Department of Space & Climate Physics

Individual Project – Final Report

Academic Year 2020 – 2021

AUTOMATED SKY SURVEY ANALYSIS METHODS

Submitted in partial fulfilment of the requirements for the award of the degree of

MSc – Space Science and Engineering

Submitted by:

Max Hallgarten La Casta

Supervised by:

Prof. Daisuke Kawata

July 2021

Word count:
11,082

Declaration

I declare the following work is my own and, where the work of others has been used, it has been clearly identified.

Acknowledgements

I would like to thank my supervisor, Prof. Daisuke Kawata, for his continued support and assistance throughout each stage of this project. My weekly meetings with him were extremely helpful, and his feedback and encouragement have greatly improved my work.

Abstract

Space telescope scheduling presents a significant challenge due to the limited resource of available telescope time caused by a combination of operational and observational constraints. Combined with high demand from astronomers for observations of different targets at different times, this results in an oversubscription for observation time where not all requests can be fulfilled. The scale of the challenge means that it is not feasible to develop these schedules manually, therefore there has been an emphasis on developing automated software tools to solve the problem. Due to restricted access to existing mission-specific scheduling tools, this project has implemented the initial development of a flexible, open-source software tool to increase accessibility to this form of planning. The tool, developed in Python and leveraging existing open-source tools, has been designed in the context of the Japan Astrometry Satellite Mission for Infrared Exploration (JASMINE). It was successfully used to confirm the expected visibility of candidate targets during a nominal mission, and to aid with the sizing of the telescope's baffle. The strengths and limitations of the software have been identified, and used to formulate recommendations for future work.

Contents

List of Figures

List of Tables

List of Listings

List of Acronyms

ANN	Artificial Neural Network
API	Application Programming Interface
ASCA	*Advanced Satellite for Cosmology and Astrophysics*
ASSAM	Automated Sky Survey Analysis Methods
CCD	Charge-coupled Device
CPU	Central Processing Unit
CSP	Constraint Satisfaction Problem
CSV	Comma-separated Values
CTE	Coefficient of Thermal Expansion
CUDA	Compute Unified Device Architecture
CXO	*Chandra X-ray Observatory*
ESA	European Space Agency
EUVE	*Extreme Ultraviolet Explorer*
GA	Genetic Algorithm
GCRF	Geocentric Celestial Reference Frame
GCRS	Geocentric Celestial Reference System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HST	*Hubble Space Telescope*

I/O	Input/Output
IAU	International Astronomical Union
ICRF	International Celestial Reference Frame
ICRS	International Celestial Reference System
ISAS	Institute of Space and Astronautical Science
JASMINE	*Japan Astrometry Satellite Mission for Infrared Exploration*
JAXA	Japan Aerospace Exploration Agency
JPL	Jet Propulsion Laboratory
LEO	Low Earth Orbit
LTAN	Local Time of the Ascending Node
NASA	National Aeronautics and Space Administration
PI	Principal Investigator
RLE	Run-length Encoding
SCRF	Spacecraft Celestial Reference Frame
SRP	Solar Radiation Pressure
SSO	Sun Synchronous Orbit
STScI	Space Telescope Science Institute
TOO	Target of Opportunity
UI	User Interface
VLBI	Very Long Baseline Interferometry

Chapter 1

Introduction

1.1 Project Objectives

The overall objective of the Automated Sky Survey Analysis Methods (ASSAM) project is automatic evaluation of observation schedules for the *Japan Astrometry Satellite Mission for Infrared Exploration* (*JASMINE*), an astrometric mission with plans to investigate the formation history of the Milky Way galaxy [1]. The astrometric observations conducted during this mission will be subject to a variety of different constraints which, when coupled with the large number of target candidates (initially 350) and the long duration of the mission (nominally three years), would be extremely difficult to schedule manually.

Automatic survey scheduling is useful for any space-based observatory mission, therefore, the project also aims to develop a general purpose tool which could be applied to a variety of different missions, not only *JASMINE*.

There have been many sky survey planning tools developed in the past for specific missions, including examples such as `Spike` used for scheduling the *Hubble Space Telescope* (*HST*) [2]. However, access to these mission-specific tools is typically restricted to those working on the mission. Due to the lack of publicly available tools, `assam` aims to be a flexible public software, following open-source software principles, expanding on existing open-source tools where required, to produce a flexible, modular tool to increase accessibility to this form of planning.

The objectives of the project are summarised into three main branches [3]:

1. Generation of target observation visibility to aid manual scheduling.
2. Full automation of sky survey scheduling.
3. Creation of a lightweight software package with high computational performance to enable rapid, iterative, mission planning.

1.2 Report Outline

This report begins with a background section containing a brief overview of astrometry and past space missions in this field, further details regarding the *JASMINE* mission, and an overview of space telescope scheduling using `Spike` as a case study. The methods used by the `assam` package are then discussed, including the existing packages and tools which are used during calculations. Finally, results generated using `assam`, specifically for *JASMINE*, are presented to demonstrate the software's capabilities, followed by a discussion of the strengths and limitations of `assam`, and recommendations for future work.

Chapter 2

Background

2.1 Astrometry

Astrometry is a branch of astronomy focused on precise measurements of the positions, velocities, and geometries of celestial bodies such as stars and planets [4]. Astrometry can be seen as the original form of astronomy: "[i]n ancient times, and until about a century ago, astronomers were essentially astrometrists, measuring the effemerides of the stars" [5].

Astrometry is the fundamental basis for all other fields of astronomy as the accurate pointing of telescopes, required for conducting astronomical studies, is dependent on astrometry [6]. Astronomy and astrophysics are heavily reliant on astrometry for understanding the dynamics of celestial bodies. Astrometric studies can be divided into two main forms of analysis: kinematic analysis, where studying and understanding the motion of the bodies themselves is the primary objective; and dynamic analysis, where the motion of the bodies is used to study the driving forces behind their motion [4].

Astrometry intrinsically provides important outputs such as reference frames, important for navigation and time-keeping, extending the applications of the field. Many of the developments in navigation and guidance are derived from astrometry: in the modern day, astrometry can be applied for space navigation but, historically, in the absence of any other references, celestial navigation was crucial for navigation at sea [6].

Astrometry does not have a requirement for a minimum measurement accuracy, nevertheless as with all experimental methods, the quality of results is greatly affected by measurement accuracy. Stability is also another important factor which affects the quality of results, particularly when using long exposures with photographic plates historically, or detectors such as Charge-coupled Devices (CCDs) more recently. Movement of the telescope can result in the capture of blurred images, reducing measurement accuracy as uncertainty is introduced into the image.

Gradual improvements in accuracy when conducting astrometry have been made in the past 400 years as technology, techniques, and general understanding improved, as illustrated in Figure 2.1. However, it was the development of space-based observatories, such as *Hipparcos* and *Gaia*, which resulted in significant improvements in the past 30 years, with angular accuracies improving by several orders of magnitude.

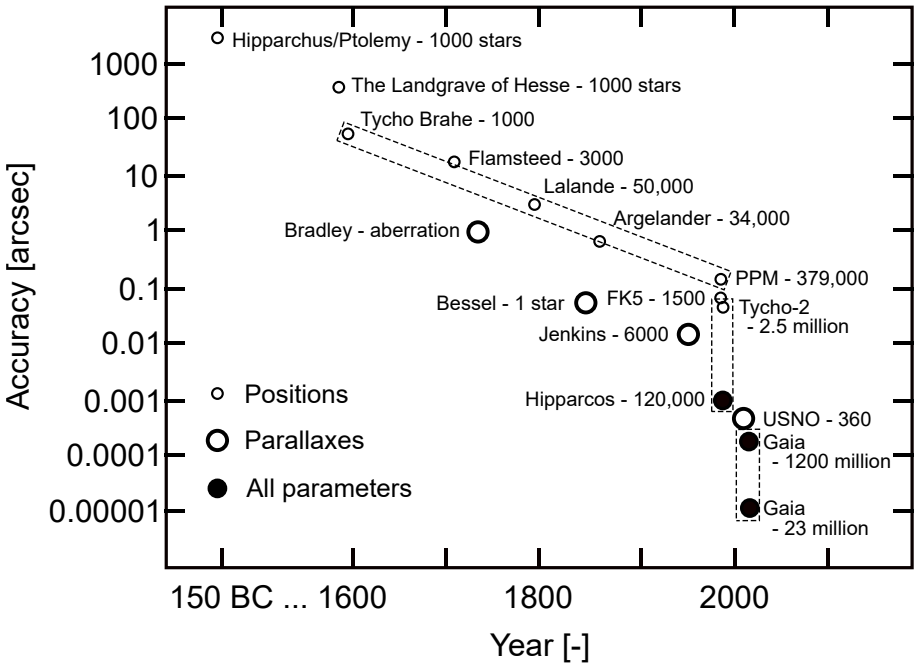


Figure 2.1: Astrometric accuracy over the past 2000 years, illustrating gradual improvements in the past 400 years before significant improvements with the development of space-based observatories with missions such as *Hipparcos* and *Gaia*. Redrawn and adapted from [7].

Ground-based astrometry is subject to several limitations which are either not present, or at least easier to mitigate, with space-based astrometry. One of the primary factors affecting ground-based measurements is the presence of the atmosphere between the observer and the target. This introduces two main effects: a systematic change in the apparent location of a target due to refraction, and a continuously varying change in the apparent location due to atmospheric turbulence which results in blurred images, effectively reducing the precision of the telescope [4]. Furthermore, the atmosphere effectively blocks light at several wavelengths, preventing observations of sources which primarily emit at these wavelengths. Ground-based measurements also need to take the Earth's rotation into account, requiring accurate knowledge of the Earth's orientation, as ground-based instruments are fixed with respect to the Earth's surface [4]. Ground-based astrometry often requires the use of measurements from multiple instruments, however errors in inter-calibration between different instruments can result in systematic errors in the combined results [4]. Nevertheless, it should be noted that space-based astrometry introduces its own disadvantages, such as the technical difficulty of developing and operating spacecraft and their instruments [4].

2.2 Past Space Missions

In the mid-1940s, the idea of a space-based observatory was proposed by Spitzer, suggesting that "such a scientific tool, if practically feasible, could revolutionize astronomical techniques and open up completely new vistas of astronomical research" [8]. The first successful space-based observatory, *OSO-2*, would not be launched until December 1968 [9]. However, the following decades saw a rapid expansion in the number and variety of space-based observatories. One of the most famous is *HST* which launched in 1990, infamously requiring several support missions for repairs [10]. *HST* is still operating, with plans to operate until at least the mid-2020s [11]. Nevertheless, this may be affected by recent issues with the spacecraft's payload computer [12]. Although many space-based observatories such as *HST* have been used for astrometry [13], there have been a limited number of missions designed specifically for this field.

Hipparcos was a European Space Agency (ESA) mission to conduct astrometry from space. The mission officially entered Phase A with feasibility studies in 1976, before full approval as an ESA mission in 1980 [14]. Launched in August 1989, the mission operated until March 1993 after which radiation damage to the spacecraft's solar panels and electronics meant that observations could no longer continue [14]. Over its mission lifetime, *Hipparcos* "provided the first very accurate catalogue of apparent magnitudes, positions, parallaxes and proper motions of 120 000 bright stars at the milliarcsec (or milliarcsec per year) accuracy level" [15].

Gaia is an ESA mission which was proposed in 1993, acting as a successor to *Hipparcos* by conducting both an expanded astrometric survey, and an astrophysical survey to determine properties of the stars "such as surface gravity and effective temperature" [16]. *Gaia* launched in December 2013 [17], completing its nominal five year mission in July 2019 [15]. Nevertheless, it is expected to continue with an extended mission until at least mid-2024 [15]. The contents of *Gaia*'s second data release included astrometric data on almost 1.7 billion sources [15], an increase by four orders of magnitude over *Hipparcos*, which is expected to grow over the following years during the extended mission. The current *Gaia* data release, EDR3, increased the number of sources to approximately 1.8 billion [18].

One of the main limitations of these missions is due to the wavelengths that their instruments can observe. For both *Hipparcos* and *Gaia*, the observable wavelengths of their instruments correspond primarily to the visible band with wavelength ranges of 375 to 750 nm and 300 to 1000 nm respectively [19, 20]. The observable wavelengths for an instrument can introduce restrictions on the regions of the sky that can be observed effectively. For example, in the case of the Galactic Centre there is large amount of interstellar dust which absorbs and scatters a large proportion the light in a process known as extinction [21]. However, the effect of interstellar dust is wavelength-dependent, and therefore by selecting other wavelengths which are less affected, such as the near-infrared, observations of these regions can be conducted [21, 22].

Rather than focusing on specific regions, both *Hipparcos* and *Gaia* conduct astrometric surveys of the entire sky, known as all-sky surveys. This survey strategy is very useful for creating large catalogues, particularly for the creation of reference frames. Another investigation strategy is to concentrate on particular regions of the sky, such as, for example, the Galactic Centre, to conduct accurate, relative astrometry. The benefit of concentrated observations of a particular region is the significant increase in the number of exposures. As astrometry typically requires data reduction through statistical analysis to obtain more accurate results, increasing the number of samples can dramatically improve accuracies.

2.3 *JASMINE*

JASMINE is a space telescope mission that was selected by the Institute of Space and Astronautical Science (ISAS) and the Japan Aerospace Exploration Agency (JAXA) in May 2019, as part of the competitive M-class programme of missions, and planned to launch in the mid to late 2020s [1]. M-class missions have a typical budget below ¥15 billion (£100 million), with a cadence of approximately five of these missions every ten years [23]. An artist's impression of the *JASMINE* spacecraft is presented in Figure 2.2.

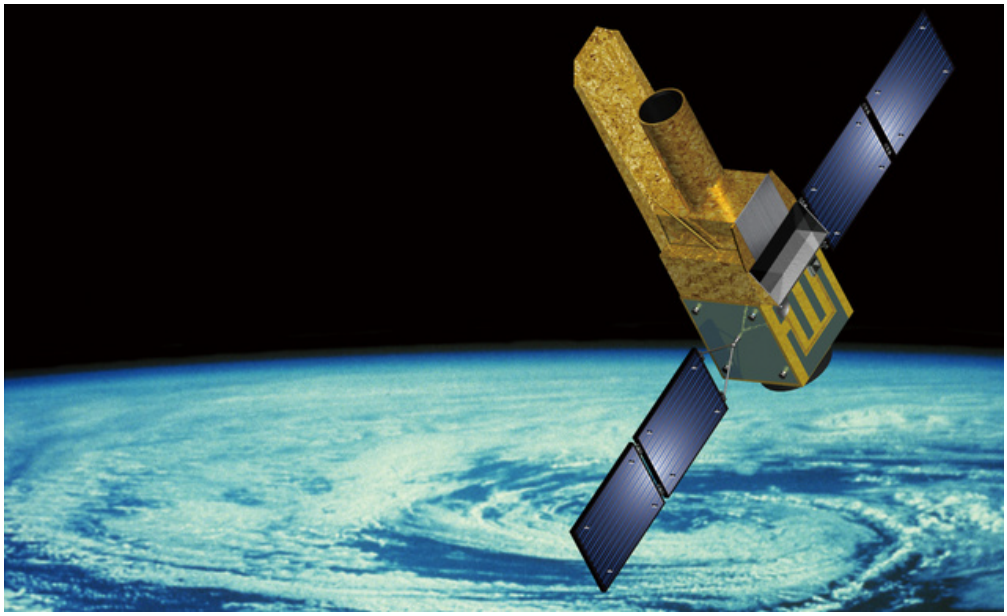


Figure 2.2: Artist's impression of the *JASMINE* spacecraft [22].

The primary objective of *JASMINE* is to conduct a detailed survey of the Galactic Centre, conducting astrometry to investigate the structure and formation history of the Milky Way's central core [1]. Further objectives include an extension of the primary objective to the Galactic mid-plane, and exoplanet hunting by observing a selection of M-type stars [1]. The distribution of targets in the sky is shown in Figure 2.3, highlighting how the Galactic Centre and Galactic mid-plane are concentrated in a much smaller region of sky than the M-type stars.

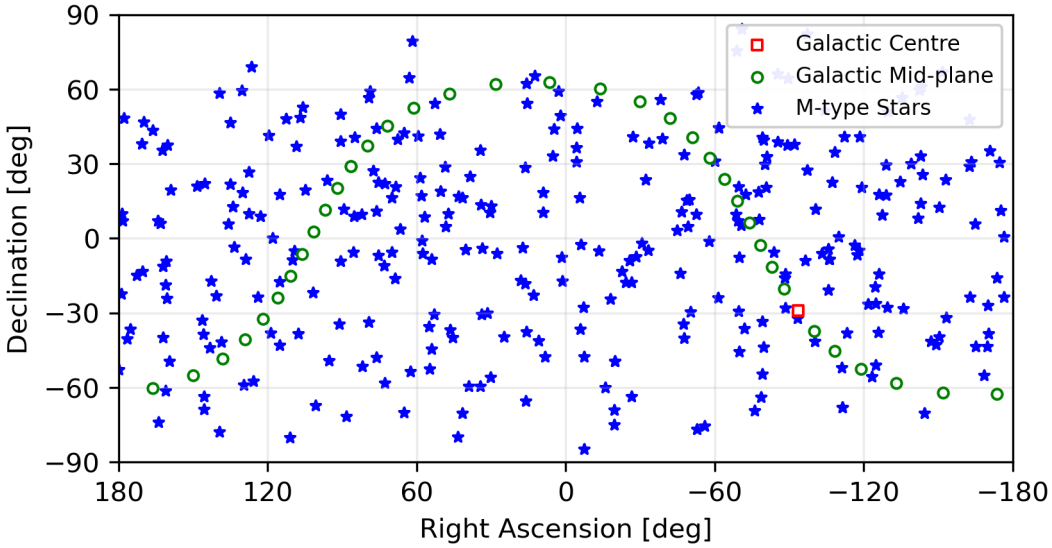


Figure 2.3: Distribution of targets in the sky. The more isotropic distribution of M-type stars in the sky compared to both the Galactic Centre and the Galactic mid-plane is highly visible. It should be noted that the markers in this plot are not to scale, and do not represent the true size or geometry of the targets. Redrawn and adapted from [3].

The launch vehicle is a major design driver for *JASMINE*, defining limits on its mass, dimensions, and orbit. One of the requirements for M-class missions is using the Epsilon launch vehicle, illustrated in Figure 2.4. This is a three stage (with an optional fourth stage) solid propellant rocket which was designed for launching scientific satellites [24]. The standard variant of the launch vehicle is capable of launching a 450 kg payload to a 500 km circular Sun Synchronous Orbit (SSO) when using the optional liquid propellant fourth stage [25]. This was increased to over 590 kg with the enhanced version of the launch vehicle, achieved by improving the fairing design to increase the amount of propellant in the second stage, and mass reductions of the structure and avionics [25].

Consequently, *JASMINE* will be launched into an SSO at around 550 km with a Local Time of the Ascending Node (LTAN) of either 06:00 or 18:00 [1]. These orbital parameters mean that *JASMINE* will be a near terminator-riding satellite, where it's ground track is close to following the Earth's terminator. One of the advantages of such an orbit is the near constant illumination from the Sun, which is beneficial for power generation when using photo-voltaic cells, but more importantly for *JASMINE*'s instruments results in a more constant thermal environment.

The design of *JASMINE*'s scientific payload had to take into account the restrictions imposed by the launch vehicle's payload fairing envelope. A modified Korsch-type telescope with a 30 cm primary mirror and 3.9 m focal length was designed, providing a highly compact overall package to meet this restriction [1]. The preliminary optical design of the telescope is presented in Figure 2.5, showing the complex configuration of six mirrors required to create this package.

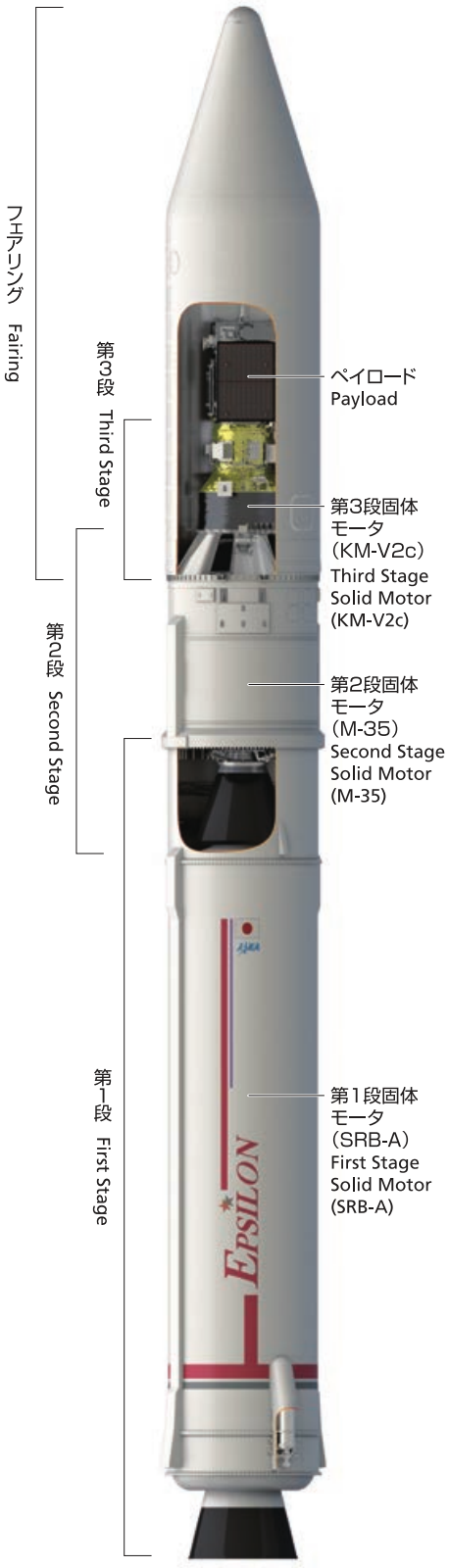


Figure 2.4: Cutaway of the Epsilon launch vehicle in its three stage configuration [25].

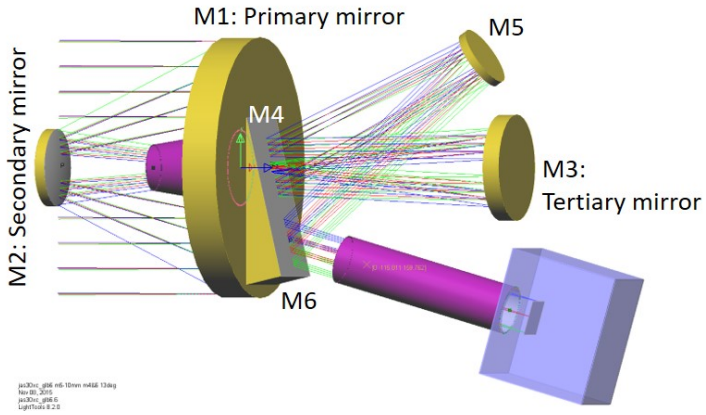


Figure 2.5: Preliminary configuration of *JASMINE*'s telescope [1].

The mission requirements for *JASMINE* state that the precision of parallaxes for brighter stars (magnitudes less than 12.5) must be equal to or better than 25 µas [1]. Nevertheless, single observations by *JASMINE* of particular stars will only achieve errors of 4 mas [1], a difference of approximately five orders of magnitude. The precision will be improved to meet the mission requirement by conducting a data reduction, using multiple exposures with a model for systematic errors to remove these errors [1].

The operating band of the telescope was driven by requirements when observing the Galactic Centre. As discussed above in Section 2.2, observing this region is heavily constrained by the amount of extinction due to interstellar dust. For *JASMINE*, the wavelength range of 1.1 to 1.7 µm, corresponding to part of the near-infrared band, was selected to take advantage of the lower extinction at these wavelengths [22].

The selection of the near-infrared band resulted in strict thermal requirements for the payload, both due to temperature limits on the instrument's detector, and thermo-structural restrictions. For example, the detector needs to be held below 180 K during observations with the temperature varying by less than 0.7 K during this time, and thermo-structural coupling must be limited to prevent deflections of greater than 10 nm of the image on the focal plane [1].

To maintain the required thermal conditions, the telescope will be installed in a "Telescope Panel Box", a box which is thermally isolated from the rest of the spacecraft, with thermal conditions maintained by an electric heater, a fixed radiator, and a fixed sun shield [1].

Thermo-structural coupling is dependent on temperature changes and the properties of the materials, specifically the Coefficient of Thermal Expansion (CTE). For *JASMINE*, materials with extremely small CTEs were selected as these materials reduce optical deflections within the telescope which would affect the precision of measurements [1]. Super-super invar was selected for the telescope structure, and CLEARCERAM for the mirrors, providing an extremely small CTE at the telescope's operating temperature [1, 26].

The performance of the thermal control system, which determines whether observations can be conducted, is heavily dependent on the pointing of the spacecraft due to the orientation of elements such as the radiator and sun shield. Additional pointing restrictions are imposed due to concerns with stray light from sources such as the Sun and scattering in the Earth's upper atmosphere entering the telescope.

The position of the Sun in the sky with respect to celestial targets from the viewpoint of *JASMINE* has a significant impact on whether observations can be conducted, due to the thermal and stray light concerns highlighted above. This impact on target observability is most apparent when considering observational conditions for the Galactic Centre, the priority target of the mission. The geometry of these constraints is illustrated in Figure 2.6 where the Sun effectively introduces a 45° avoidance cone in both directions of the Earth-Sun line where celestial targets cannot be observed. Consequently, the Galactic Centre is not observable for half of the year during either winter or summer. This observation restriction is independent of shorter-term restrictions such as the Galactic Centre being occulted by the Earth for approximately half of *JASMINE*'s orbit.

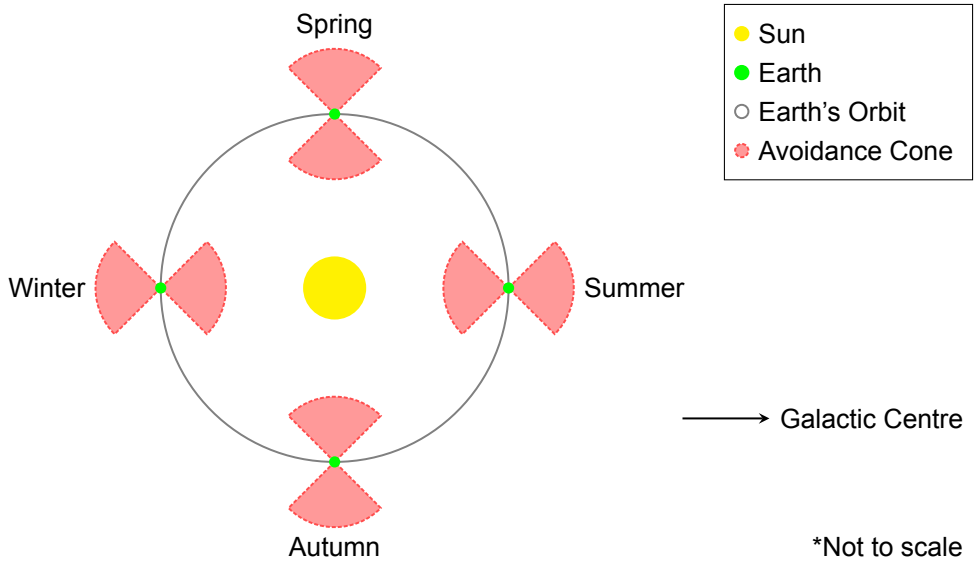


Figure 2.6: Observation constraints introduced by the Sun throughout the year. The Galactic centre cannot be observed during winter and summer due to thermal and pointing constraints introducing 45° avoidance cones in both directions of the Earth-Sun line. Redrawn from [3].

Developing space missions is a highly intensive process, both financially and technically, and there is, therefore, a constant desire to maximise the scientific return. Given that the Galactic Centre is not observable during half of the year, the mission was extended to include additional targets, including the Galactic mid-plane and a selection of M-type stars. This will allow *JASMINE* to take advantage of observations throughout the entire year.

2.4 Space Telescope Schedule Optimisation

The scheduling of space telescopes presents a significant challenge. Available observation time is a limited resource, coupled with a high demand by astronomers for observations of different targets at different times. This typically results in an oversubscription for observation time where not all requests can be fulfilled [27].

The scale of the challenge of scheduling observations means that it is not feasible to generate sky survey plans manually. There has been, therefore, an emphasis on developing software to solve the problem. Software for space telescope scheduling is typically bespoke, mission-by-mission. There have been, however, efforts to develop general tools which can be adapted to different missions. One of the most successful space telescope scheduling tools of the past three decades has been the `Spike` system.

The `Spike` system was developed by the Space Telescope Science Institute (STScI) as a general framework for space telescope scheduling, initially for use by *HST* [2]. The system was designed to schedule the 10 000 to 30 000 observations which *HST* undertakes every year, each of these subject to scientific and operational constraints [2]. Examples of both scientific and operational scheduling constraints for *HST*, and the timescales over which they must be considered, are presented in Figure 2.7.

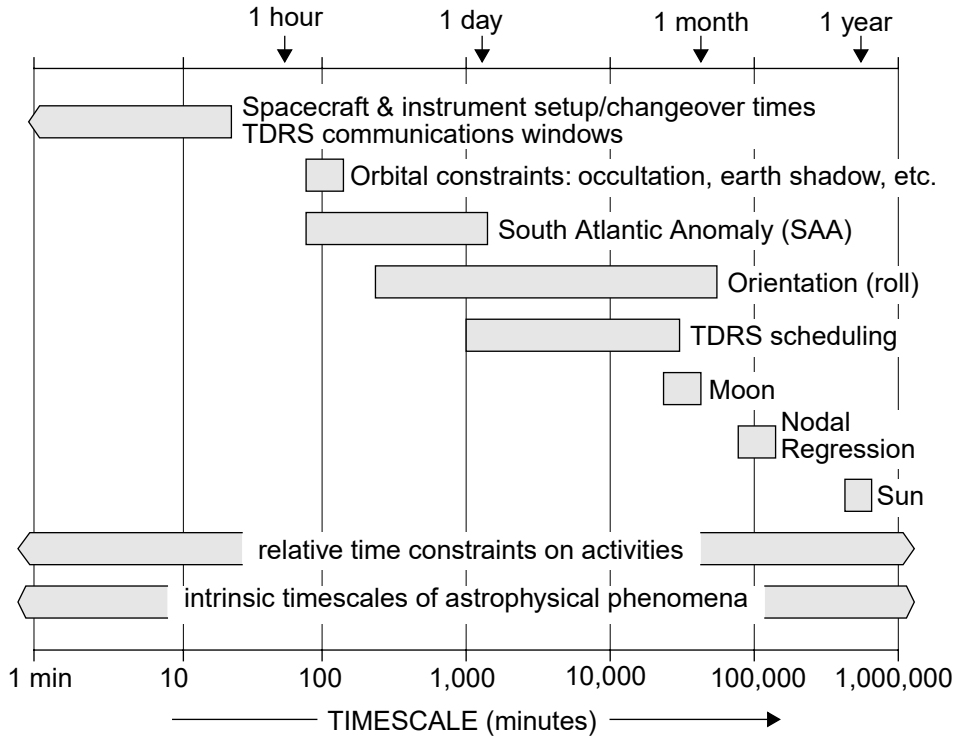


Figure 2.7: Examples of constraints considered by `Spike` for *HST* and the range of timescales where they must be considered [2].

`Spike` treats the space telescope scheduling problem as a Constraint Satisfaction Problem (CSP), representing constraints with a set of "suitability functions", and using a stochastic repair search strategy to generate schedules [2].

Two types of constraints are considered by `Spike`: feasibility constraints, which refer to constraints which cannot be violated under normal conditions, such as a minimum angular separation between a target and a particular body; and preference constraints, which can be violated, however with an objective, such as minimising the amount of scattered light from a particular body [2]. The constraints are combined into "suitability functions": continuous functions which express how suitable it would be to observe a target at a given time [2].

The search strategy used by `Spike` can be broken down into three steps: trial assignment, where an initial guess of a schedule is created which is likely to violate some constraints; repair, where heuristic methods are applied to remove constraint violations; and deconflict, where any remaining activities which violate constraints are removed until a feasible schedule remains [2]. The heuristics applied during the repair step use an Artificial Neural Network (ANN) developed specifically for `Spike` which is stochastic, therefore repeating the process can result in different and improved schedules [2].

The development of `Spike` began in the 1980s, however this continued during *HST* operations, introducing improvements prompted by lessons learned since the launch of *HST*. These improvements had a significant effect on schedule planning for *HST*: the 56 days required originally to prepare a one-week flight calendar was reduced to 11 days, and observing efficiency was increased from 25% to typically around 50% [28]. The capability to interrupt the schedule for Targets of Opportunity (TOOs) within 24 hours was also introduced, adding a rapid response capability for astronomical phenomena such as supernovae and gamma-ray bursts [28].

`Spike` was adapted for scheduling of other space telescopes, such as the *Extreme Ultraviolet Explorer* (*EUVE*) and the *Advanced Satellite for Cosmology and Astrophysics* (*ASCA*), and prototypes were developed for ground-based telescope scheduling as feasibility demonstrators [2]. More recently, `Spike` has been used for the *Spitzer Space Telescope* and the *Chandra X-ray Observatory* (*CXO*) [29]. Several major changes were made to implement the required short-term scheduling, however "[m]ost of the effort required to apply [`Spike`] to the new problems was limited to the specific domain modelling necessary, which typically involves computation related to the geometry of the satellite, Sun, target, and Earth" [2]. In the case of *EUVE*, an additional 20 days of observing time per year were gained by using `Spike` instead of the previously used search algorithm [2], illustrating the utility of a powerful scheduling tool which can be adapted to different missions.

Chapter 3

Methods

3.1 Software Architecture

The `assam` software was designed from the start to be modular. The benefit of this design philosophy is that it avoids the need for major modifications to the entire codebase if some of the methods used within one or more modules are changed. For example, changing the spacecraft propagator or scheduling algorithm could be achieved with minimal effect on the other modules.

The overall software architecture for `assam`, including the main inputs, processes, and outputs, is presented in Figure 3.1. The software is currently split into four main modules: propagator, visibility, scheduling, and visualisation.

The propagator module has two main tasks: the propagation of the spacecraft's state and of the solar bodies' states, where the solar bodies include the Sun, the major planets, and the Earth's moon. These operations are executed sequentially as the vector of times resulting from propagating the spacecraft's state is used to propagate the state of the solar bodies.

The visibility module is tasked with calculating the times at which targets of interest are visible to the spacecraft given constraints such as, for example, those presented in Section 2.3. It uses the spacecraft and solar body states previously calculated by the propagator module, in addition to a list of targets containing important parameters. The module outputs a list of target contacts, which includes information on when the targets are visible.

The scheduling module is tasked with generating the observation schedule for the spacecraft, using the list of target contacts generated by the visibility module. This process can be seen as an optimisation task, returning the schedule which maximises the scientific return of the mission. The output from this module is a subset of the originally input list of target contacts which are included in the schedule.

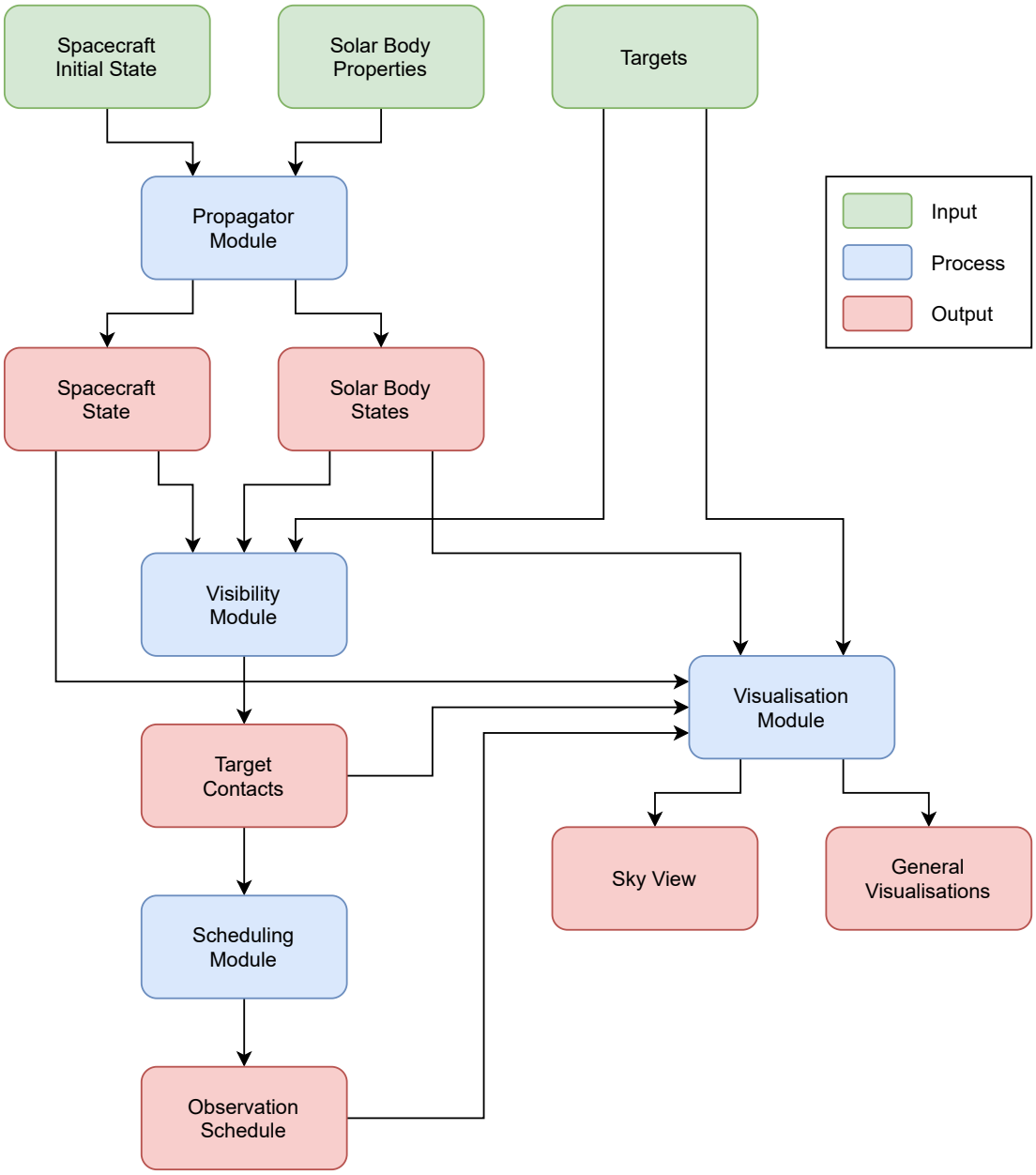


Figure 3.1: Software architecture for `assam` illustrating information flow between the modules.

The visualisation module is tasked with converting the output from the other modules into formats which are better suited for human interpretation. One of the main examples of a human understandable format is the sky view, a plot which shows the targets and constraints to scale from the viewpoint of the spacecraft, providing an effective way to understand the relative positions of the various objects at specific times.

In the following sections further detail is provided on the software used to develop `assam`, and internal functioning of the various modules.

3.2 Software

Python [30] was chosen as the primary programming language for assam. It provides several key features including a high ease of use as a very high level programming language, and its high popularity, both in general and specifically in the field of astronomy. Its popularity means that the language benefits from a significant number of highly developed third-party packages.

The main third-party Python package used by assam is astropy [31–33], an open-source community-developed package "which serves as the foundation for more specialized projects and packages" [32]. The package provides a large set of functionalities relating to astronomy.

The majority of astropy functionality used by assam comes from two sub-packages: astropy.units, and astropy.coordinates. The units sub-package provides the ability to store both the value and unit of variables instead of the traditional method of storing non-dimensional values and ensuring units consistency through comments in the code. Furthermore, the units are taken into account during calculations, ensuring consistency when using mixed units, such as, for example, radians and degrees for angles. The coordinates sub-package provides a powerful framework for transforming between various different astronomical reference frames, and enables the calculation of angular separations between coordinates. This framework for coordinate systems is heavily leveraged when calculating the visibility of targets.

Spacecraft state propagation is handled by GMAT [34], an open-source astrodynamics software package developed by the National Aeronautics and Space Administration (NASA) for mission analysis. Other propagators which were not implemented, but could be interfaced with assam, include, for example, poliastro [35], Orekit [36], and tudatpy [37].

Other third-party Python packages used by assam include pandas [38, 39] for data handling, both internally and externally; matplotlib [40, 41] and seaborn [42, 43] for data visualisation; and tqdm [44] to provide a basic User Interface (UI) in the form of progress bars.

Several methods are used to improve the computational performance of assam, primarily parallel computing and vectorisation. Many of the tasks in the various assam modules can be easily split into multiple threads without requiring shared memory, greatly increasing the computational performance. Calculations are heavily vectorised, where possible, using NumPy [45] to take advantage of significantly reduced execution times. The astropy package itself is built on NumPy, ensuring high performance [31, 32]. Hardware acceleration was investigated for performance improvements and implemented as an option, using CuPy [46], a package developed to act as a drop-in replacement of NumPy. CuPy takes advantage of the Compute Unified Device Architecture (CUDA) available on Nvidia Graphics Processing Units (GPUs), reaching a speedup of up to 270 times in specific workloads on large arrays [47].

Standard version control methods were implemented during assam's development to track changes in the codebase. Git was chosen due to its widespread use and popularity [48]. A remote repository was created on Github to enable future collaboration with multiple contributors with a central repository. The repository was kept private during initial development but is now publicly available at [49].

3.3 Propagator Module

3.3.1 Spacecraft Propagation

GMAT has an Application Programming Interface (API) for Python which allows the software to be controlled directly from a Python script. This API is, however, still at a very early stage of development and therefore is not complete nor stable enough to be used by assam. For example, many of the features necessary for assam are yet to be implemented, and furthermore, it was found to be highly sensitive to the version of Python that was being used.

Interfacing between assam and GMAT is achieved by using a combination of text files and the command-line. The propagation of the spacecraft using GMAT occurs in three phases: generation of the necessary input files, execution of GMAT, and extraction of the output files.

GMAT is based on a scripting language known as GMAT script which is used to define an entire mission. A typical script file can be broken into two sections: an initialisation section, which creates mission objects and defines their state; and a mission sequence section, which contains the list of GMAT commands that are used to execute the mission [50].

To execute an orbit propagation, assam generates a GMAT script file by importing a template script file and then replacing relevant numerical fields with the desired inputs, including start and end dates, Keplerian elements, and parameters for the export of the propagated data. An example excerpt of the template illustrating the general format is presented in Listing 3.1, highlighting examples of parameters from the template that are modified by assam.

Once the script file is generated, GMAT is executed by assam by calling the software via the command-line. Several flags are added to the command at this stage to supply the path of the script file to GMAT, suppress any Graphical User Interface (GUI) elements, and to exit GMAT once the script has been completed.

A section of the generated script file includes the commands to generate the output text file containing the propagated state of the spacecraft. The spacecraft state is exported in a fixed width format with columns corresponding to the time, the spacecraft's position, and the spacecraft's velocity. This data is imported by assam using pandas, before processing to generate the Spacecraft Celestial Reference Frame (SCRF) using astropy.

Listing 3.1: Example excerpt from the script template, including the definition of the start time and the spacecraft's Keplerian elements in red and blue respectively.

```
Create Spacecraft Spacecraft;
GMAT Spacecraft.DateFormat = UTCModJulian;
GMAT Spacecraft.Epoch = '29294.0';
GMAT Spacecraft.CoordinateSystem = EarthMJ2000Eq;
GMAT Spacecraft.DisplayStateType = Keplerian;
GMAT Spacecraft.SMA = 7000;
GMAT Spacecraft.ECC = 0;
GMAT Spacecraft.INC = 98.6;
GMAT Spacecraft.RAAN = 90;
GMAT Spacecraft.AOP = 0;
GMAT Spacecraft.TA = 0;
```

The force model chosen in GMAT was selected to minimise the number of perturbing factors but still retain important features necessary for propagating the spacecraft's state. For example, *JASMINE*'s SSO means that capturing nodal precession is crucial, therefore a low-order form of the JGM-2 gravity field model, containing the J_2 influence, is included. Other perturbing factors such as other bodies (e.g. the Moon and the Sun), atmospheric drag, and Solar Radiation Pressure (SRP) are not included, as compensating for these effects was considered an operational concern, and modelling any manoeuvres was considered outside of the scope of this project.

For numerical integration the variable-step Runge-Kutta 89 scheme was selected due to its high computational performance and accuracy upon back-propagation in Low Earth Orbit (LEO) which corresponds to the expected operating conditions of *JASMINE*. The variable-step method improves computational performance by adjusting the length of each time-step to capture gradients of the solution correctly. The performance and accuracy of the available numerical integrators is presented in Table 3.1. The error shown in the table is the residual which exists between the original state, and a state which is first propagated forward in time, and then back-propagated backwards in time. The equations of motion are identical forwards and backwards, therefore this analysis should reveal any numerical errors.

Table 3.1: Comparison of the performance and accuracy of the available numerical integrators for the LEO test case. The selected integrator has been highlighted in bold. Adapted from [50].

	Numerical Integrator						
	RKV89	RKN68	RK56	PD45	PD78	ABM	PD853
Run Time [-]	**1.53**	1.00	2.14	2.78	1.46	3.41	1.80
Error [m]	**0.003**	64.06	0.022	0.002	0.006	0.012	0.013

The propagated state from GMAT is linearly interpolated with respect to time after it has been imported by assam. This provides a number of benefits, primarily that the variable-step results from GMAT can be converted to a fixed-step. The choice of fixed time-step can provide additional benefits, for example by selecting a smaller time-step, greater temporal resolution can be achieved but at a lower computational cost than propagating the spacecraft's orbit with the same time-step. Initially a fixed time-step of five minutes was chosen, however this is a parameter that can be modified.

3.3.2 Solar Body Propagation

The solar system body states are propagated with astropy, using the interpolated time vector generated during the propagation of the spacecraft. By interfacing with jplephem [51], ephemeris data generated by the Jet Propulsion Laboratory (JPL) is used to calculate these states. This method was found to have higher computational performance than the default method used by astropy while also providing higher accuracy and precision.

The apparent angular radius of each solar body is also calculated at this stage by the propagator module, using the geometry presented in Figure 3.2. This information is used during subsequent calculations by the visibility module to determine whether targets are obscured by any solar bodies.

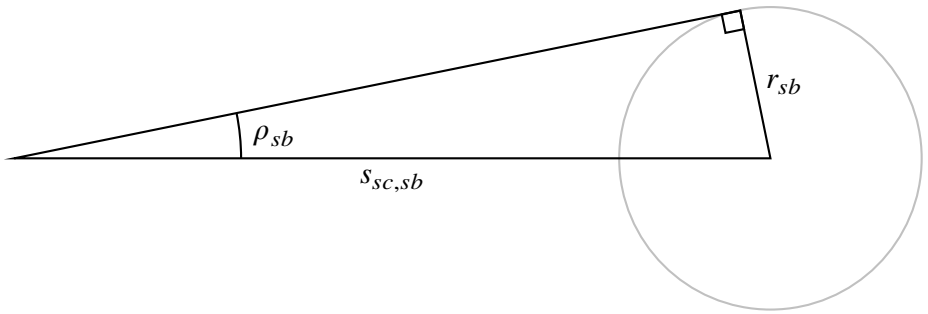


Figure 3.2: Solar body angular radius geometry. Redrawn from [3].

The apparent slant range between the spacecraft and the solar body, and the radius of the solar body are required to calculate the apparent angular radius [3]:

$$\rho_{sb} = \arcsin \frac{r_{sb}}{s_{sc,sb}}, \tag{3.1}$$

where ρ_{sb} is the angular radius of the solar body, r_{sb} is the radius of the solar body, and $s_{sc,sb}$ is the apparent slant range from the spacecraft to the solar body. The slant range between the spacecraft and the solar body varies with respect to time, therefore the angular radius of each solar body is calculated for each time-step.

3.3.3 Reference Frames

Reference frames were a key consideration during the development of `assam` due to concerns regarding compatibility between the different software used to build the tool, most critically when interfacing `GMAT` and `astropy` to generate the SCRF. Several reference frames were considered to act as the intermediary between the two software.

The International Celestial Reference System (ICRS) was adopted by the International Astronomical Union (IAU) in 1998 as the standard reference system for astronomy [52]. The frame, centred at the solar system's barycentre, is realised as the International Celestial Reference Frame (ICRF) through astrometric observations of over 3400 extra-galactic radio sources using Very Long Baseline Interferometry (VLBI) techniques [52].

The Geocentric Celestial Reference System (GCRS), realised as the Geocentric Celestial Reference Frame (GCRF), is the geocentric equivalent of ICRS [52]. The system predates ICRS as it was introduced by the IAU in 1991 [52]. The system was originally based on the predecessor of ICRS, IAU-76/FK5, however GCRS and ICRS were subsequently harmonised through the IAU-2000 Resolutions [52].

The primary reference system chosen for `assam` was GCRS. Since this system is the standard inertial coordinate system for the Earth, it is a very commonly used frame. Compatibility with different software such as propagators will subsequently be very high.

`GMAT` supports both the IAU-1976/FK5 and IAU-2000 theories [50], therefore the spacecraft's state can be exported in a variety of different systems including both ICRF and GCRF.

The choice of GCRF is also beneficial from a procedural viewpoint as it enables the use of a key feature of the `astropy.coordinates` system: the ability to create offset GCRF frames, where a new frame, aligned in rotation with GCRF but offset by a geocentric position and velocity, is created. This new reference frame is compatible with all of the `astropy.coordinates` tools.

The use of the offset GCRF means that the effects of aberration are included and calculated by `astropy`. Aberration is a phenomena that occurs with astronomical observations which results in an apparent angular offset. Two main factors contribute to this effect, firstly light-time aberration caused due to the time taken for light from the observed object to reach the observer, and secondly stellar aberration caused due to relative motion between the observed object and the observer [52]. Although these effects are typically small (below 36 mas in the case of the planets) it remains important to include these effects to ensure accurate results.

The spacecraft position and velocity, exported by `GMAT` in the geocentric GCRF, is used to create an offset SCRF as illustrated in Figure 3.3. This frame is used for all further calculations such as when calculating visibility. Since it is an `astropy` coordinate frame, the coordinates of other bodies such as planets and stars are easily transformed into the SCRF.

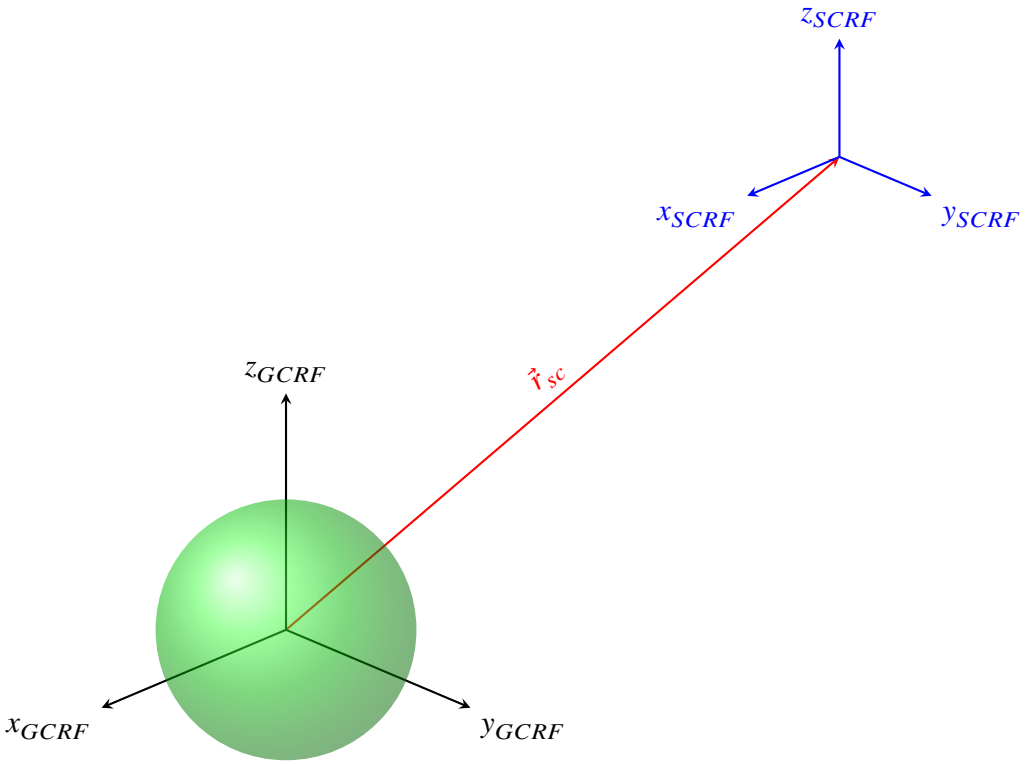


Figure 3.3: Illustration of the SCRF. The GCRF base frame (shown in black) is offset by position vector $\vec{r}_{sc}$ (shown in red) to create the SCRF (shown in blue). Redrawn and adapted from [3].

3.4 Visibility Module

3.4.1 Target Definition

Each astronomical target is defined for `assam` in a target definition file with parameters including its name, its category (e.g. M-type star), its priority, and its sub-targets. The inclusion of a category parameter allows for filtering of targets during the following analysis, and the priority parameter allows for certain targets, such as the Galactic Centre, to be prioritised during observation scheduling.

The use of sub-targets for each target enables `assam` to capture complex target geometries by combining multiple regions into one target. An example of a complex target is the Galactic Centre target, illustrated in Figure 3.4, which is comprised of a circular and rectangular region, forming a whistle-shaped region.

An example excerpt from the target definition file, which includes the definition of the Galactic Centre, is presented in Listing 3.2. This file includes both the general target information as discussed previously, and the sub-targets' parameters, including their names, reference frames, centre coordinates, and geometry.

Listing 3.2: Example excerpt from a target definition file.

```
galactic_centre:
    category: Galactic Centre
    priority: 1
    subtargets:
        region_1:
            frame: galactic
            centre: [0, 0]
            shape: circular
            width:
            height:
            angular_radius: 0.7
        region_2:
            frame: galactic
            centre: [-0.65, 0.15]
            shape: rectangular
            width: 2.7
            height: 0.3
            angular_radius:

midplane_-180:
    category: Galactic Mid-plane
    priority: 2
    subtargets:
        region:
            frame: galactic
            centre: [-180.0, 0]
            shape: circular
            width:
            height:
            angular_radius: 0.7

00064325-0732147:
    category: M-type
    priority: 3
    subtargets:
        star:
            frame: icrs
            centre: [1.679958, -7.5380623]
            shape: circular
            width:
            height:
            angular_radius: 0.3
```

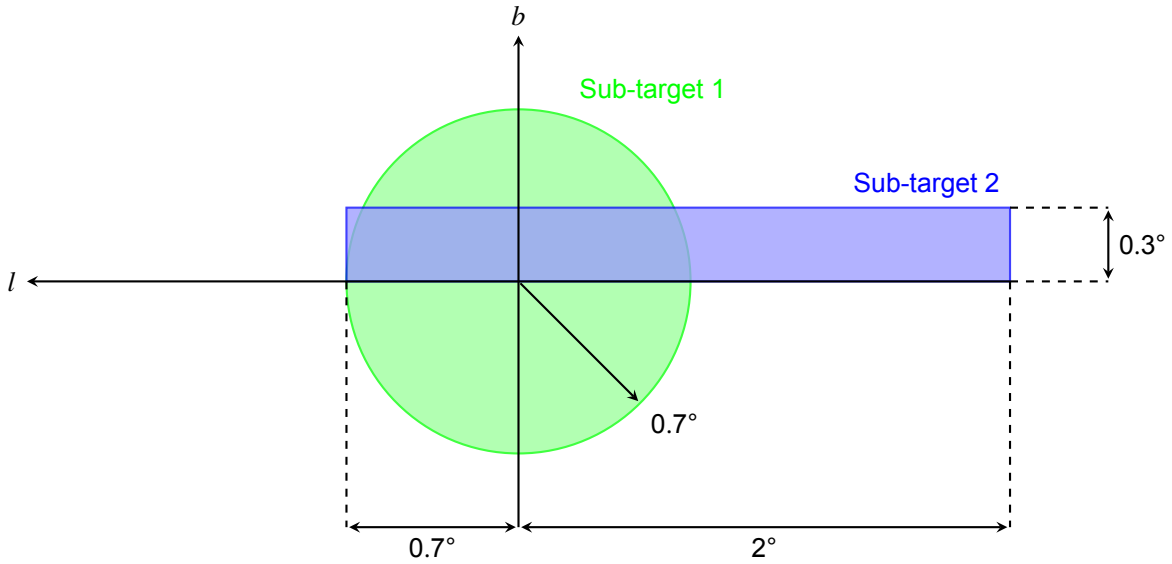


Figure 3.4: Galactic Centre target geometry. Galactic longitude and latitude are indicated by l and b respectively. Redrawn and adapted from [3, 22].

The full set of targets is imported from the target definition file and stored by `assam` as a list of target objects. Within each of the target objects, the sub-targets are stored as separate objects in a list. Certain parameters are processed by `assam` when they are imported, such as the coordinates which are stored both in SCRF and in ICRF, and the geometry. Although the target definition file supports both circular and rectangular geometries, the rectangular regions are converted to the equivalent bounding circular geometry.

3.4.2 Separation

Angular separations between coordinates are calculated using `astropy`. After confirming that the two coordinates exist within the same coordinate system, `astropy` calculates angular separation between two sets of coordinates by using the Vincenty formula, a formula designed for geodesy to calculate the distance between two points on a spheroid [53, 54]. The inverse method can be simplified for the spherical case to calculate angular separations between points on the celestial sphere:

$$\theta = \arctan \frac{\sqrt{\left(\cos\phi_2 \sin\left(\lambda_2 - \lambda_1\right)\right)^2 + \left(\cos\phi_1 \sin\phi_2 - \sin\phi_1 \cos\phi_2 \cos\left(\lambda_2 - \lambda_1\right)\right)^2}}{\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos\left(\lambda_2 - \lambda_1\right)}, \quad (3.2)$$

where θ is the angular separation between the points, λ_1 and λ_2 are the longitudes of the first and second points respectively, and ϕ_1 and ϕ_2 are the latitudes of the first and second points respectively. This formula was chosen during the development of `astropy` over alternatives: while it requires more evaluations of trigonometric functions, “[it] is stable at all distances, including the poles and antipodes” [53].

3.4.3 Visibility Constraints

Targets are subject to a variety of visibility constraints which affect whether certain regions of the sky are visible or not from the perspective of the space telescope. Visibility constraints are broken into two categories: physical constraints and mission constraints.

Physical

Physical constraints refer to those constraints which are dictated by an object physically occulting a target. For example, solar bodies such as the Earth will block light from a target it is occulting from the perspective of the space telescope. For solar bodies, their angular radii are calculated previously as discussed in Section 3.3.2.

The separation geometry used for calculating visibility subject to physical constraints is presented in Figure 3.5. Under this scheme, a target is considered visible if the two bounding circles of the regions do not intersect with each other, and thus one body does not occult the other. The target is visible if the following equation is satisfied [3, 55]:

$$\theta \geq \rho_1 + \rho_2, \tag{3.3}$$

where θ is the angular separation between the centres of the circular regions, and ρ_1 and ρ_2 are the regions' angular radii.

Mission

Mission constraints refer to those constraints which are not necessarily linked to physical constraints such as a solar body occulting a target. For example, *JASMINE*'s stray light and thermal constraints with respect to the Sun are considered mission constraints as they are not a result of the Sun's size in the sky.

The separation geometry used for calculating visibility subject to mission constraints is presented in Figure 3.6. Mission constraints can be seen as an extension of physical constraints, where the solid circular region of the physical constraint is replaced by an annulus with both inner and outer radii.

For mission constraints, it is simpler invert an expression of the conditions under which the target is not visible. The target is not visible if the following equation is satisfied [3]:

$$\rho_1 - \rho_2 < \theta < \rho_2 + \rho_3, \tag{3.4}$$

where θ is the angular separation between the centres of the circular regions, ρ_1 and ρ_3 are the inner and outer radii of the mission constraint, and ρ_2 is the angular radius of the target.

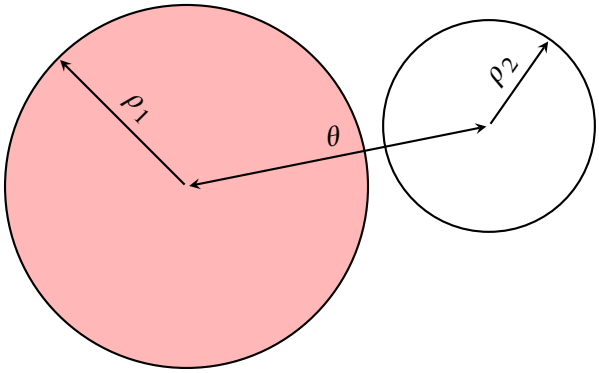
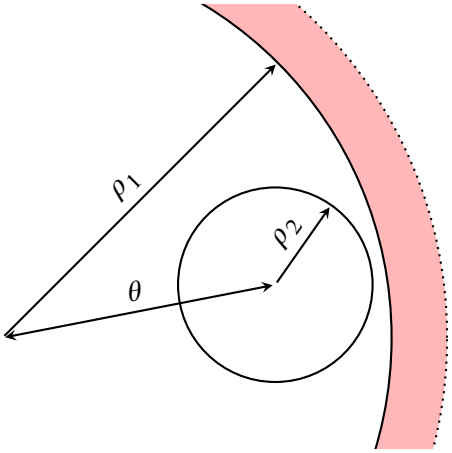
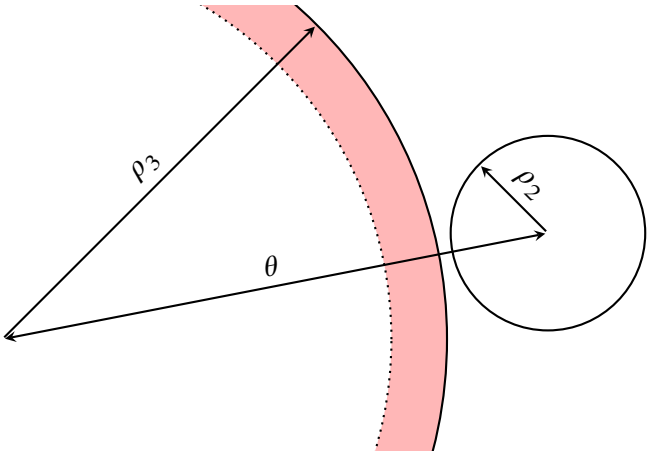


Figure 3.5: Separation geometry for physical constraints. The target would not be considered visible if it intersected with the red zone. Redrawn and adapted from [3, 55].



(a) Inner geometry.



(b) Outer geometry.

Figure 3.6: Separation geometry for mission constraints. The target would not be considered visible if it intersected with the red zone. Redrawn from [3].

An example of the use of mission constraints, as discussed above, is enforcing the pointing restrictions due to thermal and stray light concerns due to the Sun. The mission constraint provides a high level of flexibility, due to the option of a hollow centre through the inclusion of an inner radius. This flexibility is used to create the Sun's mission constraint as illustrated in Figure 3.7. The avoidance cone pointing outwards from the Sun requires the option of the inner radius so as to not restrict pointing which is normal to the Earth-Sun line.

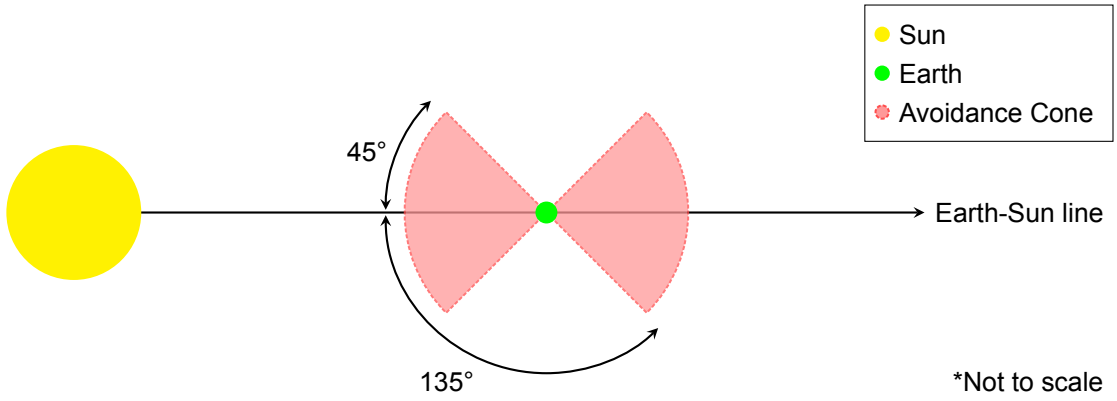


Figure 3.7: Observation constraint for *JASMINE* with respect to the Sun as discussed in Section 2.3. The 45° constraints in both directions of the Earth-Sun line can be broken into two mission constraints, one from 0° to 45°, and another from 135° to 180°.

3.4.4 Target Contacts

For each time-step, `assam` iterates through combinations of every target and its underlying sub-targets, and visibility constraints to calculate which targets are visible at the corresponding time. For each target, the visibility of each sub-target is combined through logical conjunction to form a vector of Booleans which contains whether the target is visible for each time-step, referred to as the target's visibility vector. An example plot of a visibility vector is presented in Figure 3.8 showing the square wave-like nature of the visibility.

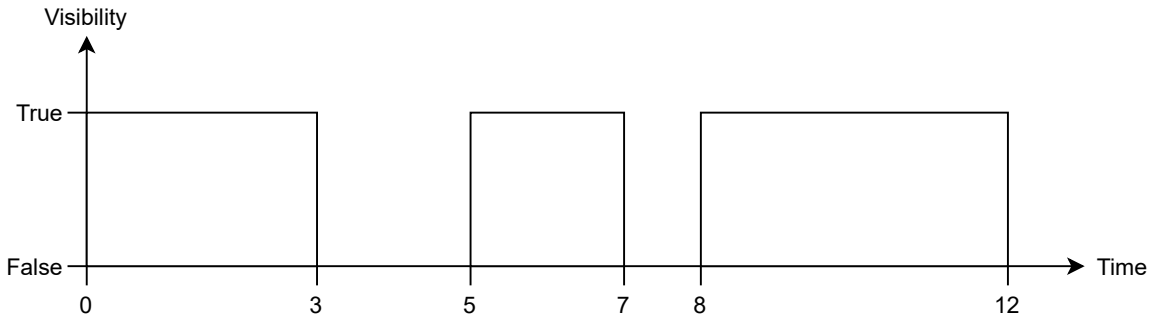


Figure 3.8: Example plot of a visibility vector, where the target is visible at a given time if the corresponding element of the visibility vector is true.

The visibility vector for each target is then converted into a series of target contact objects, where a target contact is defined as a continuous period where the target is visible. This is achieved by applying a Run-length Encoding (RLE) scheme to each of the visibility vectors which extracts the start and end times for each run of continuous state, either visible or not. These runs are filtered to only include runs where the target is visible. These visible runs are then used to generate the target contact objects. An example of this process is presented in Table 3.2.

Table 3.2: Extracted runs by applying the RLE scheme to the example visibility vector presented in Figure 3.8. The examples in bold, where the target is visible, would be used to generate the target contacts.

Index	Visibility	Start Time	Duration	End Time
1	**True**	**0**	**3**	**3**
2	False	3	2	5
3	**True**	**5**	**2**	**7**
4	False	7	1	8
5	**True**	**8**	**4**	**12**

3.4.5 Target Statistics

The final task of the visibility module is to compile a table containing overall statistics for each target. These statistics include the number of contacts, the total visibility, the percentage visibility, the mean contact duration, the standard deviation of contact durations, and the minimum and maximum contact durations.

The overall statistics table also includes the mean coordinates of each target. These are calculated by converting the sub-targets' coordinates into Cartesian form, taking the mean of these values, and converting back from Cartesian form into angular form:

$$\bar{\alpha} = \arctan \frac{\frac{1}{n}\sum_i^n \sin \alpha_i}{\frac{1}{n}\sum_i^n \cos \alpha_i}, \tag{3.5}$$

$$\bar{\delta} = \arctan \frac{\frac{1}{n}\sum_i^n \sin \delta_i}{\frac{1}{n}\sum_i^n \cos \delta_i}, \tag{3.6}$$

where $\bar{\alpha}$ and $\bar{\delta}$ are the mean right ascension and declinations respectively, α_i and δ_i are the sub-targets' right ascension and declinations respectively, and n is the number of sub-targets.

The mean coordinates for each target are currently only used by `assam` for the overall target statistics, however they are calculated earlier in the visibility module, during the target definition phase when the targets are imported from the target definition file.

3.5 Scheduling Module

The scheduling module currently implements a dynamic programming method for telescope scheduling as described by Goodrich and Tamassia [27] and outlined below. This method operates on a list of contacts, each with a start time s_i, finish time f_i, and benefit b_i which quantifies the scientific gain for including the observation.

The list of target contacts is initially unsorted with respect to time as the list is generated by appending the contacts from each target into one list. To break down the overall scheduling problem into smaller sub-problems in time, the overall list of target contacts is sorted in ascending order by finish time.

The concept of a predecessor p_i is introduced and calculated, which for each contact i is the latest finishing contact before it without a conflict (i.e. the largest index j for which $f_j < s_i$). For `assam`, the predecessor is calculated by stepping backwards in time from the currently considered contact, and comparing the start and end times, halting when the first non-conflicting contact is found. Conducting this search backwards in time avoids unnecessary comparisons of contacts which could have a large temporal separation from the currently considered contact.

The scheduling problem can be broken down into smaller sub-problems which can be defined in terms of a parameter B_i: the maximum benefit which can be achieved with the first i contacts in the sorted list. In this case, the targets are indexed from one, therefore B_0, which represents the initial solution where no contacts are included, will be zero.

The sub-problems can be defined as a recursive problem:

$$B_i = \max\left(B_{i-1}, B_{p_i} + b_i\right), \tag{3.7}$$

where B_i is the current maximum benefit, B_{i-1} is the previous maximum benefit, B_{p_i} is the maximum benefit of the current contact's predecessor, and b_i is the benefit gained from the current contact. Effectively as `assam` iterates through each of the contacts, it is calculating whether the overall benefit improves by adding the current contact to a previous solution without conflicts, or if the solution corresponding to the previous maximum benefit remains optimal.

The benefit for each contact is calculated by `assam` as follows:

$$b_i = \frac{f_i - s_i}{P_i}, \tag{3.8}$$

where b_i is the current contact's benefit, s_i and f_i are the start and finish times respectively of the contact, and P_i is the priority of the current contact's corresponding target as defined in the target definition file. The benefit of a contact is therefore proportional to its duration, encouraging the selection of long contacts where several exposures can be taken, and inversely proportional

to the priority of the target as lower values of priority correspond to the most important (e.g. a priority value of one corresponds to the most important target).

An example of assigned priorities is shown in Listing 3.2 where the Galactic Centre target is given a priority of one, as it is the priority target for *JASMINE*, and examples of the Galactic mid-plane and M-type star targets are given priorities of two and three respectively.

The main advantage of this scheduling method over alternatives is its low complexity with respect to the number of contacts. The running time of the algorithm is $O(n)$ where n is the number of contacts, therefore the algorithm scales extremely well as the number of contacts increases. For comparison, a brute-force algorithm which evaluates all possible subsets scales by $O(n2^n)$ [56], therefore this would method require approximately 2^n times longer to execute. This is a significant result when considering that the number of contacts for *JASMINE* over a year approaches the order of 10^6, therefore brute-forcing a solution would effectively be impossible.

The built-in `Python` list sorting method, used for the initial sort of the contacts by end time, uses the Timsort algorithm [57]. This algorithm has a worst-case performance of $O(n \log n)$, however performance can be better as any existing ordering is exploited [57].

An example of this dynamic programming method, applied to a set of contacts with varying benefits, is presented in Figure 3.9. From the input set of seven contacts, a solution containing four of these contacts was produced, with an overall benefit of 18, which was optimal.

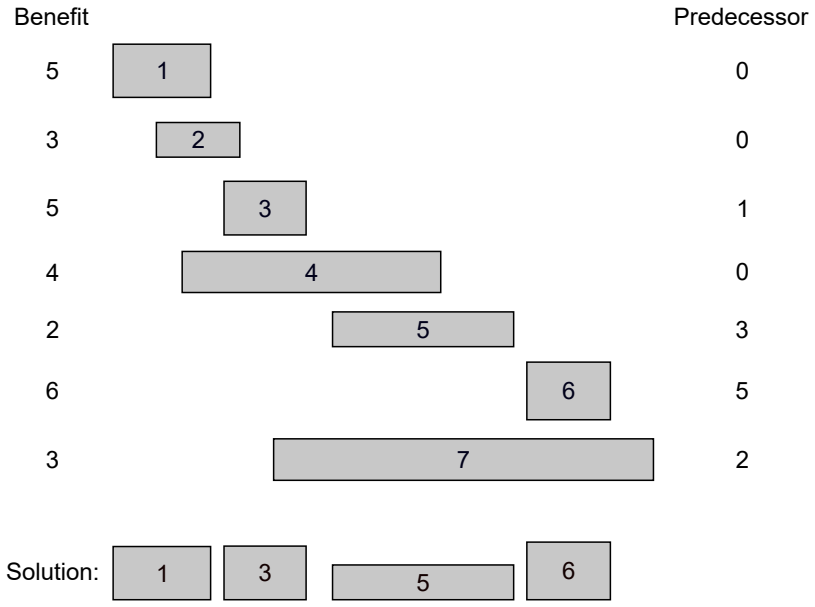


Figure 3.9: Example of a telescope scheduling problem. Each target contact has a start and end time shown by its left and right boundaries, with its benefit and predecessor listed on the left and right hand sides respectively. Redrawn and adapted from [27].

3.6 Visualisation Module

3.6.1 Sky View

The sky view visualisation shows the view of the sky from the perspective of the spacecraft, highlighting the visible and invisible regions of the sky, and the locations of the targets. These visualisations serve to illustrate the times and positions of targets that are obstructed by either physical and/or mission constraints, as discussed in Section 3.4.3.

The sky views are generated by calculating the angular separation between the targets and visibility constraints, and a structured mesh of points covering the entire sky. This structured mesh is effectively a large set of pseudo-targets with angular radii of zero. The visibility at each of these points is calculated by applying Equation 3.3 in the case of physical constraints and targets, and Equation 3.4 in the case of mission constraints. This results in a two dimensional array of Booleans for each target and each constraint where the elements are true if it intersects with a target or a constraint. The arrays are combined with a logical disjunction to form a single array for the targets and another for the constraints at each time-step.

An example of the sky view generation is presented in Figure 3.10 for a physical and a mission constraint, and two targets. This example highlights one of the main limitations of the method which is the high amount of visual aliasing that can be introduced due to the sampling strategy. This can lead to jagged edges, and inaccurate capture of the objects' shapes. Nevertheless, this is not critical as the generated plots provide qualitative information which does not affect the accurate visibility calculations conducted in the visibility module.

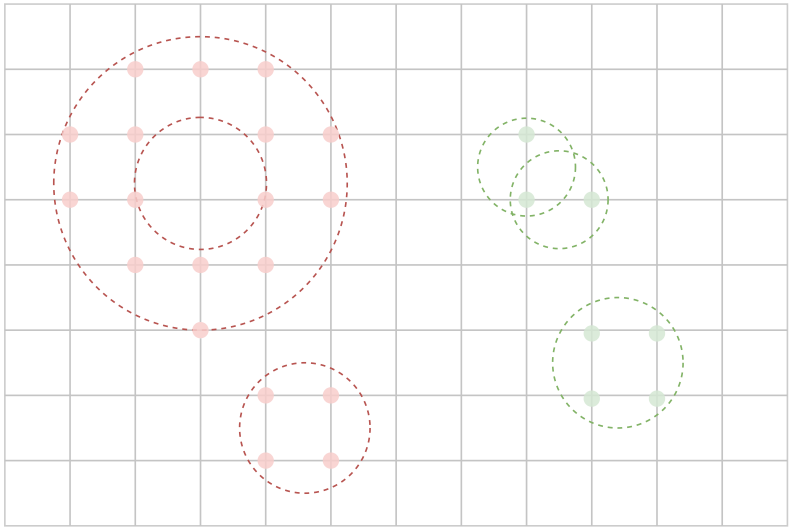


Figure 3.10: Example of a structured mesh of pseudo-targets used to generate the sky views. The constraints and targets are illustrated by red and green dashed circles respectively. The array elements which are true are illustrated with coloured dots.

Computational performance was identified as a major limitation of this rendering method due to the large number of separation calculations that have to be conducted. For example, a mesh of the full sky with 0.5° by 0.5° resolution includes 260 281 pseudo-targets, which for *JASMINE*'s 350 targets would require in excess of 10^7 separation calculations at each time step. Hardware acceleration, using CUDA cores through `CuPy`, was used to improve performance by taking advantage of the high performance provided by GPUs over conventional processing on Central Processing Units (CPUs). Using the GPU instead of the CPU required rewritten code to implement the separation calculations discussed in Section 3.4.2, however this was trivial due to `CuPy` acting as a drop-in replacement for `NumPy`.

3.6.2 General Visualisations

General visualisations are generated by the visualisation module to enable interpretation of the overall target statistics calculated by the visibility module, as discussed in Section 3.4.5. These include scatter plots and box plots of the statistics. The use of `seaborn` with data structures from `pandas` means that the creation of new visualisations in the future will be relatively trivial.

Chapter 4

Results

4.1 Spacecraft Orbit

An example of *JASMINE*'s orbit at four times of the year between, and including, the vernal equinox and the summer solstice is presented in Figure 4.1. One of the key behaviours of the spacecraft's orbit is highly visible in this plot which shows the nodal precession of the orbit about the z-axis due to the non-sphericity of Earth.

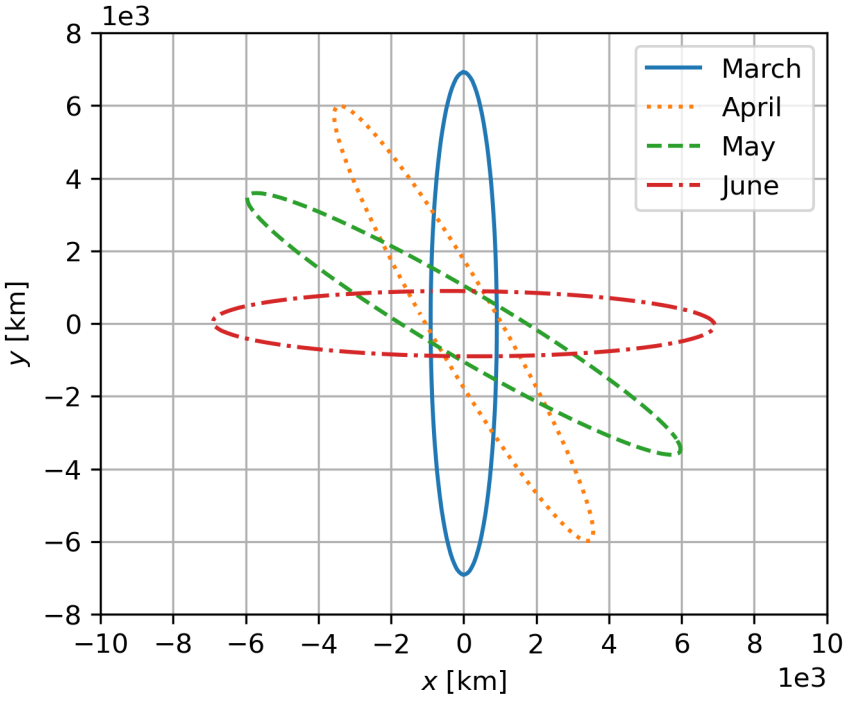


Figure 4.1: An example of one of *JASMINE*'s orbits at four times during the year in the GCRF. This illustrates the precession of the spacecraft's orbit about the celestial pole (z-axis) due to the J_2-term, matching the Sun's movement with respect to the celestial sphere.

The key feature of SSOs is this nodal precession, however specifically at a rate matching that of the Sun's movement against the celestial sphere. This rate corresponds to the Sun's movement of 360° per year. It was therefore expected that *JASMINE*'s orbit would precess by 90° during a quarter year as seen in Figure 4.1. These results act as a confirmation that the low-order gravity model specified in the `GMAT` script file is successfully capturing the J_2-term which is required for the spacecraft's orbit to precess.

4.2 Short-term Visibility

An example of the short-term visibility of the Galactic Centre on the vernal equinox, using the parameters in Table 4.1 and Listing 4.1, is presented in Figure 4.2. The binary state of the visibility is shown by the two possible states: true or false. The Galactic Centre shows a periodic variation between visible and invisible with a period of approximately 95 min, corresponding to the orbital period of the spacecraft.

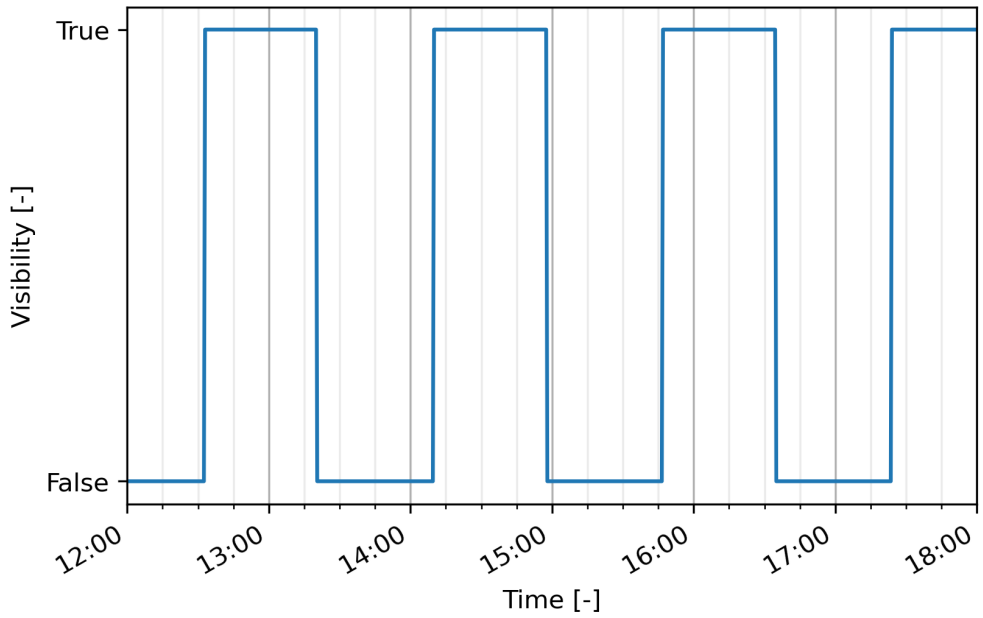
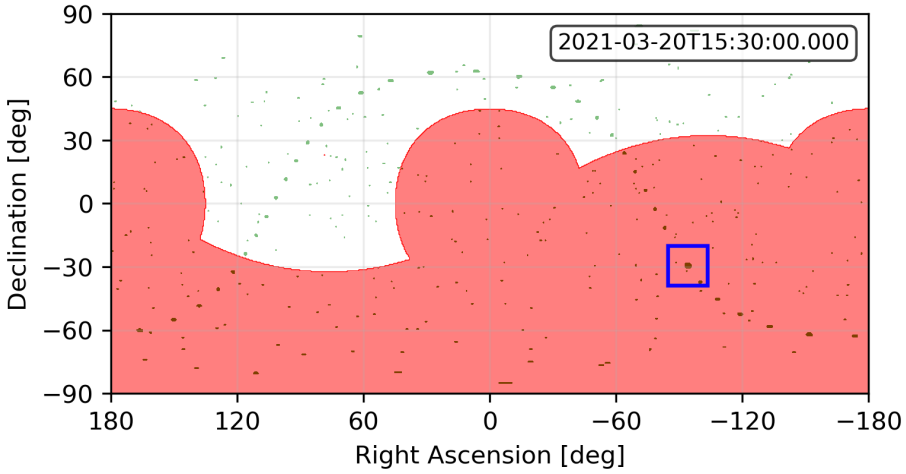
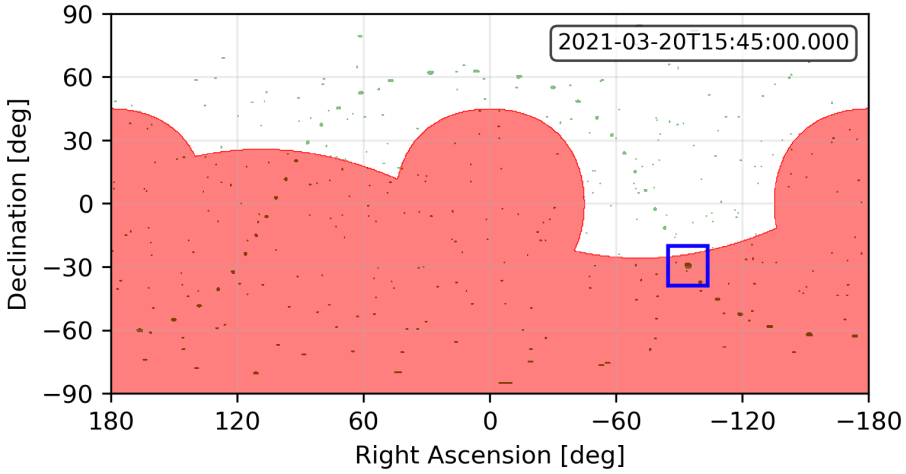


Figure 4.2: Galactic Centre visibility on the vernal equinox.

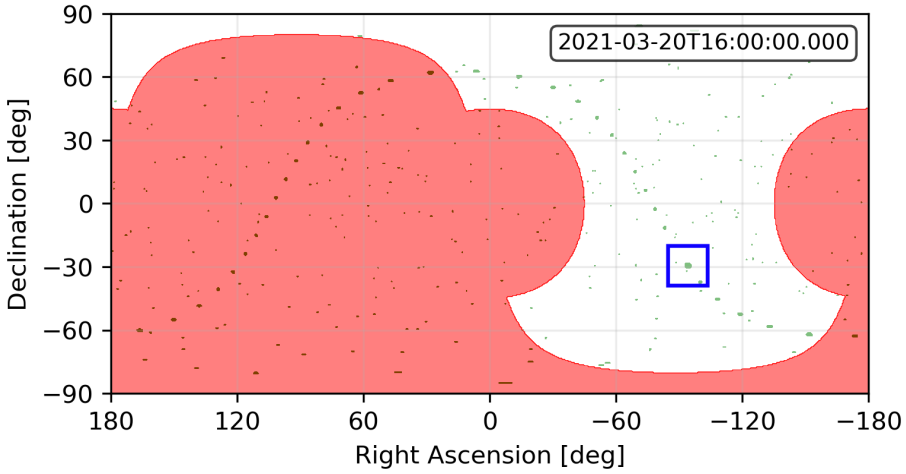
The sky view from the perspective of the spacecraft at three times during the vernal equinox is presented in Figure 4.3. The Sun's mission constraint can be seen as the two circular regions in red, with angular radii of 45° and centred at ($\alpha = 0°$, $\delta = 0°$) and ($\alpha = \pm 180°$, $\delta = 0°$) as expected due to the intersection of the ecliptic and the celestial equator on this day. The primary difference at the three time-steps was the movement of the largest constraint, the mission constraint of the Earth, as the spacecraft orbited the Earth. Between 15:45 and 16:00, the spacecraft moves sufficiently for the Galactic Centre to no longer be intersecting with the Earth's mission constraint and therefore the target becomes visible as also shown in Figure 4.2.



(a) 15:30. The Galactic Centre is not visible.



(b) 15:45. The Galactic Centre is not visible.



(c) 16:00. The Galactic Centre is visible.

Figure 4.3: Sky view on the vernal equinox at various times, illustrating the Earth's mission constraint moving to unobstruct the Galactic Centre ($\alpha = -93°$, $\delta = -29°$), highlighted by the blue box. The physical and mission constraints are shown in red, and the targets in green.

4.3 Long-term Visibility

The annual visibility between two consecutive vernal equinoxes, using the parameters in Table 4.1 and Listing 4.1, of *JASMINE*'s targets is presented in Figure 4.4a. All of the targets were visible at some point during the year with every target having a minimum visibility above 20% during the year. Several targets had significantly higher annual visibility, reaching near 50%. This indicated that these targets had no major obstructions except for the Earth which would block the targets for approximately half of the spacecraft's orbit.

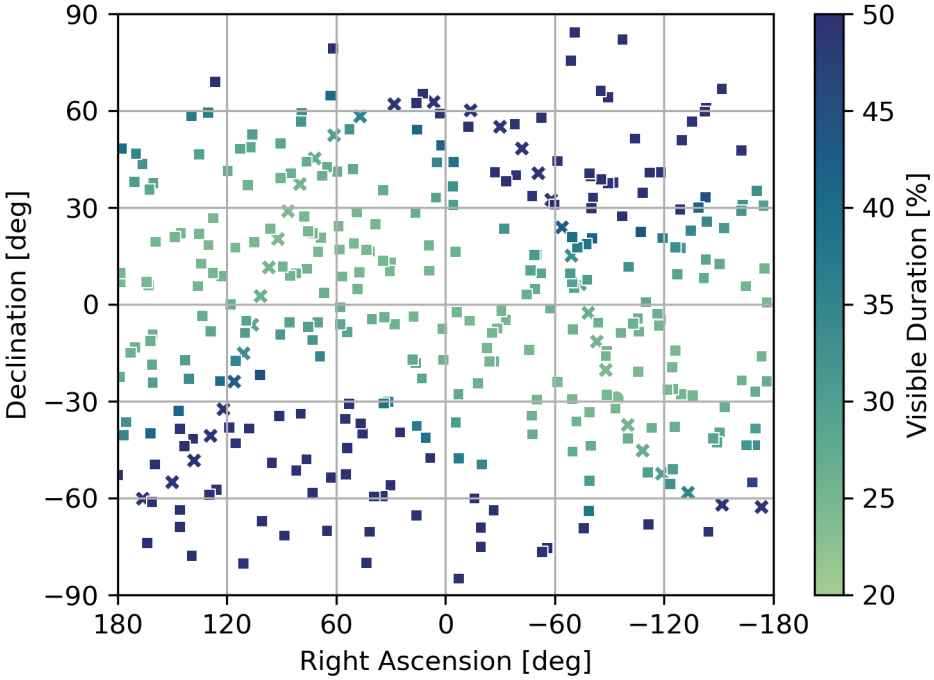
The targets' positions in the sky appeared to highly affect their annual visibility. A sine-like band can be seen passing through the sky, in which targets have significantly lower visibility, closer to one quarter of the year. This significant reduction occurred due to the mission constraint imposed by the Sun. Due to the Sun's apparent movement in the sky, this resulted in a moving band across the sky resulting in a period without visibility. The inclination of the ecliptic with respect to the equator of around 23.5° [52] resulted in the sine-like movement of the Sun.

The annual visibility of targets, broken down by target category, is presented in Figure 4.5a. The Galactic Centre had the lowest visibility of any category at around 24%. The Galactic Mid-plane had a median visibility of 30%, and M-type stars had a median visibility of 30%. Both of these categories had similar minimum and maximum visibility of around 24% and 50%.

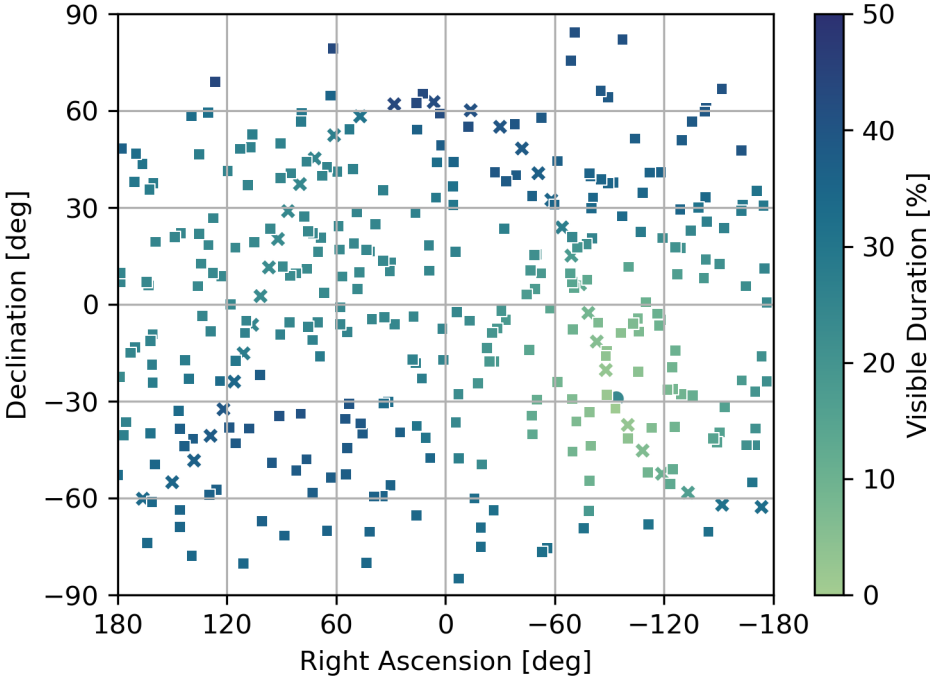
Although all three categories had targets which were subject to the Sun's mission constraint at some point during the year, their minimum visibility varied between each category, increasing from the Galactic Centre to the M-type stars. This can be explained by the size of the targets which decreased in radius from the largest (the Galactic Centre) to the smallest (the M-type stars). The smaller radii meant that the Sun had to have a smaller separation with the target before preventing an observation, therefore there was a smaller period during the year where these targets were obstructed.

The annual visibility of *JASMINE*'s targets was also investigated for the scenario where the priority target, the Galactic Centre, is always observed when it is visible. Under these conditions, the other targets were considered invisible to *JASMINE* during times where the Galactic Centre was visible as these observation periods were used exclusive for the Galactic Centre.

The annual visibility, as a function of the position of the target, for this scenario is presented in Figure 4.4b. The annual visibility for most of the targets reduced significantly, particularly for targets near the Galactic Centre, whose visibilities dropped below 5% of the year. The general reduction in annual visibility is highly visible in Figure 4.5b, with the median visibility dropping by around 8 and 5 percentage points for the Galactic Mid-plane and M-type star targets respectively. The significant reduction in target visibility near the Galactic Centre was also highlighted, with minimum visibilities of the targets reducing to 3% and 1% respectively.

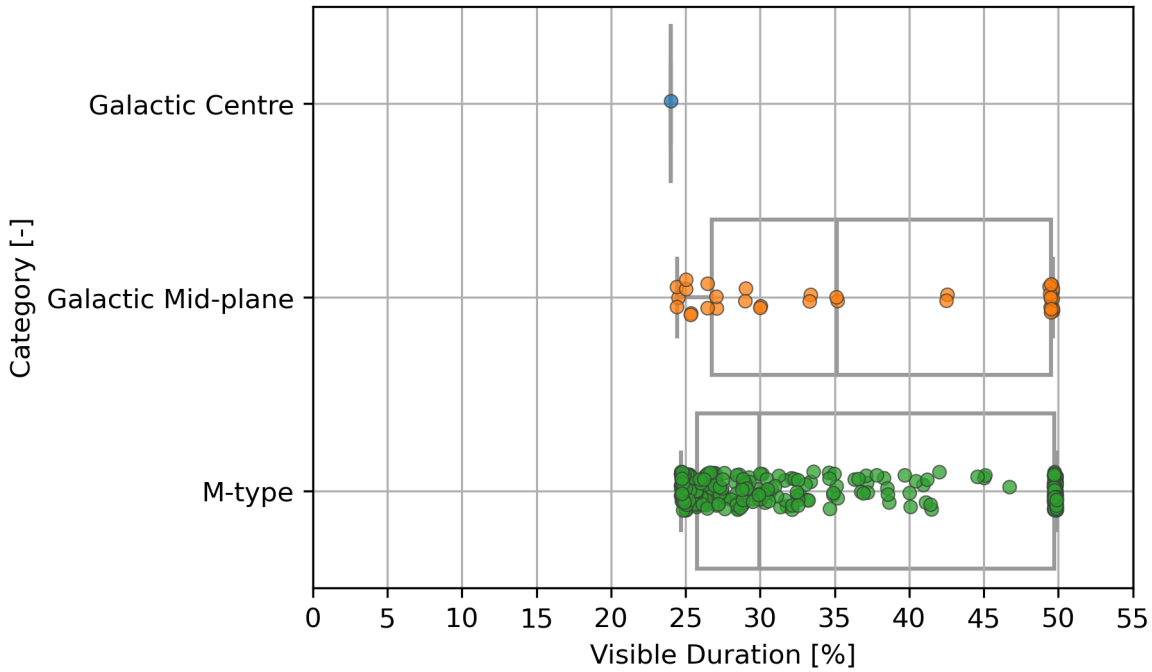


(a) Raw.

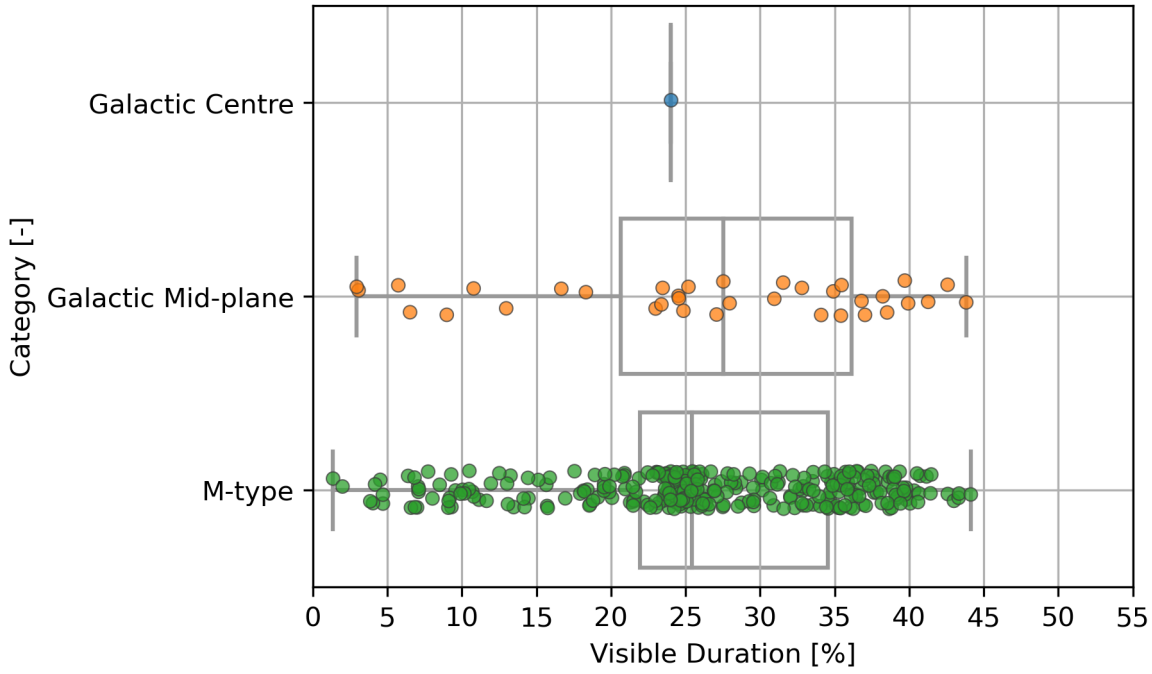


(b) Processed by removing periods where the Galactic Centre is visible.

Figure 4.4: Target visibility over a period of one year as a function of location in the sky.



(a) Raw.



(b) Processed by removing periods where the Galactic Centre is visible.

Figure 4.5: Target visibility over a period of one year broken down by category. The box plots indicate the median, quartiles, and minimum and maximum of each category. The points are spread vertically within each category to aid visualisation of their concentration with respect to the visible duration.

4.4 Visibility Dependence on Earth Avoidance Angle and Altitude

One of the key considerations during the design of a space telescope is the size of the baffle. This is the component of the telescope that prevents unwanted stray light from entering the telescope's optics. Its design can have a significant impact on the telescope's performance.

The size of the baffle is an important property of *JASMINE*'s telescope as avoiding the refracted and scattered light passing through the Earth's atmosphere was identified as an important requirement for accurate observations. The sizing of the baffle can be related to an avoidance angle with Earth: an additional angular offset added to the existing physical constraint, which can be modelled as a mission constraint whose outer angular radius is the sum of the physical constraint radius and the avoidance angle. The Earth's physical constraint, and hence the total mission constraint including the avoidance angle, varies with altitude due to the angular radius of Earth varying with slant range. A relationship between the mission constraint (including the avoidance angle), the spacecraft's altitude, and the Galactic Centre visibility was investigated as it was important to understand for the development of the baffle.

The annual visibility of the Galactic Centre as a function of the Earth's mission constraint and the spacecraft's altitude is presented in Figure 4.6. A direct parameter sweep of the spacecraft's altitude and mission constraint with respect to the Earth was conducted. This parameter sweep required a high number of function calls from `assam` for each configuration, therefore to reduce the overall execution time, only the time between March and June was evaluated. Nevertheless, the results are still applicable for an entire year due to the similarity of the Earth-Sun geometry with the remaining three quarters of the year.

The results indicate that annual visibility of the Galactic Centre remains near constant with respect to the spacecraft's altitude but has a strong relationship with the mission constraint, with visibility increasing as the mission constraint reduces. This shows that to increase annual visibility of the Galactic Centre, the mission constraint must be reduced. This relationship was expected as reducing the mission constraint results in the Earth blocking a smaller amount of the sky, hence reducing the period that it can block targets such as the Galactic Centre.

Although the altitude does not directly affect the annual visibility of the Galactic Centre, it does affect the annual visibility if the avoidance angle is kept constant: increasing the altitude decreases the mission constraint for a given fixed avoidance angle, increasing the visibility. For example, for the nominal altitude of 550 km and avoidance angle of 25°, the expected annual visibility is around 21.5%, however increasing the altitude to 650 km would result in a visibility closer to 22%. Alternatively, changing the avoidance angle directly, and hence the mission constraint, could increase visibility at a fixed altitude: for example, decreasing the avoidance angle to 20° would increase visibility to approximately 23% at 550 km.

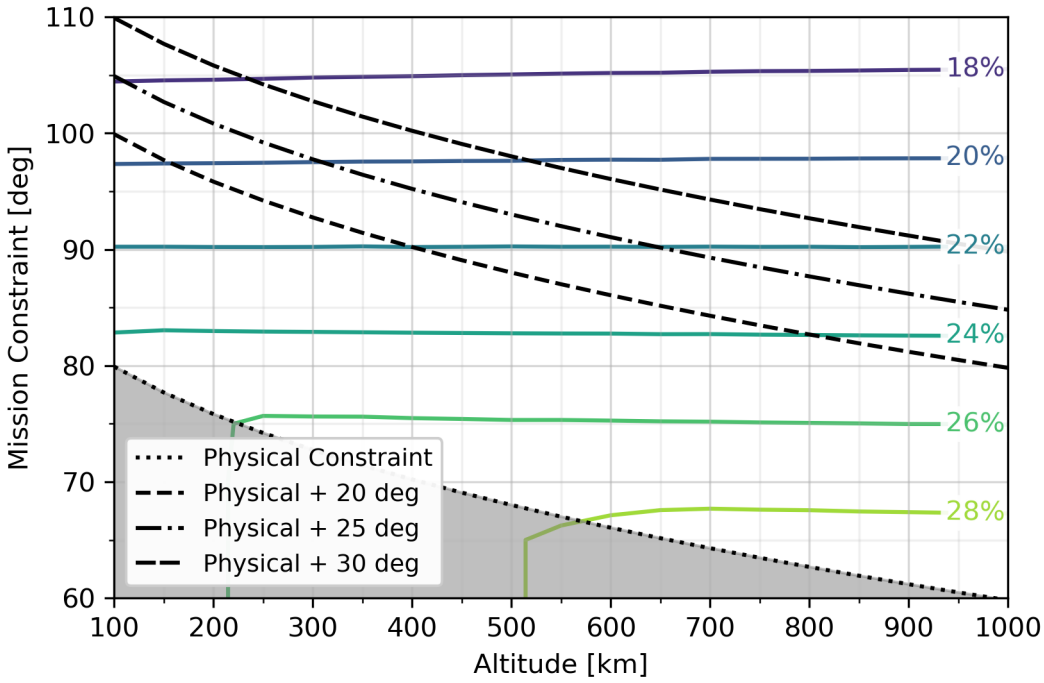


Figure 4.6: Galactic Centre visibility between the vernal equinox and summer solstice as a function of the Earth's mission constraint and the spacecraft's altitude. The coloured, solid contours are the expected annual visibility, with percentages shown on the right hand side of the figure. The dashed lines are lines of constant avoidance angle added to the Earth's physical constraint shown as the dotted line and greyed-out area at the bottom of the figure.

The altitude of *JASMINE* is driven heavily by the performance of the Epsilon launch vehicle as discussed in Section 2.3. It would, therefore, be unlikely that this aspect of the mission could be modified to change the Galactic Centre visibility. Consequently, the only option which remains for improving Galactic Centre visibility would be to reduce the avoidance angle with Earth. This would require a larger baffle to improve rejection of stray light, increasing the mass and volume, and potentially the cost, of the telescope.

Table 4.1: Parameters used by `assam`.

Parameter	Short-term	Long-term
Start Time [-]	2021-03-20 12:00	
End Time [-]	2021-03-20 18:00	2022-03-20 12:00
Time Step [min]	1	5
Semi-major axis [km]	6921	
Eccentricity [-]	0	
Inclination [deg]	97.57	
Right ascension of the ascending node [deg]	90	
True anomaly [deg]	0	

Listing 4.1: Solar bodies definition file. Due to a change in nomenclature, soft radius in the file is a legacy term which refers to mission constraints. Mission constraints of 0° to 45° and 135° to 180° were applied with respect to the Sun, and 0° to 90° with respect to the Earth.

```
sun:
    included: True
    radius: 696340000
    soft_radius: [[0,45],[135,180]]
mercury:
    included: True
    radius: 2439700
    soft_radius: []
venus:
    included: True
    radius: 6051800
    soft_radius: []
earth:
    included: True
    radius: 6371000
    soft_radius: [[0,90]]
moon:
    included: True
    radius: 1737100
    soft_radius: []
mars:
    included: True
    radius: 3389500
    soft_radius: []
jupiter:
    included: True
    radius: 69911000
    soft_radius: []
saturn:
    included: True
    radius: 58232000
    soft_radius: []
uranus:
    included: True
    radius: 25362000
    soft_radius: []
neptune:
    included: True
    radius: 24622000
    soft_radius: []
```

Chapter 5

Discussion

5.1 Strengths and Limitations of the Software

5.1.1 Strengths

The primary objective of the ASSAM project, to generate target observation visibility, is successfully satisfied by the current implementation of `assam`. As shown in Sections 4.2 and 4.3, the software can evaluate the visibility of targets on both a short- and long-term basis to aid with space telescope scheduling. Furthermore, the secondary objective, to automate the scheduling of the observations, is satisfied by the scheduling module being able to calculate an optimal solution to maximise the scientific return of the mission. Overall, the software is producing results as expected, with orbit propagation capturing key features of *JASMINE*'s orbit, and both short- and long-term visibility matching expectations.

One of the main strengths of `assam` is its computational performance, satisfying the third main objective of the ASSAM project which focuses on performance. A nominal *JASMINE* mission, analysed for a duration of one year with 350 targets, can be scheduled in less than half an hour, using a four core, eight thread Intel i7 4770 running at 3.7 GHz. It is expected that future work will further improve the performance of the software.

The flexibility planned for `assam` was achieved as the software is not mission-specific, and can be adapted to a variety of different scenarios and requirements. The main requirements of *JASMINE*, such as constraints with respect to the Sun, are not hard-coded into the software, therefore adapting `assam` to work with other missions is easily achieved by modifying the software's configuration files. This flexibility is improved by the modular architecture of the software which will allow for future developments to introduce new techniques and models, improving the capabilities of `assam` both generally and for specific use cases.

5.1.2 Limitations

The current implementation of visibility is binary: a target is considered either fully observable, or not at all. Nonetheless, there are scenarios where a certain amount of violation of constraint may be acceptable, or in the case of larger targets, it may be possible to observe a target with multiple overlapping exposures as is the strategy for *JASMINE* when observing the Galactic Centre [1]. Although this approach is conservative and safe, in the sense that it will not try to schedule any contacts which are less than ideal, it introduces an inefficiency which could be reduced to improve overall telescope utilisation.

The implementation of the scheduling module is one of the main limitations of the current version of `assam`. Although the dynamic linear programming method has very high computational performance, a trade-off exists with some features which could improve the generated schedule. For example, the current scheduling method considers the target contacts indivisible in the sense that, if a contact is selected, the observation must be conducted for the entirety of its duration. Nevertheless, a scenario could exist in which a higher overall scientific return could be achieved by observing several targets for less than their total contact durations.

One of the limitations of the scheduling module is that it does not consider the dynamics of the spacecraft with regards to pointing. Currently, the algorithm can "snap" between targets instantaneously, instead of considering the time it would take for the spacecraft to slew between targets and also to stabilise before conducting observations. Additionally, an operational concern when slewing space telescopes, and spacecraft more generally, with reaction or momentum wheels is momentum build-up [58]. The excess momentum must be periodically "dumped" using reaction thrusters to avoid saturation of the wheels, therefore minimising propellant usage is desirable to maximise mission lifetime [58]. This may affect scheduling strategy and may need to be considered for future implementations.

The scheduling module does not consider operational constraints which may affect observation periods. For example, there may be periods of downtime for housekeeping activities to ensure that the spacecraft is operating correctly, and operational procedures such as orbital manoeuvres. Furthermore, capacity restrictions of the on-board scientific data storage, and downlink periods may require *JASMINE* to temporarily halt science operations.

5.2 Recommendations for Future Work

During the development of `assam`, several features and improvements were identified that would extend the functionality and performance of the software. Nevertheless, these were considered out of scope during the initial development and are, therefore, presented in this section as recommendations for future work.

5.2.1 Refactoring

The main recommendation for future work, which would be required for several of the recommendations outlined below, would be to conduct a refactoring of the codebase. This would provide three main improvements: greater code simplicity and readability, greater scope for extending the software's functionality, and potentially improved computational performance by conducting optimisations during the refactoring process.

An important benefit of conducting a refactor will be the introduction of unit testing for the software. The codebase for `assam` currently contains many sections which could be broken down into multiple functions. Refactoring in this way would greatly simplify the process of generating unit tests, and using them would ensure that the software is functioning correctly.

An example of where the codebase could be simplified can be seen with the visibility calculations discussed in Section 3.4.3. The mission constraint calculation is effectively an extension of the physical constraint calculation, therefore a physical constraint could be expressed as a mission constraint by setting the inner angular radius, ρ_3, to zero. Combining these into a single function would reduce the complexity of the codebase, improve readability, and potentially allow for future extension to include other geometries. Consolidating functionality also prevents repeated sections of code in other parts of the codebase. For example, the visibility calculations required to generate sky views in the visualisation module uses repeated code, therefore two instances of the same functionality exist in the codebase.

5.2.2 Visibility Calculation Improvements

The present handling of visibility constraints could be reworked to provide even further flexibility to model different scenarios for different missions. The centre coordinates for constraints are currently fixed to solar bodies, however this could be extended to arbitrary fixed or moving points in the sky. Furthermore, mission constraint radii are currently static but could be brought in line with physical constraints which can vary with time.

The `astropy` affiliated package `regions` [59] was initially considered during the development of `assam` to handle intersection calculations between different geometries [3], other than the circle-circle intersections discussed in Section 3.4.3. Nevertheless, the early stage of development of this package means that many of the required features have yet to be implemented, therefore it has not been used [3]. It was for this reason that rectangular regions are approximated using their bounding circle in the current implementation of `assam`. Future work could include the introduction of further geometries to improve the flexibility of the software, and to remove the error introduced by the bounding circle approximation.

5.2.3 Scheduling Improvements

The limitations regarding scheduling discussed in Section 5.1.2 highlight that improvements can be made with regards to the scheduling algorithm. Future work could concentrate on improving both the technique used for scheduling, and the underlying models used to capture factors such as visibility and spacecraft dynamics. It has been shown that CSP-based methods combined with Genetic Algorithms (GAs) can be highly effective for scheduling space telescopes [2, 58] due to the ability to consider a large number of factors and metrics on a global scale in a way not possible with the current dynamic linear programming algorithm used by `assam`.

5.2.4 Performance Improvements

The use of hardware acceleration resulted in performance improvements when generating the sky views with the visualisation module. Nevertheless, the significant gains in computational performance that were suggested for certain workloads were not achieved. Hardware acceleration can be bottlenecked due to the transfer of data between the CPU and the GPU. For this reason, performance increases significantly for large arrays where the time lost due to data transfer between the CPU and GPU is offset by the significantly faster calculations on the GPU. The current implementation of the visualisation module calls the hardware accelerated separation function for each target and each constraint, therefore each execution is conducted on a relatively small array. Batching these calculations into fewer, or even a single call, could yield greater computational performance. GPU batching could also be considered for the visibility module which may yield large improvements in performance when conducting runs over long time periods with small time steps.

5.2.5 File Handling

One aspect of `assam` which should be considered for future work is the current implementation of file handling. Currently file paths are hard coded into the various functions which require Input/Output (I/O), such as when importing the target definition file, as relative paths from the main file. It may be worthwhile implementing a separate sub-package within `assam` to handle I/O to provide better functionality by allowing for filepaths to be specified, and to allow for the introduction of other common file types such as Comma-separated Values (CSV).

The output from `assam` primarily takes the form of plots generated by the visualisation module. Nevertheless, it would be beneficial to allow for data to be exported in file formats such as CSV. This would provide a robust method both to store results and to enable further post-processing as required by the user. Outputting to files at key milestones during execution would allow for partial backups of the software's state, safeguarding against any unexpected interruptions which could result in a loss of results.

Updating the file handling for GMAT should be considered a priority. The working files for GMAT, including both the modified script file and the output spacecraft's state, are currently stored in a directory within the propagator module's own directory. The primary effect of this is that multiple instances of the propagator module cannot execute at the same time as they will attempt to access and modify the same files within this directory. The GMAT interface should be updated to use temporary directories, both to enable parallel execution of multiple GMAT instances if required, but also to prevent the use of working files within the assam package's directory.

5.2.6 UI

A major lesson during the initial years of *HST* operations using Spike related to the process of receiving requests from Principal Investigators (PIs). PIs would propose observations using a text file and a simple tool for syntax checking, however "while the PI would see an error-free program, problems would arise [...] with [Spike]" [28]. One of the main advancements was to develop tools, including a GUI, to improve the proposal process to avoid "[t]he continual back-and-forth between PI and STScI staff [which] slowed the planning and scheduling process and could delay implementation of a program by weeks" [28]. The lessons learned from the initial years of using Spike highlight the importance of UI, particularly when facing users who have not developed the software and do not have, nor require, a deep understanding of its operation. This should be a consideration during future development of assam.

Chapter 6

Conclusions

The overall objective of the ASSAM project has been satisifed: the `assam` tool has been developed. It has been demonstrated that it is able to evaluate observations schedules automatically, both for the general case, and specifically for the *JASMINE* mission. This includes the three main objectives of the project: manual and automatic scheduling capability; and lightweight, high performance software.

The `assam` tool has been used in the context of the *JASMINE* mission to calculate both short- and long-term visibility of the candidate targets, confirming the expected results. Additionally, the long-term visibility of the priority target, the Galactic Centre, was calculated as a function of the Earth's mission constraint and the spacecraft's altitude, providing results to aid with the development of the telescope's baffle, a critical component for reducing the effects of stray light.

The strengths and limitations of the software have been discussed, with recommendations for future work presented to extend the functionality and performance of `assam`.

Bibliography

[1] N. Gouda, 'JASMINE,' *Scholarpedia*, vol. 6, no. 10, p. 12 021, 2011. [Online] Available at: http://www.scholarpedia.org/article/JASMINE [Accessed 8th March 2021].

[2] M. D. Johnston and G. E. Miller, 'SPIKE: Intelligent Scheduling of Hubble Space Telescope Observations,' *Intelligent Scheduling*, pp. 391–422, 1994.

[3] M. Hallgarten La Casta, *Automated Sky Survey Analysis Methods: Interim Report*, Internal Access Only, University College London, 2021.

[4] J. Kovalevsky, *Modern Astrometry*, 2nd ed. Springer-Verlag Berlin Heidelberg, 2002, ISBN: 9783642076190.

[5] A. Vallenari, 'The Future of Astrometry in Space,' *Frontiers in Astronomy and Space Sciences*, vol. 5, p. 11, 2018.

[6] J. Kovalevsky and P. K. Seidelmann, *Fundamentals of Astrometry*, 1st ed. Cambridge University Press, 2004, ISBN: 0521642167.

[7] E. Høg, '400 years of astrometry: from Tycho Brahe to Hipparcos,' *Experimental Astronomy*, vol. 25, no. 1, pp. 225–240, 2009.

[8] L. Spitzer Jr., 'Report to project rand: Astronomical advantages of an extra-terrestrial observatory,' *Astronomy Quarterly*, vol. 7, no. 3, pp. 131–142, 1990.

[9] NASA, 'NASA's First Stellar Observatory, OAO 2, Turns 50,' 11th December 2018. [Online] Available at: https://www.nasa.gov/feature/goddard/2018/nasa-s-first-stellar-observatory-oao-2-turns-50 [Accessed 22nd June 2021].

[10] NASA, 'About - Hubble History Timeline | Full Text,' 13th October 2020. [Online] Available at: https://www.nasa.gov/content/goddard/hubble-timeline-full-text [Accessed 7th July 2021].

[11] J. Foust, 'Hugging Hubble longer,' *The Space Review*, 15th June 2020. [Online] Available at: https://www.thespacereview.com/article/3965/1 [Accessed 22nd June 2021].

[12] NASA, 'Operations Underway to Restore Payload Computer on NASA's Hubble Space Telescope,' 18th June 2021. [Online] Available at: https://www.nasa.gov/feature/goddard/2021/operations-underway-to-restore-payload-computer-on-nasas-hubble-space-telescope [Accessed 22nd June 2021].

[13] G. F. Benedict *et al.*, 'Astrometry with Hubble Space Telescope Fine Guidance Sensors–A Review,' *Publications of the Astronomical Society of the Pacific*, vol. 129, no. 971, p. 012 001, 2016.

[14] F. van Leeuwen, 'The Hipparcos Mission,' *Space Science Reviews*, vol. 81, no. 3, pp. 201–409, 1997.

[15] D. Hobbs *et al.*, 'All-sky visible and near infrared space astrometry,' *Experimental Astronomy*, pp. 1–61, 2021.

[16] Gaia Collaboration *et al.*, 'The Gaia mission,' *A&A*, vol. 595, A1, 2016.

[17] ESA, 'ESA PR 44-2013: Liftoff for ESA's billion-star surveyor,' 19th December 2013. [Online] Available at: https://sci.esa.int/s/WnDJjbA [Accessed 11th June 2021].

[18] Gaia Collaboration *et al.*, 'Gaia Early Data Release 3 - Summary of the contents and survey properties,' *A&A*, vol. 649, A1, 2021.

[19] M. A. C. Perryman *et al.*, 'The Hipparcos Satellite Operations,' in *The Hipparcos and Tycho Catalogues*, vol. 2, ESA Publications Division, 1997, ISBN: 9290923997. [Online] Available at: https://www.cosmos.esa.int/documents/532822/552851/vol2_all.pdf/ae20969e-b6e9-47eb-8057-7b2cdb1111c4 [Accessed 13th June 2021].

[20] M. A. C. Perryman *et al.*, 'GAIA: Composition, formation and evolution of the Galaxy,' *A&A*, vol. 369, no. 1, pp. 339–363, 2001.

[21] S. Nishiyama *et al.*, 'The Interstellar Extinction Law toward the Galactic Center. II. V, J, H, and K_s Bands,' *The Astrophysical Journal*, vol. 680, no. 2, pp. 1174–1179, June 2008.

[22] D. Kawata *et al.*, 'JASMINE: Near-Infrared Astrometry and Time Series Photometry Science,' *Astronomical Society of Japan*, in preparation.

[23] T. Yamada, *Future Space Science Program of JAXA*, 12th December 2019. [Online] Available at: https://www.nao.ac.jp/for-researchers/naoj-symposium2019/presentations/09.NAOJ_sympo20191212_Yamada.pdf [Accessed 8th March 2021].

[24] JAXA, *Epsilon Launch Vehicle User's Manual*, rev. A, July 2018. [Online] Available at: https://global.jaxa.jp/projects/rockets/epsilon/pdf/EpsilonUsersManual_e.pdf [Accessed 2nd June 2021].

[25] JAXA, *Epsilon Launch Vehicle*. [Online] Available at: https://www.jaxa.jp/projects/pr/brochure/pdf/01/rocket07.pdf [Accessed 2nd June 2021].

[26] S. Utsunomiya, T. Kamiya and R. Shimizu, 'Development of CFRP mirrors for space telescopes,' in *Material Technologies and Applications to Optics, Structures, Components, and Sub-Systems*, J. L. Robichaud, M. Krödel and W. A. Goodman, Eds., International Society for Optics and Photonics, vol. 8837, SPIE, 2013, pp. 206–211.

[27] M. T. Goodrich and R. Tamassia, *Algorithm Design and Applications*. Wiley, 2015, ISBN: 9781118335918.

[28] D. S. Adler, D. K. Taylor and A. P. Patterson, 'Twelve years of planning and scheduling the Hubble Space Telescope: process improvements and the related observing efficiency gains,' in *Observatory Operations to Optimize Scientific Return III*, P. J. Quinn, Ed., International Society for Optics and Photonics, vol. 4844, SPIE, 2002, pp. 111–121.

[29] STScI. 'Spike Planning and Scheduling Software,' [Online] Available at: https://www.stsci.edu/scientific-community/software/spike [Accessed 10th July 2021].

[30] Python Software Foundation, *Python*, version 3.9.6, June 2021. [Online] Available at: https://www.python.org/ [Accessed 6th July 2021].

[31] Astropy Collaboration *et al.*, 'Astropy: A community Python package for astronomy,' *Astronomy & Astrophysics*, vol. 558, A33, October 2013.

[32] Astropy Collaboration *et al.*, 'The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package,' *The Astronomical Journal*, vol. 156, no. 3, p. 123, September 2018.

[33] T. Robitaille *et al.*, *astropy/astropy: v4.2.1*, version v4.2.1, April 2021. [Online] Available at: https://doi.org/10.5281/zenodo.4670729 [Accessed 24th June 2021].

[34] NASA, *GMAT R2020a*, version R2020a, July 2020. [Online] Available at: https://sourceforge.net/projects/gmat/files/GMAT/GMAT-R2020a/ [Accessed 29th June 2021].

[35] J. L. C. Rodríguez *et al.*, *poliastro/poliastro: poliastro 0.15.0 (Earth edition)*, version v0.15.0, May 2021. [Online] Available at: https://doi.org/10.5281/zenodo.4763538 [Accessed 24th June 2021].

[36] CS Group, *Orekit*, version 10.3.1, June 2021. [Online] Available at: https://github.com/CS-SI/Orekit [Accessed 2nd July 2021].

[37] G. H. Garrett *et al.*, *tudatpy*, version 0.5.23-rc, May 2021. [Online] Available at: https://github.com/tudat-team/tudatpy [Accessed 2nd July 2021].

[38] W. McKinney, 'Data Structures for Statistical Computing in Python,' in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 56–61.

[39] J. Reback *et al.*, *pandas-dev/pandas: Pandas 1.2.5*, version v1.2.5, June 2021. [Online] Available at: https://doi.org/10.5281/zenodo.5013202 [Accessed 24th June 2021].

[40] J. D. Hunter, 'Matplotlib: A 2D graphics environment,' *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[41] T. A. Caswell *et al.*, *matplotlib/matplotlib: REL: v3.4.2*, version v3.4.2, May 2021. [Online] Available at: https://doi.org/10.5281/zenodo.4743323 [Accessed 24th June 2021].

[42] M. L. Waskom, 'seaborn: statistical data visualization,' *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

[43] M. L. Waskom *et al.*, *mwaskom/seaborn: v0.11.1 (December 2020)*, version v0.11.1, December 2020. [Online] Available at: https://doi.org/10.5281/zenodo.4379347 [Accessed 24th June 2021].

[44] C. da Costa-Luis *et al.*, *tqdm: A fast, Extensible Progress Bar for Python and CLI*, version v4.61.1, June 2021. [Online] Available at: https://doi.org/10.5281/zenodo.4937831 [Accessed 24th June 2021].

[45] C. R. Harris *et al.*, 'Array programming with NumPy,' *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[46] R. Okuta *et al.*, 'CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations,' in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, 2017. [Online] Available at: http://learningsys.org/nips17/assets/papers/paper_16.pdf [Accessed 24th June 2021].

[47] P. A. Entschev, 'Single-GPU CuPy Speedups,' *Medium*, 23rd July 2019. [Online] Available at: https://medium.com/rapids-ai/single-gpu-cupy-speedups-ea99cbbb0cbb [Accessed 20th June 2021].

[48] Synopsys. ‘Compare repositories,’ [Online] Available at: https://www.openhub.net/repositories/compare [Accessed 6th July 2021].

[49] M. Hallgarten La Casta, *maxhlc/assam: Initial release of assam*, version v0.1.0, July 2021. [Online] Available at: https://doi.org/10.5281/zenodo.5085605 [Accessed 9th July 2021].

[50] The GMAT Development Team, *General Mission Analysis Tool (GMAT) User Guide*, NASA Goddard Space Flight Center, Greenbelt, Maryland, 2020.

[51] B. Rhodes, *jplephem 2.15*, version 2.15, September 2020. [Online] Available at: https://pypi.org/project/jplephem/ [Accessed 29th June 2021].

[52] D. A. Vallado, *Fundamentals of Astrodynamics and Applications*, 4th ed. Microcosm Press, 2013, ISBN: 9781881883180.

[53] The Astropy Developers, *Astropy Documentation*, 12th May 2021. [Online] Available at: https://docs.astropy.org/en/stable/index.html [Accessed 2nd July 2021].

[54] T. Vincenty, ‘Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations,’ *Survey Review*, vol. 23, no. 176, pp. 88–93, 1975.

[55] M. Hallgarten La Casta, *Automated Sky Survey Analysis Methods: Initial Report*, Internal Access Only, University College London, 2021.

[56] M. T. Goodrich and R. Tamassia, *Dynamic Programming: Telescope Scheduling*, 2015. [Online] Available at: https://www.ics.uci.edu/~goodrich/teach/cs260P/notes/TelescopeSchedule.pdf [Accessed 26th June 2021].

[57] T. Peters, *[Python-Dev] Sorting*, 20th July 2002. [Online] Available at: https://mail.python.org/pipermail/python-dev/2002-July/026837.html [Accessed 7th July 2021].

[58] M. E. Giuliano and M. D. Johnston, ‘Multi-Objective Evolutionary Algorithms for Scheduling the James Webb Space Telescope,’ in *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008)*, AAAI, 2008, pp. 107–115.

[59] L. Bradley *et al.*, *regions*, version v0.4, June 2019. [Online] Available at: https://github.com/astropy/regions/ [Accessed 5th July 2021].