with the following orbit data formats and types, and attitude data formats: Ground Station, BOX, OPM, OEM, Database Record, Reference Orbit,
Attitude Bias Offset (Roll, Pitch, and Yaw), and Look-up-table. However, these were not yet tested.

The main implementations that are going to be presented in this report were the development and testing of functionalities of the GPT able to integrate into the workflow the Consolidated Prediction Formats for the orbit data formats, and Attitude Ephemeris Message for the attitude data formats. Debugging and testing were necessary for Orbit Ephemeris Message and Two Line Elements for the orbit data, and Nadir-pointing and Look-up-table for the attitude data format.

IV. Mathematical description

In sections IV-A and IV-B the main reference frames and time systems will be explored. A thorough comprehension of the differences between spatial and time frames is important as the goal of the GPT is to be able to handle different reference frames and time systems. In section IV-C, the Chebyshev Polynomials are discussed from a mathematical point of view. This is followed by section IV-D which explains how they are used in a common method of interpolating ephemeris data points. The main mathematical and orbital parameters that are computed in the GPT are presented in section IV-E. Section IV-F and section IV-G respectively explain the main orbit and attitude data formats implemented.

A. Reference systems and frames

There are two main types of reference systems in use, the first is based on celestial points that are approximated fixed thus providing a quasi-inertial system, and the second is co-rotating with Earth. Note that Reference Systems and Reference Frames are two different concepts. The first one is a theoretical definition with models and standards. The second one is a “practical implementation” through observations and a set of reference coordinates.

1) Conventional Celestial Reference Systems: Conventional Celestial Reference System (CRS) is quasi-inertial, centered at Earth’s center of mass. Its x-axis is along the mean equinox at J2000.0 epoch, the z-axis is perpendicular to the equator plane at J2000.0 epoch, and the y-axis follows the right-hand rule. It is usually referred as Earth-Centered Inertial (ECI) frame. Its practical implementations are known as Celestial Reference Frame (CRF) where the coordinates are determined by extragalactic radio sources. The mean equator and equinox were defined by the International Astronomical Union (IAU) in 1976 with 1980s nutation series. [13]

EME2000 is a CRF, so it does not rotate with respect to the stars, and it has an almost non-accelerating origin (though the velocity is not zero). The x-axis is towards the Vernal equinox, the y-axis is normal to the mean equator of date at epoch J2000 TD, approximately Earth’s spin axis at that epoch. It is usually also called J2000 or ICRF as the differences are very small.

ICRF is defined by adopted locations of 295 extra-galactic radio sources, and it is very similar to J2006 with a difference of 0.1 arc second, with no conversion usually required. [14]

The True Of Date (TOD), or True of Date Earth Equator, is a quasi-inertial coordinate system reference of Earth’s true equator at the current epoch. The current epoch is defined by the Spacecraft epoch. TOD has the x-axis that points towards the true vernal equinox at the current epoch, the y-axis points towards the true rotation axis at the current epoch and the z-axis follows the right-hand rule. Thus, the directions of the coordinate system axes are time-dependent and include nutation and precession effects. [15]
2) Conventional terrestrial reference systems: Conventional Terrestrial Reference System (TRS) is co-rotating with Earth in its diurnal rotation and is also called Earth-Centered Earth-Fixed (ECEF). A mathematical model takes into account the temporal variation of the point’s position due to geophysical effects such as plane motion and Earth’s tides. The x-axis is towards the intersection between the equatorial plane and the mean Greenwich meridian; the z-axis is in the direction of Earth’s rotation defined by the Conventional Terrestrial Pole; the y-axis follows the right-hand rule. [16]

Practical implementations are known as Terrestrial Reference Frame (TRF) and include:

- International Terrestrial Reference Frame (ITRF)
- World Geodetic System 84 for the GPS
- Parametry Zemli 1990 used by GLONASS
- Galileo Terrestrial Reference Frame for the Galileo GNSS

ITRF is updated every few years to have it as precise as possible. The ITRF 2000 coincides with World Geodetic System 84 for only a few centimeters, and in the GPT code the two reference frames are considered the same. The Galileo GTRF was defined as ITRF 2005.

B. Time systems

The International Atomic Time (TAI) has the initial epoch matching to 0h0m0s of UT2 scale of January 1st 1958. Its second coincide with an ET second, which is “the duration of 9192631770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of Cesium 133 atom”. [17]

It is kept by several atomic clocks around the world, thus providing a statistical time obtained with a weighted average. The difference between ET and UT2 in January 1st 1958 was of 32.184 seconds. In 1967 TAI substituted ET as the SI second. So there is a constant offset scale between TAI and ET (now called Terrestrial Dynamical Time (TDT)) of 32.184 seconds. [17]

The Universal Time Coordinated (UTC) is a compromise between TAI and UT1 (solar time), kept closer than 0.9 seconds with respect to UT1 to follow the Earth’s rotation variation. This is accomplished by adding or subtracting leap seconds from TAI, a number which is provided by the International Earth Rotation Service (IERS). This is relevant as all the time from TAI, a number which is provided by the International

The GPS Time is a continuous timescale maintained by the Galileo Central Segment and synchronized with TAI with a nominal offset below 50 nanoseconds. The GST start epoch is 0h UTC on August 22nd 1999. [17]

C. Chebyshev Polynomials: theoretical description

The Chebyshev Polynomials are a set of orthogonal polynomials defined as solutions to the Chebyshev Differential equations. [18] The \( n^{th} \) polynomial of the 1\(^{st}\) kind is defined as:

\[
T_n(x) = \cos(n \arccos x)
\]

So the first terms are:

\[
\cos 0 \theta = 1 \\
\cos 1 \theta = \cos \theta \\
\cos 2 \theta = 2 \cos^2 \theta - 1 \\
\cos 3 \theta = 4 \cos^3 \theta - 3 \cos \theta
\]

with a recurrence relation [19] of \( T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \) for a range of \(-1 \leq x \leq 1\).

Chebyshev Approximation Formula [20] is defined as:

\[
c_j = \frac{2}{N} \sum_{k=1}^{N} f(x_k)T_j(x_k)
\]

\[
= \frac{2}{N} \sum_{k=1}^{N} f(\cos \frac{\pi(k - \frac{1}{2})}{N}) \cos \frac{\pi(k - \frac{1}{2})}{N}
\]

Then:

\[
f(x) \approx \sum_{k=0}^{N-1} c_kT_k(x) - \frac{1}{2}c_0
\]

When truncated, the error of the approximation is spread smoothly over the interval \([-1,1]\). Chebyshev interpolation minimizes the problem of Runge’s phenomenon (the oscillation of the edges when using equispaced points for polynomial interpolation).

D. Representation of planetary ephemerides

Here the main concepts will be provided of the mathematical description of the interpolation of Planetary Ephemerides through Chebyshev Coefficients. The main reference document is [19], dating to 1989 when the JPL DE (Development Ephemeris) in use was DE102. While as of 2021 the data sets in use are DE440 and DE441, which have better accuracy over the Outer Planets orbital elements and longer time epochs [21], the mechanisms behind interpolating data with Chebyshev Polynomials from JPL DE have not changed.

The Jet Propulsion Laboratory (JPL) has been providing NASA and all users with high-precision planetary and lunar ephemerides since the 1970s [19] in support of spacecraft navigation. The numerical integration computes the instantaneous state of the solar system as polynomials in the form of position, velocity, and acceleration. Saving this information at each time
step would lead to a huge file in size, thus the necessity for obtaining polynomial representations of the position and velocity over a certain time span.

Chebyshev polynomials are valuable as they are stable during evaluation, they give near-minimax representation, and it is possible to estimate the interpolation error. Any given function \( f(x) \) has an approximate \( N \)th degree expansion of:

\[
f(x) = \sum_{n=0}^{N} a_n T_n(x)
\]

and when differentiated

\[
f(x) = \sum_{n=1}^{N} a_n \dot{T}_n(x)
\]

As \( a_n \) serve to define the function \( f(t) \) as a polynomial, it is necessary to determine \( a_n \).

The error (interpolation error between interpolated and integrator-supplied values, not the degree of accuracy of the integrated ephemeris with the true state of the Solar System) can be estimated. An arbitrary function has the exact representation when using an infinite Chebyshev expansion defined as:

\[
f(x) = \sum_{n=0}^{\infty} a_n T_n(x)
\]

Thus, when a function is approximated to the \( N \)th degree, the maximum error \( \epsilon \) is:

\[
\epsilon = \left| \sum_{n=N+1}^{\infty} a_n T_n(x) \right| \leq \sum_{n=N+1}^{\infty} |a_n| |T_n(x)| \leq \sum_{n=N+1}^{\infty} |a_n|
\]

Empirical evidence [19] using DE102, showed that the maximum expected interpolation error is about \( 1/10 \)th of the highest retained coefficient \( a_N \).

The inertial coordinate frame is connected to the International Celestial Reference System (ICRS) with the implementation of the Third Realization of the International Celestial Reference Frame (ICRF) (ICRF3). The orbits of the inner planets are tied to ICRF3 via Very Long Baseline Interferometry measurements of Mars-orbiting spacecraft with an accuracy of 0.2 milliarcseconds while the orbits of Jupiter and Saturn are tied to ICRF3 via Very Long Baseline Array measurements of the Juno and Cassini spacecraft [21].

The Solar System Barycenter is computed considering the Sun, the barycenter of eight planetary systems, the Pluto-system barycenter, 343 asteroids (which represent \( 90\% \) of the total asteroid-belt mass), 30 Kuiper Belt Objects (KBO) and a KBO ring representing the main Kuiper belt. With respect to the previous JPL DE, the Solar Inertial Motion has shifted of \( \approx 100 \) km due to the addition of KBO. Nevertheless, the Earth is not affected due to its orbit around the Sun and not the Solar System Barycenter, to the first order. [21]

JPL DE series are integrated using the Barycentric Dynamical Time (TDB) (defined relative to the Barycentric Coordinate Time (TCB)). All the data are time tagged in UTC. The conversion that JPL uses is to go from UTC to TAI, from TAI to TT, and from TT to TDB with this latter being represented by the conversion formula depicted in [21].

E. GPT parameters of interest

Firstly, the orbit and attitude data are read and converted or interpolated as necessary. Then the rotation matrix of the LCT local frame with respect to the orbital frame is computed taking into consideration also mounting matrices. Thus, the software is ready to compute the lines of sight: all the relative position vectors of the other spacecraft or ground station, Sun and Moon positions, etc. Those are computed in TOD and converted into the LCT frame for each spacecraft. [22]

The user can provide the Visibility Service through the JSON input file and also two rotation matrices that correlate the LCT position with respect to the Spacecraft Body Frame. \( LCT_{MOUNTING} \) expresses the LCT mounting while \( LCT_{CORRECTION} \) can be added to model discrepancies in the mounting, discovered for instance with later calibrations once the spacecraft is already in orbit. These are important because the Spacecraft Body Frame might not coincide with the mounting of the LCT; for example, it could be placed “upside down” on the “belly” of the spacecraft (as seen from LVLH if it is the body frame). The integration of these two matrices into the logic of the GPT will be discussed in section V-A.

\[
LOS_{Main \rightarrow Sun} = \frac{REL_{Sun}}{2} - \frac{R_{Sun}^{TOD}}{2}
\]

\[
LOS_{Main \rightarrow Target} = -REL_{Target}^{TOD}
\]

If no rotation matrices are present, the Line of Sight (LoS) going from the main (sc1) to the target (sc2) objects is the simple difference of position vectors (defined as TOD) which actually gives the relative position of the target with respect to the main:

\[
REL_{sc1}^{TOD} = R_{sc1}^{TOD} - R_{sc2}^{TOD}
\]

\[
REL_{sc2}^{TOD} = -REL_{sc1}^{TOD}
\]

To get the LoS in the LCT frames, \( REL_{sc1}^{TOD} \) and \( REL_{sc2}^{TOD} \) need to be multiplied by the rotation matrices in the correct
order. Taking as an example the first one, to get from TOD to the LCT frame, the sequence is:

TOD → Orbital frame (LVLH) → Spacecraft Body Frame → → Attitude Matrix → LCT Mounting.

Integrating in these calculations the displacement of the LCT from the center of mass is explained in section V-A.

The Argument of Latitude (AOL) is necessary when the spacecraft is in a circular orbit since the argument of perigee ω and the true anomaly ν are not defined. Thus, the location of the spacecraft is defined by the AOL ν, the angle measured between the ascending node \( \bar{n} \) vector and the position vector of the spacecraft \( \bar{R} \). Thus:

\[
\cos u = \frac{\bar{n} \cdot \bar{R}}{|\bar{n}|} \tag{11}
\]

The distance between Sc1 and Sc2 is defined as:

\[
\text{Dist} = |R_{Sc2} - R_{Sc1}| \tag{12}
\]

Azimuth and elevations are two angular measurements in a spherical coordinate system. Azimuth is the horizontal angle from a cardinal direction (usually north), and elevation is the angle between the object and the observer’s local horizon. Those are important output parameters that are to be provided to the LCT of both ground station and spacecraft. They are defined in the reference frame of the Laser Communication Terminal.

The azimuth of Sc2 with respect to Sc1 is defined between −180° and +180° as:

\[
AZI_{Sc1} = \arctan \left( \frac{LOS_{Sc1 \rightarrow Sc2}(y)}{LOS_{Sc1 \rightarrow Sc2}(x)} \right) \tag{13}
\]

The elevation of Sc2 with respect to Sc1 is defined between −90° and 270° as:

\[
ELE_{Sc1} = \arctan \left( \frac{LOS_{Sc1 \rightarrow Sc2}(z)}{\sqrt{LOS_{Sc1 \rightarrow Sc2}(x)^2 + LOS_{Sc1 \rightarrow Sc2}(y)^2}} \right) \tag{14}
\]

The Sun angle is defined as the angle between the LOS from the main object to the Sun, and the LOS from the main object to the second object. Thus:

\[
\alpha = \arccos \left( \frac{LOS_{Sc1 \rightarrow Sun} \cdot LOS_{Sc1 \rightarrow Sc2}}{|LOS_{Sc1 \rightarrow Sun}| \cdot |LOS_{Sc1 \rightarrow Sc2}|} \right) \tag{15}
\]

The acronyms Sc1 or Sc2 can represent one station and one spacecraft, or both spacecraft. The same computations are done to find the Moon angles. Figure 9 illustrates the Sun angle.

The grazing altitude is the minimum distance of the LOS between the two spacecraft to the Earth’s surface. It is defined as:

\[
GALT = \frac{-R_{Sc1} \times (R_{Sc2} - R_{Sc1})}{|R_{Sc2} - R_{Sc1}|} - R_{Earth} \tag{16}
\]

The grazing altitude exists if both objects are spacecraft. It is only of relevance if the LOS can get close to the Earth’s surface, so only if the angle between Sc1 and Sc2 with the vertex in the center of the Earth is greater than 90°. Otherwise, the grazing altitude is set to the minimum altitude of the two spacecraft. [22] Figure 9 illustrates the Grazing altitude.

The grazing angle of Sc1 is the angle between the LOS and the Earth’s surface seen from the Sc1. So it is:

\[
\beta = \arctan \left( \frac{GALT}{\sqrt{R_{Sc1}^2 - (R_{Earth} + GALT)^2}} \right) \tag{17}
\]

The same equation applies to the grazing angle of Sc2.

The Point Ahead Angle (PAA) is the angle between the LOS at the moment of transmission and the LOS at the moment of signal reception (Point Ahead LOS). The PAA is the angle offset with respect to the LOS that the terminal needs to point to so that the transmitted signal arrives at the receiver. [22] The PAA needs to be computed, as it could be a physical constraint of the driver that moves the LCT while following the other object. Thus, it is a constraint that can be given in the JSON input file.

The time of transmission at the computation step \( i \), the Signal Transmission Time (STT\(_i\)), is calculated from the distance between the two terminals and the speed of light. The position of Sc2 at time \( t_i \) and \( t_{i+1} \) is \( P(t_i) \) and \( P(t_{i+1}) \). Then linear interpolation is used to compute the position \( P(t_i + STT_i) \). From the classic formula:

\[
\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0} \Rightarrow y = y_0 \left( \frac{x - x_0}{x_1 - x_0} \right) + y_1 \left( \frac{x_1 - x}{x_1 - x_0} \right) \tag{18}
\]

Which brings:

\[
P(t_i + STT_i) = P(t_{i+1}) \cdot \frac{STT_i}{\Delta t} + P(t_i) \cdot \left( 1 - \frac{STT_i}{\Delta t} \right) \tag{19}
\]

Now that the position of Sc2 at the moment of signal reception is known, the relative position with respect to Sc1 can be calculated together with its Point Ahead Line of Sight (LOS\(_{Sc1 \rightarrow Sc2} \rightarrow \)) using the method described above. Figure 10 illustrates the Point Ahead Angle in case of Ground-to-Satellite Link (GSL) without considering the LCT displacement (for that, see section V-A). Note that the red line with its arrows represents the signal at departure from the Ground Station and at the arrival to the main object.

Finally, the PAA can be computed:

\[
PAA_{Sc1} = \arccos \left( \frac{LOS_{Sc1 \rightarrow Sc2} \cdot LOS_{Sc1 \rightarrow Sc2}}{|LOS_{Sc1 \rightarrow Sc2}| \cdot |LOS_{Sc1 \rightarrow Sc2}|} \right) \tag{20}
\]
F. Orbit data formats

The orbit data formats can be described in three main categories. First, there are those which provide analytical information, thus the orbit has to be propagated. Then there can be numerical information that needs to be propagated into the future, too. The third kind is orbit ephemeris, a set of orbit data that spans epochs from which it is possible to interpolate the desired data. In this section, the main characteristics of the data formats will be discussed, as this implementation in the GPT is explained in section V.

- Analytical
  - TLE: Two Line Elements
- Numerical Integration
  - OPM: CCSDS Orbit Parameter Message
- Orbit Ephemeris
  - CPF: Consolidated Prediction Formats
  - OEM: CCSDS Orbit Ephemeris Message

1) TLE: Two Line Elements: Two Line Elements are “general perturbation mean elements constructed by a least squares estimation from observations of a satellite’s orbit”. [23] The mean values are built by removing periodical perturbation forces such as the oblateness of Earth, atmospheric drag, and lunar and solar gravitational effects. Once the TLE is created, and eventually distributed to the users, in order to have the orbit of a spacecraft those elements need to be considered again, and the orbit must be propagated until the desired time span. They generally provide enough accuracy for a day after they are published and in the case of optical communication, it is thought that it might not be sufficient.

A derived format is an ASCII-formatted text file containing TLEs for multiple objects with the optional name of the spacecraft in the first line (line 0). The example that follows is the TLE of the Zarya Module on the ISS from a TLE file requested from Celestrack.org on the 24th of February 2023:

```
ISS (ZARYA)
1 25544U 98067A 23055.36715531 .00017001 00000+0 31285-3 0 9996
2 25544 51.6387 167.3561 0005418 22.9195 99.0673 15.49284681384295
```

The format description, from [24], is displayed in Table I.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Line Number of Element Data</td>
</tr>
<tr>
<td>03-07</td>
<td>Satellite Number</td>
</tr>
<tr>
<td>08</td>
<td>Classification (U=Unclassified)</td>
</tr>
<tr>
<td>10-11</td>
<td>International Designator (Last two digits of launch year)</td>
</tr>
<tr>
<td>12-14</td>
<td>International Designator (Launch number of the year)</td>
</tr>
<tr>
<td>15-17</td>
<td>International Designator (Piece of the launch)</td>
</tr>
<tr>
<td>19-20</td>
<td>Epoch Year (Last two digits of year)</td>
</tr>
<tr>
<td>21-32</td>
<td>Epoch (Day of the year and fractional portion of the day)</td>
</tr>
<tr>
<td>34-43</td>
<td>First Time Derivative of the Mean Motion</td>
</tr>
<tr>
<td>45-52</td>
<td>Second Time Derivative of Mean Motion (Leading decimal point assumed)</td>
</tr>
<tr>
<td>54-61</td>
<td>BSTAR drag term (Leading decimal point assumed)</td>
</tr>
<tr>
<td>63</td>
<td>Ephemeris type</td>
</tr>
<tr>
<td>65-68</td>
<td>Element number</td>
</tr>
<tr>
<td>69</td>
<td>Checksum (Modulo 10)</td>
</tr>
</tbody>
</table>

The TLE reference frame is an Earth-centered inertial called TEME: True Equator Mean Equinox. The z-axis points towards the true rotation axis at the current epoch and the x-axis points towards the mean vernal equinox at the current epoch. A detailed definition can be found in [25].

2) OPM: CCSDS Orbit Parameter Message: An OPM contains the position and velocity of a single object at a specified epoch, with optionally Keplerian elements. It is recommended as suited for a mix of automated and human interaction and does not require high-fidelity dynamic modeling.

OPM has a header with information about the OPM version, comments, creation date, and originator. The OPM Metadata has as mandatory entries the Object Name, its ID, the origin of reference frame (always assumed Earth in the GPT), the Reference Frame¹ and the Time Systems². The OPM Data has the Epoch, the Position, and Velocity vector elements in units [km] and [km/s]. When chosen to provide optional Osculating Keplerian Elements, none or all parameters defined in [26] must be given.

¹Chosen between EME2000, GCRF, GIC, ICRF, ITRF2000, ITRF-93 & -97, MCI, TDR, TEME, T0D.
²Chosen between GMST, GPS, MET, MRT, SCLK, TAI, TCB, TDB, TCG, TT, UT1, UTC.
The OPM requires the use of a propagation technique, allows modeling of any number of maneuvers, and contains an optional covariance matrix to express the uncertainty of the orbit state. [26] For the propagation, the OPM can have additional optional information such as the spacecraft’s maneuver planning data, mass, solar radiation pressure area and coefficient, drag area, and coefficient.

3) CPF: Consolidated Prediction Format: This ephemeris-like orbit data set format comes from the International Laser Ranging Service, and it contains the position and velocity of a single object at multiple epochs within the time span. The baseline for interpolation of the CPF predictions is a 10-point (9th order) Lagrange interpolation algorithm, which allows for records with variable time spacing. [27]

Prior to June 2008, the satellite laser ranging stations used the standard “Tuned IRV” prediction format. The ILRS Predictions Formats Study Group created a Consolidated Prediction Format (CPF) for laser ranging that accurately predicts positions and ranges for a much wider variety of laser ranging targets than had been previously possible. [6] The three target classes are:

1) Passive retro-reflectors (including earth orbiting, in lunar orbit, and on the Moon);

2) Asynchronous transponders;

3) Synchronous transponders.

As described in [27], CPF format 2.0, together with comments, is composed of data headers and ephemeris entries. Each line is categorized by a parameter called record type. It occupies the first two characters of each line and can be H1, H2, up to H5 for headers, or 10, 20, up to 70 for ephemeris entries. H9 and 99 represent respectively the end of the header and the ephemeris trailer.

The header contains information about the date and time of production of the ephemeris, the format version (1 or 2), the satellite ID as COSPAR ID, SIC or NORAD ID, the start and end date and time of the ephemeris, and other minor information. The main ephemeris record types are “10” for Position, “20” for Velocity, and “30” for Corrections, followed by other information such as transponder data, offset from the center of the main body, rotation angle of offset, and Earth orientation if space-fixed reference frame is in use. An Earth-orbiting artificial satellite CPF example, as the main object in use for the GPT, is provided in the appendix.

Each ephemeris entry includes three position coordinates and they are spread over multiple records for a day, usually the time between adjacent entries is constant (variable entry spacing is possible for high eccentricity satellites). Typical values are 1 minute for LEO, 15 minutes at the Moon, and hours for the planets. For orbits around the Moon or further, the Euclidean Space assumptions do not hold anymore due to distances and masses. The relativistic corrections are needed as the difference between the true range and Euclidean distance gives error ranges up to hundreds of microseconds for the Moon. Omitting relativistic correction gives range errors of about 50 nsec. The stellar aberration effects are necessary as the aberration is between 1 and 2 second of arc at the Moon position, 30 or more arc-seconds at Mars’ and asteroids’ position. [27]

The coordinate system used in the CPF format is ITRF. Earth orientation information is supplied in case the predictions are presented in an inertial space-fixed reference system. Position and velocity fields in the case of CPF computed with solar system barycenter have entries for vectors to fire time from geocenter to target (record 10-1) and from bounce time from target to geocenter (record 10-2).

Entry 30 is about corrections, necessary for all targets computed from a solar system ephemeris, which are relativistic and aberration. Aberration can be a light-time aberration, which applies to all targets (including Earth and satellites) and is implicit in the state vector, and stellar aberration for targets computed from solar system ephemerides. Near-Earth artificial satellites are computed in ECI and do not require stellar aberration. The in-bound and out-bound relativistic corrections are due to geodetic curvature when using a solar system ephemeris and computing in a solar system barycentric frame. For a round trip to Mars, this correction can be 200 m.

4) OEM: CCSDS Orbit Ephemeris Message: The Orbit Ephemeris Message describes the position and velocity of a single object at multiple epochs. A message can come with a covariance matrix 6 x 6. They allow higher fidelity or precision than OPM and allow interpolation instead of propagation. [26]

The orbit data is provided in the form of a series of Cartesian state vectors of position, velocity, and optionally, acceleration. An OEM consists of the following sections: header, metadata, ephemeris data, optional covariance matrix data, and optional comments.

The metadata section provides among the data the object name, its ID, the origin of the reference frame, the reference frame (chosen between ICRF, ITRF, TOD, EME2000, TDR, and GRC), the reference epoch, the time system (chosen between UTC, TAI, TT, GPS, TDB, TCB), and start and end of the time span covered by ephemeris data. Information about the recommended interpolation method and degree can be provided, but it is up to the user to decide on those.

Each set of ephemeris data, including the time tag, must be provided on a single line. The order in which data items are given shall be fixed: Epoch, x, y, z, ˙x, ˙y, ˙z, ¨x, ¨y, ¨z.

G. Attitude data formats

For the look-up-tables (as defined in IV-G3), Nadir-pointing, and constant offset angles, the heritage code was already

3It is the apparent shift in the position of stars due to the motion of the Earth in its orbit around the Sun, especially for stars away from the ecliptic [28]
there, but it was not yet tested in the framework of the GPT. The scope of the internship was to test those and add support for handling Attitude Ephemeris Message. Here, only the main characteristics of data formats will be discussed, as the new implementation is further explained in section V.

1) LVLH for satellites: The default orbit frame in the GPT is local-vertical local-horizon (LVLH) Earth pointing and is defined with its origin at the Spacecraft’s center and its axes as [29]:

   - z-axis: Vector pointing in the opposite direction to the position vector (points to center of Earth);
   - y-axis: Vector pointing in the opposite direction to the orbit normal (the orbit normal is the cross product of position and velocity);
   - x-axis: Vector perpendicular to the y- and z-axes, forming a right-handed coordinate system;

Sometimes it is also referred to as T,−N,−R, (ordered as x,y,z) as are tangential, opposite to normal, and opposite to radial vector. It is preferred to have satellites that are Nadir-pointing, and it is used for the ISS. LVLH is displayed in Figure 11.

2) Attitude frame for ground stations: The default attitude reference frame for Ground Stations is the Local Tangent Plane Coordinate (LTP) which is defined as:

   - z-axis: Vector pointing to the Zenith (or Up) direction with respect to the tangent plane to the Earth;
   - y-axis: Vector pointing to the East, tangent to the parallel passing through the Ground Station;
   - x-axis: Vector pointing to the North, tangent to the passing meridian passing through the Ground Station;
   - origin: Center of the coordinates of the Ground Station.

Note that it is not a right-hand rule coordinate frame. Figure 12 shows the default attitude frame for Ground Stations.

3) Look-up-table: A look-up-table file contains orbit-periodic attitude data. For each step’s orbit position, defined by the argument of latitude [degrees] the corresponding roll, pitch, and yaw offset angles (describing the offset from the orbital frame or local-Nadir local-horizontal frame) are provided in radians. Current implementation expects a latitude sampling of 2 degrees. [22] This means that the look-up table is based on the default Nadir-pointing frame.

4) Constant offset: Attitude can also be specified as constant offset angles from the orbit frame LVLH. It is expressed through a sequence of three Euler angles: roll pitch yaw. The values are in degrees.

5) Attitude ephemeris message: The Attitude Ephemeris Message (AEM) is one of two CCSDS-recommended Attitude Data Messages in ASCII text format. It represents attitude data for a single object and is suited for high fidelity or higher precision dynamic modeling as it allows any number of torques. It requires the use of an interpolation technique, it is fully contained as no additional information is required when inertial reference frames are specified. If local orbital reference frames are specified, they must be used in conjunction with an OEM. [30]

The data format is similar to the OEM, with a header, metadata, optional comments, and attitude data. Metadata must specify the two reference frames with respect to which the rotations transform the attitude and the rotation direction of the attitude, specifying from which frames the transformation is. The attitude types supported by the AEM are the following:

   - Quaternions (with or without derivative or rate)
   - Euler angles (with or without rate)
   - Spin (for Spin stabilized spacecraft, with or without nutation)

Quaternions and Euler angles were implemented but the derivative or the rate, being secondary attitude formats and less urgent, were left to be implemented in the future. The same was chosen for the spin. In the case of quaternions, the scalar portion placement must be specified whether it is first or last. In the case of the Euler angles, the rotation sequence of Euler angles rotates from the first to the second reference frame (123,321, etc.). The order of the transformation is from left to right, where the leftmost integer represents the first rotation axis. The AEM standard does not allow for less than three Euler rotation axis (212 or 213 with a final rotation of 0 deg is allowed, but not just a sequence dictated by “21”).

Time systems are consistent with the OEM, reference frame values can be ICRF, ITRF, EME2000, TOD, LVLH, body, or instrument frames. Time step duration can vary within an AEM but the time system cannot. The keyword of $ATTITUDEE_{DIR}$ specified the direction of the attitude rotation.
and can be A2B when the attitude parameters specify a rotation from reference frame A to B or can be B2A for the opposite case. The implementation is explained in section V-C together with its flowchart.

V. SOFTWARE DESIGN AND IMPLEMENTATION

The main work of the internship has been divided into two different parts: developing or integrating new features from scratch, discussed in this chapter, and validating and testing the already existing subroutines as well as my own work, presented in the next chapter.

The software design and the implementation of new functions or capabilities added to the GPT will be discussed in this chapter. First, the implementation of the Laser Communication Terminal displacement from the center of mass of the spacecraft is going to be presented. Then the conversion from CPF to OEM is going to be briefly presented as it made use of only pre-existing functions, and it did not represent a big portion of the work. Lastly, the subroutine that converted AEM to attitude matrices is discussed in detail.

A. Implementation of LCT displacement

The previously implemented version of the software assumed that the Laser Terminal had the same center of mass as the spacecraft Center of Mass (CoM). For small satellites, it also does not coincide, but the difference between the LCT and the CoM is negligible, compared to the width of a laser beam. For large objects, such as the ISS, the displacement is too large to make that assumption. The input data that represents the LCT displacement is given in the input files of the GPT as x, y, and z coordinates of the Center of the LCT (CoL) with respect to spacecraft’s CoM in body frame coordinates. The quickest and less intrusive implementation was determined to be the one described below.

First, the vector of the displacement is given in body frame, so it is \( \text{DISP}_{\text{SCB}} \), and it needs to be added to the other position vectors of the Main and Target objects that are in TOD \( (R_{\text{main}}^{\text{TOD}} \text{ and } R_{\text{target}}^{\text{TOD}}) \). An example might be the case of computing the relative position of the Target with respect to the LCT displacement. Already implemented in the GPT software was the relative position of the Target with respect to Main. The solution, illustrated in Figure 13, is a simple vector addition in the correct reference systems:

\[
\begin{align*}
\text{REL}_{\text{target}}^{\text{TOD}} + R_{\text{main}}^{\text{TOD}} &= R_{\text{target}}^{\text{TOD}} \\
\text{REL}_{\text{target}}^{\text{TOD}} &= R_{\text{target}}^{\text{TOD}} - R_{\text{main}}^{\text{TOD}} \\
\text{DISP}_{\text{lct}}^{\text{TOD}} + \text{REL}_{\text{lct}}^{\text{TOD}} &= \text{REL}_{\text{main}}^{\text{TOD}} \\
\text{REL}_{\text{lct}}^{\text{TOD}} &= \text{REL}_{\text{main}}^{\text{TOD}} - \text{DISP}_{\text{lct}}^{\text{TOD}}
\end{align*}
\]

Another interesting case is when considering the vector of the relative position of the Sun or the Moon with respect to one object, in this case, the Main. These vectors need to be extrapolated and checked against the constraints, as the Sun could interfere with the laser during daylight and the Moon could interfere with it during nighttime. Thus, two of the constraint inputs are the minimum allowable angle between the Line of Sight of the other object and the Line of Sight of the Sun or the Moon. Here, the mathematical construction is similar to the relative position vector of Target with respect to Main. Figure 14 illustrates the vectors.

In conclusion, the LCT vector expressed in Spacecraft Body Frame needs to be converted to TOD for each time step. When the attitude is not expressed as AEM, the conversion follows:

\[
LCT_{\text{TOD}}^{\text{disp}} = [TOD2ORB]^{T} \times [ORB2SCB]^{T} \times [ATT\_MATRIX]^{T} \times LCT_{\text{SCB}}^{\text{disp}}
\]

In this way the LCT\_CORRECTION and LCT\_MOUNTING, discussed in section IV-E, are applied later with the final position vector that already encompass the LCT displacement. A different approach needs to be followed when the attitude comes from an AEM file, as the attitude matrix is already defined as TOD2SCB (see section V-C). So the LCT position in TOD is expressed as:

\[
LCT_{\text{TOD}}^{\text{disp}} = [ATT\_MATRIX]^{T} \times LCT_{\text{SCB}}^{\text{disp}}
\]

B. Implementation of CPF orbit data

Instead of implementing from scratch a subroutine that could extrapolate the CPF data and prepare it for the GPT, it was decided to convert the CPF file to an OEM file. The version...
of the OEM is 1.0 with the default reference frame ICRF in UTC. The default data points of sample points used in the fit of the Chebyshev interpolation, as well as the number of the Chebyshev coefficients, was 12. Then, once the CPF has been converted to an OEM file thanks to the Flight Dynamics libraries, the latter could be read and the ephemeris interpolated as necessary.

C. Implementation of AEM rotation matrices

In the GPT, the AEM is supported for Euler angles and Quaternions. At first, they are linearly interpolated to obtain an ephemeris with all data points necessary for internal calculations. Then, for each time step, the attitude data is transformed to an attitude matrix such as the Direct Cosine Matrix through classical algorithms [31]. Also, the direction (A2B or B2A) is considered in this step. Figure 15 shows the flowchart of the subroutine that generates the attitude matrix from the AEM file given as input. When the attitude is expressed as an AEM, the ATT_MAT_AEM subroutine is called from the main code, and at first it checks if the AEM input data exists or not. It reads it with the use of the READ_AEM subroutine, and it returns an AEM object, that can be used by the Flight Dynamics libraries. The code checks if both the type of attitude (only Quaternions and Euler angles as of June 2023) and the reference frames are implemented. Depending on whether it is Quaternions or Euler angles, those are interpolated linearly or with a custom order Lagrange interpolation. Afterward, the Direct Cosine Matrices (DCM) are created. To have the subroutine ATT_MAT_AEM in the least invasive way possible with respect to the already existing code, the attitude matrix in DCM should be from TOD to Spacecraft Body Frame (SCB). Thus, a tree of options, shown at the bottom of Figure 15, was created to take into consideration all the possible combinations of the reference frames (TOD, ITRF, LVLH, EME2000) and conversion matrices to get TOD to Spacecraft Body Frame.

The goal was to implement the AEM with minimum disruption of the code. For example, the previous existing code, which computed a position vector of the Target with respect to the Main Frame, transformed the TOD position to LVLH (below or ORB), then to SCB, then apply the attitude matrix of Main and then multiply it with the LCT mounting and correction matrices as discussed in section V-A.

\[ R_{\text{target}} = [LCT_{\text{correction}}] \times [LCT_{\text{mounting}}] \ldots \times [ATT_{\text{MATRIX}}] \times [ORB2SCB] \ldots \times [TOD2ORB] \times R_{\text{TOD}}^{\text{target}} \]

This meant that for having the attitude matrix placed in the same position, it had to also encompass the \([ORB2SCB]\) and \([TOD2ORB]\) transformations. So the attitude needed to be expressed as rotations from TOD to SCB thus having:

\[ R_{\text{main}}^{\text{target}} = [LCT_{\text{correction}}] \times [LCT_{\text{mounting}}] \ldots \times [ATT_{\text{MATRIX}}] \times R_{\text{TOD}}^{\text{target}} \]

The supported reference frames were TOD, ITRF, LVLH and EME2000. Apart from the first case, a conversion was applied to have the attitude matrix as TOD2SCB.

An example can be the AEM defined with direction A2B, SCB as Frame A, and EME2000 as Frame B. Given that the transformation matrix available is EME2TOD, the attitude matrix is calculated as:

\[ [ATT_{\text{MATRIX}}] = [ATT_{\text{MATRIX}}^{\text{EME2000}}]^{\text{T}} \ldots \times [EME2TOD]^{\text{T}} \]

All cases are considered at the bottom of the flowchart in Figure 15.

VI. VALIDATION

One of the requirements of the Flight Dynamics team before releasing a new software version, is code validation. The official testing of both Fortran and Python codes will be discussed in section VII. In this section, two examples of the debugging process implemented for the TLE and the LCT displacement will be presented. Something similar was done for the other orbit and attitude data formats, like the OPM, CPF, Nadir-pointing, AEM, LUT, and BIAS offsets.

A. TLE debugging and validation

As a test case, a Cubesat orbiting in a LEO polar orbit at around 500 km of altitude and a DLR Ground Station were used. To validate the results of the GPT, it was necessary to compare the visibility windows with an external tool. NASA General Mission Analysis Tool (GMAT) 2018 was selected for being an open-source software for space mission design, optimization, and navigation. One interesting function is called “ContactLocator” as it allows finding visibility windows only for Satellite-Ground Station. Note that since the tool cannot take into consideration exclusion zones or particular pointing limitations of the Laser Terminal, the input parameters in the GPT were changed to be able to have the same setup of the problem.

The CubeSat laser terminal was assumed to have a full sphere of freedom to imitate the capabilities of the GMAT Event Locator. The Ground Station had the same Latitude, Longitude, and Altitude in an Ellipsoid model of the Earth. The CubeSat TLE was created for the 18th of November 2021⁴ and was converted into approximated Keplerian parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMA</td>
<td>6905.00</td>
</tr>
<tr>
<td>ECC</td>
<td>0.00110</td>
</tr>
<tr>
<td>INC</td>
<td>97.4789</td>
</tr>
<tr>
<td>RAAN</td>
<td>20.8993</td>
</tr>
<tr>
<td>AOP</td>
<td>291.7827</td>
</tr>
<tr>
<td>TA</td>
<td>357.1582</td>
</tr>
</tbody>
</table>

⁴The TLE and other information about the satellite were changed or omitted to maintain confidentiality.
Fig. 15. Flowchart of ATT_MAT_AEM subroutine
The requested visibility windows time-span went from the 20th November 2021 at 12:50:56 UTC for 24 hours.

The propagation in GMAT was done using the following parameters. The integration used RungeKutta89 with an initial step size of 60 sec, an accuracy of $10^{-11}$, a minimum step size of 0.001 sec, and a maximum step size of 2700 sec. The force model saw the Earth as the primary body, described by a JGM-2 gravity model with a degree of 4/70 and an order of 4/70. No tides were considered. The drag was a spherical MSISE90 model and the Solar Radiation Pressure model was a Spherical SRP. No other point masses were considered.

The number of visibility windows of the test case coincides. Both the start and stop time and the duration are quite similar between GMAT and GPT. The differences might be due to several factors: three days old TLEs with respect to the Visibility Request Start Date of 20th of November 2021, slightly different propagation methods, and numerical error in converting TLE data to Keplerian data. The data is shown in Table II.

<table>
<thead>
<tr>
<th>Date</th>
<th>Start Window</th>
<th>End Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 Nov 2021</td>
<td>18:17:23</td>
<td>18:21:08</td>
</tr>
<tr>
<td>20 Nov 2021</td>
<td>19:48:38</td>
<td>20:00:23</td>
</tr>
<tr>
<td>21 Nov 2021</td>
<td>07:20:06</td>
<td>07:26:10</td>
</tr>
<tr>
<td>21 Nov 2021</td>
<td>08:52:26</td>
<td>09:04:14</td>
</tr>
</tbody>
</table>

A second validation test to verify higher precision accuracy was done with an update TLE created on the 2nd of March 2023. The input for GMAT was Cartesian coordinates converted by the GPT. In this way, the initial conditions fully coincided. The propagation had the same set-up condition as the previous test but was run for 7 days.

For both GMAT and GPT, 42 windows were found with an offset of 0.5 or 1 sec at the first passes and 10 or 15 seconds at the last passes after 7 days. This is still due to slightly different ways of propagating the TLEs and also shows that their accuracy and reliability quickly decrease after a couple of days.

Figure 16 shows the argument of latitude drifting towards east due to the Earth’s oblateness and the influence of the Second Zonal Harmonics (east as the orbit is retrograde at 97 deg).

B. LCT displacement validation

To test the LCT displacement, first, a displacement along the x-axis only was set to +7600 meters. No change of reference was added, the default attitude still was LVLH. This meant that the LCT was ahead of the spacecraft along its orbit. The offset of 7600 meters was selected as it is the distance that a LEO spacecraft moves through in approximately 1 second in LEO. The resulting visibility windows were all starting and ending 1 second before the control case was done with an OEM, no displacement, and Weilheim Ground Station (DLR ground station near Munich). A more extreme test was done with an offset of 76000 meters. The spacecraft was expected to take around 10 seconds, with greater discrepancies due to the curvature of the orbit. The visibilities found were starting and ending around 8 to 10 seconds early with respect to the OEM control case.

The second test was done with a displacement along the z-axis set to −10000 meters. This means that the LCT orbit was higher of 10 km higher. This means that the orbit radius increases from approximately $r = 6870$ km to $r = 6880$ km. The orbit period goes from 94.2 minutes to 94.4 minutes: the orbit takes 13 seconds more. The passages found were from 6 to 14 seconds, with a median of 8 seconds. The most important element was that each pass started before and ended after the reference epoch of the control case mentioned above.

VII. Official testing and code coverage

After developing software, the important task of testing must not be underestimated as it is complex and requires a great deal of time for setting up the tests, running those, confronting the results with a reference, and eventually debugging. Both the testing of Fortran core code and the Python interface scripts are presented briefly below.

A. Fortran official testing

The Fortran executable expects different text input files from which the data is extracted: one configuration file with start and stop times, main and target description files with the constraints data, and files with orbit and attitude data for both main and target. Some tests used the same input data as previous software products from Flight Dynamics. The input format was adapted to be compatible with the GPT, and the tests were run with the goal of having consistent results regarding the number and start and stop of each visibility windows. [22] The TLE
results were checked with GMAT as presented in section VI-A, and in the same test had the goal of demonstrating the correct number of visibility windows.

Other missions were simulated in a group encompassing all orbit data formats. These tests had the goal of producing similar results when starting from the same orbit converted by already tested Flight Dynamics libraries. For the orbit data, the same orbit was described by TLE, OEM, OPM, and CPF. The attitude expressed as Look-Up-Table and BIAS Offset were checked running the same simulation of a previous Flight Dynamics software [22]. The AEM capabilities were tested with respect to the same attitude expressed with different combinations of Quaternions, Euler angles, LVHL, EME2000, ICRF (internally treated as EME2000) and TOD.

Many different error codes were tested, which are triggered in case of missing configuration or other files, in case of invalid or unsupported time systems or reference frames, typos in OEM or other orbit files as well as attitude files, and other internal errors that would bring the code to a stop. The goal of the Fortran testing was to reach 90% of code coverage of the Visibility Service, a value set internally by the Flight Dynamics team, for both nominal and error test cases. Overall, around 35 different missions and 45 different error codes were either reused or developed anew to reach the 90% code coverage goal.

B. Python and microservice official testing
The work during the month of May and June was focused on debugging the already existing Python scripts and adding those features that enabled the new capabilities like the AEM, solar panels, and Masks described as polygons (see section VIII) and others. After the first debugging process, the same tests developed for the Fortran code were translated into the JSON request files and run through the microservice. The responses were checked against the same results provided by the Fortran code and translated by a different script from simple text files to a JSON format. This achieved the verification of the Python script sequence that enable converting the single JSON file into several text files, calling the Visibility Service Fortran executable, and converting the output text file into the JSON response.

VIII. Case study: LCT onboard ISS
To explore the different capabilities of the GPT, two test cases were prepared and will be presented below. One of the two LCTs is installed on the ISS on the outside of the Columbus Module. The first mission is establishing visibility links during a ground-to-satellite link to the DLR Oberpfaffenhofen Institute of Navigation and Communication, while the second mission will be with an inter-satellite link with one of the EDRS satellites. The mission requirements dictate which capabilities are necessary from the GPT, and they will exemplify why those features were developed in the first place. Following the GPT terminology, the MAIN object will be in both missions the LCT on the Columbus Module, while the TARGET object will be the one varying.

For both test cases, the Visibility Windows List request will be done for a time frame between the 5th of May 2023 at 05:00:00 and the 7th of May 2023 at 05:00:00 UTC. The step size propagation will be of 1 second, and the minimum duration of the visibility windows of 1 minute.

MAIN object as LCT on Columbus
The LCT is installed outside the Columbus module, on its external science rack. The TLE publicly available on Celestrak.org provides the orbit data of the Zarya Module, the first and oldest module of the ISS. Since the station covers almost the dimensions of a football field, the Columbus module displacement must be taken into consideration. From [32] the displacement from the Zarya module can be inferred from the length and width of each of the ISS modules. The assumptions made are that the TLE is with respect to the CoM of the Zarya module and that its CoM is the same as the center of volume. The final displacement displayed in Figure 17, in LVHL reference frame (in use on the ISS as defined by [33]), is:

- T: 27 m;
- −R: 1 m;
- −N: −10 m.

The other constraints defined in the object description file were:

- Minimum grazing altitude: 100 km
- Minimum grazing angle: 0°
- Reverse pointing: false
- Minimum Sun angle: 2°
- Minimum Moon angle: 0°
- Maximum Point-ahead-angle 1.0°
- Check Exclusion zones: false
- Check Solar Panels: false
- Check Exclusion polygons: false
- Azimuth: from −180° to +180°;
- Elevation: from −90° to 270°.

The orbit of the ISS is defined by the TLE of the Zarya module on the 5th of May 2023, at 04:38:59.843 UTC time:

<table>
<thead>
<tr>
<th>ISS (ZARYA)</th>
<th>255440</th>
<th>90067A</th>
<th>23125.19374818 −0.00169740 00000+0 −30424−2 0 9997</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 255440 90067A</td>
<td>25540 51.6388 181.6167 0006015 309.1398 89.1969 15.49870072395123</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This test case includes in the simulation also Masks that cover partially the vision of the LCT, described in the software as polygons. There are only three: two permanent, called “poly1” and “poly4”, and one flexible called “flex1”, where the first two, on the right of the LCT field of view, could represent other experiments that are installed on the ram side of ISS.
and somehow block the visibility, and the flexible one could be the rotation of the starboard solar arrays. Both time and coordinates of the polygons are fictitious and do not represent true operational data on ISS.

The polygons are described by a list of coordinates of azimuth and elevation with respect to the local frame of the LCT (in this case still LVLH as the LCT Mounting matrix was set to the identity). The two permanent polygons are represented by the coordinates displayed in Table III. The flexible polygons’ capture and release times are presented in Table IV. A visual representation can be found in Figure 19. No LCT correction matrix is considered, as it is added to help fine calibration of the LCT mounting after the launch in space. The maximum point ahead angle is 0.1°, which is usually never exceeded.

The attitude is expressed as an AEM of constant values that represent a rotation matrix equivalent to 123 rotation of Euler angles of respectively 2, 4, and 6°. The header and the metadata of the AEM are as follows:

```plaintext
CCSDS_AEM_VERS = 1.0
COMMENT 'Fictitious AEM created for demonstration purposes'
CREATION_DATE = 2023-05-05T00:00:03.126
ORIGINATOR = GSOC
META_START
OBJECT_NAME = LCTonISS
OBJECT_ID = 9999999
CENTER_NAME = Earth
REF_FRAME_A = SC_BODY
REF_FRAME_B = LVLH
ATTITUDE_DIR = A2B
TIME_SYSTEM = UTC
START_TIME = 2023-05-05T00:00:00.00000
STOP_TIME = 2023-05-07T00:00:00.00000
ATTITUDE_TYPE = QUATERNION
QUATERNION_TYPE = FIRST
INTERPOLATION_METHOD = LINEAR
INTERPOLATION_DEGREE = 1
META_STOP
```

ISS’s official attitude frame is LVLH. Thus, the orbit to spacecraft body rotation matrix can be kept as the identity matrix as the default orbital frame is LVLH as well.

The last important factor to be considered is the position of the LCT with respect to the ISS itself. After the displacement from the Zarya module, the LCT attitude matrix is still in the LVLH frame, as shown in Figure 18. From the same picture, it is possible to see how the Node 2 and Japanese Lab, as well
as the Integrated Truss Structure (IST), could block the view of the LCT. This problem could be addressed in many ways: the solar panels could be added in the “panel function” of the GPT or with flexible polygons-masks. For now, this will be excluded as we can assume that the Solar Panels are in the horizontal position. The modules on the back and side of the LCT will be modeled with the fixed Exclusion Zones (though the same result could be achieved, limiting the LCT minimum and maximum angle).

Assuming that the Columbus module and those behind it block some visibility of the LCT, the azimuth is limited from 0° to +180° and the elevation from −90° to +270° thus excluding the “right semi-sphere” (as seen from L VLH).

A second exclusion zone can model the Integrated Truss Structure blocking the elevation view from −60° (the Truss can be seen in the background of Figure 18) to +20° and the azimuth from +120° to +180° (60° from the intersection between the IST and the parallel to US LAB), and from −135° to −180° (around 45° in the other direction). Note that the exclusion zones, expressed in degrees, is a four-sided surface on a sphere, specified by two azimuthal and elevation pairs and the range in between each pair is the exclusion zones. It is a simplified version of the full polygons-masks. This is why, taking into consideration that the azimuth values can go from −180° to +180°, it is necessary to split up the exclusion zone of the IST. Thus, the exclusion zones are three:

<table>
<thead>
<tr>
<th>EXCLUSION ZONES</th>
<th>AZI 1 → AZI 2</th>
<th>ELE 1 → ELE 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+180</td>
<td>−90 → +270</td>
</tr>
<tr>
<td>+120</td>
<td>+180</td>
<td>−60 → +20</td>
</tr>
<tr>
<td>−135</td>
<td>−180</td>
<td>−60 → +20</td>
</tr>
</tbody>
</table>

**TABLE III**

Polygons describing masks that cover the visibility of the LCT on the ISS.

<table>
<thead>
<tr>
<th>Poly 1</th>
<th>Poly 2</th>
<th>Flex 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AZI</td>
<td>ELE</td>
<td>AZI</td>
</tr>
<tr>
<td>150</td>
<td>−5</td>
<td>123</td>
</tr>
<tr>
<td>145</td>
<td>5</td>
<td>147</td>
</tr>
<tr>
<td>140</td>
<td>10</td>
<td>155</td>
</tr>
<tr>
<td>120</td>
<td>20</td>
<td>172</td>
</tr>
<tr>
<td>110</td>
<td>20</td>
<td>172</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>168</td>
</tr>
<tr>
<td>85</td>
<td>15</td>
<td>160</td>
</tr>
<tr>
<td>75</td>
<td>15</td>
<td>152</td>
</tr>
<tr>
<td>65</td>
<td>30</td>
<td>143</td>
</tr>
<tr>
<td>60</td>
<td>30</td>
<td>131</td>
</tr>
<tr>
<td>40</td>
<td>30</td>
<td>118</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>113</td>
</tr>
<tr>
<td>40</td>
<td>−5</td>
<td>113</td>
</tr>
<tr>
<td>90</td>
<td>−5</td>
<td>123</td>
</tr>
</tbody>
</table>

**TABLE IV**

Flexible polygons capture and release times.

| 2023-05-05 | 05:00:00 | Capture | Flexible Mask 1 |
| 2023-05-05 | 06:00:00 | Release | Flexible Mask 1 |
| 2023-05-05 | 10:30:00 | Capture | Flexible Mask 2 |
| 2023-05-05 | 16:00:00 | Capture | Flexible Mask 1 |
| 2023-05-05 | 17:30:00 | Release | Flexible Mask 1 |
| 2023-05-06 | 12:45:00 | Capture | Flexible Mask 1 |
| 2023-05-06 | 14:45:00 | Release | Flexible Mask 2 |
| 2023-05-06 | 21:00:00 | Release | Flexible Mask 1 |

A. Ground-to-satellite link

The Ground Station will be the DLR Institute of Navigation and Communication at the site of Oberpfaffenhofen. The Laser Terminal was set to have almost full degrees of freedom to cover the sky, excluding the zenith pointing. The azimuth and elevation constraints are:

- Azimuth: from −180° to +180°;
- Elevation: from 0° to 85°.

The geodetic coordinates defined in the WGS-84 reference frame of the optical ground station are:

- Latitude: 48.0848° N;
- Longitude: 11.2780° E;
- Altitude: 653.0 m.

The attitude of the ground station is the local-tangential-plane coordinate rotation matrix discussed in section IV-A. No exclusion zones, solar panels nor masks are considered. The maximum range distance for the LCT is 100 000 km and the maximum point ahead angle is 0.1 deg, which is usually never exceeded.

The results, coming from 172801 steps of the 48 hours of visibility requests, gave as result the visibilities in Figure 20:

<table>
<thead>
<tr>
<th>Poly 1</th>
<th>Poly 2</th>
<th>Flex 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AZI</td>
<td>ELE</td>
<td>AZI</td>
</tr>
<tr>
<td>150</td>
<td>−5</td>
<td>123</td>
</tr>
<tr>
<td>145</td>
<td>5</td>
<td>147</td>
</tr>
<tr>
<td>140</td>
<td>10</td>
<td>155</td>
</tr>
<tr>
<td>120</td>
<td>20</td>
<td>172</td>
</tr>
<tr>
<td>110</td>
<td>20</td>
<td>172</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>168</td>
</tr>
<tr>
<td>85</td>
<td>15</td>
<td>160</td>
</tr>
<tr>
<td>75</td>
<td>15</td>
<td>152</td>
</tr>
<tr>
<td>65</td>
<td>30</td>
<td>143</td>
</tr>
<tr>
<td>60</td>
<td>30</td>
<td>131</td>
</tr>
<tr>
<td>40</td>
<td>30</td>
<td>118</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>113</td>
</tr>
<tr>
<td>40</td>
<td>−5</td>
<td>113</td>
</tr>
<tr>
<td>90</td>
<td>−5</td>
<td>123</td>
</tr>
</tbody>
</table>

Fig. 20. Results from Ground-to-satellite link simulation

This shows that each visibility window happens every hour and a half, which is expected for a LEO spacecraft. The link duration varies from a minimum of 1 minute up to 10 minutes, in line with NASA predictions for Oberpfaffenhofen. [34]
**B. Inter-satellite link**

The need for reliable, European, high-speed data relay satellites saw the launch into space of the European Data Relay Satellite System. It consists of two payloads onboard two different geostationary satellites that relay information to and from non-geostationary satellites, spacecraft and stations. EDRS-A is a payload hosted on a satellite operated by Eutelsat known as EUTELSAT 9B EAST and consists of an optical inter-satellite link and a Ka-band inter-satellite link. EDRS-C is a GEO satellite developed by OHB and consists of the same communication payload as EDRS-A. [35] Together with providing relay services to the whole fleet of Sentinel satellites, the EDRS also provides the same service to the Columbus Module onboard the ISS through a Ka-band terminal, nicknamed “ColKa”, installed during a spacewalk in January 2021. [36] A similar device, now optical, might be installed approximately in the same position in the future. This is why this test case was created.

Assuming to be using EDRS-A, orbiting in a Geostationary orbit at 9 degrees east. The BOX coordinates will be defined as:

- Longitude: 9.0°
- Altitude: 35786.0 km
- Max latitude Offset: 0.1°
- Max Longitude Offset: 0.1°

In this test case, the LCT has no constraints, so no exclusion zones or solar panels that block the view. The minimum and maximum elevation angles are going to model the LCT terminal on the GEO satellite so that is mounted in the belly of the satellite, pointing in the Nadir direction and having the bottom half sphere of visibility pointing to the center of the Earth:

- Minimum Azimuth: −180°
- Maximum Azimuth: +180°
- Minimum Elevation: −90°
- Max Elevation: 0°

The same effect could be obtained by having the elevation constraints between 0 and 90° and using the LCT Mounting matrix (previously set as the identity matrix) to have it represent a 180° rotation along the first (T) or second axis (−N) and having in the first case:

\[
LCT_{\text{mounting}} = \begin{bmatrix}
1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & -1
\end{bmatrix}
\]

In this way, though no LCT mounting matrix is defined, the azimuth and elevation constraints limit accordingly to the requirements stated above the LCT visibility field. The attitude is the default one: Nadir-Pointing. The maximum distance of the LCT range is 50 000 km and the maximum point ahead angle is 0.1°, which is usually never exceeded.

The results, coming from 172 801 steps of the 48 hours of visibility requests, gave, as a result, the visibilities in Figure 21. The results represent approximately 800 minutes of communication between the ISS with EDRS in a time span of 2 days. This means that using optical communication with a data rate of 100 Mbps, almost 2.5 terabytes of data could be transferred per day.
IX. Conclusions and future work

The rise of laser communications in the space sector, together with the shift of GSOC from mission specific to multi-mission software, brought the Generic Planning Tool into existence. Its purpose is to support the Mission Planning team at DLR in planning and executing satellite-to-ground and inter-satellite links. Its two services are developed through a mixture of programming languages.

This enables both the high performances of the Flight Dynamics Fortran libraries derived from the strong heritage within the field and automates the process with a network protocol thanks to the Python scripting and the Microservices architecture. As the two test cases involving the ISS showed, the optical link, in case of the absence of obstacles or adverse meteorological conditions, yields a far higher data transfer rate than traditional radio transmissions, paving the way to new exciting opportunities both from a science and engineering point of view.

All the goals defined before the beginning of the internship were achieved. Goal (i) was completed by integrating CPF, OEM, and AEM into the prototype. Goal (ii) saw the successful testing of TLE, CPF, OPM, and OEM for the orbit data, and Nadir-pointing, LUT, BIAS Offset, and AEM for the attitude. The Fortran core libraries were tested extensively reaching more than 90% of code coverage. Collectively both the Fortran and the Python code were tested to verify the correct workflow of the microservice. Thus a new version of the software was released. Finally, goal (iii) was completed as the design of the workflow of the architecture of the Link Support Service was reviewed.

Further work on the GPT could see an expansion of the input orbit and attitude data formats, though the number of time systems, reference frames, and data formats are very large. What has been implemented for now represents the most common orbit and attitude formats for optical communication together with those currently in use at GSOC. Future work could see the addition of the Venus minimum angle (which has the same grounds as the Sun and Moon minimum angle). Up to debate was the addition of visibility windows defined in GPS time, together with the already developed UTC times.

Any big change related to the Visibility Service could be done following some mission-specific requirements, thus developing each time some ad hoc add-ons. As of June 2023, the Link Support Service was in its definition and conceptual phase: design choices will need to be discussed, weighing the pros and cons of different development strategies. The process could be accelerated by basing the development on existing mission-specific software, though each design iteration will see the participation of the Mission Planning group of GSOC to establish the requirements from their side.

Acknowledgments

Working on this master’s thesis in DLR GSOC has been a precious experience that enriched my personal and professional life. My gratitude goes to Dr. Ralph Kahle for believing in me and for offering me this opportunity, to Frederic Schoutetens for the wise guidance and support throughout the whole internship, and to all the Flight Dynamics team for their warm welcome.

I would like to thank Dr. Gunnar Tibert, my university supervisor, for his support of this thesis and especially for the motivation, knowledge, and passion that I always felt in his courses.

I would also like to thank all the great friends with whom I studied, worked, and lived this part of my life at home, in Padova, and in Stockholm, whose bonds literally defy the boundaries of time and space. Finally, I dedicate this thesis to my parents and to all my family, for their everlasting encouragement and support during my studies in Italy and in Sweden.
REFERENCES


[34] NASA. Spot the Station · International Space Station. https://spothestation.nasa.gov/sightings/view.cf?country=Germany & region=None & city=Columbus_Control_Center_Oberpfaffenhofen. Online; accessed 22 June 2023.

