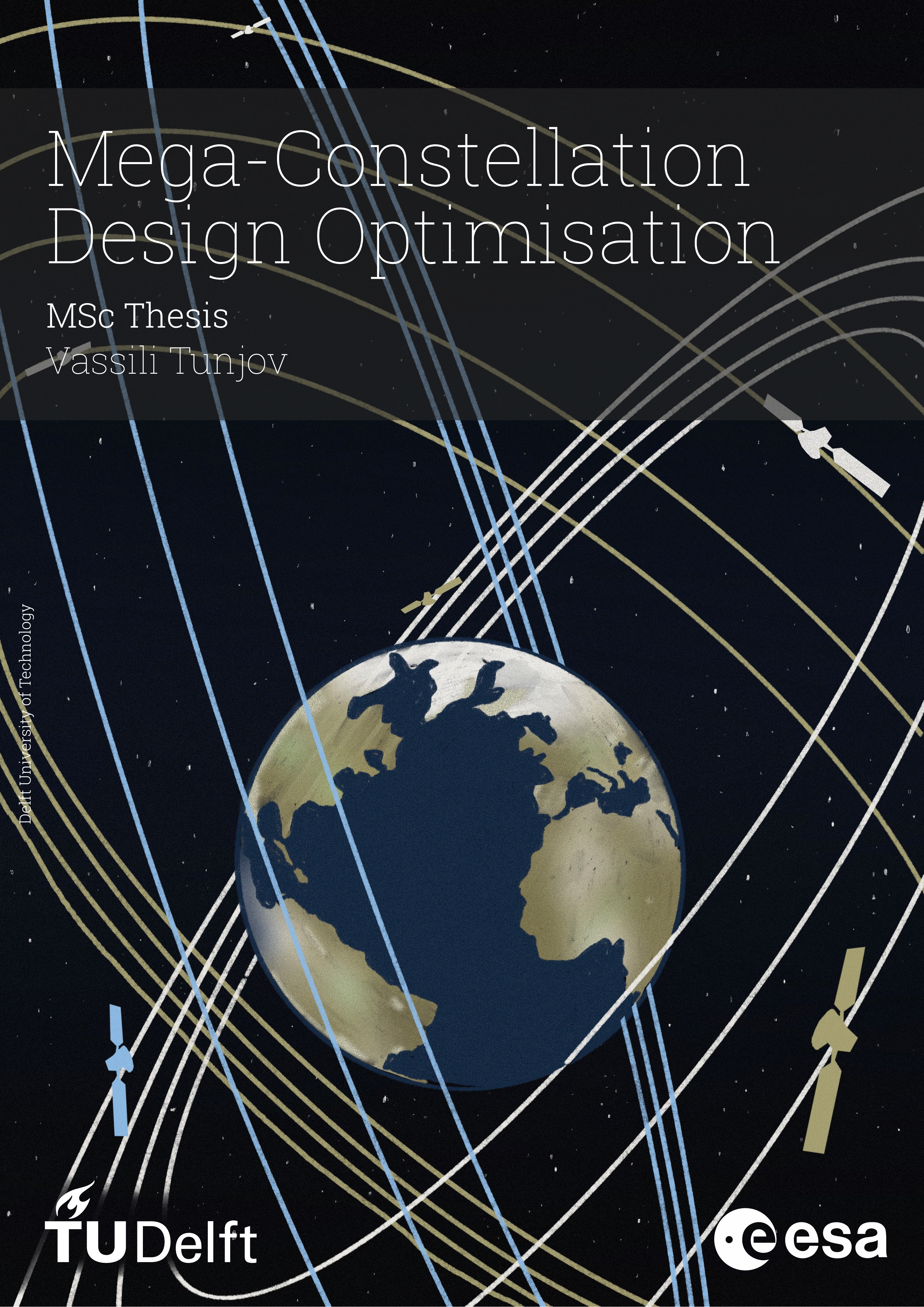
Mega-Constellation Design Optimisation

MSc Thesis

Vassili Tunjov



Delft University of Technology

TU Delft

esa

Mega-Constellation Design Optimisation

by

Vassili Tunjov

to obtain the degree of Master of Science in Aerospace Engineering
at the Delft University of Technology,
to be defended publicly on Friday April 5, 2024 at 13:30.

Student number:	4548485	
Project duration:	January 1, 2023 – April 5, 2024	
Thesis supervisors:	Jian Guo	TU Delft
	Michael Khan	ESA
	Arnaud Boutonnet	ESA

Cover image credits: Vera Gabets

Preface

This Master's thesis is a collaborative effort between TU Delft and the Mission Analysis Section at ESOC/ESA. It aims to develop methodologies and tools for mega-constellation design, supporting future studies and mission analyses.

This work marks the culmination of my seven-year academic journey at TU Delft.

I am deeply grateful to Arnaud and Florian for their belief in me. Their initial internship offer at the Mission Analysis Section proved invaluable. I extend my thanks to everyone I met there for making it a formative experience. Special thanks to Michael, whose guidance extended beyond the thesis, encompassing valuable career advice, his insights into other aspects of the field, and his entertaining (though unmentionable here) jokes that helped lighten the mood.

My gratitude also extends to Amedeo and Lorenzo for their support during my ESOC internship.

Professor Jian, I am thankful for accepting me as your thesis supervisee.

Words cannot express my appreciation for the unwavering support of my dear friends. A special thank you to Kate, a loyal and supportive friend who remained by my side despite all difficulties, you are truly amazing. Ilya, my friend since arriving in Delft, this would have been impossible without you. Gleb, your invaluable support - especially during the pandemic's challenges - helped me persevere. Makars and Arturs, a shout-out to you for the amazing House of Culture and everyone who was part of that gang! Finally, thank you, Veronika, for coming to my life and supporting during the thesis' final, the most critical stages.

Kate, a loyal companion who stuck by me despite all difficulties, you are truly amazing. Ilya, my friend since arriving in Delft, this would have been impossible without you. Gleb, your invaluable support during the pandemic's challenges helped me persevere.

I also want to thank my friends from Estonia (Oks, Evert, Alina K., Alina S., Maksim, Matvei, Dasha, everyone from VitaTiim, and many more) who consistently believed in me, supported, and cheered me up whenever I felt down.

Thanks everyone from Lindy Hop society who helped me ease the tough days of thesis work.

Looking back, I am amazed by how many people supported me throughout this challenging stage of my life. While I cannot possibly acknowledge each and every one, I want you all to know how grateful I am for your encouragement and belief in me during this journey.

Lastly, to my family: my sisters Maria, Marina, and Valentina, my brother Georgi, my dad, and especially my mom, whose enduring support, especially at the beginning before I even moved to the Netherlands, fuelled my determination. Thank you!

Vassili Tunjov
Delft, May 2024

Summary

Given the rapid growth in popularity of mega-constellations for telecommunication purposes, this thesis aims to identify methods for designing orbital layouts of such constellations. Traditional optimisation methods for regularly sized constellations do not scale due to the number of satellites involved. Therefore, this thesis investigates the design of mega-constellations and how to achieve an optimal layout within a reasonable timeframe.

Initially, the figures of merit most commonly used in mega-constellation design were identified. The most important figure of merit used in all types of missions, except for Earth observation, is visibility, i.e., the number of satellites visible from a point on the ground. For Earth observation missions, the revisit time is more relevant than visibility. Thus, the thesis focused on mission objectives where visibility is the main figure of merit, such as Satcom, Satnav, IoT, etc. There is also more literature and data available on such constellations, including Starlink, OneWeb, Project Kuiper, and others. Consequently, two closely related figures of merit were selected for this study: minimum visibility and mean visibility. The former ensures an N-fold uninterrupted coverage, while the latter provides a general metric of how many satellites are visible over time. Also, the main focus was on Walker constellations as this is the most common geometry used for mega-constellations.

The visibility computation begins with constellation propagation. Due to the large number of satellites in a mega-constellation, existing tools used at ESOC, such as Godot, are insufficient. Therefore, a self-written tool was developed, named `mcdo` (*Mega-Constellation Design Optimisation*), which utilises the basic functionality of Godot and takes advantage of NumPy by applying vectorisation to notoriously slow Python. This approach resulted in a decrease in computational time by a factor of 100x with respect to `godot.cosmos.BallisticPropagator`.

Moreover, the visibility computation was accelerated with the utilisation of graphic processing units (GPU) and simplifications such as North-South symmetry, longitude averaging, or estimating visibility for a single time instance. Depending on the methods and simulation setups used, a reduction of 400x-120,000x in computational time was achieved for visibility computation.

Additionally, the parametric analysis of large Walker constellations yielded some valuable discoveries. For instance, the number of planes P and phasing parameter F do not influence the mean visibility but may have a significant effect on minimum visibility. This means that P and F can be omitted when designing for the mean visibility. It was also revealed that the mean visibility curve scales linearly with the number of satellites N, allowing to avoid propagation of very large constellations, and to scale up the mean visibility curve from smaller constellations instead. Parametric analysis of other parameters provided a general insight into their effect on visibility.

The improved computational efficiency of visibility computation for large Walker constellations enabled the application of multi-shell constellation design. A case study was set up with a requirement of uninterrupted coverage of 50 satellites over European latitudes (35-70 deg), assuming that all shells were at the same altitude of 700 km with a goal to minimise N. The analysis revealed that for two- and three-shell layouts, the minimum N was attained when higher-inclination shells had more satellites than lower-inclination ones. Moreover, methods for design acceleration were discussed.

The *building blocks* method was proposed for designing mega-constellations with more than three shells. This method starts from placing shells at high inclinations and then gradually lowers the inclination of the subsequent shells. It takes advantage of visibility properties of Walker constellations: the higher-inclination shells can cover both high and low latitudes, while the lower-inclination shells can only cover low latitudes. This method reduces the computational time drastically: by a factor of 150x for three-shell layouts and even more for larger numbers of shells.

The presented research offered valuable insights into initial phases of design of the orbital layout of mega-constellations. The `mcdo` tool and the obtained results were already applied for internal projects at ESA, demonstrating their relevance and usefulness.

Contents

1 Introduction

Over the past decade, mega-constellations have emerged as a result of the rapid evolution of technology and changing communications needs. These constellations consist of hundreds or even thousands of small satellites in low Earth orbit (LEO). Compared to the traditional geostationary orbits (GEO) used by most communications satellites and medium Earth orbits (MEO) used by all satellite-based navigation systems, the LEO orbits offer a number of advantages. These advantages include lower latency and improved signal strength which results in higher data rates, enhanced positioning and increased resilience against interference. Moreover, they are not only suitable for communications applications, but also for other purposes that require high-resolution imagery and high signal-to-noise ratio (SNR) for radio frequency transmissions. These include remote sensing, navigation, the Internet of Things, and national security applications. The latter includes signal intelligence (SigInt), optical and radar reconnaissance, or disaster monitoring, planning and relief [1]. Furthermore, the payload size of a spacecraft in LEO can be significantly reduced compared to those in MEO or GEO without compromising performance because of reduced link budget requirements.

The advantages of large LEO constellations have attracted the attention of both commercial companies and governments. While businesses are intrigued by the commercial opportunities offered by large LEO constellations, governments are seeking to use them for strategic purposes such as sovereign communications and navigation. This growing interest has led to an increasing number of mega-constellations being deployed in LEO from only 3 commercial small satellites (with mass <500 kg) in 2012 to 2,402 smallsats in 2022 [2]. Although 76% of these launched satellites are operated by Starlink and OneWeb, the number of smallsat operators grows as well despite a sharp decrease in 2020 due to Covid-19 pandemic [2].

The deployment of large-scale constellations presents unique engineering and technical challenges. The design of these intricate systems covers various aspects, including orbital layout design, spacecraft design, system deployment, constellation operation and control, satellite disposal, and regulatory requirements. To ensure scalability, resilience, and sustainability, while maintaining cost-effectiveness, the design of mega-constellations necessitates optimisation. This involves factors such as the number of satellites, orbital parameters, payload size, and resource allocation in space and on the ground, among others.

It is essential to recognise the inherent interdependence of the various aspects of mega-constellation design. It is an iterative process where the optimal solution for one element may not translate directly to the system as a whole.

The orbital layout design of a mega-constellation is a critical factor that directly influences its performance and sustainability. The layout dictates the number and distribution of satellites, as well as the altitude, inclination, and phasing of their orbits. The design process must consider various factors, including the desired coverage area, data transmission requirements, and the operational lifespan of the satellites. Consequently, determining the optimal orbital layout is crucial to ensure that the mega-constellation delivers efficient and sustainable services that meet user requirements, while minimising the number of satellites to be deployed and the environmental impact of the constellation.

This optimised layout serves as a foundational element upon which the design of other subsystems can be further refined, ultimately leading to a mega-constellation that effectively balances performance, sustainability, and cost-effectiveness.

The task of determining an optimal layout for a mega-constellation is complicated by the extremely high number of satellites involved. While there is no universally accepted definition of a mega-constellation, it is generally agreed that a constellation with more than 100 satellites qualifies as such. However, scalability can pose a problem; a method that works for a few hundred satellites may become computationally burdensome when the number increases to tens of thousands of satellites. This necessitates the development of scalable methods that can be applied to the design of constellations with thousands or even tens of thousands of satellites.

A literature survey on constellation design has been conducted. Analytical methods for continuous global, regional, and zonal coverage employ the Streets-of-Coverage (SoC) concept studied by Luders [3], Ullock [4], Beste [5], and Rider [6]. Walker extensively studied Star and Delta layouts [7–9] which are now the most commonly used layouts for mega-constellations. Ballard introduced Rosette constellations [10] that are similar to Walker Delta. Furthermore, Rider and Adams [11], and Lang [12] studied various layouts single- and multifold coverage. All these methods consider circular orbits in LEO. However, methods for geosynchronous constellations [13] and constellations in elliptical orbits [14, 15] also exist.

For regional or complex coverage requirements, some researchers explore mathematical approaches using set and group theory [16–20]. Mortari introduced Flower constellations [21, 22], which were later generalised to 2D [23], 3D [24], and 4D Flower constellations [25].

With the development of computational capabilities, semi-analytical methods became more popular [26, 27]. Later, numerical optimisation algorithms were applied to constellation design for various applications such as GNSS [28], Earth observation [29–36], military [37], space situational awareness [38], Internet-of-Things (IoT) [39], and other applications requiring global or regional coverage [40–45].

The aforementioned research papers involve "conventional" numbers of satellites, which usually do not exceed a few tens. However, there is a noticeable lack of published studies focusing on the design of orbital layouts for mega-constellations with only a few articles publicly available. Most published research papers focus on utilising satellites in LEO for GNSS aplications [46–49] and involve no more than 500 satellites. There is also one paper focusing on IoT [50] involving up to 1000 satellites. Finally, one paper considers a constellation for global connectivity with more than 5000 satellites [51].

Private companies and governmental institutions often keep their design and optimisation process confidential due to its potential value as a trade secret or even a military secret. Commercial companies, in particular, may be influenced by their respective governments' desire to utilise space systems for military applications, as exemplified by the recent use of Starlink in Ukraine[1]. However, final layouts can be found on the web as they are filed to the Federal Communications Commission (FCC) and International Telecommunication Union (ITU), but motivation for certain design choices is typically not specified there.

The existing academic research and publicly available data do not address aspects that are of interest from a mission analysis perspective, such as how to design a mega-constellation consisting of thousands of satellites given specific mission objectives and requirements. Therefore, the **main research question** is formulated as follows:

How does selection of the mission objective affect the optimal design of mega-constellations?

To help answer it, the main question is divided in 3 subquestions where each subquestion will be answered separately in consecutive order:

Subquestion 1: *What are the figures of merit for evaluating mega-constellation design for different mission objectives?*

[1]https://www.reuters.com/business/aerospace-defense/pentagon-buys-starlink-ukraine-statement-2023-06-01/, accessed 14 May 2024

The aim of answering this subquestion is to identify the mission objectives that justify the deployment of mega-constellations and the corresponding figures of merit (FoM) that measure the performance of a constellation layout. A FoM is a quantitative or qualitative metric that can be used to compare different layouts and assess their suitability for a given mission objective. Since different mission objectives may have different FoMs, a comprehensive list of FoM is compiled and mapped to the relevant mission objectives.

Subquestion 2: *How can the computational time of the optimisation problem be reduced?*

This subquestion explores the methods that can be applied to reduce the computational time of the optimisation problem. Due to the large number of satellites and the multiple parameters for the multi-shell layout, the optimisation problem is expected to be computationally expensive and time-consuming. Therefore, various techniques must be examined to facilitate the design process and make it more efficient and scalable.

Subquestion 3: *How can the defined figures of merit be optimised for a multi-shell constellation layout?*

This subquestion investigates the behaviour of the defined figures of merit within the orbital layout design of mega-constellations. In particular, the focus is on the multi-shell constellation layouts, which consist of multiple orbits with different altitudes and inclinations. The advantages and disadvantages of the multi-shell layout compared to the single-shell layout have to be analysed with respect to the FoM.

By answering these questions, scalable and reasonable (in terms of computational time) methods will be found which would allow to get a better insight into mega-constellation design and create tools that would help design a mega-constellation.

The research methodology begins with a review of the definition, existing and planned systems, and published methods for mega-constellation orbital layout design. This includes an examination of the design of other subsystems and the figures of merit (FoM) that are used to quantify mega-constellation performance, as detailed in Chapter 2.

Following this, time-efficient methods of propagating and converting the orbits of satellites from Keplerian to Cartesian system are developed and verified, as presented in Chapter 3.

Next, methods that accelerate visibility computation are defined. This encompasses the development, implementation, and verification of numerical visibility estimation methods by means of vectorisation and massive parallelisation. Various types of simplifications are also introduced for visibility computation acceleration, along with a parametric analysis providing a comprehensive insight into the visibility of a mega-constellation. This is discussed in Chapter 4.

The methodology then proceeds to analyse the effect of multiple shells in a mega-constellation on visibility by applying it to a defined test mission scenario. Additionally, methods of accelerating the optimisation process of multi-shell design are defined and analysed, including a novel method. The new method is applied to improve constellation design described in [51]. This is explained in Chapter 5.

Finally, the conclusions drawn from the research and recommendations for further research are presented in Chapter 6.

2
Mega-Constellation Overview[1]

Before delving into the concept of a mega-constellation, it is essential to define what a constellation is. This report adopts the definition provided by Wood [52]:

"*A number of similar satellites, of a similar type and function, designed to be in similar, complementary, orbits for a shared purpose, under shared control.*"

However, there is no universally accepted definition of a mega-constellation. Various sources use different criteria and terminology to describe such systems. For instance:

Airbus[2]: *Mega-constellations are systems utilising hundreds to tens of thousands of satellites in Low Earth Orbit (LEO) to deliver low latency broadband data services anywhere on the planet.*

Arthur K. Lacombe [53]: *A constellation that is composed of several hundreds and thousands of satellites orbiting the Earth.*

These definitions lack precision and consistency. Airbus appears to restrict mega-constellations to communication services only, excluding other applications such as navigation, earth observation, or ISR (intelligence, surveillance, reconnaissance). Additionally, neither Airbus nor Lacombe provide a specific number of satellites that define a mega-constellation. This ambiguity extends to other terms used interchangeably, such as *non-geostationary orbit (NGSO) constellation*, *large-scale constellation*, *large LEO constellation*, *LEO communication (navigation) constellation*, or *mega satellite network*. None of these terms imply a minimum number of spacecraft in a mega-constellation either.

Therefore, the list below will briefly discuss four potential ways to define a mega-constellation based on the number of satellites it is comprised of.

1. **Subjective Perception (>100 satellites)**
   This definition aligns with that from ESO but is highly subjective and can vary significantly from one person to another.
2. **Exceeding Literature Standards (>200 satellites)**
   Literature on constellation design presented in Chapter 1 only uses the term "*constellation*". Thus, a mega-constellation could be defined as a constellation with a number of satellites exceeding that mentioned in the literature.
3. **Beyond Software Capabilities (>1000 satellites)**
   From a mission analysis perspective, a mega-constellation could be defined as a constellation with a number of satellites that existing software tools struggle to handle efficiently. The number of 1000 satellites is based on values for STK[3] and FreeFlyer[4] with a large uncertainty margin.

[1]This chapter builds upon the literature study report with additional information and modifications tailored to this specific report.
[2]https://securecommunications.airbus.com/en/meet-the-experts/mega-constellations-in-space-revolutionising-satellite-industry, accessed 14 May 2024
[3]https://www.ansys.com/products/missions/ansys-stk, accessed 14 May 2024
[4]https://ai-solutions.com/freeflyer-astrodynamic-software/, accessed 14 May 2024

4. **Extremely Large Constellations (>10000 satellites)**
   This definition considers a number of satellites that is one order of magnitude higher than in the previous definition.

For the purposes of this project, a mega-constellation will be defined as a LEO satellite constellation with more than 1000 satellites. However, it should be noted that the methods developed during the project will also be applicable to smaller constellations, although these methods may not be the most accurate solutions. This is primarily due to the fact that the simplifications and assumptions incorporated may not provide accurate enough results for constellations of a smaller size.

This chapter begins by describing typical constellation geometries for mega-constellations in Section 2.1. Furthermore, it provides an overview of both existing and planned systems that incorporate mega-constellations for a variety of applications, as detailed in Section 2.2. Section 2.3 presents a survey of published academic research on mega-constellations emphasising on methods relevant for this thesis. Finally, Section 2.4 identifies and selects the figures of merit based on the literature review and publicly available data on existing systems.

2.1. Typical Geometries

A mega-constellation typically comprises one or more layers, often referred to as ***shells***. While there is no rigid definition of a shell, it is generally characterised as a group of satellites within the same system, orbiting at the same altitude and inclination in a circular orbit. Some entities further distinguish shells by the frequency band (e.g. Project Kuiper [54]), implying that a shell consists of satellites at the same altitude and inclination, operating within the same frequency band. However, for the purpose of this thesis, a shell will be defined solely by its **altitude** and **inclination**.

Two prevalent types of geometries for mega-constellation shells are the Walker Delta and Walker Star (Streets-of-Coverage), both studied and described by John Walker [7–9]. These geometries are depicted in Figure 2.1 and will be discussed in this section. There exist other less common constellation configurations, such as the one used by Spire [55], which do not have specific names. However, due to their infrequent applications, these configurations fall outside the scope of this project.

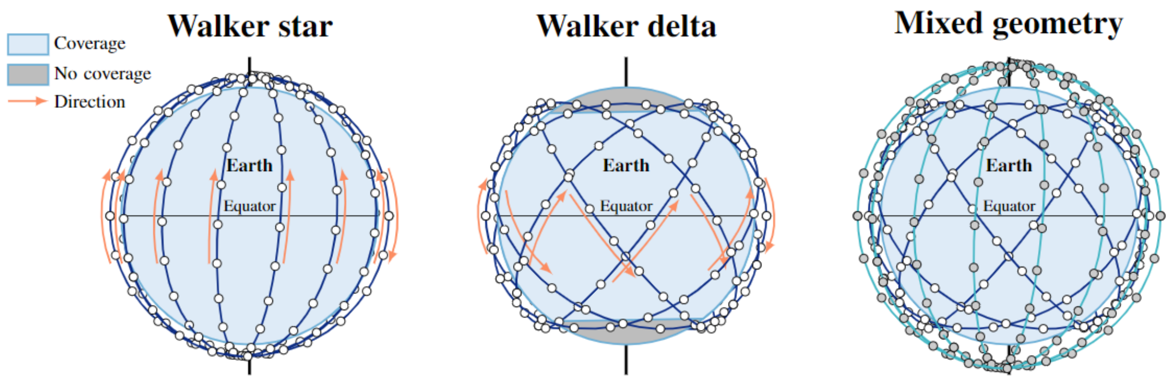


Figure 2.1: Walker Delta and Walker Star geometries typically used as mega-constellation shells [56].

2.1.1. Walker Delta

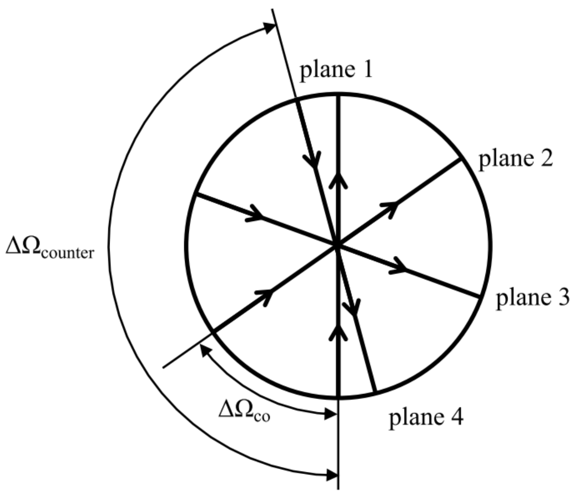
The Walker-Delta pattern is a symmetric constellation of T satellites with P planes each containing S evenly distributed satellites with intervals of $360^{\circ}/S$. Ascending nodes of each plane are uniformly distributed along the equator such that the intervals between the nodes are $360^{\circ}/P$. The three parameters are defined such that $T = P \cdot S$.

Phasing of satellites in adjacent planes is computed by multiplying F with $360^{\circ}/T$. For Delta patterns, F is an integer on interval $[0, P-1]$. For example, in a constellation $60^\circ : 30/5/2$, $F = 2$ meaning that phase difference between adjacent planes is $2 \cdot (360^\circ/30) = 24^\circ$.

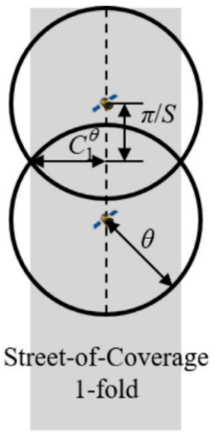
The advantage of Walker-Delta constellation is that they have full symmetry in longitude which allows to easily adjust latitudinal coverage by changing inclination of the constellation. This makes Delta patterns a suitable geometry for large non-polar constellations.

2.1.2. Walker Star (Streets-of-Coverage)

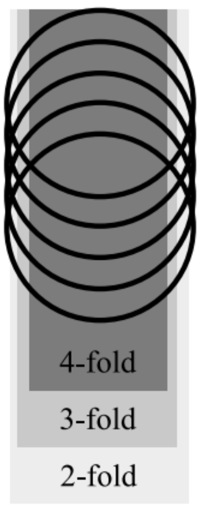
The Walker Star configuration comprises N satellites, distributed across P nearly polar orbit planes, each containing S satellites. This arrangement ensures continuous global coverage and is also known as Streets-of-Coverage (SoC). Figure 2.2a provides a top view (from the North Pole) of the SoC constellation. In this setup, half of the satellites move northward while the other half move southward at any given time. The ascending nodes of satellites are contained in a region of only 180 degrees as opposed to 360 degrees for Delta.



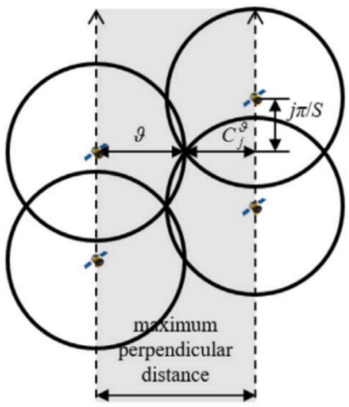
(a) North Pole view of SoC.



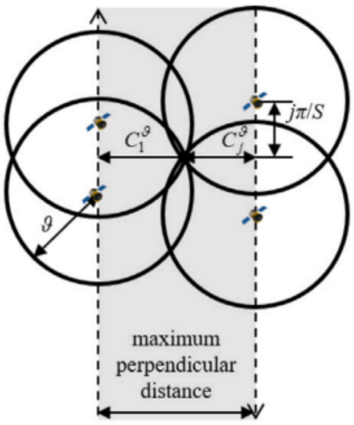
(b) One-fold SoC.



(c) Multi-fold SoC.



(d) Co-rotating planes interface.



(e) Counter-rotating planes interface.

Figure 2.2: General concepts of Street-of-Coverage (SoC) constellation [57].

The half-width C_j of a Street of Coverage for a j-fold coverage, as illustrated in Figure 2.2b and Figure 2.2c, can be computed using the equation below [17]:

$$\cos C_j = \frac{\cos\theta}{j\pi/S} \tag{2.1}$$

The maximum distances between co-rotating planes (planes moving in the same direction, see Figure 2.2d) $\Delta\Omega_{\text{co}}$ and counter-rotating planes (planes moving in the opposite direction on seams, see

Figure 2.2e) $\Delta\Omega_{\text{counter}}$ are given by these equations [17]:

$$\Delta\Omega_{\text{co}} = 2\,\text{arcsin}\left(\sin\frac{\theta + C_j^\theta}{2} / \sin i\right) \tag{2.2}$$

$$\Delta\Omega_{\text{counter}} = 2\,\text{arcsin}\left(\sin\frac{\pi - C_1^\theta - C_j^\theta}{2} / \sin i\right) \tag{2.3}$$

Assuming that inclination i is near-polar, i.e. $85° \leq i \leq 95°$, it can be approximated that $\sin i \approx 1$, which simplifies the equations to

$$\Delta\Omega_{\text{co}} = \theta + C_j^\theta \tag{2.4}$$

$$\Delta\Omega_{\text{counter}} = \pi - C_1^\theta - C_j^\theta \tag{2.5}$$

A (near-)polar SoC constellation must satisfy the following condition as demonstrated by [57]:

$$jP(P-1) \leq N \tag{2.6}$$

If θ is not known a priori, a minimum half-width angle ϑ can be computed with the following equation [57]:

$$(P-1)\,\text{arcsin}\left(\sin\frac{\vartheta + C_j^\vartheta}{2} / \sin i\right) = \text{arcsin}\left(\sin\frac{\pi - C_1^\vartheta - C_j^\vartheta}{2} / \sin i\right) \tag{2.7}$$

Equation (2.7) can be solved numerically using a numerical solver. This will provide the minimum half-width angle required for continuous global coverage. It should be noted that an iterative process may be required if P and S are not given as one will need to iterate through those to find the optimal values. More commonly, θ is specified based on the altitude h and the minimum elevation angle ε_{min}, as determined by the project team for the mission requiring a mega-constellation. Consequently, the task of a mission analyst would be to find the optimal values for P and S, with an aim to minimise N.

2.2. Existing systems

Numerous constellations are currently undergoing deployment or have plans for deployment in the near future. These constellations provide a valuable resource for researchers, as they offer publicly accessible data and can be utilised as benchmarks for verifying the accuracy of newly developed algorithms.

It is important to note that this section provides an overview of some of the mega-constellations projects available. However, it is not an exhaustive list, mainly for two reasons. Firstly, due to the lack of information available in English and in the public domain, it is not feasible to include all mega-constellations in this section. Secondly, to maintain a reasonable size, this section has focused on the most notable constellation designs. The constellations described below showcase diverse applications and orbital layouts, offering an informative glimpse into the different designs and their unique features.

The dynamic nature of satellite constellation design presents a challenge for reporting accurate and up-to-date information. In this work, data on existing and planned systems primarily stemmed from two sources: the Federal Communications Commission (FCC) and the International Telecommunication Union (ITU).

However, discrepancies may exist between data from these sources due to several factors.

- **FCC filings:** The complex and bureaucratic nature of FCC filings can make it difficult to navigate and identify the most recent information.
- **ITU filings:** While generally more accessible, searching by company name in ITU filings is not possible. Instead, reliance on call signs, which may not always be readily available, restricts search options.

This section attempts to provide the most up-to-date information on mega-constellations **as of January 2024**. However, it should be noted that due to the aforementioned reasons, the information may be outdated or differ from other sources.

2.2.1. Iridium

Iridium was one of the first operators of a relatively large constellation in LEO, before the emergence of mega-constellations. Although its size is smaller than what is commonly defined as a “mega-constellation”, it was the largest LEO constellation at the time and remained so until the 2010s, when the demand for higher data rates and lower latency increased. Iridium’s orbital layout does not seem to have any direct influence on the design of current or planned systems, but it demonstrated the capabilities of Satcom constellations in LEO and contributed to the rise of mega-constellations.

Initially, it was planned to have 77 satellites (hence, the name corresponding to number 77 in the periodic table of the elements) [58]. However, later it was found that 66 satellites would be sufficient to achieve the mission objective. The current layout or the Iridium constellation is shown in Table 2.1 [59]. In 2007, Iridium announced the development of a new generation of satellites called Iridium NEXT. These satellites were designed to replace the first generation of Iridium satellites [60]. Despite the upgrade, the orbital layout of the constellation remained unchanged, with the new satellites being placed in the same orbits as their first-generation counterparts [60].

Table 2.1: Iridium / Iridium NEXT. Total 66 satellites [59].

Altitude [km]	Inclination [deg]	Planes	Satellites per plane	Total # of satellites
780	86.4	6	11	66

2.2.2. Starlink

Starlink is a satellite internet constellation developed by SpaceX that aims to provide high-speed internet access to remote and underserved areas around the world. Starlink is the largest constellation in orbit with 5,806 satellites launched as of January 2024 [61]. The initial plan for Starlink was to have two generations of satellites, Gen1 and Gen2, each with their own distinct orbital layout designs [62–64]. However, the total number of satellites for the fully deployed system has changed over time.

Given the several years it takes to deploy the constellation, the spacecraft design continuously improves. As a result, different generations of satellites may differ in design, capabilities, and technical specifications. As of January 2024, it is planned to be around 30,000 satellites, down from the initial plan of approximately 42,000 Gen1 and Gen2 satellites [65], although this number may vary as SpaceX adjusts its constellation plans. Also, based on the ITU filing from 7 December 2023 [65], the latest filing as of January 2024, the constellation will no longer distinguish between Gen1 and Gen2, but will consist of a single design with a total of 29,988 satellites.

The orbital arrangement of the constellation is depicted in Table 2.2. The ultimate goal is to construct a constellation that contains satellites in both Very Low Earth Orbit (VLEO) and Low Earth Orbits (LEO). The constellation’s design includes shells that cover mid-latitudes with inclinations of 33-53 degrees, and a shell of satellites in Sun-Synchronous Orbits (SSO), operating across Ku- and Ka-bands [64]. Notably, in addition to these shells, there are also shells with inclinations larger than SSO, which raises intriguing questions regarding SpaceX’s motivations for placing satellites in these orbits.

Table 2.2: Starlink. Total 29,988 satellites [65].

Altitude [km]	Inclination [deg]	Planes	Satellites per plane	Total # of satellites
340	53.0	48	110	5,280
345	46.0	48	110	5,280
350	38.0	48	110	5,280
360	96.9	30	120	3,600
525	53.0	28	120	3,360
530	43.0	28	120	3,360

535	33.0	28	120	3,360
604	148.0	12	12	144
614	115.7	18	18	324

2.2.3. OneWeb

As of January 2024, OneWeb is the second-largest communication constellation, operating in a manner similar to Starlink, with plans to deploy their constellation in two distinct phases. Below is a description of these phases.

Phase 1

Phase 1 of OneWeb constellation was completed in April 2023[5]. To achieve full global coverage, 588 satellites have been deployed while the other 60 have been put in parking obit as spares, according to OneWeb [66]. Satellites are of smallsat class with a mass of 150 kg [67].

Table 2.3: OneWeb Phase 1. Total 588 satellites + 60 spare satellites [66].

Altitude [km]	Inclination [deg]	Planes	Satellites per plane	Total # of satellites
1200	87.9	12	49	588

Phase 2

Phase 2 constellation will be much larger compared to Phase 1. In addition to 648 deployed satellites, some 6,372 satellites will be deployed in LEO at altitude of 1200 km [68]. Including Phase 1 satellites, the constellation will consist of 3 shells with the same altitude but different inclinations as shown in Table 2.4.

Table 2.4: OneWeb Phase 2. Total 6,372 satellites [68].

Altitude [km]	Inclination [deg]	Planes	Satellites per plane	Total # of satellites
1200	87.9	36	49	1764
1200	55.0	32	72	2304
1200	40.0	32	72	2304

2.2.4. Project Kuiper

Project Kuiper is a communications constellation planned by Amazon. It will consist of 3,230 satellites operating in Ku- and V-band. The constellation is planned to consist of multiple shells three of which will be regular Walker-Delta shells while the other two will consist of a large number of planes as shown in Table 2.5.

Table 2.5: Project Kuiper. Total 3,230 satellites [54].

Altitude [km]	Inclination [deg]	Planes	Satellites per plane	Total # of satellites
590	33.0	782	1	782
610	42.0	1,292	1	1,292
630	51.9	1,156	1	1,156

[5] https://oneweb.net/resources/completing-constellation, accessed 14 May 2024

2.2.5. Spire

Spire is an Earth observation constellation that uses 3U CubeSats called Lemur-2 and Lemur-3. These are improved versions of the first generation satellites Lemur. They are more relevant for our analysis because the constellation has a larger number of satellites than the first generation [55].

The constellation has 56 orbital planes with 32 satellites evenly distributed in each plane. A notable feature of this constellation is that it does not follow the Walker shells pattern, which is uncommon for such a large constellation. Moreover, none of the 32 orbital planes have the same inclination and altitude simultaneously. The inclinations of the planes vary from 0° to 98° (SSO), and the altitudes range from 300 km to 720 km. Additionally, 5 planes have eccentric orbits, which is also unusual for a mega-constellation.

The orbital layout is not included in this report because it is too large to be displayed here. The interested reader can refer to the ITU filing for this constellation [55].

2.2.6. Space Systems

Space Systems is a proposed constellation for IoT applications by SpinLaunch [69]. It will operate in Ku-, Ka-, and V-bands. The constellation will have 1190 satellites that will not follow the Walker patterns, but instead they will "*use exactly the same orbit with different right ascensions so as to create the appearance of a fixed line of satellites in the sky*" [69].

Table 2.6: Space Systems. Total 1,190 satellites [69]

Altitude [km]	Inclination [deg]	Planes	Satellites per plane	Total # of satellites
830	55.0	1190	1	1,190

2.2.7. Kepler

Kepler Communications is a communications constellation that has a smaller size than comparable communications constellations, only 140 satellites [70]. It operates in Ku-band (G2S and S2G), and it has a distinctive feature of using optical inter-satellite links[6].

Table 2.7: Kepler Communications. Total 140 satellites [70].

Altitude [km]	Inclination [deg]	Planes	Satellites per plane	Total # of satellites
600	90.0	7	20	140

Kepler Communications also plans to deploy the second constellation consisting of 360 satellites [71].

Table 2.8: Kepler Communications. Total 360 satellites [71].

Altitude [km]	Inclination [deg]	Planes	Satellites per plane	Total # of satellites
600 $\pm$ 50	89.5	12	30	140

2.2.8. E-Space

E-Space is an ambitious project aimed at launching a mega-constellation of over 300,000 CubeSats, each weighing around 10 kg, to enable the application of Artificial Intelligence of Things[7]. What sets this constellation apart is its unique design, which includes a collision capture mechanism that allows the satellite to capture any object it collides with and then sacrificially de-orbit[8]. The project is the

[6] https://kepler.space/network/, accessed 14 May 2024
[7] https://www.e-space.com/pages/about, accessed 14 May 2024
[8] https://www.e-space.com/pages/sustainability, accessed 14 May 2024

brainchild of Alan Walker, the founder of OneWeb and O3b Networks, and promises to revolutionise the satellite industry. There are 2 configurations of the constellation submitted to the ITU by Rwanda Space Agency briefly described below.

First configuration

The complete configuration of the constellation submitted by Rwanda Space Agency to ITU [72] is displayed in Table 2.9. It comprises 27 shells, one of which is equatorial with a single plane, while the remaining 26 are scattered among inclinations ranging from 24 to 98 degrees. Each shell is made up of 36 planes, separated in RAAN by 5 degrees, spanning a range of 180 degrees. Remarkably, each plane consists of 360 satellites, spaced at phase intervals of 0.5 degrees, forming only half a circle (i.e., 180 degrees).

Table 2.9: E-Space. First configuration. Total 337,320 satellites [72].

Altitude [km]	Inclination [deg]	Planes	Satellites per plane	Total # of satellites
550	0.0	1	360	360
553.6	24.0	36	360	12,960
557.2	27.0	36	360	12,960
560.8	30.0	36	360	12,960
564.4	33.0	36	360	12,960
568	36.0	36	360	12,960
571.6	39.0	36	360	12,960
575.2	42.0	36	360	12,960
578.8	45.0	36	360	12,960
582.4	48.0	36	360	12,960
586.0	51.0	36	360	12,960
589.6	54.0	36	360	12,960
593.2	57.0	36	360	12,960
596.8	60.0	36	360	12,960
600.4	63.0	36	360	12,960
604.0	66.0	36	360	12,960
607.6	69.0	36	360	12,960
611.2	72.0	36	360	12,960
614.8	75.0	36	360	12,960
618.4	78.0	36	360	12,960
622.0	81.0	36	360	12,960
625.6	84.0	36	360	12,960
629.2	87.0	36	360	12,960
632.8	90.0	36	360	12,960
636.4	93.0	36	360	12,960
640.0	96.0	36	360	12,960
643.6	98.0	36	360	12,960

Second configuration

The second proposed configuration is notably smaller than the first, yet it remains the largest of its kind. It comprises seven shells, totalling 52,080 satellites [73]. Like the first configuration, the second features one equatorial shell with a single plane, accompanied by inclined Walker constellations. Each shell is designed to contain 240 satellites that are evenly distributed across 360 degrees in-plane, with a phase angle separation of 1.5 degrees.

Table 2.10: E-Space. Second configuration. Total 52,080 satellites [73].

Altitude [km]	Inclination [deg]	Planes	Satellites per plane	Total # of satellites
580	0.0	1	240	240
580	33.0	36	240	8,640
587.2	54.0	36	240	8,640
594.4	66.0	36	240	8,640
601.6	72.0	36	240	8,640
608.8	88	36	240	8,640
616	98	36	240	8,640

2.3. Previous Research

This section provides a concise overview of prior research related to the design of mega-constellation orbital layouts. It briefly presents the published work, the methodologies employed, and the mission objectives and figures of merit.

Despite the scarcity of published research on the design of large LEO constellations, especially those with more than 1000 satellites, this section presents the relevant research available.

He, Hugentobler 2018

He and Hugentobler [46] proposed a method to optimise a large LEO Walker constellation comprising two or three shells at different altitudes for Satellite Navigation (SatNav) applications. However, they only considered visibility as a figure of merit, neglecting other commonly used FoMs such as Geometric Dilution of Precision (GDOP) or Precise Point Positioning (PPP). The total number of satellites ranged from 80 to 160.

Ge et al. 2020

Ge et al. [47] also proposed a method to optimise a constellation for SatNav applications. They identified the best constellation configurations consisting of 1, 2, or 3 shells based on GDOP and PPP. Instead of using numerical optimisers, they applied a grid search method. The total number of satellites ranged from 120 to 240.

Ma et al. 2020

Ma et al. [48] presented another work on SatNav constellation optimisation. They used visibility and the standard deviation of satellite distribution as FoMs. The aim was to maximise the number of satellites in view and ensure uniform distribution across the globe. They used numerical optimisation techniques such as mixed-integer non-linear programming. The number of satellites ranged from 100 to 200.

Arnas et al. 2021

Arnas et al. [25] present a methodology for designing satellite constellations with high temporal and geometric symmetries, called 4D Lattice Flower Constellations (4D LFC).

This 4D LFC methodology is applicable to any number of satellites, even those placed at varying semi-major axes, distinguishing it from 3D LFCs. The method, however, suggests that both the total number of satellites and the number of satellites in each shell are predefined. It also assumes known inclinations for each shell. Essentially, this methodology aids in distributing satellites within shells and in determining shell altitudes for missions or situations (such as interaction or interference with other missions) where coordination between satellites at different semi-major axes is important.

The paper presents a method for an Earth Observation mission involving a constellation of 5,000 satellites distributed across 5 Sun-Synchronous Orbit (SSO) shells at different altitudes, with 1,000 satellites in each shell. Given the number of shells, their inclinations, and the numbers of satellites in each shell, the paper shows how satellites in each shell can be distributed to minimise collision risk. The figure of merit for this objective is the minimum distance between satellites.

The 4D LFC method is suitable for designing mega-constellations, but only when the number of shells and satellites per shell are already determined. In such cases, the 4D LFC method can ensure collision avoidance. However, this project aims to find the optimal values for these parameters, which are essential inputs for the 4D LFC method. Therefore, this method is not applicable for the scope of this project.

Kak, Akyildiz 2021

Kak and Akyildiz [50] presented a framework for designing large-scale CubeSat constellations for Internet of Things (IoT) applications. The framework is scalable for up to 1000 satellites. They used a numerical optimiser to find the optimal layout, and as figures of merit, they used coverage and connectivity, i.e. the total number of Inter-Satellite Links (ISLs). However, coverage was estimated using Voronoi regions and Delaunay triangulation. This coverage estimation method will be discussed in detail in Section 4.1.

Jia et al. 2022

The work of Jia et al. [51] is particularly noteworthy as it will be partially used as a reference in this project. They designed a mega-constellation comprising approximately 5,600 satellites.

The objective was to create a mega-constellation with uniform distribution of the number of satellites in view, or *N asset coverage* as referred to in Jia's work. A schematic overview of this method is described below and summarised in Figure 2.4.

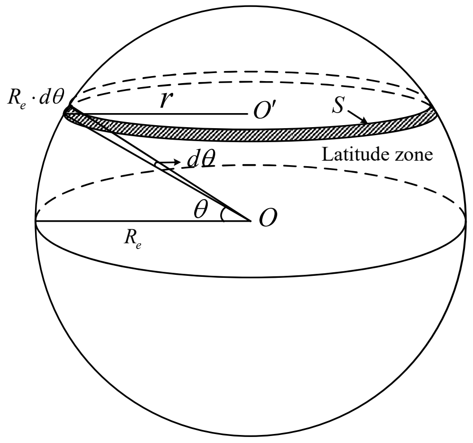


Figure 2.3: Latitude zone on Earth's surface [51].

Step 1: Split the globe in m latitude zones (stripes) of areas S_k with a width of $\Delta\delta = \dfrac{\pi}{m}$. An example of such a zone is shown in Figure 2.3. The number of shells is the same as the number of latitude zones, and each shell is a Walker constellation. The inclination of each shell j that has to reach latitude zone j is $\dfrac{\pi}{2}$, $\dfrac{\pi}{2} - \Delta\delta$, $\dfrac{\pi}{2} - 2\Delta\delta, \ldots, \dfrac{\pi}{2} - \left(\dfrac{m}{2} - 1\right)\Delta\delta$.

Step 2: For each shell C_j with the total number of satellites N_{C_j}, compute the average density of SSPs, $\bar{D}_{k,C_j}$, in each latitude zone k. Start from the shell C_1 with the highest inclination, $i_1 = \dfrac{\pi}{2}$.

This will correspond to SSP density distribution in latitude as shown by the purple bar plots in Figure 2.4. It can be seen that SSPs do not go beyond $\delta_{\text{max},j} = \pm i_j$ with $i_j = 90° - (j-1)\Delta\delta$.

Step 3: Repeat Step 2 for the next shell C_{j+1} with $i_{j+1} = 90° - j\Delta\delta$.

Step 4: Sum densities for each shell as shown in Figure 2.4 by the green bar plots, $\bar{D} = \sum_{j=1}^{m/2} \bar{D}_{C_j}$. As inclination decreases, higher-latitude zones remain unaffected by the lower-inclination shells.

Step 5: Repeat steps 2-4 until the last latitude zone $k = m/2$.

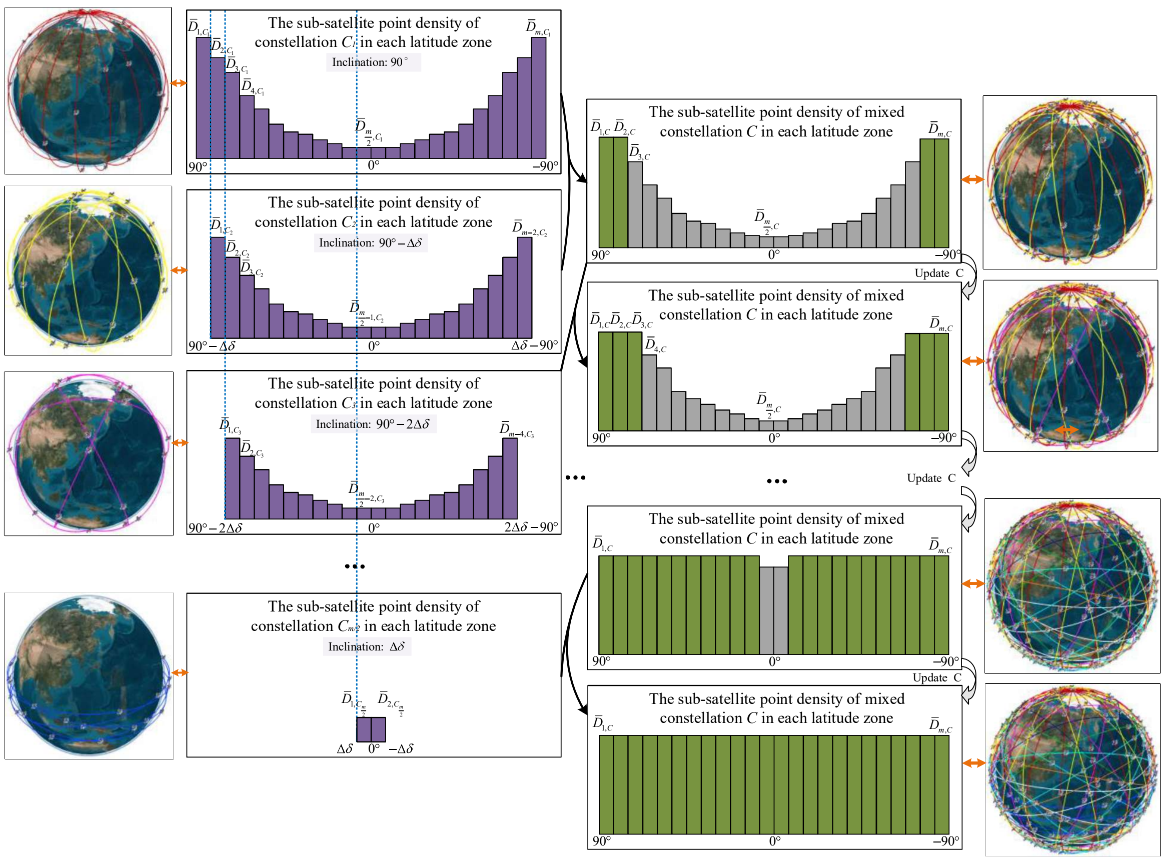

The sub-satellite point density of constellation C_1 in each latitude zone
Inclination: 90°
The sub-satellite point density of constellation C_2 in each latitude zone
Inclination: 90° − Δδ
The sub-satellite point density of constellation C_3 in each latitude zone
Inclination: 90° − 2Δδ
The sub-satellite point density of constellation $C_{m/2}$ in each latitude zone
Inclination: Δδ
The sub-satellite point density of mixed constellation C in each latitude zone
Update C
90°
0°
−90°

Figure 2.4: Schematic approach of constellation design with uniform N asset coverage as presented by Jia et al. [51].

In essence, Jia's method involves examining the distribution of sub-satellite points and "filling" the globe with a selected number of shells to achieve uniform distribution. The "filling" process starts from higher latitudes and goes towards the equator. This provides the uniform average distribution of satellites which, in turn, corresponds to the uniform N asset coverage.

Jia also used inter-satellite links to estimate latency and incorporated collision avoidance in the constellation design. However, these aspects are beyond the scope of the current project. For now, the focus is only on the visibility computation, that is, the number of satellites in view (N asset coverage).

The final design will also be used for software verification, as shown in Section 4.3. This approach is an effective means of verifying the tool due to the number of shells that comprise the mega-constellation and the total number of satellites.

This design will also serve as a reference when a novel method for designing a uniform distribution for a mega-constellation is introduced in Section 5.3. The new method will use the similar approach of designing a mega-constellation by starting from high latitudes and going towards the equator. However, only the number of satellites in view will be considered. The aim is to improve the N asset coverage of Jia's design by adjusting the orbital layout while preserving the same number of satellites.

2.4. Figures of Merit

The mission objectives necessitating large Low Earth Orbit (LEO) satellite constellations are derived from an extensive literature review [25, 28–51, 74–80]. These objectives are categorised into four main areas, as outlined below:

- **Satellite communications (Satcom)**
  This objective aims to provide global broadband phone and internet access to remote and underserved areas.
- **Satellite navigation (Satnav)**
  This objective aims to provide space-based navigation and positioning system for terrestrial and orbital users.
- **Remote sensing**
  This objective includes all types of observation of emitted radiation at a distance (as opposed to in-situ observations). It covers Earth observation for environmental and scientific purposes as well as space-based intelligence, surveillance and reconnaissance (ISR).
- **Internet of Things (IoT)**
  IoT overlaps with Satcom and aims to provide global coverage and connectivity for IoT devices. The major difference between the IoT and Satcom is the amount of data that has to go through satellites. IoT satellites have to handle much lower data rates, which means that their link budget requirements are lower than those of Satcom. This, in turn, may affect the number of satellites, their size, or their orbits (e.g. lower elevation angles may offer better coverage).

As shown from the existing methods in Section 2.3, there are multiple figures of merit used to evaluate performance of a mega-constellation. These figures of merit will be summarised in this section.

2.4.1. Visibility

Visibility, defined as the number of satellites observable from a ground-based point, is typically the most important Figure of Merit (FoM) when designing the orbital layout of a constellation. This importance is underscored by both academic [46, 50, 51] and industrial [54, 62, 63, 66, 68, 69] sectors. This FoM is applicable to all applications, with the exception of Remote Sensing (RS), where the primary interest lies in revisit time.

For visibility, typically mission analysts are interested in two FoMs: mean visibility and minimum visibility. The two are explained below.

- **Mean visibility**
  This is how many satellites an observer will have in view *on average*. This FoM helps estimate how satellites are distributed across the area of interest. Although averaging the visibility over time does not show its variation, it can be assumed that the variance will be reasonably low, making this FoM a good representation.
- **Minimum visibility**
  This indicates the smallest number of satellites visible from a point on the ground. The minimum visibility is of interest as it often pertains to the requirement of uninterrupted coverage by a specific number of satellites, a condition that cannot be guaranteed by the average visibility.

Visibility is computed by the given satellite position and minimum elevation angle. It is depicted in Figure 2.5 how it is determined whether a satellite is in view. The following condition must be met (assuming no obstruction by terrain or buildings):

$$\varepsilon \geq \varepsilon_{\text{min}} \tag{2.8}$$

where ε is the real elevation angle of the satellite with respect to the ground point while ε_{min} is the minimum elevation angle.

Furthermore, the Earth central angle can be expressed as

$$\theta = \arccos\left(\frac{R_e}{R_e + h}\cos\varepsilon_{\text{min}}\right) - \varepsilon_{\text{min}} \tag{2.9}$$

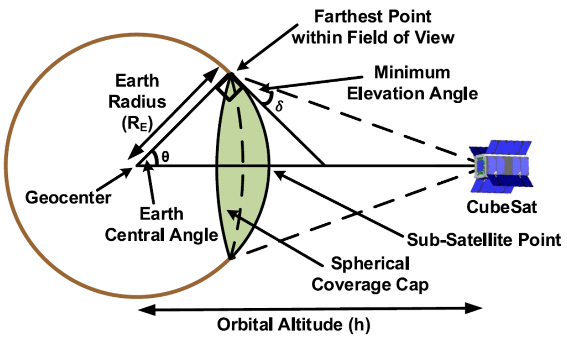


Figure 2.5: Coverage and visibility geometry of a satellite [50]. Note that the figure uses different notation for the minimum elevation angle: in this report ε is used to denote the elevation while in the figure it is shown as δ. Also note that orbital altitude h measure from Earth's surface to the satellite while in the figure it is shown incorrectly.

Given the time constraints for this thesis and the importance of visibility computation in the design of mega-constellation orbital layouts, visibility will be the primary FoM for this project.

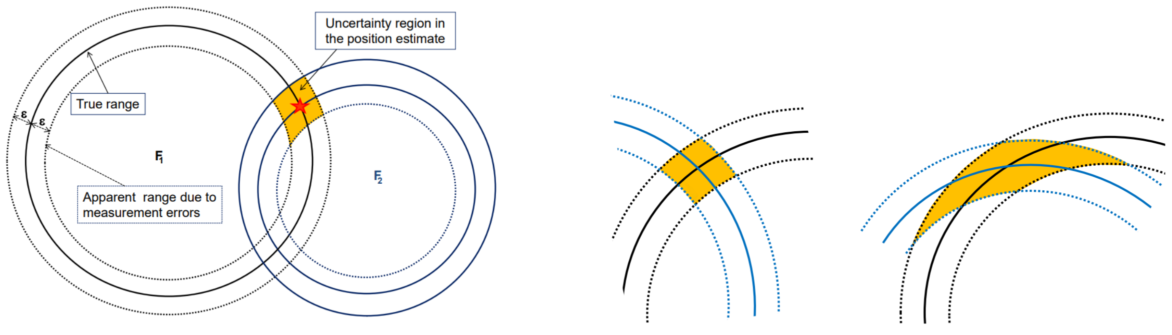
2.4.2. Revisit time

Revisit time refers to the duration between successive observations of the same point on Earth by a satellite (either single or comprising a constellation). This is a common figure of merit for Earth Observation (EO) missions or other Remote Sensing (RS) missions, such as space-based ISR. Numerous authors have utilised revisit time as a FoM in their mission design [29, 31–34, 36, 37, 42, 43, 75, 78, 81].

For EO missions, the satellite's field of view is typically quite narrow, especially when compared to other objectives like broadband connectivity or IoT. To analyse the ground coverage by a large number of satellites, a high-resolution grid of ground points would be necessary, complicating the analysis. While there is an example of such a mission, Planet Labs, which has launched over 500 satellites[9], all other current mega-constellation projects are deployed (or planned) for applications other than Earth Observation. Therefore, revisit time will not be a consideration within the scope of this thesis.

2.4.3. Dilution of precision

Dilution of Precision (DOP) is the quantitative FoM showing how the geometry of satellites affects the positioning error. Shown in Figure 2.6 is the example of how positioning accuracy is influenced by measurement errors and the geometry of the satellites. DOP is particularly relevant for global navigation applications and was used by various authors for constellation [28, 76, 77] and mega-constellation design [47–49].


Uncertainty region in the position estimate
True range
ε
ε
F_1
F_2
Apparent range due to measurement errors

(a) Position uncertainty region due to errors of measurements. **(b)** Example of the satellite geometry effect on positioning uncertainty.

Figure 2.6: Example of DOP on 2D surface [82].

[9]https://www.newspace.im/constellations/planet-labs, accessed 14 May 2024

DOP computation begins with the geometry matrix **G** [82]:

$$\mathbf{G} = \begin{bmatrix} a_1 & b_1 & c_1 & 1 \\ a_2 & b_2 & c_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ a_n & b_n & c_n & 1 \end{bmatrix} \tag{2.10}$$

where a_i, b_i and c_i represent the direction cosines of satellite position parameters and ones refer to the clock error parameter.

The covariance matrix **P** of the error of the estimate is given by the following equation [82]:

$$\mathbf{P} = \sigma^2(\mathbf{G}^T\mathbf{G})^{-1} \tag{2.11}$$

where σ^2 is the variance of the measurements (prefit residuals).

DOP parameters are defined in matrix **Q** with x, y, z representing directions in 3D space and t representing time [82]:

$$\mathbf{Q} = (\mathbf{G}^T\mathbf{G})^{-1} = \begin{bmatrix} q_{xx} & q_{xy} & q_{xz} & q_{xt} \\ q_{xy} & q_{yy} & q_{yz} & q_{yt} \\ q_{xz} & q_{yz} & q_{zz} & q_{zt} \\ q_{xt} & q_{yt} & q_{zt} & q_{tt} \end{bmatrix} \tag{2.12}$$

The GDOP can then be calculated [82]:

$$GDOP = \sqrt{q_{xx} + q_{yy} + q_{zz} + q_{tt}} \tag{2.13}$$

It is important to note that the calculation of GDOP implies that the parameters comprising the geometry matrix **G** are derived from the positions of satellites that are in view. This means that GDOP inherently includes visibility.

2.4.4. Positioning Accuracy

To determine positioning accuracy, an error metric can be used. For example, root mean square error (RMSE) can be used to determine accuracy of position that is computed using either code or carrier-based methods. Code and carrier-based positioning are used for GNSS applications only and require additional data and simulations than just satellite propagation. While this provides additional information about constellation performance, this also becomes much more time consuming in terms of computational time. In this subsection concepts will only be introduced briefly. A more detailed guide how the two are computed can be found in ESA handbook on GNSS data processing [82].

Code-Based Positioning (SPP)

Code-based positioning, also referred to as Standard Point Positioning (SPP) is a method of computing receiver state based on single-frequency code pseudorange measurements. SPP method converges very quickly to a final solution, requiring only a few iterations. However, it comes at a cost of relatively low accuracy of a few meters. There does not seem to be any published research that uses SPP as a figure of merit. This has likely to do with the low accuracy provided by SPP which negates the necessity for a LEO constellation.

Carrier-Based Positioning (PPP)

Carrier-based positioning, or Precise Point Positioning (PPP), is a method of determining receiver state based on dual-frequency carrier phase measurements. These can provide much an accuracy of a few centimetres but require significantly more computational effort. Besides, PPP requires additional considerations of precise satellite orbits and clocks, relativistic effects, atmospheric effects, antenna biases and orientation, and Earth deformation effects. It can take tens of minutes to a few hours for a PPP algorithm to converge. Thus, designing a mega-constellation layout using PPP can be extremely time-consuming. There are, however, multiple published studies that aim to design a GNSS constellation enhanced by LEO satellites [49, 83, 84].

2.4.5. Latency

In the context of constellation design, latency is defined as the time required for a signal to traverse from one ground-based transmitter to another. LEO constellations can offer significantly lower latency times compared to GEO-based communication systems and ground-based systems. Ground-based internet systems rely on fiber optics, which incur a 50% higher delay due to a higher refraction index and consequent slower speed of light [85]. Modelling latency requires the implementation of networking algorithms and the allocation of ground gateways, thereby complicating the analysis.

Latency is typically used for network topology optimisation and is essential for optimising the network infrastructure on the ground as well as for determining the optimal routing for signals in space. Although the optimal routing will depend on the distribution of satellites (number of planes and satellites per plane) within the constellation, latency as a FoM will be utilised when the number of satellites is more or less established, i.e., in the later phases of the design. Numerous studies focus on latency in mega-constellations, not only for communication services but also for Earth observation tasks such as disaster monitoring and emergency response [86–89]. However, it is not used as a defining FoM for the orbital layout of a mega-constellation.

2.4.6. Signal-to-noise ratio

The Signal-to-Noise Ratio (SNR) is a parameter that quantifies the clarity of the received signal. It is defined as the ratio of the power of a signal to the power of the background noise. While this FoM is particularly important, especially for SatCom, it is typically employed in link budget estimations. These estimations, in turn, are used for payload optimisation and gateway allocation. Once these requirements are estimated, they can be passed on to mission analysts who will then utilise other FoMs, such as minimum visibility.

SNR also accounts for the required data throughput, which may influence routing (and hence, latency) as satellites have limited throughput capacity. This limitation implies that the signal would need to be redirected, necessitating multi-path routing. Although this may influence the design of the orbital layout, SNR is not used as the primary FoM, as shown by published studies that examine SNR [90–93]. Therefore, the orbital layout is not optimised based on the SNR. It is typically calculated beforehand and used, for instance, to define the minimum visibility requirement, or SNR is computed once the orbital layout is determined.

2.4.7. Conclusion

This section explored the typical figures of merit and applications for mega-constellations. However, due to the time constraints of the thesis, it is not feasible to implement all of these. Consequently, only a select few can be chosen for further analysis.

According to FCC filings from private companies [54, 62–64, 66, 68–71], visibility is the primary figure of merit employed in the industry for the initial design of the orbital layout. One is typically interested in the minimum visibility to ensure uninterrupted coverage. However, mean visibility gives a measure of the “typical” service level, while minimum visibility ensures service availability even under less-than-ideal conditions. Both these metrics together provide a comprehensive view of the coverage performance of the constellation, making them of utmost interest in mega-constellation design.

2.5. Summary and Conclusions

This chapter presented an overview of existing mega-constellations systems and relevant scientific research. From this, the typical figures of merit for mega-constellation design were identified. Mean and minimum visibility were selected as the main focus FoMs for this thesis.

Key Takeaways

- Mega-constellations typically consist of one or multiple shells that are represented by Walker Delta or Walker Star configurations
- Mean and minimum visibility (number of satellites in view from the ground) are the primary figures of merit used in mega-constellation design

3 Constellation Propagation

Chapter 2 demonstrated that the analysis of a mega-constellation requires the orbit propagation for all satellites. The challenge with such a large constellation lies in its sheer number of satellites. While orbit propagation for small constellations only takes fractions of a second, it can increase to minutes, or even tens of minutes, for mega-constellations with thousands of satellites. This dramatically impacts computational efficiency, especially during optimisation processes involving thousands or millions of design iterations.

The Godot[1] library, currently in use at ESOC, is not well-suited for the very large constellations considered in this thesis. Not only is it cumbersome to define thousands of satellites in Godot, but their propagation also consumes an excessive amount of computational time. Therefore, a more robust tool is required, one that can define mega-constellations more efficiently and propagate them faster.

It is also important to note that merely propagating the satellites and determining their final positions is insufficient. To ensure seamless coverage, the satellite orbits must be retrieved at relatively small time intervals (e.g., 30 seconds or 60 seconds). This requirement implies that accurate numerical integrators with adaptive step size are unsuitable for this task, because their efficiency would be reduced by the fixed step size. Consequently, more efficient methods of propagating large constellations are needed.

This chapter will address the challenge of large constellation propagation by showing how computational time can be reduced. Section 3.1 describes the implementation of an analytical model that yields an acceptably accurate result. The conversion to the reference system, which is further used for visibility computation, is presented in Section 3.2. Moreover, Section 3.3 shows how this model can be optimised at the software level. Finally, software verification is presented in Section 3.4, where a comparison of accuracy and computational time for a fully vectorised analytical approach and numerical method is also shown.

3.1. Dynamics

During the initial design phase, the accuracy requirements for satellites' orbit propagation are relatively lenient. There is no immediate need to implement high-fidelity models of spacecraft dynamics, as an error margin of a few tens of kilometres is acceptable for visibility purposes. This flexibility allows for the implementation of less accurate but significantly faster analytical propagation methods, thereby reducing computational time considerably.

3.1.1. Point-Mass (Keplerian Orbit)

The simplest approach to implementing an analytical solution is through the use of a Keplerian orbit, which assumes point-mass dynamics. This is achieved in Keplerian coordinates using the equations outlined below to propagate a Keplerian orbit around a point-mass [94].

[1] `https://godot.io.esa.int/`, accessed 14 May 2024

The Keplerian state $\mathbf{x}_{\text{Kep}}$ at a given time instance t_i is defined by six elements:

$$\mathbf{x}_{\text{Kep}} = [a \quad i \quad e \quad \Omega \quad \omega \quad \theta]^T \tag{3.1}$$

In an analytical solution around a point-mass, the first five elements remain constant, with only the true anomaly θ subject to change. The change in true anomaly can be computed by first calculating the mean motion n [94]:

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

Here, M represents the mean anomaly, which can be computed using the following formula [94]:

$$M = M_0 + n(t - t_0) \tag{3.3}$$

The eccentric anomaly E can then be found using the equation [94]:

$$M = E - e \sin E \tag{3.4}$$

Finally, the true anomaly θ is computed with the equation [94]:

$$\tan\frac{\theta}{2} = \left(\frac{1+e}{1-e}\right)^{1/2} \tan\frac{E}{2} \tag{3.5}$$

If the orbit is assumed to be perfectly circular, this calculation can be simplified. In this case, the eccentricity is equal to zero ($e = 0$), which results in all three anomalies being equal [94]:

$$\theta = E = M \tag{3.6}$$

This implies that the true anomaly of a spacecraft in a circular orbit can be simply computed with the propagated time t and mean motion n [94]:

$$\theta = \theta_0 + n(t - t_0) \tag{3.7}$$

3.1.2. J2 Perturbation

Among the various perturbations, the J2 perturbation is the most dominant for the altitude range that is usually considered by projects involving mega-constellations: 500-800 km. While atmospheric drag can slow down a spacecraft at low altitudes, for the purpose of this study, it is assumed that satellites either do not experience this effect or counteract it using thrusters. In other words, the effect of atmospheric drag on satellite movement is considered negligible, even at low altitudes.

In a Walker constellation, all satellites experience the same perturbation, effectively cancelling out the J2 effect. Despite this, the decision was made to implement the J2 effect due to its relevance in multi-shell constellations. Unlike single-shell constellations, multi-shell constellations do not cancel out the J2 effect as different shells experience varying perturbations due to J2. Furthermore, the implementation of the J2 effect is a straightforward and relatively quick process. Importantly, it has a negligible impact on computational time, further justifying its inclusion as it poses no significant drawback.

The advantage of incorporating J2 perturbation is that it does not require numerical propagation and can be computed analytically. However, certain assumptions must be made. When J2 perturbation is added, it results in osculating semi-major axis a, eccentricity e, and inclination i if computed numerically. For an analytical approach, one can assume mean values for these three elements that remain constant, while the other three elements, the right ascension of the ascending node Ω, the argument of perigee ω, and the true anomaly θ, will change due to J2 [94].

$$\frac{da}{dt} = \frac{di}{dt} = \frac{de}{dt} = 0 \tag{3.8}$$

The changes in the argument of periapsis ω and the longitude of the ascending node Ω due to J2 perturbation can be computed as follows [94]:

$$\Omega = \Omega_0 + \frac{d\Omega}{dt}(t - t_0) \tag{3.9a}$$

$$= \Omega_0 - \frac{3}{2}J_2 n \left(\frac{R_E}{a}\right)^2 \left(1 - e^2\right)^{-2} \cos i \cdot (t - t_0) \tag{3.9b}$$

$$\omega = \omega_0 + \frac{d\omega}{dt}(t - t_0) \tag{3.10a}$$

$$= \omega_0 + \frac{3}{2}J_2 n \left(\frac{R_E}{a}\right)^2 \left(1 - e^2\right)^{-2} (2 - \frac{5}{2}\sin^2 i) \cdot (t - t_0) \tag{3.10b}$$

Due to change in ω, the true anomaly θ will shift as well [94]:

$$M = M_0 + \frac{dM}{dt}(t - t_0) \tag{3.11a}$$

$$= M_0 + \frac{3}{4}J_2 n \left(\frac{R_E}{a}\right)^2 \left(1 - e^2\right)^{-3/2} (3\cos^2 i - 1) \cdot (t - t_0) \tag{3.11b}$$

In the case where the eccentricity is zero, $e = 0$, the following equations hold [94]:

$$\Omega = \Omega_0 - \frac{3}{2}J_2 n \left(\frac{R_E}{a}\right)^2 \cos i \cdot (t - t_0) \tag{3.12}$$

$$\omega = \omega_0 + \frac{3}{2}J_2 n \left(\frac{R_E}{a}\right)^2 (2 - \frac{5}{2}\sin^2 i) \cdot (t - t_0) \tag{3.13}$$

$$M = M_0 + \frac{3}{4}J_2 n \left(\frac{R_E}{a}\right)^2 (3\cos^2 i - 1) \cdot (t - t_0) \tag{3.14}$$

For circular orbits, where $\theta = M$, the true anomaly θ is computed in the same manner as the mean anomaly M [94]:

$$\theta = \theta_0 + \frac{3}{4}J_2 n \left(\frac{R_E}{a}\right)^2 (3\cos^2 i - 1) \cdot (t - t_0) \tag{3.15}$$

Figure 3.1 presents a comparison of the analytical and numerical propagation of point-mass + J2 dynamics. As previously discussed, the semi-major axis a, the eccentricity e, and the inclination i remain constant in the analytical propagation, while they osculate in the numerical propagation. The longitude of the ascending node Ω is assumed to be non-osculating but with a constant linear change rate. This rate changes in the same manner as in numerical propagation as seen from Figure 3.1.

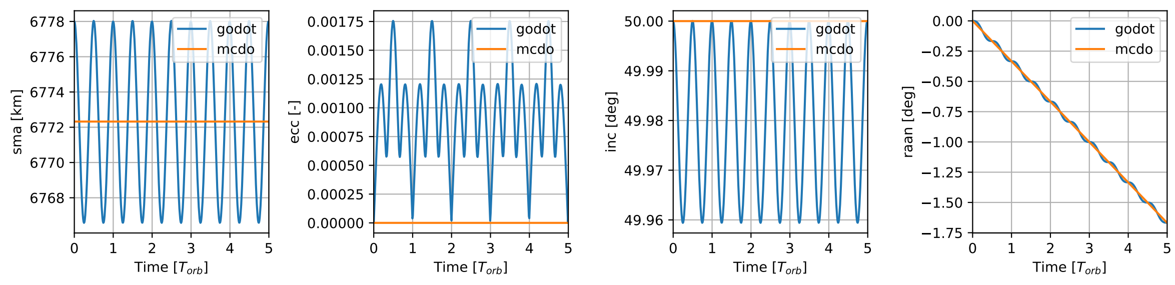


Figure 3.1: Comparison of propagated Keplerian elements of analytical and numerical models with point-mass + J2 dynamics.

It is expected that this will lead to a certain degree of error in the final position compared to the more accurate numerical propagation. However, a detailed discussion on this will be presented in Section 3.4.

3.2. Keplerian to Cartesian

Furthermore, since visibility is computed based on satellite elevation with respect to the observation point on the ground, one needs to compute the elevation. The elevation is defined as the angle above the local horizon of the observation point. This angle can be computed by finding the angle between two vectors: vector from Earth centre to the observation point and vector from the observation point to the satellite.

The easiest way to compute the angle between the two is to have both vectors in Cartesian coordinates. This, however, requires that satellite states are converted from Keplerian to Cartesian coordinates. Namely, for visibility computation, there is no need to use the velocity, so retrieving Cartesian position only would be sufficient. But it is a good practice, nevertheless, to get the full state vector of a spacecraft because if more figures of merit are considered later, the velocity can be of use as well.

The Cartesian elements are computed as shown below [95].

1. Compute the semi-latus rectum p using the semi-major axis a and eccentricity e [95]:

$$p = a \cdot (1 - e^2) \tag{3.16}$$

2. Compute the radial distance R from the focus to the point on the orbit using the semi-latus rectum p and true anomaly θ [95]:

$$R = \frac{p}{1 + e \cdot \cos\theta} \tag{3.17}$$

3. Compute the velocity components in the radial and transverse directions, V_r and V_ϕ [95]:

$$V_r = \sqrt{\frac{\mu}{p}} \cdot e \cdot \sin\theta \tag{3.18}$$

$$V_\phi = \sqrt{\frac{\mu}{p}} \cdot e \cdot (1 + \cos\theta) \tag{3.19}$$

4. Compute the position components x, y and z in the orbital plane [95]:

$$x = R \cdot \big(\cos\Omega \cdot \cos(\omega + \theta) - \sin\Omega \cdot \sin(\omega + \theta) \cdot \cos i\big) \tag{3.20}$$

$$y = R \cdot \big(\sin\Omega \cdot \cos(\omega + \theta) + \cos\Omega \cdot \sin(\omega + \theta) \cdot \cos i\big) \tag{3.21}$$

$$z = R \cdot \big(\sin(\omega + \theta) \cdot \sin i\big) \tag{3.22}$$

5. Compute the velocity components V_x, V_y and V_z in the orbital plane [95]:

$$V_x = V_r \cdot x - V_\phi \cdot \big(\cos\Omega \cdot \sin(\omega + \theta) + \sin\Omega \cdot \cos(\omega + \theta)\cos i\big) \tag{3.23}$$

$$V_y = V_r \cdot y - V_\phi \cdot \big(\sin\Omega \cdot \sin(\omega + \theta) - \cos\Omega \cdot \cos(\omega + \theta)\cos i\big) \tag{3.24}$$

$$V_z = V_r \cdot z + V_\phi \cdot \big(\cos(\omega + \theta) \cdot \sin i\big) \tag{3.25}$$

3.3. Software Implementation and Vectorisation

One of the main decisions in the software implementation of models for this project was the choice of programming language. Python emerged as the preferred choice due to several compelling reasons:

1. **Compatibility with Godot:** Despite its limitations in handling large constellations, Godot is extensively utilised in the Mission Analysis Section. Its fundamental features, such as axes rotation, state conversion, and state retrieval, are highly beneficial. Notably, Godot's Python interface allows users to tap into its core functionality, which is primarily written in C/C++. This feature facilitates the offloading of computational tasks to low-level languages, thereby retaining Python's advantages of readability, simplicity, and flexibility.
2. **Ease of Development:** Python is renowned for its ease of writing and learning, along with its capability to integrate other languages seamlessly. It has a large and active developer community that contributes numerous useful libraries and tools for a variety of purposes.

3. **Popularity in the space exploration industry:** Python's widespread use in the space exploration industry is an added advantage. This popularity implies that potential users of this software are more likely to be familiar with Python.

Despite its versatility, Python has a notable limitation: it is an interpreted language. Unlike compiled languages such as C/C++, Java, or Fortran, Python code is executed line by line by an interpreter, rather than being converted into machine code by a compiler. This process introduces significant overhead, reducing computation speed.

To enhance Python's performance, particularly for iterative (for/while loops) and memory-intensive (large arrays) operations, a solution needs to be found. *Vectorisation* presents itself as a viable option. This technique operates on entire arrays or matrices instead of individual elements, reducing the number of loops and function calls. It also enables the use of optimised libraries, such as *NumPy* [96], which implement vectorised operations in C.

By employing vectorisation, it is possible to substantially improve the speed and efficiency of Python code. Furthermore, it enhances the conciseness and readability of the code.

NumPy allows to do various element-wise operations with just one line of code, e.g. element-wise array multiplication would look like this:

```python
### Pure python implementation
a = [1, 2, 3]
b = [7, 8, 9]

product = len(a) * []

for i in range(len(a)):
    product[i] = a[i] * b[i]

### NumPy implementation

a = np.array([1, 2, 3])
b = np.array([7, 8, 9])

product = a * b
```

NumPy provides the capability to apply vectorisation not only to one-dimensional arrays but also to multi-dimensional arrays. However, the feasibility of vectorisation is contingent upon the operation's compatibility with this process. In cases where vectorisation is not applicable, a high-level Python implementation remains necessary.

As demonstrated in the equations presented in Section 3.1 and Section 3.2, vectorisation is applicable in this context. Each equation corresponds to an individual satellite at a specific time epoch, indicating no dependency on other satellites or time instances.

Numerical propagation in Godot for simple dynamics can be executed using the object for propagation of a massless spacecraft, called `godot.cosmos.BallisticPropagator`[2]. However, this method is notably slower than analytical propagation, and setting up the simulation is quite cumbersome. For the simulation setup, each satellite must be individually specified in the configuration file for the `godot.cosmos.Universe`[3], either manually or through a programmatic approach. This process becomes particularly burdensome when propagating the orbits of thousands of satellites. For this thesis, the `BallisticPropagator` was implemented for verification. It will also be evaluated in terms of computational time and accuracy.

As an alternative, the `godot.core.astro.keplerPropagate`[4] function offers analytical point-mass propagation in Godot. Its limitation lies in the fact that it accepts only 2D arrays as input, whereas a 3D array input would be preferable for this thesis. Such 3D arrays would contain the state of each

[2]https://godot.io.esa.int/godotpy/api_reference/generated/godot.cosmos.BallisticPropagator.html, accessed 14 May 2024

[3]https://godot.io.esa.int/godotpy/api_reference/generated/godot.cosmos.Universe.html, accessed 14 May 2024

[4]https://godot.io.esa.int/godotpy/api_reference/generated/generated/godot.core.astro.keplerPropagate.html, accessed 14 May 2024

satellite at every epoch, with dimensions of $T \times N \times 6$, where T represents the number of epochs, N the number of satellites, and 6 the elements of the Keplerian state vector. Despite this, the use of `godot.core.astro.keplerPropagate` necessitates Python for-loops, which adds computational overhead and reduces efficiency. This method, known as the *non-vectorised* implementation, was initially employed in this study.

As a result, a software tool that propagates satellites positions was created as a Python module and was given a name `mcdo` which stands for “Mega-Constellation Design Optimisation”. The analytical propagation model described in Section 3.1 and state conversion described in Section 3.2 were implemented primarily using Python and NumPy. Basic functions from `godot.core.astro`[5] and `godot.core.tempo`[6] are used for simulation setup.

Performance improvement of the vectorised implementation with respect to Godot and non-vectorised implementation will be shown in Section 5.4. Specifically, the comparison with Godot utilises the `BallisticPropagator`. This object propagates the state of a massless spacecraft for a given dynamical model. This work considers only Earth point-mass and J2 perturbation. For clarity, throughout the context of constellation propagation, the use of word *Godot* will refer to `BallisticPropagator` object, unless otherwise specified.

It is important to note that this comparison also encompasses state conversion (or retrieval in the case of Godot). The computational time for these processes is calculated collectively due to the differing implementation methods between the `mcdo` module and Godot. The `mcdo` module propagates states of all satellites before converting the states to Cartesian, whereas Godot necessitates the propagation of each satellite individually. Following this, Godot retrieves Cartesian states for each given epoch before proceeding to the next satellite.

3.4. Verification

Verification is completed in order to confirm the correctness of propagation results. It can be seen from Figure 3.2 that the positional error does not exceed 30 km over 15 full orbital periods which is a bit less than 24 hrs for $a = 6778$ km (approximately 400 km altitude). Verification for more altitudes – up to 2000 km – is shown in Appendix A. Due to the fact that visibility computations will only depend on the positions of the satellites, there is no need to analyse velocity. The difference in position is explained by the fact that for analytical model, mean non-osculating values are used.

Figure 3.2 shows how osculating elements can differ. The argument of perigee and true anomaly are not shown as they are undefined for a circular orbit. However, it is clearly seen that using mean values yields acceptable results. Thus, the propagation of satellite orbits with the `mcdo` module can be considered verified.

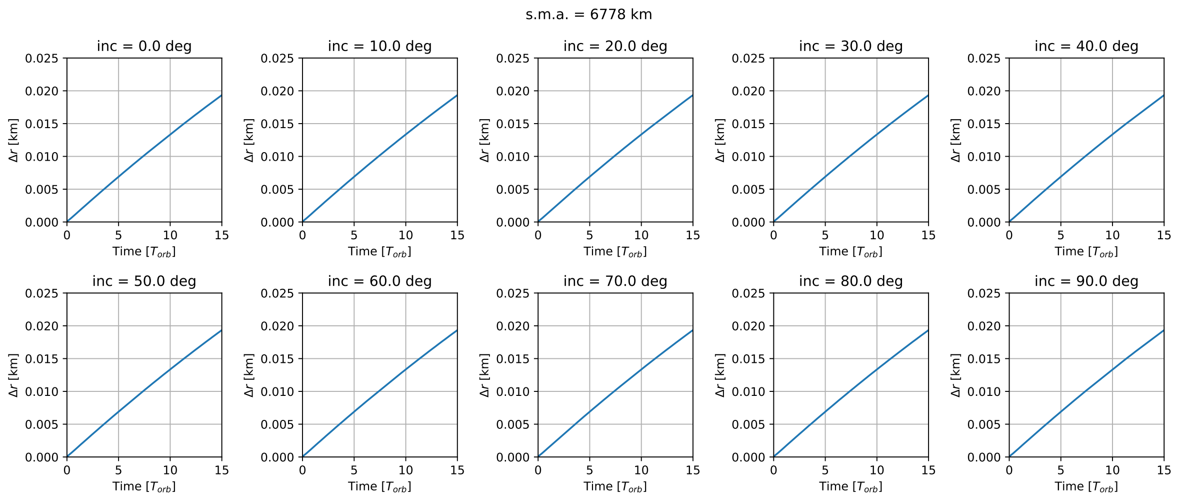
3.5. Summary and Conclusions

This chapter described how analytical method can be implemented for propagation of mega-constellations consisting of thousands of satellites with simple dynamics: point-mass and J2 effect. It also demonstrated how NumPy can be utilised to accelerate notoriously slow Python performance by means of vectorisation. Finally, the accuracy of this approach was discussed with respect to numerical propagation. It was shown that the analytical propagation introduces a positional error of up to 25 meters with point-mass dynamics and up to 30 km with point-mass + J2 dynamics.

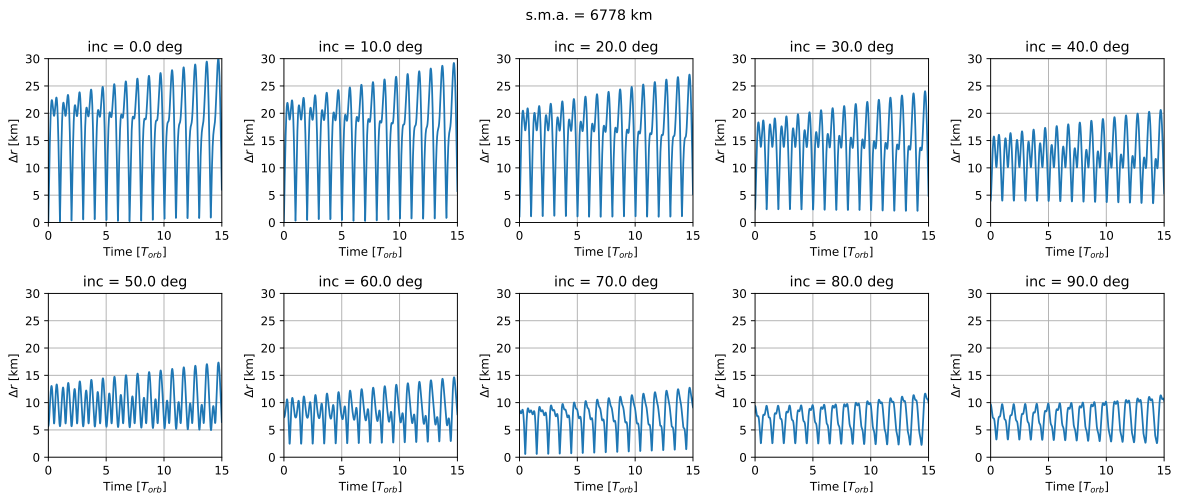
Key Takeaways

- Analytical approach for point-mass and J2 dynamics introduces a positional error of up to 25 km with respect to numerical propagator in Godot
- Vectorisation can be applied to accelerate state propagation and conversion from Keplerian to Cartesian

[5] `https://godot.io.esa.int/godotpy/api_reference/generated/godot.core.astro.html`, accessed 14 May 2024
[6] `https://godot.io.esa.int/godotpy/api_reference/generated/godot.core.tempo.html`, accessed 14 May 2024



(a) Point-mass only.



(b) Point-mass + J2.

Figure 3.2: Positional error of analytical model with respect to numerical model (`BallisticPropagator` in Godot).

Limitations and Recommendations

- This tool is not applicable anymore if higher accuracy or higher model fidelity is required.
- CUDAjectory can be used for very fast *numerical* propagation with higher fidelity dynamics [97].

4
Coverage Estimation

In the domain of satellite constellation design, *coverage* refers to the geographical area on the Earth's surface that can be serviced by a satellite or a constellation of satellites at a given instant. *Uninterrupted* (continuous) coverage would mean that the area can be serviced at any given moment, while *interrupted* coverage implies that the coverage is not continuous (discontinuous) [18].

Coverage can be categorised into two types [18]:

1. **Global Coverage**
   This refers to a scenario where the entire Earth's surface is serviced by at least one satellite from the constellation at all times. This is a critical requirement for systems such as the Global Positioning System (GPS) or Satcom, which necessitate a global service.
2. **Regional Coverage**
   This refers to a scenario where a specific geographical region on Earth is serviced by at least one satellite from the constellation at all times. This is typically employed when the service is intended for a specific region, such as a particular country or continent.

The process of designing a satellite mega-constellation to achieve optimal coverage is a complex task that requires the consideration of numerous factors. These include the number of satellites, their respective orbital parameters (such as altitude and inclination), and the overall configuration of the constellation.

A significant challenge in the design of mega-constellations is the estimation of coverage, which is known to be a computational bottleneck [50, 98]. This is primarily due to the extensive number of satellites, grid points, and time instances that require evaluation for coverage. As the number of these elements increases, the computational time grows rapidly.

This issue gets worse further when one aims for high-resolution coverage on the ground, as it results in a significant increase in the number of grid points. Consequently, the computational demands for coverage estimation in the design of satellite mega-constellations present a significant challenge that requires further investigation and optimisation.

As discussed in Chapter 2, the coverage of a mega-constellation is estimated using visibility as the primary figure of merit. Consequently, this chapter is dedicated to the computation of visibility within the context of mega-constellation design. Section 4.1 introduces existing methodologies and research conducted on coverage estimation, with a particular emphasis on visibility. Section 4.2 then describes the implementation of parallel computing principles for visibility computation acceleration. The verification of this parallelised visibility computation is presented in Section 4.3. Upon successful implementation, Section 4.4 discusses various simplifications that contribute to a reduction in computational time, along with an analysis of their impact on accuracy. Finally, Section 4.5 presents a parametric analysis, illustrating the influence of parameter variation on the visibility of a mega-constellation.

4.1. Existing Methods

Numerous methods have been proposed to enhance the efficiency of coverage computation. However, these methods often involve simplifications that may lead to a decrease in accuracy, or they may be too complex to implement within the thesis' time constraints.

This report enumerates several methods previously introduced by researchers. However, it does not delve deeply into the specifics of these methods, particularly those that were not utilised in this thesis. Instead, the focus is on a newly introduced numerical method that has demonstrated a significant performance improvement compared to non-optimised numerical methods.

Grid Point Method

The Grid Point method is the most straightforward approach to computing coverage. This method involves applying a grid of points to the target area, typically selected uniformly in both longitudinal and latitudinal directions. The grid resolution is determined based on the mission objective and accuracy requirements. Once this is established, the visibility at each point is computed.

Visibility in the Grid Point method is calculated based on the satellite's elevation from a given point and its field of view. This implies that not only must the satellite be visible from the ground, but the observation point must also be within the satellite's field of view. Consequently, other parameters such as the satellite antenna field of view and pointing offset must be considered, or reasonable assumptions must be made. For this study, it is assumed that satellites always point in the nadir direction (towards the Earth's center). Another assumption is that satellites move in circular orbits, which means that the field of view can be expressed equivalently to the minimum elevation angle, as illustrated in Figure 2.5.

The advantage of this method is its high accuracy, easy implementation, and universality. The grid does not have to be uniform which allows to select any region of interest. However, this also the most demanding approach in terms of computational time.

The time complexity of this method is $\mathcal{O}(T \cdot M \cdot N)$, where T is the number of time instances, M is the number of grid points, and N is the number of satellites.

Latitude Stripe Method

The Latitude Stripe method [99] is a technique used to calculate the global coverage for a satellite constellation. As its name implies, the method divides the target region into narrow latitude stripes. All satellites are assumed to be in the same orbit. Thus, it allows to compute the coverage from each satellite only in the first orbital period, and then simply "drift" it to obtain the coverage of each latitude stripe. The method allows to obtain instantaneous coverage rate as well as accumulated coverage rate.

While this method is noticeably faster than the Grid Point method, the disadvantage of this method is that the output is averaged along longitude resulting in the mean visibility as a function of latitude. While the mean visibility is deemed sufficient in many cases, for this project, we are also interested in the minimum visibility (as outlined in Section 2.4), which cannot be computed with the Latitude Stripe method.

The time complexity of this method is $\mathcal{O}(T \cdot M_{\text{lat}} \cdot N)$, where T is the number of time instances, M_{lat} is the number of latitude stripes, and N is the number of satellites [99].

Voronoi Diagram

This method is based on Voronoi regions and Delaunay triangles. These are shown in Figure 4.1 [100]. Voronoi diagrams are a way of dividing surface into different regions based on a set of points. Each point gets its own region, which consists of all the locations that are closer to that point than to any other. As shown in Figure 4.1, all points within the highlighted red region are closer to vertex O than any other vertex.

Delaunay triangulation is a technique closely related to Voronoi diagrams. It is a triangulation of a set of points in the plane such that no point is inside the circumcircle of any triangle. The circumcircle of a triangle is defined as the circle that passes through all three vertices of the triangle. Delaunay triangulations maximise the minimum angle of all the angles of the triangles in the triangulation, avoiding skinny triangles.

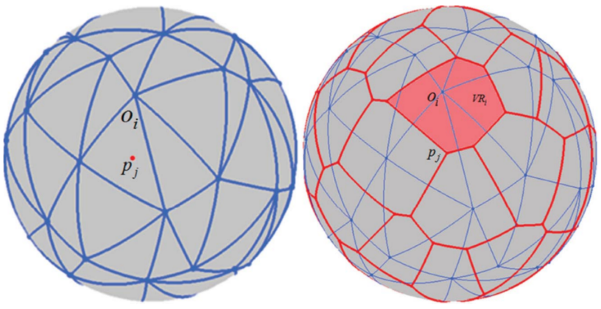


Figure 4.1: Delaunay triangulation (left) and Voronoi regions (right) on spherical surface [100].

The connection between Voronoi diagrams and Delaunay triangulation is that the circumcenters of the triangles in the Delaunay triangulation are the vertices of the Voronoi diagram. In the case of visibility calculation, a sub-satellite point (SSP) is a vertex of a Delaunay triangle on Earth's surface. Referring to Figure Figure 4.2, SSPs of three satellites are represented by points A, B, and C. It is assumed that all satellites cover the same area on the ground, which is a spherical cap with radius d. Point P is the centre of the circumcircle around ΔABC with radius r. Point P' is defined as the point that minimises the maximum of the distances to A, B, and C. In other words, P' is the point that is as close as possible to all three SSPs while keeping the furthest distance to any of them as small as possible. If P' is located within ΔABC, then P and P' are the same point. To ensure coverage of ΔABC, it is necessary that P' is covered, which implies $d \geq r$ [100].

This concept can be extended to a constellation, resulting in multiple triangles with radius r_j. It is only necessary to compute points P'_j and radii r_j to determine whether full coverage is achieved. If d exceeds the maximum radius of all the triangles, then the target region is fully covered [100]:

$$d \geq \text{max}\{r_1,\ r_2,\ \ldots,\ r_n\}$$

.

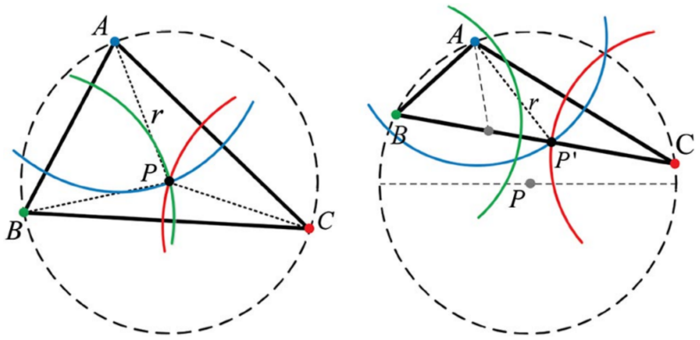


Figure 4.2: Circumscribed triangle [100].

The disadvantage of this method, however, is that it is only capable of determining whether the target area is fully covered by single-fold coverage. This implies that this method will not provide insight into the distribution of satellite visibility across latitude and longitude. However, given the analysis of large Walker constellations, coverage of significantly more than just one satellite is expected. Therefore, this method is not suitable for this project.

The time complexity of this method is $\mathcal{O}(T \cdot N)$, where T is the number of time instances and N is the number of satellites. It can be noticed that there is no dependency on the grid points on the ground [98].

2D Map

This method of coverage estimation, proposed by Gong et al. [98], is based on 2D map analysis. It is reported to be computationally faster than the one based on Voronoi diagrams, as shown in Figure 4.3, and it can estimate multi-fold coverage. Moreover, it can be applied to multi-shell constellations by computing the coverage performance of each shell separately and then stacking them together.

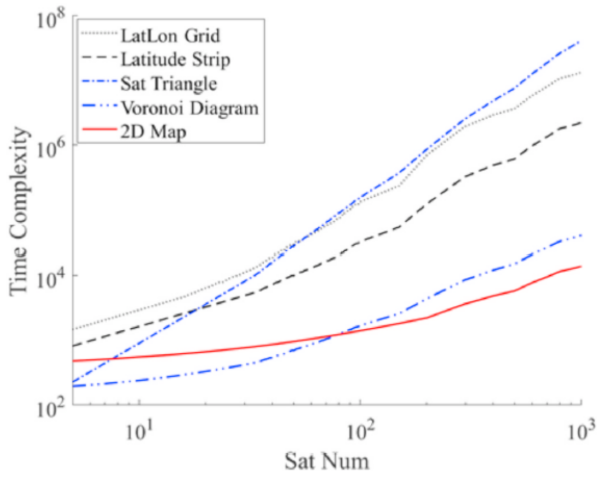


Figure 4.3: Computational performance comparison of 2D map-based method with other methods [98].

The method is developed specifically for Walker-Delta constellations and takes advantage of their symmetry. A detailed description of the method is not presented here, and interested readers can refer to the original work [98].

The disadvantage of this method is that it can only be applied to Walker Delta constellations, due to their symmetrical properties. The visibility of Walker Star constellations, which also comprise mega-constellation shells, cannot be estimated as they are not symmetrical in longitude. Additionally, while the gain in computational time is very significant, the underlying mathematics appears quite complicated and could take too long to implement. Given the time constraints of the project, it was decided not to proceed with this method.

The time complexity of this method is $\mathcal{O}(M_{\text{lat}} \cdot N)$, where M_{lat} is the number of grid points in latitudinal direction, and N is the number of satellites [98].

Selected Method: Grid Point

The decision to implement a purely numerical computation of visibility, based on grid points and satellite positions at each time step, was driven by three primary factors:

1. **Accuracy**
   Among the methods compared in Figure 4.3, this approach offers the highest level of accuracy. While the latitude strip method provides an averaged visibility across longitude, Voronoi diagrams only indicate if an area is entirely covered, and the 2D method assumes uniform satellite distribution within a Walker constellation, the Grid Point method avoids any simplifying assumptions within the calculations. This ensures accurate results even with higher fidelity models of constellation propagation.
2. **Universality**
   This method can be applied to any type of orbit, provided that the visibility computation is based solely on the minimum elevation angle. It does not necessitate the satellites to have the same Earth central angle (like Voronoi diagram method) or to be in a Walker constellation (like 2D Map method), thereby allowing its application to other domains (e.g., Earth observation) and having satellites at different altitudes.
3. **Ease of Implementation**
   The calculation of visibility using this method is straightforward, simplifying software implementation and reducing development time.

Furthermore, the implementation of this method in the `mcdo` module will be described. The angle γ (refer to Figure 4.4) between two vectors, $\vec{OG}$ and $\vec{GS}$, is computed from their dot product. The vector $\vec{OG}$ represents the position of a grid point G with respect to the Earth's center O, while $\vec{OS}$ represents the position of a satellite S with respect to O.

$$\vec{GS} = \vec{OS} - \vec{OG} \tag{4.1}$$

Furthermore, the angle γ is computed:

$$\vec{OG} \cdot \vec{GS} = |OG| \cdot |GS| \cdot \cos\gamma \tag{4.2a}$$

$$\gamma = \arccos \frac{\vec{OG} \cdot \vec{GS}}{|OG| \cdot |GS|} \tag{4.2b}$$

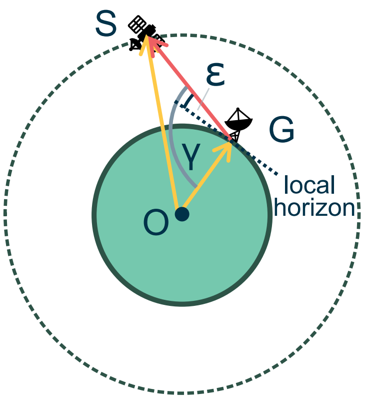


Figure 4.4: Schematic representation of elevation angle computation.

The elevation ε is then computed consequently:

$$\varepsilon = \gamma - 90^\circ \tag{4.3}$$

A satellite is considered in view (visible) if the following condition is met:

$$\varepsilon \geq \varepsilon_{\text{min}} \tag{4.4}$$

This method allows for the visibility of each satellite to be determined independently of others. This means that given the positions of a single grid point G and one satellite S, the visibility of S from G is not influenced by, nor does it influence, other visibilities. This allows for each visibility estimation to be performed independently. Given that all these estimations can be computed independently, it suggests that these operations can be parallelised, implying that they can be executed simultaneously.

4.2. Parallelised Visibility Computation

As demonstrated in the preceding section, the Grid Point method for visibility computation is inherently parallelisable. This parallelisation has the potential to significantly reduce computational time, thereby allowing for better optimisation. More layouts can be analysed within a shorter timeframe if elevation evaluations are executed concurrently.

However, typical CPUs have only a few cores (4 or 8), which limits the benefits of single-CPU parallelisation. Therefore, alternative acceleration methods using multi-CPU and GPU are explored. These hardware types are more suitable for the Single Instruction Multiple Data (SIMD) model, which is applicable to the Grid Point method for visibility computation. Appendix B describes the concepts of parallel computing and how they relate to the Grid Point method. This section focuses on the actual implementation of parallelised visibility computation and discusses the hardware and software choices made during the development and implementation process.

4.2.1. Hardware Setup

Computational speed depends not only on methods used but also on the hardware. Obviously, more powerful hardware will reduce computational time. The problem, however, can be is that development turns out to be more complex on such devices. Thus, selection of the hardware is a trade-off between availability, development, and potential gain in computational time. Since the project is a collaboration between TU Delft and ESOC, the author has access to resources of both. However, they operate in different environments, thus, it is better to select just one.

TU Delft has a supercomputer with 11424 CPU cores and 204800 CUDA cores[1]. However, only 12% of these resources are allocated to the aerospace engineering faculty. This means that around 1370 CPU cores and 24600 CUDA cores are shared among all the faculty researchers, including professors, postdocs, PhDs, students and others. Students are usually limited to one CPU node with 48 cores for up to 24 hours. Moreover, the waiting times after submitting jobs can range from seconds to days, which reduces the efficiency of the work. Finally, the supercomputer uses the workload manager Slurm, which requires some training (and time) before using it.

ESOC also has machines with multi-CPU and GPU capabilities. Most of the time, at least one machine is available for running simulations with multiple cores, which means that there is no need to wait in a queue. Additionally, there are several machines with low-end GPUs (NVIDIA Quadro P2000) that are often idle. Although these GPUs are slower than high-end ones, they can be used for development and testing. There is also a high-end GPU (NVIDIA Tesla V100) with 5120 cores that can be used for more demanding tasks. The ability to run jobs immediately makes the development process more efficient.

Table 4.1: Computational resources available at ESOC.

	mash2	mash6	masd1	masd2	dclb1	dclb3	dclb5
CPU	48 cores	64 cores	48 cores	112 cores	8 cores	8 cores	8 cores
GPU	-	-	-	5120 cores	1024 cores	1024 cores	1024 cores

Therefore, it was decided that ESOC's computational resources were to be used for this project. Namely, machine *dclb1* was primarily used for development and analysis as it showed to have sufficient resources for the needs of the project. Machine *masd2* was primarily used for benchmarking, and not so much for computations. This is due to slower CPU on masd2, cumbersome data transfer between machines, and high occupancy of this machine as it is used by other members of the section.

Table 4.2: Computational resources available at ESOC.

	CPU	GPU
dclb1	Intel(R) Xeon(R) W-2125 CPU @ 4.00GHz	NVIDIA Quadro P2000
masd2	Intel(R) Xeon(R) Platinum 8176 CPU @ 2.10GHz	NVIDIA Tesla V100-PCIE-16GB

4.2.2. Software Implementation

The software for this project was primarily developed in Python. The motivation for this choice is discussed in Section 3.3. This section will provide a description of how the software was implemented and why certain choices were made.

Figure 4.5 shows the general approach to coverage estimation, which is based on visibility computation. This approach can be divided into four parts:

[1] https://doc.dhpc.tudelft.nl/delftblue/, accessed 14 May 2024

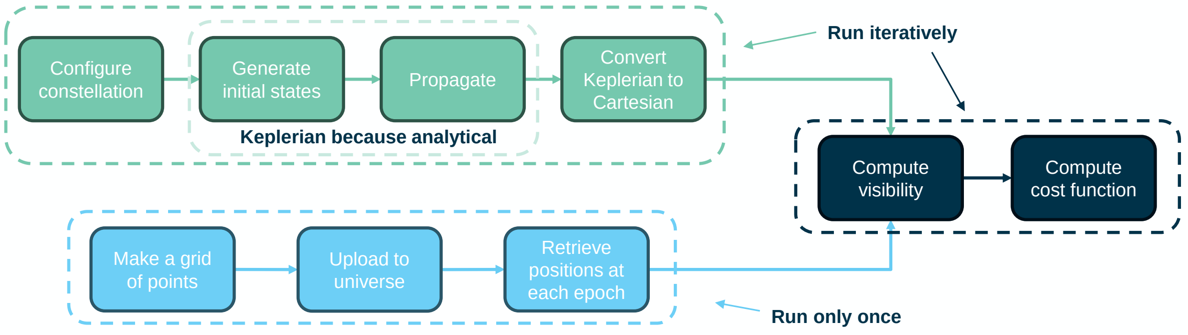


Figure 4.5: Approach to optimisation process of a mega-constellation based on coverage estimation.

1. **Constellation propagation**
   Constellation is configured and propagated analytically, with an option of J2 perturbation.
2. **Retrieving positions of grid points**
   Given that the method is purely numerical, the points used for coverage estimation can be arbitrary. They do not have to form a uniformly distributed grid, nor do they need to be located on the ground or be stationary (e.g., representing airborne aircraft). The retrieval of positions can be implemented by the user. While there are helper functions in the software, they assume that the points are stationary and rotate with the Earth. The positions must be in the ICRF reference frame with the origin at the Earth's centre.
3. **Visibility computation**
   Visibility computation is largely delegated to the GPU. However, some tasks are still executed on the CPU. This will be described in more detail in Section 4.2.2. For a large constellation and high grid resolution, this is the most demanding task and constitutes the main bottleneck of the optimisation process.
4. **Cost function estimation**
   The cost function is defined by the user but is typically less demanding than visibility computations. To be precise, cost function estimation is not a part of the `mcdo` module. Thus, it is entirely up to the user how this function is computed.

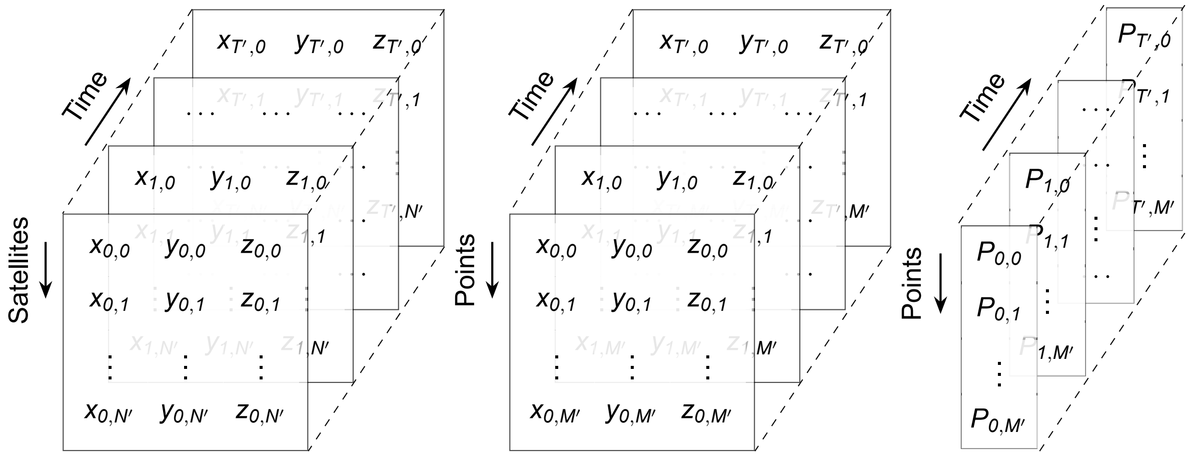
As shown in Figure 4.5, a typical optimisation process would involve reconfiguring the constellation by adjusting its parameters and then recomputing the visibility and cost function for the new layout. If one assumes that the grid points and their positions remain unchanged for each iteration, then the positions can be retrieved only once and stored in memory. All subsequent iterations would not require recomputing point positions, as it is a quite time-consuming process.

The constellation propagation in `mcdo`, implemented using NumPy, is most efficient when all time instances are propagated "in one go", as this approach minimises the overhead associated with transitioning between low-level C and Python. Therefore, in terms of computational time, it is more efficient to propagate all satellites for the entire propagation period first, and then compute visibility. This is preferable to computing visibility at each time step, which would necessitate the introduction of for-loops, thereby reducing the efficiency of the code.

Consequently, a function was developed for the visibility computation in `mcdo` that accepts two arrays – representing the positions of satellites and grid points – as input, and computes the visibility at each grid point, i.e., the total number of satellites in view from each point at each time instance.

As depicted in Figure 4.6, both input arrays are three-dimensional, containing the positions of N satellites ($N' = N - 1$) and M grid points ($M' = M - 1$). The output array, which can also be considered three-dimensional with one dimension of length 1, stores the number of satellites P at each epoch. All three arrays share a "Time" axis of length T, representing discrete time steps from 0 to $T' = T - 1$.

Figure 4.7 illustrates the computation of visibility at each epoch t. For each grid point, an iteration over all satellites is required. During each iteration, the elevation angle ε is computed as previously described. If


Time
Satellites
Points
$x_{T',0}$ $y_{T',0}$ $z_{T',0}$
$x_{1,0}$ $y_{1,0}$ $z_{1,0}$
$x_{0,0}$ $y_{0,0}$ $z_{0,0}$
$x_{0,1}$ $y_{0,1}$ $z_{0,1}$
$x_{0,N'}$ $y_{0,N'}$ $z_{0,N'}$
$x_{0,M'}$ $y_{0,M'}$ $z_{0,M'}$
$P_{T',0}$
$P_{1,0}$
$P_{0,0}$
$P_{0,1}$
$P_{0,M'}$

(a) Input array of satellite positions **(b)** Input array of grid points positions **(c)** Output visibility array

Figure 4.6: Input and output

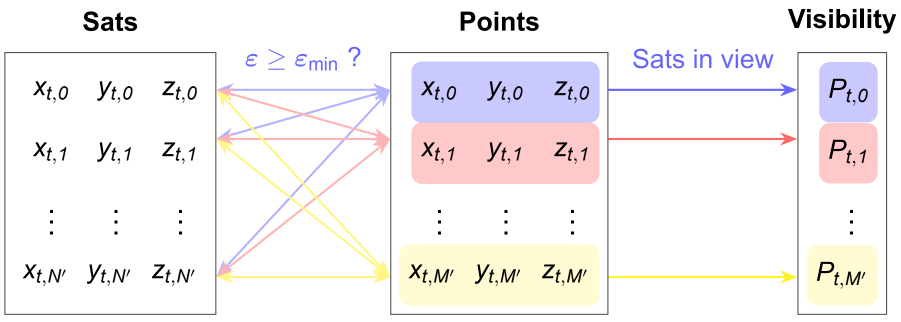

Sats
Points
Visibility
$\varepsilon \geq \varepsilon_{\min}$?
Sats in view
$x_{t,0}$ $y_{t,0}$ $z_{t,0}$
$x_{t,1}$ $y_{t,1}$ $z_{t,1}$
$x_{t,N'}$ $y_{t,N'}$ $z_{t,N'}$
$x_{t,M'}$ $y_{t,M'}$ $z_{t,M'}$
$P_{t,0}$
$P_{t,1}$
$P_{t,M'}$

Figure 4.7: Calculation of number of satellites in view for a single time instance

a satellite is in view, $+1$ is added to the output array, i.e. $P_{i,j} \mathrel{+}= 1$. The colours in Figure 4.7 correspond to the same grid points. The numbers $P_{i,j}$ in the output array indicate the number of satellites in view at the given time instance i and point j. A limitation of this approach is that it does not provide information about which specific satellites are in view – only their count.

The evaluation is performed for each epoch, as illustrated in Figure 4.6c. A key advantage of this operation is that each evaluation is independent of the others, enabling parallel execution. This parallelisation can significantly accelerate visibility computation as multiple processes perform this calculation simultaneously. In fact, such operations are called *embarrassingly parallel* as they have no dependence one each other [101].

The primary computational gain for visibility computation was achieved by utilising a GPU, which is an ideal candidate for embarrassingly parallel operations. The code was developed using the C/C++ CUDA API. However, given that the `mcdo` module is written in Python, an interface between the two languages was necessary. For this purpose, *pybind11* was used, a C++ module that provides seamless interoperability between C++ and Python [102]. Figure 4.8 illustrates the visibility computation algorithm and the type of hardware each subprocess is handled on. The functions depicted in blue are already implemented in either CUDA or *pybind11*, and thus, cannot be modified.

The GPU kernel (indicated in yellow in Figure 4.8) was authored by the writer of this report as part of this thesis work. For launching the CUDA kernel, a three-dimensional grid and block sizes were used, with one dimension corresponding to the time axis, another to satellites, and the third to grid points. This approach was chosen as it simplifies data indexing and the retrieval of corresponding values from arrays. Understanding how this function was implemented helps in later analysis of computational time and occurring phenomena.

```
// Number of blocks and threads
```

```
dim3 numBlocks(128, 128, 64);
dim3 threadsPerBlock(16, 16, 4);

// Launch kernel
VisibleSats<<<numBlocks, threadsPerBlock>>>
(dev_sats, dev_points, dev_visibility, N_steps, N_sats, N_points, min_elevation);
```

It is important to note that values for variables `numBlocks` and `threadsPerBlock` were based on experience of other developers[2,3] and were not optimised further as the selected values showed already a huge decrease in computational time which was considered sufficient for the purposes of this study.

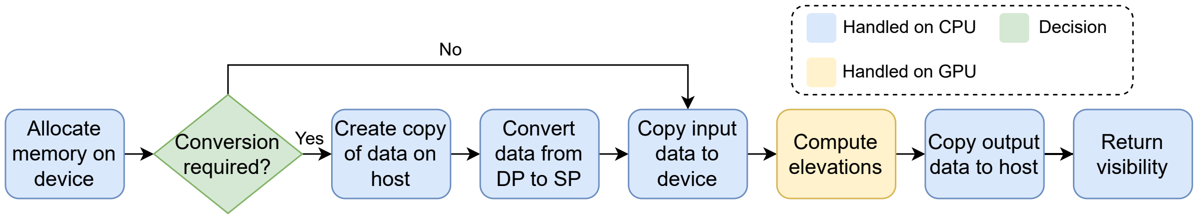


Figure 4.8: Function computing visibility broken down into subprocesses.

Visibility computation is performed with single-precision (SP) floating points (FP32), as low-end GPUs (such as Nvidia Quadro P2000) are designed to handle single-precision floating operations. They also support double-precision (DP) operations with FP64, albeit less efficiently in terms of computational time. A more detailed analysis of computational performance will be presented in Section 5.4. More modern GPUs are capable of handling double precision more efficiently. However, the analysis showed that the probability of an error is 10^{-5}, which implies that 1 in $100,000$ visibility computations with FP32 will differ by 1 with respect to calculations with FP64. This means that one entry in the output array (see Figure 4.6c) will be, for example, $P_{i,j} = 9$ or $P_{i,j} = 11$ for FP32 computation instead of $P_{i,j} = 10$ for FP64. This is considered acceptable for the purpose of this thesis.

4.3. Verification

Verification was done in multiple steps:

1. **Elevation angle computation with SP and DP, verified with Godot**
   For a single satellite and a single grid point, the elevation was computed at multiple epochs for both single and double precision. It was verified that `mcdo` computes elevation angles with acceptable accuracy for both precision types.
2. **Visibility at one point with multiple satellites, verified with Godot**
   For a single grid point, the visibility of multiple satellites was computed using `mcdo` and Godot. This verified that all satellites computed to be in view with Godot were also visible with `mcdo` computation.
3. **Visibility at multiple points, verified with Godot**
   The number of points was increased to ensure accurate computation across a set of points. This was also verified with Godot.
4. **Average and minimum visibility over propagation period, verified with [51] and [98]**
   Final verification of visibility computation for single and multi-shell constellation layouts was conducted using prior research published by Jia [51] and Gong [98]. This verified that the visibility of entire constellations over a globally distributed grid of points was correctly computed.

As depicted in Figure 4.9, the mean visibility computed by `mcdo` closely matches that computed by Gong [98]. Minor variations between the curves can be attributed to slight misalignment of the background image, which occurred because the curves were superimposed on an image sourced from

[2]https://forums.developer.nvidia.com/t/how-to-choose-how-many-threads-blocks-to-have/55529, accessed 14 May 2024
[3]https://stackoverflow.com/questions/4391162/cuda-determining-threads-per-block-blocks-per-grid, accessed 14 May 2024

Gong's work [98]. Additionally, the minimum visibility is similar to that of Gong, with some minor deviations due to differences in propagation settings and grid settings. Gong employed a higher fidelity propagation model, a higher resolution grid, and a longer propagation period. Despite these differences, the final comparison shows that the results are remarkably similar, verifying the use of mcdo for computation of visibility of single shells.

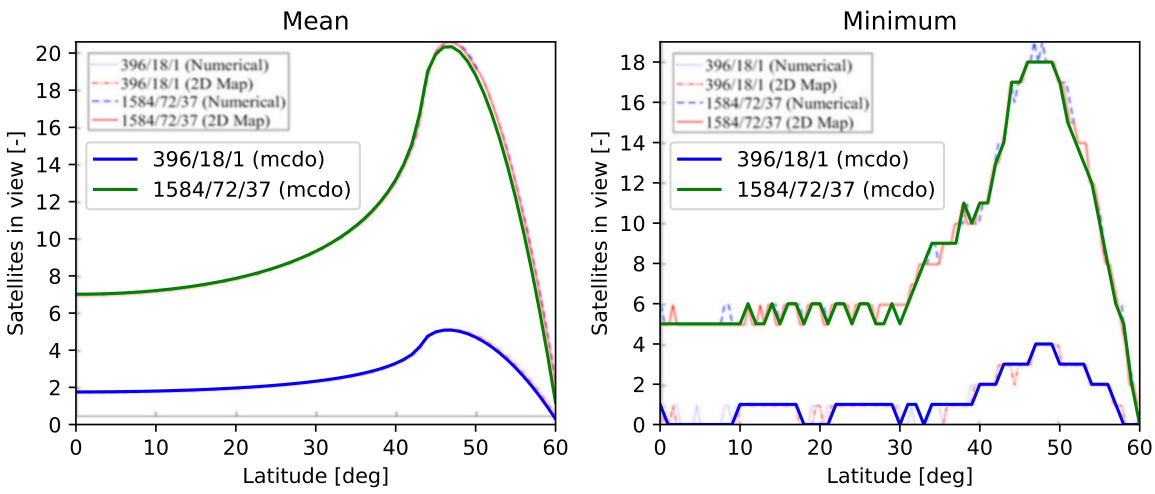
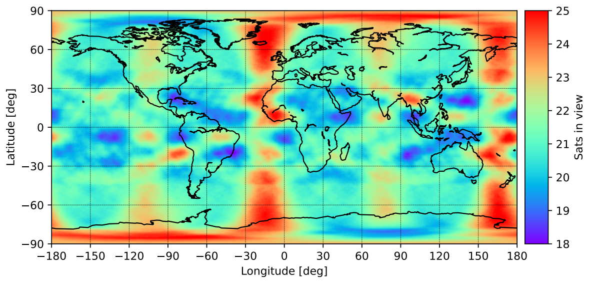


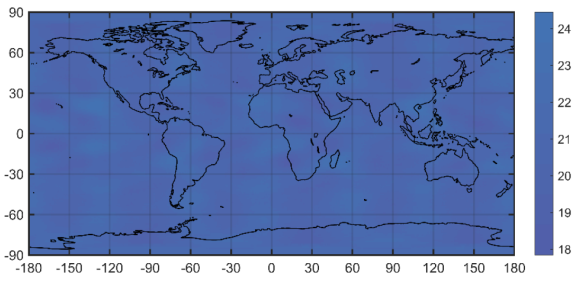
Figure 4.9: Verification with 2D map method [98].

Jia's [51] mega-constellation design can be used for verification of visibility computation for multiple shells. Unfortunately, only the number of satellites *N* and planes *P* are specified for the final design, while the phasing parameter *F* is omitted. As will be demonstrated in Section 4.5, *F* can significantly impact visibility. Therefore, it is anticipated that the results may vary.

The comparison of visibility is shown in Figure 4.10. Unfortunately, the invariant colour choice in [51] does not allow for a proper visual comparison. Nevertheless, it is noted that the number of satellites in view is "*18∼25, with little variation in the number of duplicates covered across global regions*" [51]. It can be observed that visibility computed with mcdo is very close to the 18-25 range, with some variations, possibly due to the unspecified *F*.



(a) Visibility of final multi-shell mega-constellation layout presented in [51] computed with mcdo.



(b) Visibility of final multi-shell mega-constellation layout presented in [51].

Figure 4.10: Verification of visibility computation for multi-shell mega-constellations. Propagation time $T = 96.69$ min. Grid resolution: 1° × 1°.

It is important to clarify that the objective of this verification was to calculate the mean visibility over time for the constellation presented in [51], given the same input conditions. The simulation period was limited to a single orbital period, $T = 96.69$ min (same as in [51]). For a constellation comprising so many shells, this duration is insufficient for the visibility to even out in the longitudinal direction. If the simulation period were extended appropriately, there would be no variation in the longitudinal direction.

Despite the short propagation period and the colours being difficult to distinguish, the visibility numbers of 18-25 satellites in view mentioned in [51] align with those obtained with `mcdo`. This comparison allows to conclude that the result is acceptable and the visibility computation with `mcdo` is verified.

4.4. Simplifications

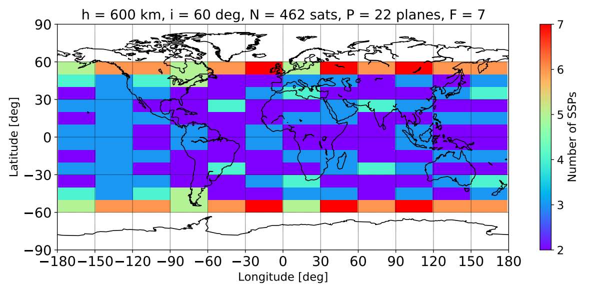
The computational time of visibility computation can be significantly reduced by applying simplifications. Given the high number of satellites in similar orbits, it is anticipated that certain assumptions can be made, which would greatly enhance the efficiency of calculations. Naturally, this will compromise accuracy. Therefore, the objective of this section is to determine the extent to which simplifications can be applied while staying within acceptable accuracy requirements.

4.4.1. Walker Constellation Symmetry

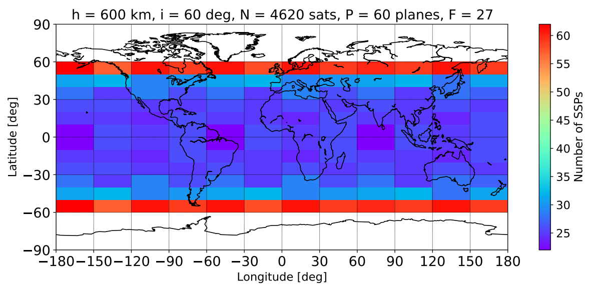
Walker constellations are known for their symmetry, as both planes and satellites within planes are evenly distributed. This symmetry allows for the assumption that large Walker constellations exhibit nearly identical distributions in the Northern and Southern hemispheres of the Earth. In the case of planes, their even distribution in longitude suggests that similar effects are observed at all longitudes, and only the latitude value is of significance.

North-South symmetry

North-South symmetry refers to the assumption that the coverage pattern in the Northern hemisphere is identical to that in the Southern hemisphere of the Earth. This assumption is derived from the fact that satellites constituting Walker constellations are evenly distributed within their respective planes. Also, the planes are evenly distributed along longitude. This results in the same geometric distribution of satellites in both hemispheres, which in turn implies that visibility of these satellites will also have the same distribution on both sides of the equator. Note that it does not mean that the satellite positions are mirrored with respect to the equator but rather that visibility statistics will be nearly identical for both hemispheres.



(a) $N = 462$ sats, $P = 22$ planes, $F = 7$


h = 600 km, i = 60 deg, N = 4620 sats, P = 60 planes, F = 27
Latitude [deg]
Longitude [deg]
Number of SSPs

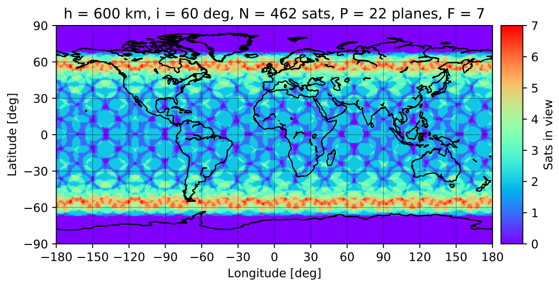
(b) $N = 4620$ sats, $P = 60$ planes, $F = 27$

Figure 4.11: Number of sub-satellites points across the globe for smaller and larger constellations for an arbitrary selected epoch.

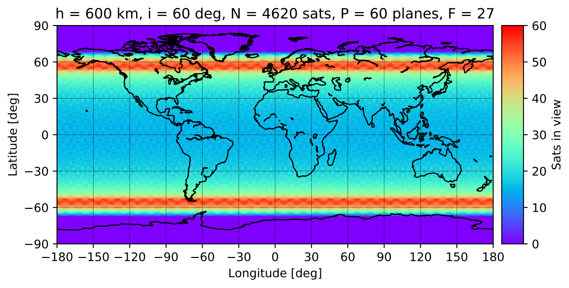
First, the distribution of satellites around the globe can be examined. Shown in Figure 4.11 are the distributions of sub-satellite points of a large Walker constellation in geodetic coordinates. It can be observed that the distribution of SSPs are identical around the equator. As if it is mirrored. This symmetry holds for an even number of planes, while for an odd number of planes or satellites it is not mirrored. However, the distribution is still similar.

Furthermore, instantaneous visibility for reference epoch t_0 is plotted in Figure 4.12. It can also be observed that North-South hemispheres have the same visibility distribution.

Finally, plotted in Figure 4.13 are visibility curves for selected longitudes at single time instance. It can be seen that those are distributed very similarly which confirms that North-South symmetry is a valid assumption for visibility computation of Walker constellations.



(a) $N = 462$ sats, $P = 22$ planes, $F = 7$


h = 600 km, i = 60 deg, N = 4620 sats, P = 60 planes, F = 27
Latitude [deg]
Longitude [deg]
Sats in view

(b) $N = 4620$ sats, $P = 60$ planes, $F = 27$

Figure 4.12: Number of satellites in view at each point across the globe for smaller and larger constellations at $h = 600$ km, $i = 60$ deg for reference epoch t_0. Grid resolution: 1° × 1°.

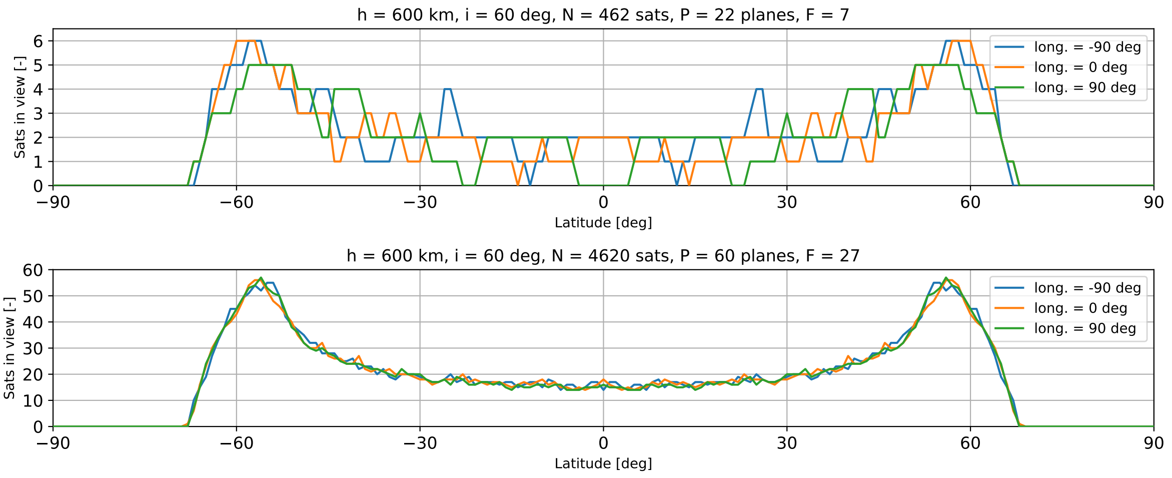


Figure 4.13: Visibility as a function of latitude for a number of selected longitude values.

Longitude averaging

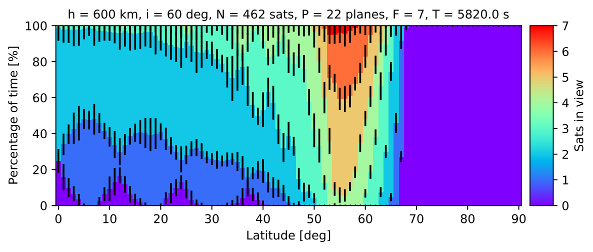
Walker constellations, by definition, distribute their planes uniformly around the globe in the longitudinal direction. This distribution allows for the assumption that visibility statistics over time will be consistent at each meridian line, regardless of longitude, if points on a single meridian line are considered.

This assumption, combined with the North-South symmetry, allows to consider grid points at a chosen longitude ranging only from 0 to 90 degrees. The range can be further reduced based on the mission objective and requirements. However, this approach requires a sufficiently long propagation period to identify potential coverage gaps. As a general guideline, a propagation time of 10 orbital periods is recommended.

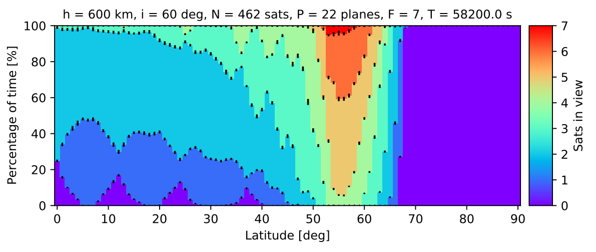
To verify the assumption of longitude averaging, comparisons were made between visibilities at various longitudes. This was achieved using a grid of 120 points in the longitudinal direction (3° resolution) and 91 points in latitude (1° deg resolution). For each meridian line, the distribution of visibility[4] over time was computed. Additionally, for each latitude value, the mean value of each fraction was computed, along with its variance.

Figure 4.14 depicts the visibility distribution with different colours corresponding to the numbers of satellites in view. The black bars correspond to the maximum range over all 120 meridian lines, i.e. difference between maximum and minimum values. A drastic drop in range can be observed when the propagation period is increased from one orbital period to ten full orbital periods. This observation verifies the assumption that a single meridian line can represent the entire globe, provided the propagation period is sufficiently long.

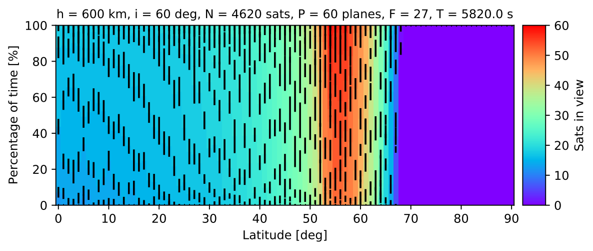
[4]The term *visibility distribution* refers to the fraction of time that a specific number of satellites is in view.



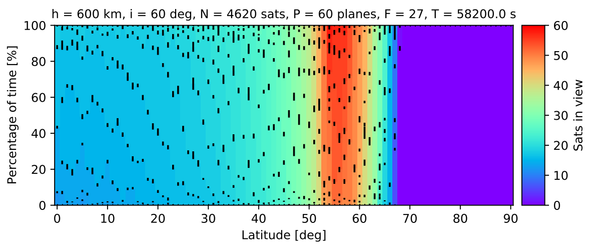
(a) $N = 462$ sats, $T = 5820.0$ s



(b) $N = 462$ sats, $T = 58200.0$ s



(c) $N = 4620$ sats, $T = 5820.0$ s

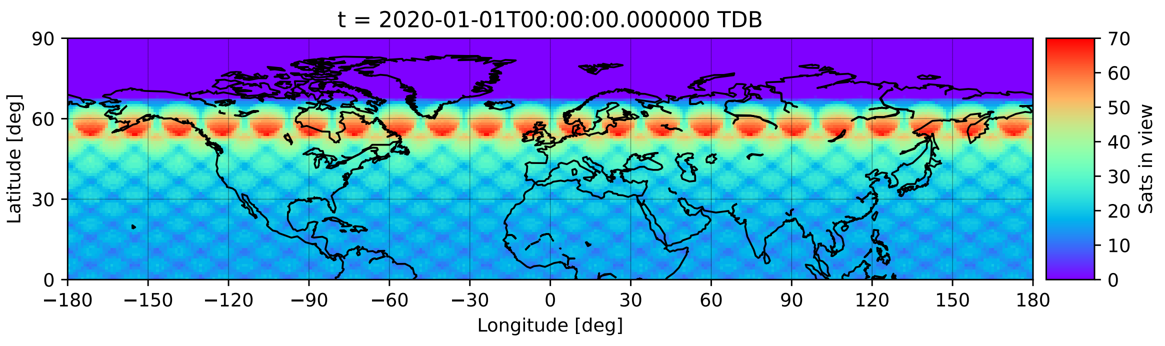


(d) $N = 4620$ sats, $T = 58200.0$ s

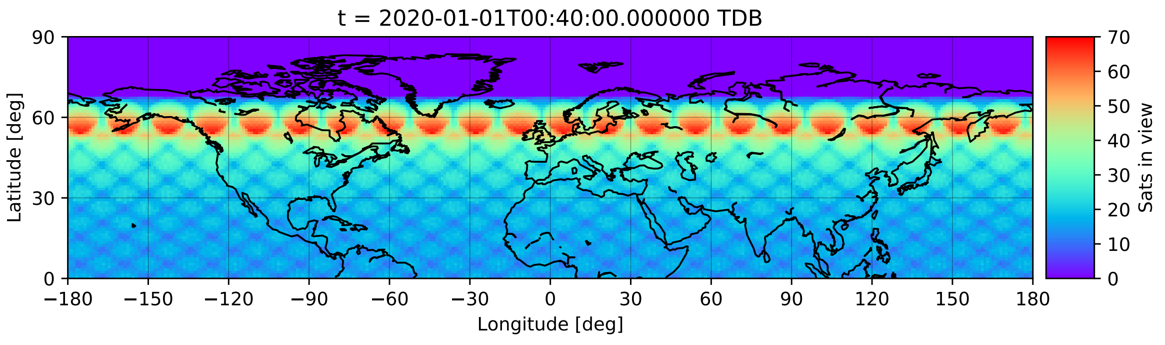
Figure 4.14: Distribution of visibility in time. Black bars correspond to the maximum range.

4.4.2. Walker Constellation Dynamics: Snapshot

A non-obvious observation was made during visibility analysis. When propagated in time, it can be seen that the global pattern of visibility does not significantly change. Instead, it rather drifts in Eastern direction, as shown in Figure 4.15, i.e. it shifts in longitudinal direction while in latitudinal direction the distribution stays more or less the same. Assuming that the whole pattern of the visibility does not change over time, one can analyse visibility distribution at a single time instance like taking *a snapshot*.



(a) Visibility at t_0.



(b) Visibility at $t_1 = t_0 + 40$ minutes.

Figure 4.15: Visibility of a mega-constellation at different epochs.

It was shown that due to symmetrical properties of large Walker constellations, the latitude range from 0 to 90 degrees can be used. For initial analysis, one can look at such snapshots of only one hemisphere. It can be just visibility at t_0 (initial state of the constellation) which means that the constellation does not even need to be propagated.

For propagation period of $T = 24$ hours ($\Delta t = 60$ s) and the number of satellites $N = 4620$ satellites, the following analysis was conducted to measure influence on accuracy due to the snapshot simplification:

Step 1: Compute the mean visibility for each epoch (i.e. each snapshot)
For each epoch the mean visibility was computed as a function of latitude. This results in 1441 arrays, each corresponding to the mean visibility.

Step 2: Compute the minimum visibility for each epoch (i.e. each snapshot) For each epoch the minimum visibility was computed as a function of latitude. This results in 1441 arrays, each corresponding to the minimum visibility.

Step 3: Compute the standard deviation of mean visibility for all epochs
With the mean visibility computed at each epoch, it was computed how visibility function varies by calculating the standard deviation. Based on 1441 epochs, the standard deviation was computed at each latitude value.

Step 4: Compute maximum difference in minimum visibility
For each latitude it was computed what is the maximum difference in minimum visibility to estimate how much minimum visibility can differ. Based on 1441 epochs, the maximum difference was computed at each latitude value.

Figure 4.16 displays the standard deviations and maximum differences in minimum visibility for different latitudes. The mean visibility has a low standard deviation, around 0.2 satellites, across all latitudes. This means that snapshot simplification is a good approximation for the mean visibility, which is typically a few tens of satellites.

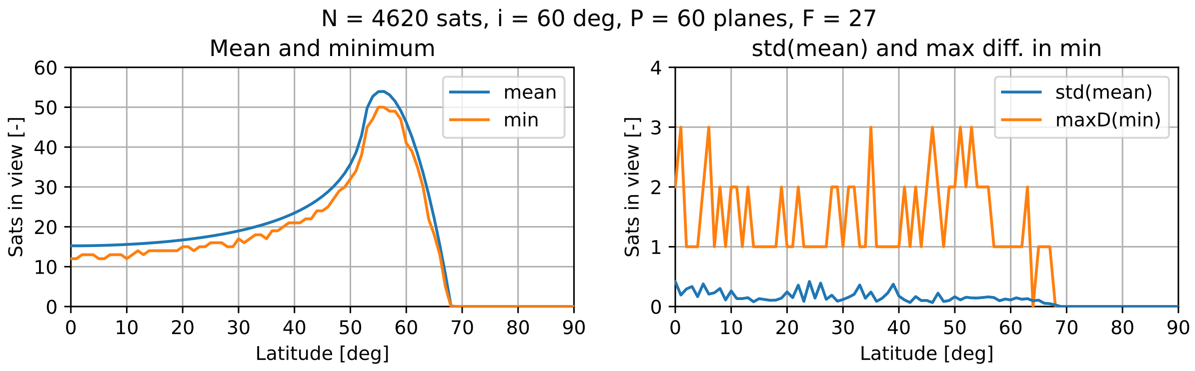


Figure 4.16: Analysis of visibility computation using snapshot simplification. Figure on the left shows mean and minimum values for the full simulation (all epochs and grid points). Figure on the right shows the standard deviation of the mean visibility and maximum difference in minimum visibility.

The minimum visibility has a higher variation as indicated by the maximum difference. This is partly due to the fact that the minimum visibility is an integer number, so any fractional difference is rounded up or down. Some latitudes have a constant minimum visibility over the entire propagation period, with the maximum difference of zero. Others have a maximum difference of up to 3 satellites in view, which may be significant or not, depending on the required fidelity of the analysis.

4.5. Parametric analysis

The visibility of a large Walker constellation is influenced by several parameters. These include the parameters defining the Walker constellation itself, and an additional parameter that defines visibility - the minimum elevation angle. To gain a deeper understanding of how these parameters affect visibility, a parametric analysis was conducted. In this analysis, visibility was computed as a function of a single parameter, while all other parameters were held constant.

This section presents the parametric analysis for the number of planes P, phasing parameter F, number of satellites N, inclination i, and minimum elevation angle ε with altitude h. Given the assumption of a circular orbit and the definition of visibility, h and ε can be expressed equivalently.

4.5.1. Number of Planes and Phasing Parameter

The number of planes P and the phasing parameter F influence the distribution of satellites within the constellation, which in turn affects coverage. The number of planes P is intrinsically linked with F, meaning they cannot be considered separately. Only the combination of the two defines the layout of satellites within the constellation. Therefore, it is interesting to examine how coverage is affected and what the optimal P and F are for given i, N and h.

The initial aim was to examine all possible parameter combinations. However, this was not feasible because each plane in a Walker constellation has the same number of satellites S, which implies that N must be a multiple of P. Finding $N-P$ combinations was not easy to automate and required manual adjustment. This was time consuming, so only 5 values of N were chosen. For each value N_i, a variation of ± 5 satellites was allowed, given the high number of satellites. Therefore, P and F were analysed with only a few selected h-i-N combinations. Despite this limitation, this approach proved efficient in reaching the conclusions discussed further.

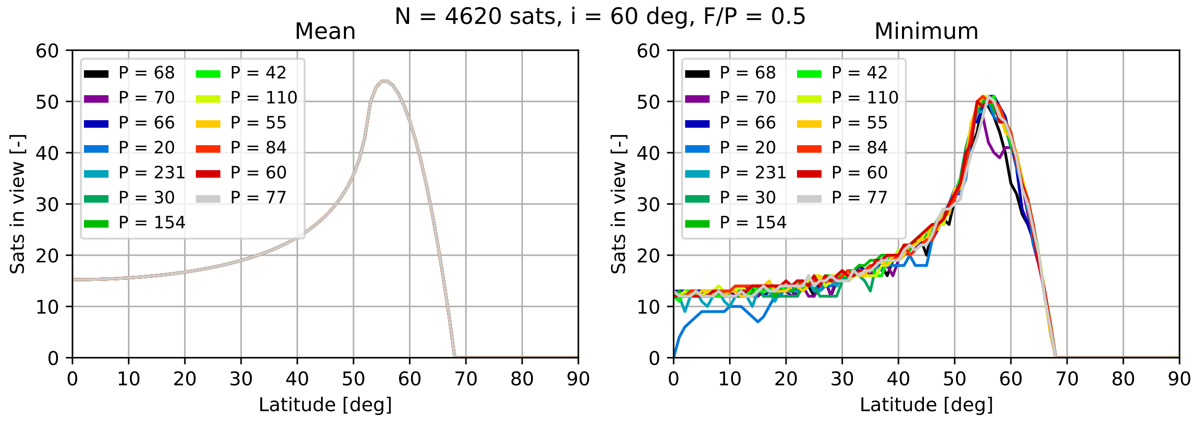


Figure 4.17: Mean and minimum visibility for the given constellation as a function of P with a fixed F/P.

Another limitation was the number of values for F that could be analysed. By definition of a Walker constellation, F must be an integer. This ensures that satellites of the first reference plane and the last plane have the same phasing difference as all other adjacent planes. The value for F ranges from 1 to $P-1$. After this, satellites of adjacent planes align, and the relative positions for $F=P$ are the same as $F=0$. However, since P can vary significantly for large constellations, so can F. To reduce the number of simulations, it was decided that for each selected P, F would be expressed as a fraction of P rather than an absolute value. Thus, the parameter F/P was selected as a representation of the effect of phase on constellation coverage. The corresponding values from $F/P=0.1$ to $F/P=0.9$ with a step of 0.1 were analysed.

In the initial analysis, the mean and minimum visibility are plotted as functions of P, with all other parameters held constant. Figure 4.17 illustrates the visibility for different layouts. The mean visibility of a Walker constellation appears to be independent of the satellite distribution within the constellation. However, the minimum visibility is significantly influenced by both P and F. Notably, Figure 4.17 reveals noticeable drops (e.g. for $P=70$ planes) in minimum coverage when the number of planes varies. Similar plots corresponding to other values of N can be found in Appendix C.1.

The initial assumption was that for a large number of satellites, the phasing would not affect visibility across longitude, as it would be smoothed out by the sheer quantity of satellites. This "smoothing-out" effect was not expected to occur at very low or very high values of P, as this would imply large gaps between satellite planes (for low P) or large gaps between satellites within one plane (for high P). Therefore, it was initially hypothesised that $P \approx \sqrt{N}$ would be an optimal or near-optimal solution, ensuring an even distribution of satellites around the globe. An optimal phasing parameter was considered to be $F/P = 0.5$, implying that satellites in adjacent planes cover the gaps between satellites within the same plane.

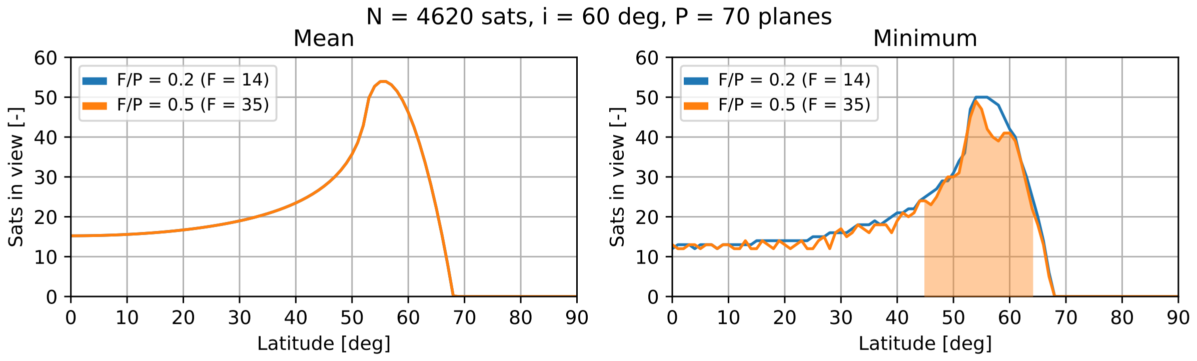


Figure 4.18: Mean and minimum visibility for the given constellation as a function of F/P with a fixed P. The shaded region is an example of the area under curve.

However, this assumption was disproven by the data in Figure 4.18, which shows a significant drop in minimum visibility for $F/P = 0.5$. The only difference between the layouts in Figure 4.18 is the phasing F between adjacent planes, clearly demonstrating the significant impact of this parameter on coverage.

To quantify this phenomenon, the area under the curve was calculated, as shown in Figure 4.18. The visibility curve peaks at a latitude of approximately $\varphi_{\text{peak}} = i - 5°$ for the given inclination i and minimum elevation angle ε. For non-polar constellations, the inclination is selected to optimise coverage not only at the target latitude but also at adjoining latitudes. Thus, the coverage of the latitude range $i - 15 < \varphi < i + 5$ was analysed.

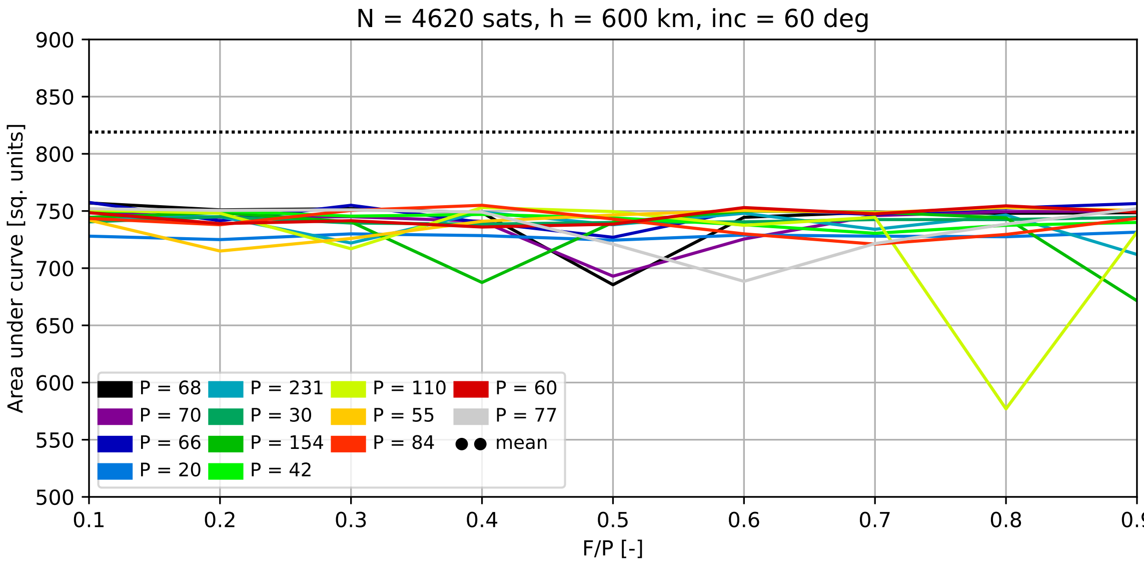
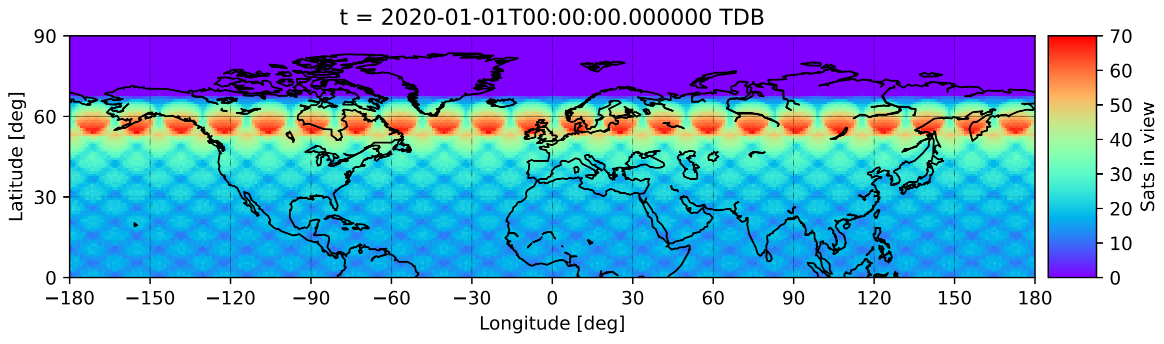
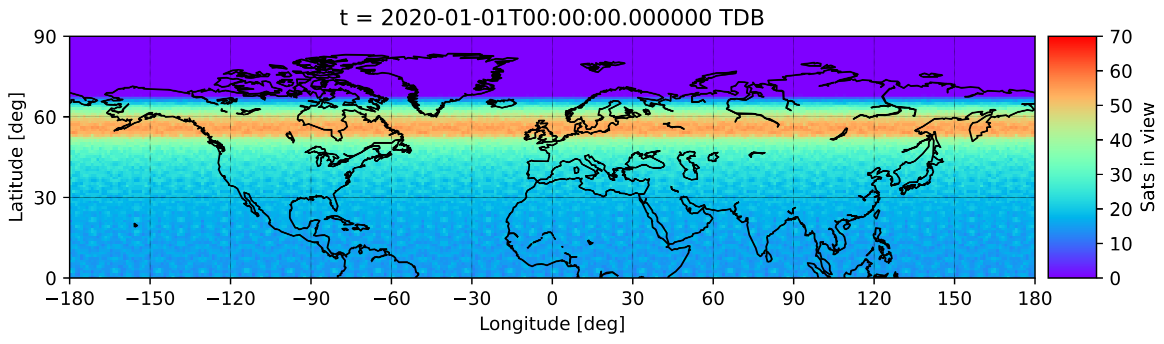


Figure 4.19: Area under visibility curve for different layouts.

This area was computed for various values of P and F/P. Figure 4.19 presents the area under the curve for various layouts. Several observations can be made from this plot. Firstly, the mean visibility appears to be unaffected by P and F/P, consistent with previous plots. However, this plot provides a more general demonstration of this phenomenon. Secondly, the plot clearly shows that the minimum visibility is highly dependent on the layout. Significant changes in minimum visibility can occur, as exemplified in Figure 4.19 for $N = 4620$ satellites, $P = 110$ planes, and $F/P = 0.8$. A substantial drop in the value of the area under the curve corresponds to low values of minimum visibility in the range of interest. The plots, however, do not readily reveal a trend for the minimum visibility. Outliers appear to emerge somewhat randomly, occasionally causing significant drops in the minimum visibility curve, as demonstrated in the aforementioned example. This is also observed with other layouts which can be found in Appendix C.1.



(a) $F = 88$.



(b) $F = 44$.

Figure 4.20: Visibility at reference epoch t_0 for $N = 4620$ satellites, $P = 110$ planes, $h = 600$ km, $i = 60°$.

To understand why these drops occur, visibility at t_0 is plotted for $N = 4620$ and $P = 110$ with $F = 88$ and $F = 44$. Figure 4.20 presents two layouts where only F is varied. For $F = 88$, areas of very high visibility and significantly lower visibility coexist at the same latitude. In contrast, for $F = 44$, the visibility distribution is more uniform, with less variation in the number of visible satellites. The effect of the phasing parameter F is illustrated in Figure 4.21, which clearly shows that while mean visibility remains unaffected by F, the minimum number of satellites in view and the maximum range of visibility are significantly influenced by the phasing parameter.

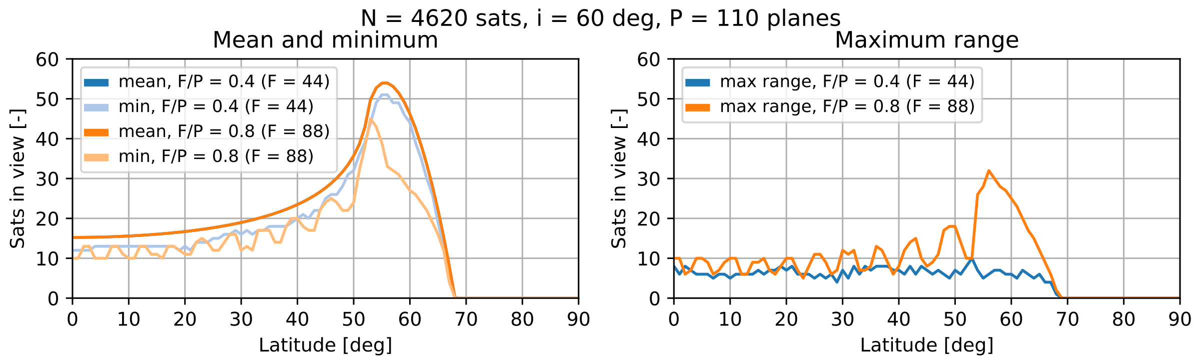
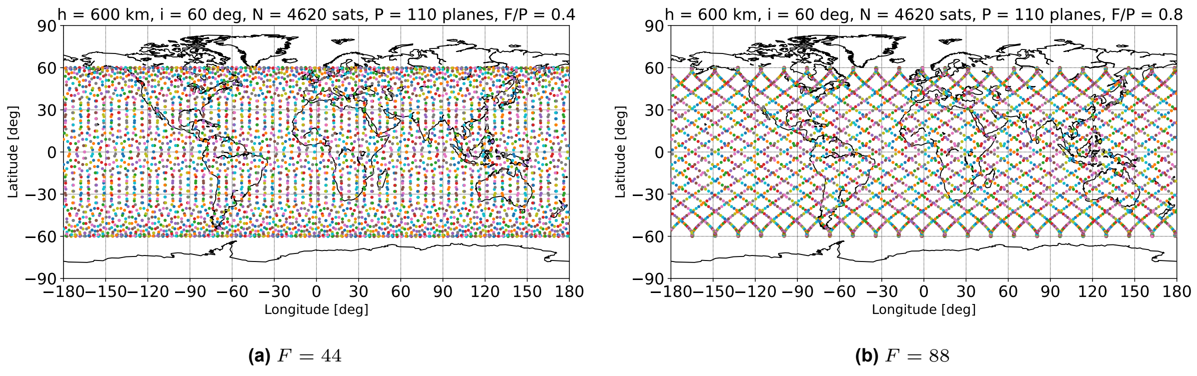


Figure 4.21: Visibility variation. Effect of F on visibility.

The reason for such variation becomes evident when the sub-satellite points are plotted. Figure 4.22 demonstrates how satellite distribution is affected by F. However, it remains unclear how to select optimal P and F to achieve a more uniform satellite distribution without constellation propagation.

When plotting mean and minimum visibility for various layouts in Figure 4.19, it can be observed that



h = 600 km, i = 60 deg, N = 4620 sats, P = 110 planes, F/P = 0.4
Latitude [deg]
Longitude [deg]
(a) $F = 44$
h = 600 km, i = 60 deg, N = 4620 sats, P = 110 planes, F/P = 0.8
Latitude [deg]
Longitude [deg]
(b) $F = 88$

Figure 4.22: Sub-satellite points at reference epoch t_0

the difference between the mean and minimum visibility is approximately 10%. This suggests that one can assume the minimum visibility to be approximately 10% lower than the mean. A key advantage of the mean visibility is its independence from P and F, as previously demonstrated.

Figure 4.19 also demonstrates that for a fixed N, there exists a $P - F$ combination that ensures no significant drops in minimum visibility if P is sufficiently high. Therefore, it can be assumed that for any given N (± 5 sats), optimal $P - F$ values can be found such that the difference between the mean and minimum visibility will be no more than 10%.

Given that constellation requirements are typically based on minimum visibility, a mission analyst could apply a 10% margin to the requirement and design for mean visibility instead. This approach would eliminate the dependence on P and F, while maintaining visibility close to the requirement. Initially, the optimal i and N would need to be determined for each shell. Subsequently, P and F could be optimised to enhance minimum visibility. This approach will be explored further in Chapter 5, which discusses the multi-shell constellation design.

4.5.2. Number of Satellites

The number of satellites N was varied from 500 to 5000 satellites in a single shell. The analysis of P and F revealed that the mean visibility remains unaffected by these parameters, and the minimum visibility is approximately 10% lower than the mean when distributed uniformly. Consequently, for a large Walker constellation with a number of satellites N, the focus was placed on the mean visibility to avoid unfavourable combinations of P and F that could result in substantial gaps in minimum visibility.

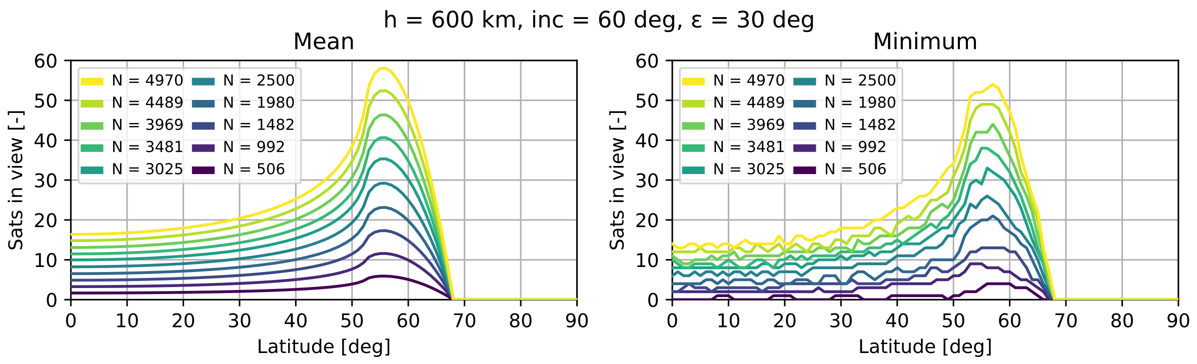


Figure 4.23: Parametric analysis, varying N.

Figure 4.23 shows that with changing N, the only parameter affected is the **scaling** of the mean visibility function. The highest latitude at which satellites remain visible, as well as the latitude of peak visibility, stay constant. Given that this is the mean visibility, it can be interpreted as if additional satellites form an independent shell. As the previous subsection demonstrated that the distribution of satellites is irrelevant for mean visibility, this comparison is valid. Since both shells are independent, it follows that the sum of their means will be equal to the mean of their sum. This suggests that the visibility distribution scales linearly with N.

Figure 4.24 presents the scaled visibility, where the visibility for a constellation with only $N_0 = 506$ satellites was computed numerically. For all higher values of N_i, the visibility was simply scaled with the ratio $\kappa_i = N_i/N_0$. The scaled visibility curves match the propagated curves identically. The difference between actual visibility and scaled visibility, shown in the right-hand plot of Figure 4.24, is negligibly small, leading to the conclusion that the mean visibility can be computed by simply scaling the visibility curve. This is also applicable to other layouts which can be found in Appendix C.2.

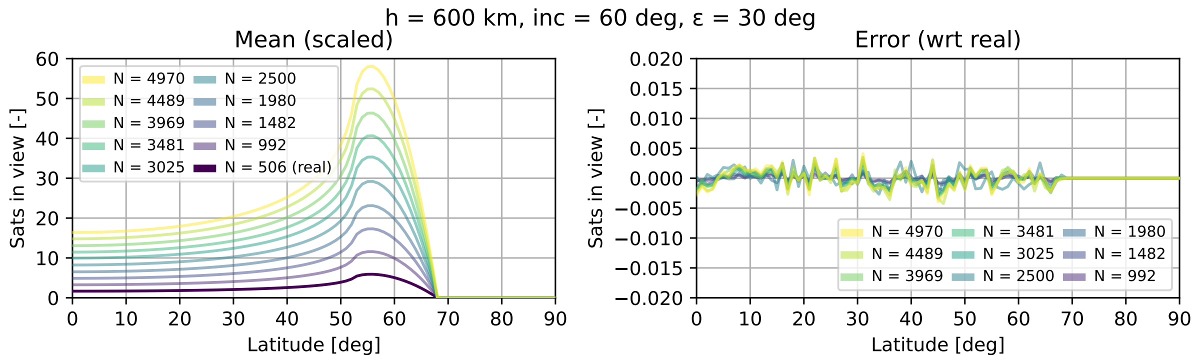


Figure 4.24: Mean scaled visibility and error with respect to actual simulation

This effect allows to avoid propagating and computing an excessively high number of satellites in one shell if the interest lies in the mean visibility. Instead, a shell with a lower number of satellites can be propagated with subsequent visibility computation. This approach will significantly reduce computational time.

Additionally, this effect implies that if a mega-constellation is designed for mean visibility initially, there is no need for repeated propagation of the constellation. Instead, the constellation can be propagated just once with a fixed N_0 and a set of i (e.g., ranging from 0 to 90 degrees), and then the visibility can be simply scaled to a desired N to meet the requirement. Chapter 5 will discuss how this method can be applied to constellation design and how the constellation design is affected by this method.

4.5.3. Inclination

The inclination is anticipated to influence visibility in a relatively predictable manner. Satellites cannot go beyond latitude $\varphi = i$, which implies that visibility cannot extend beyond $\varphi \leq i + \theta$. Figure 4.25 depicts the visibility curves for various inclinations, with all other parameters held constant.

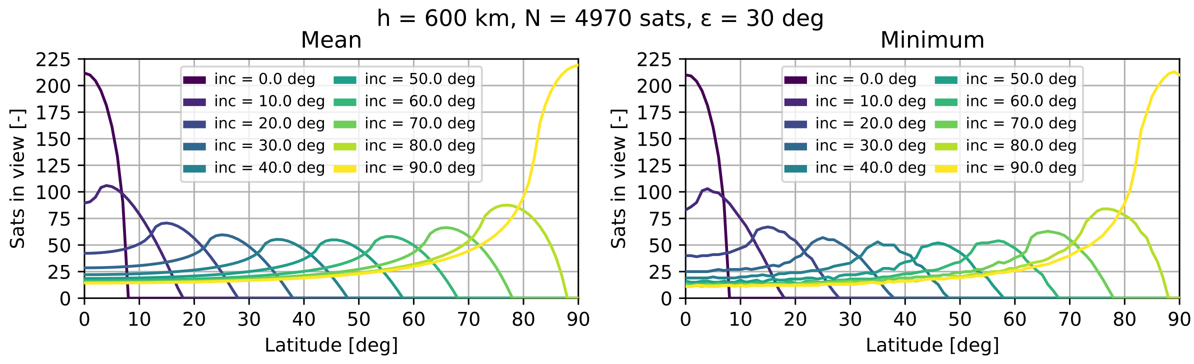


Figure 4.25: Parametric analysis, varying i.

The peak values for visibility are observed to decrease until approximately $i = 45°$ and then increase consistently. This can be explained by the higher concentration of satellites per unit surface area at low inclinations compared to mid-latitudes, where the surface area is still significantly large but satellites are more dispersed. At higher inclinations, satellites reach higher latitudes where the planet's surface area is much smaller, resulting in an increase in the number of satellites per unit surface area. Appendix C.3 shows that this is observed for other layouts where N and ε_{min} are varied.

At certain inclinations and minimum elevation angles, satellites cover the entire latitude range from 0 to 90 degrees. This is because satellites do not need to be directly over the pole to observe it, allowing them to see the pole and even the opposite side of the globe (relative to the pole) from their respective inclinations. This is why, among other reasons, Iridium and Oneweb have their satellites at 86.9 degrees and 87.9 degrees, respectively. This visibility feature helps to avoid significant congestion at the pole and results in a lower probability of satellite collisions.

As previously discussed, the visibility distribution for higher inclinations also covers lower inclinations. Therefore, in the case of a multi-shell constellation, it should be noted that the constellation design should commence from higher inclinations and proceed downwards. This is because lower-inclination shells are influenced by the higher inclination shells, while the reverse is not true. This will be discussed in more detail in Chapter 5.

4.5.4. Minimum Elevation Angle and Altitude

The minimum elevation angle is an important factor in the design of mega-constellations as it directly influences satellite visibility and hardware design. For example, a higher elevation angle means that a smaller portion of the sky, and hence fewer satellites, are visible. Furthermore, the minimum elevation angle requirement also drives the design of both the satellites and ground-based equipment such as gateways or user terminals.

The minimum elevation angle and altitude are interconnected, as they both contribute to the Earth's central angle. As observed from Equation (2.9), there are two variables in the Earth's central angle equation. This implies that for the angle θ to remain constant, the altitude h and minimum elevation angle ε_{min} must change proportionally.

While this section will explore the influence of the minimum elevation angle and altitude on visibility of mega-constellations, it is important to highlight that these parameters are not typically optimised during the design phase of the orbital layout. Rather, they are predefined parameters. These parameters are usually dictated by the constraints of hardware design or by regulatory requirements imposed by authorities such as the Federal Communications Commission (FCC) or the International Telecommunication Union (ITU).

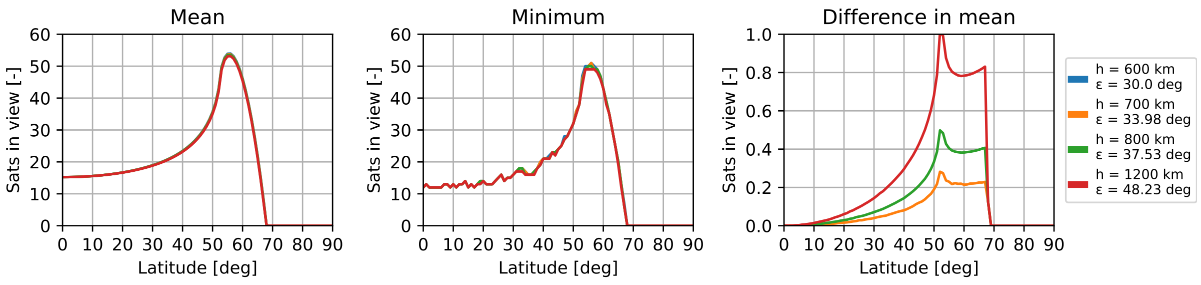


Figure 4.26: Visibility for fixed θ and varying h and ε_{min}. The rightmost image show the difference in mean visibility with respect to the reference distribution: visibility at $h = 600$ km and $\varepsilon_{\text{min}} = 30$ deg.

This is verified by the visibility curves depicted in Figure 4.26. If the angle θ remains constant, the visibility curves show insignificant change. A slight difference in the mean visibility can be seen in the rightmost image. However, these differences are considered negligible and do not contradict the overall trend. These differences can be attributed to numerical error, the Earth's oblateness, or varying propagation periods when expressed in orbital periods (the propagation period was the same $T = 24$ hours for each layout, which means that the number of orbits differs for all layouts).

This can be plotted for θ as a function of h and ε_{min}, as shown in Figure 4.27. This plot aligns well with Figure 4.26, demonstrating that the values of h given in the example correspond closely to ε_{min}. This confirms the previous assertion that the two are proportional, and thus the parametric analysis can be conducted on only one of them. For this report, h was held constant and ε_{min} was varied.

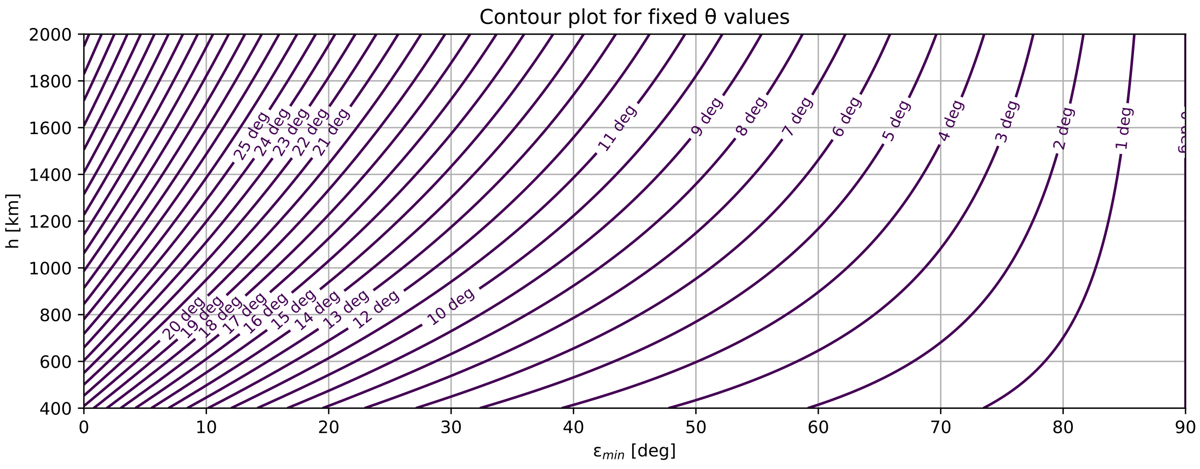


Figure 4.27: Contour plot for θ as a function of h and ε_{min}

Moreover, we can look at the visibility curves as functions of minimum elevation angles for a fixed h, as shown in Figure 4.28 (more figures for other layouts can be found in Appendix C.4). It can be observed that the visibility curve scales up with a decreasing elevation angle, which is expected as lower minimum elevation allows more satellites to be observed from the ground. More notably, however, the peak value of the visibility curve, along with the "bell", shifts more towards the equator, peaking at lower latitudes compared to higher ε_{min}. This can be explained by the fact that grid points at lower latitudes start seeing more satellites from higher latitudes, which increases their weight and shifts the distribution of visibility to lower latitudes. Conversely, the higher latitude points decrease their weight as their field of view expands to a lower number of satellites.

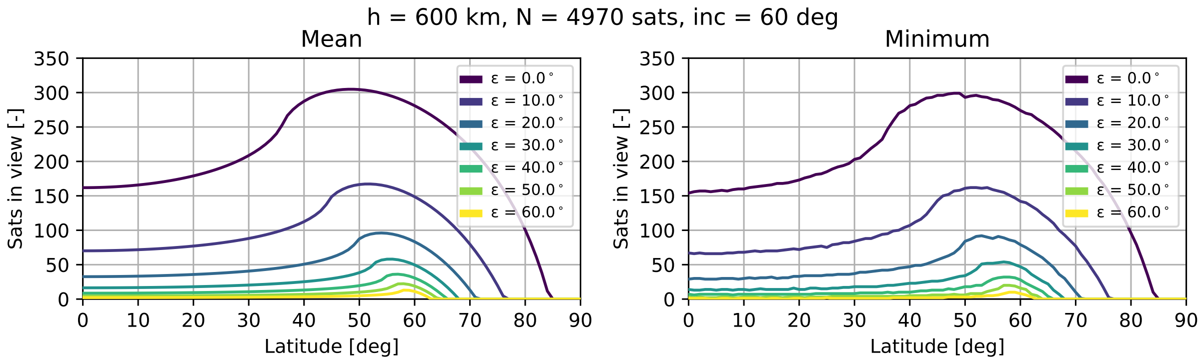


Figure 4.28: Parametric analysis, varying ε_{min}.

The figures show that the visibility curves for various minimum elevation angles scale non-linearly. This scaling is linked to the visibility cone, as observed from a ground-based observation point. As the minimum elevation angle decreases, the visibility cone expands, thereby capturing more satellites. Interestingly, this scaling also corresponds to the Earth's central angle. This correlation is logical, as the ground area, expressed by the Earth's central angle, is 2θ, as seen from Figure 2.5. This suggests that the maximum visibility scales in a similar manner to θ with changing ε_{min}.

However, we cannot use the exact formula for θ, as it corresponds to an angle, while our interest lies in the number of satellites in view. If this equation is scaled with a scaling coefficient C_1, as provided by Equation (4.5), to match the peak visibility value for the reference $\varepsilon_{\text{min, ref}} = 30°$, the scaled curve shows behaviour similar to the numerical data (see Figure 4.29).

$$y_{\text{scaled}} = C_1 \cdot \left(\arccos\left(\frac{R_e}{R_e + h} \cos \varepsilon_{\text{min}} \right) - \varepsilon_{\text{min}} \right) \tag{4.5}$$

Despite this, the difference in values remains significant for $\varepsilon_{\text{min, ref}} < 30°$. Therefore, it is not recommended to use this exact formulation of the equation.

There could be several reasons for this discrepancy. Firstly, satellites are not evenly distributed in the sky, implying that the number of satellites in view does not scale linearly with θ. Secondly, the simulation involves dealing with a 3D Earth model rather than a 2D one, adding complexity to the analysis. A more detailed quantitative analysis of this phenomenon is beyond the scope of this project, but it could serve as an intriguing topic for future research.

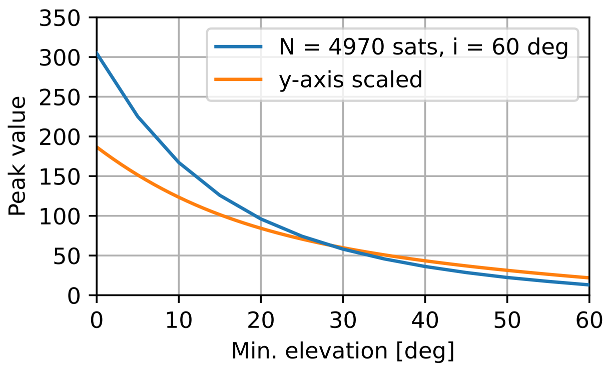


Figure 4.29: Peak visibility value as a function of minimum elevation angle ε_{min}

While the observations made may not seem particularly noteworthy, this analysis provides valuable insights into how visibility changes with variations in h or ε_{min}. Such insights could prove beneficial for mission analysts in the design of mega-constellation missions as it would allow to adjust parameters more efficiently in the process.

4.6. Summary and Conclusions

This chapter showed provided an overview of existing methods for visibility computation, and selected one for this thesis: Grid Point method. This method relies on elevation angle of satellites with respect to grid points on the ground. The advantage of this method is its versatility and straightforward implementation. The disadvantage of this method is its time complexity. However, this can be overcome by utilising GPUs due to the embrassingly parallel nature of this method. Visibility computation augmented by GPUs was implemented in the `mcdo` tool and verified with Godot and other methods proposed by other researchers. Furthermore, simplifications for large Walker constellations were analysed that accelerate visibility computation without compromising accuracy too much. North-South symmetry, longitude averaging, and snapshot simplifications were considered, and their accuracy and limitations were analysed. Finally, a parametric analysis of large Walker constellations was conducted to see the effect of parameters defining a Walker constellation on mean and minimum visibility.

Key Takeaways

- Embarrassingly parallel nature of the selected Grid Point method allows for GPU utilisation which substantially accelerates visibility computation
- Using data in FP32 format is sufficient for visibility computation
- This computation method can be applied to any number of satellites and any grid resolution but memory limitations must be considered
- Simplifications:
  - North-South symmetry is always used for Walker constellations due to their symmetry
  - Longitude averaging (grid points on a single meridian line) is accurate if the propagation period is sufficiently long (at least 10 full orbital periods is recommended)
  - Snapshot simplification does not require propagation but does require a grid over the entire globe
- Parametric analysis of large Walker constellations:

  - P and F do not affect mean visibility but may have a significant effect on minimum visibility
  - The minimum visibility is generally 5-10% lower than the mean visibility
  - The visibility of a shell spans from the equator and peaks at latitude values slightly below the inclination value
  - Mean visibility scales linearly with the total number of satellites
  - Altitude and minimum elevation angle can be expressed equivalently via a single parameter: Earth central angle

Limitations and Recommendations

- Implementation of visibility computation provides only the number of satellites in view without mentioning what exact satellites are visible at each grid point.
- Current implementation assumes that the satellite antenna is pointing nadir and its field of view can be express equivalently with the minimum elevation angle. In the future, satellite's field of view and antenna pointing offset can be included.
- Visibility can be expressed as a continuous function of latitude, inclination, and Earth central angle with the use of an interpolator.
- The tool can also be used for Earth observation but this would require a very fine grid due to a very narrow camera/sensor angle of EO satellites payload.

5 Multi-shell Mega-constellation Design

Building on our understanding of large Walker constellations that form mega-constellations, it is of interest to explore efficient strategies for their combination. The primary objective is to minimise the cost function within a reasonable timeframe.

Insights from Chapter 4 show that the average visibility remains unaffected by the number of planes P and the phasing parameter F. Consequently, the strategy adopted is to first focus on the average visibility and subsequently optimise the minimum visibility. This implies that in the initial phase of the optimisation process, P and F will not be considered as design variables.

Generalising methods for constellation design optimisation can be challenging. Therefore, it is helpful to define a test case for design and optimisation. For this project, we will focus on a constellation intended for broadband connectivity, which would need to provide coverage in Europe. The requirements for this study case are summarised in Table 5.1 and explained below.

Table 5.1: Requirements for a study case.

Parameter	Requirement	Rationale
Latitude	35-70 deg	Europe latitudes
Sats in view	Minimum: 50 at all times	Large number that will help test the limits of the `mcdo` tool and methods
Altitude	700 km	Avoid atmosphere drag, avoid overly populated region of 700-950 km, easier spacecraft disposal
Min. elev. angle	30 deg	Typical value for broadband mega-constellation systems

The visibility requirement is to maintain at least 50 satellites in view at any given moment (minimum visibility) within the latitude band between 35 and 70 degrees, corresponding to Europe. The requirement of 50 satellites stems from the fact that fulfilling this requirement will necessitate several thousands of satellites. This will help in testing the `mcdo` tool and new methods, pushing them to their limits with such a high number of satellites.

As demonstrated in Section 4.5, the minimum visibility is typically 10% lower than the average visibility. Consequently, it is decided to transform the minimum visibility requirement into an average visibility requirement, setting it at an average of 55 satellites in view. Due to this approach of the effects of P and F on the design will be avoided. Once the optimal shells are identified, P and F for each shell can be optimised based on the minimum visibility requirement. Due to the time constraints of this thesis, finding optimal P and F is left as a recommendation for the future work.

The minimum elevation angle and altitude are considered fixed and are thus not subject to optimisation. The altitude is set at 700 km, and the minimum elevation angle is 30 degrees. These are typical values for broadband constellations. While the altitudes shown in Section 2.2 are slightly lower (Kuiper, Starlink, Kepler) or substantially higher (OneWeb), the rationale for a 700 km altitude is to avoid highly populated altitudes occupied by other large constellations and avoid altitudes above 700 km up to 950-1000 km, which are densely populated with Earth observation satellites. Additionally, a lower altitude decreases orbital decay time and facilitates easier spacecraft disposal.

Mega-constellation optimisation can involve various cost functions, incorporating different parameters typically dictated by customer requirements. This requires finding a method to integrate these parameters in a way that aligns with customer needs and requirements. However, the specific customer preferences are unknown. Hence, only one parameter is considered in the cost function.

For this project, the cost function is defined as the total number of satellites in the mega-constellation, denoted as N_{tot}. This definition is chosen because:

- The launch and manufacturing costs of all satellites constitute the largest portion of the total cost [103]. Therefore, reducing the number of satellites by a few hundreds will significantly lower the expenses.
- It is a single objective, meaning that there is no need to deal with weights for other objectives or to compute a Pareto front.
- It is a very descriptive and easy-to-understand parameter, providing clear insight into the optimisation process.

This chapter introduces methods for optimising mega-constellations through the simple permutation of shells, as detailed in Section 5.1. Additionally, Section 5.2 elaborates on the acceleration of the optimisation process to decrease computational time and allow to find more optimal solutions. Subsequently, Section 5.3 presents a novel approach to mega-constellation design that is based on visibility requirements. Lastly, Section 5.4 provides an analysis of the computational time associated with the various methods and setups utilised in this project.

5.1. Shell Permutation

The most direct approach to finding the optimal solution involves iterating through all possible combinations. Given that the developed method of visibility computation enables faster calculations, there is no immediate need to apply mathematical optimisation algorithms. This suggests that shells at different inclinations with varying numbers of satellites can simply be selected and permuted collectively. This method will henceforth be referred to as the *permutation method*.

The assumption is made that all shells are positioned at the same altitude of 700 km. This assumption is based on the likelihood that the altitude of satellites is predetermined by payload or regulatory requirements. In reality, the altitude separation between shells can be relatively small, for instance, 10 km but for the initial analysis this difference is considered negligible. This altitude difference is observed in constellations such as Kuiper (initial design) and Starlink while OneWeb positions all shells at the same altitude. Thus, this assumption is considered valid.

To estimate how the number of possible combinations, and consequently the computational time, will scale, an initial examination of the number of layouts for each individual shell *j* can be conducted. Since all permutations of N and i are considered, this is equivalent to:

$$n_j = \text{len}(N_{\text{range, j}}) \cdot \text{len}(i_{\text{range, j}}) \tag{5.1}$$

The number of possible combinations can be computed as follows:

$${}_nC_r = \frac{n!}{r!(n-r)!} \tag{5.2}$$

where *r* represents the number of shells and *n* is defined by Equation (5.1). It can be observed that if r is increased, then ${}_nC_r$ grows drastically, especially if one aims for sufficiently high resolution (which implies that ΔN and Δi should be sufficiently low). This can escalate very quickly, and examples will be demonstrated in the following subsections.

5.1.1. One shell

Before exploring the design of multi-shell constellations, the feasibility of a single-shell solution is examined. Based on the parametric analysis presented in Section 4.5, the inclination and the number of satellites for a single shell are estimated and shown in Table 5.2.

Table 5.2: Setup for initial one-shell constellation design.

Parameter	Min. value	Max. value	Increment	Total values
Inclination i	60 deg	71 deg	1 deg	12
Number of sats N	$9,000$ sats	$11,000$ sats	100 sats	21
Total one-shell layouts				252 layouts

- **Number of steps:** $T = 1441$
  Propagation period of $t = 24$ hours with an increment $\Delta t = 60$ seconds
- **Number of points:** $M = 10,920$
  Grid of points with 120 points in longitudinal direction (3 deg resolution) and 91 points in latitudinal direction (1 deg resolution)

Figure 5.1 shows that a single-shell constellation is highly inefficient, as it provides excessive coverage in the latitude range of 40 to 68 degrees. The mean visibility reaches 150 satellites in view at 58 degrees latitude, which is 2.7 times more than the requirement. A typical visibility curve has a "bell" shape with a "tail", which suggests that using multiple shells can reduce the peak and flatten the curve, thus meeting the requirement more closely and optimising the constellation design.

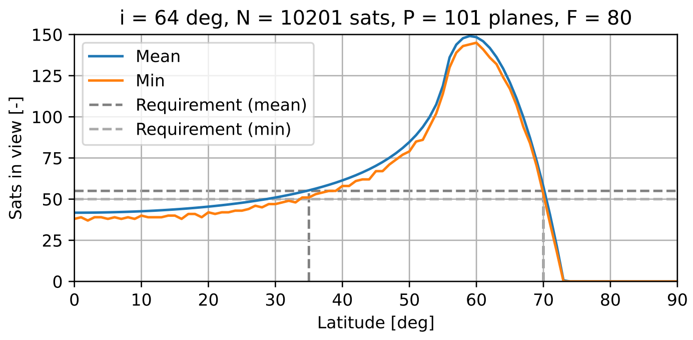


Figure 5.1: Visibility for one-shell mega-constellation.

5.1.2. Two shells

Drawing from the parametric analysis in Section 4.5, we can estimate the initial range for both shells to achieve the desired visibility. Consequently, the search ranges for N and i are set to the values shown in Table 5.3. For inclination, there is no need to exceed this inclination as it would result in unnecessary overcrowding of satellites at high latitudes, while the coverage requirement is up to 70 degrees. For the same reason, the lower bound of inclinations is set: shells with an inclination of 35 degrees are expected to have a visibility peak at approximately 30 degrees latitude, which is outside the target service area. As for the number of satellites, the parametric analysis showed that for the given constellation parameters, namely h, ε_{min}, and inclination range, this is a sufficient number of satellites, especially for an initial estimate.

- **Number of steps:** $T = 1441$
  Propagation period of $t = 24$ hours with an increment $\Delta t = 60$ seconds

Table 5.3: Setup for initial two-shell constellation design.

Parameter for each shell	**Min. value**	**Max. value**	**Increment**	**Total values**
Inclination i	35 deg	80 deg	5 deg	10
Number of sats N	3000 sats	6000 sats	500 sats	7
Total two-shell layouts				$2,415$ layouts

- **Number of points:** $M = 10,920$
  Grid of points with 120 points in longitudinal direction (3 deg resolution) and 91 points in latitudinal direction (1 deg resolution)

From the given ranges and Equation (5.2), it is possible to compute the number of possible combinations. The inclination range includes 10 values, while the range for N includes 7 values, yielding $n = 70$ combinations of $r = 2$ shells. Substituting these numbers for both shells into Equation (5.2), it is found that ${}_{70}C_2 = 2415$ two-shell constellation layouts over which it is necessary to iterate.

It should be noted that for the first iteration, the search grid is relatively coarse to maintain a reasonable computational time. Based on the results of this iteration, it will be possible to estimate the search range for subsequent iterations. This search will be refined to find a more optimal solution.

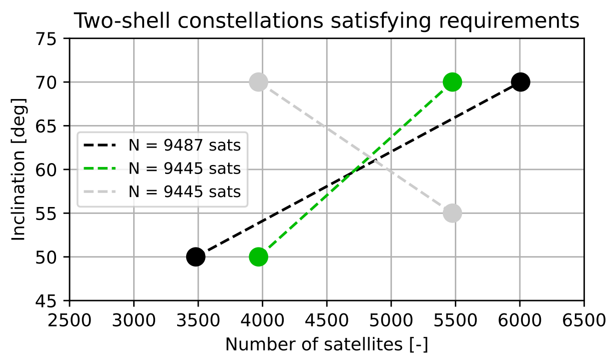


Figure 5.2: Two-shell constellation layouts satisfying requirements. One point represents a shell. Two shells comprising the same mega-constellation are connected by a dashed line.

The mega-constellations satisfying the requirements with 55 satellites in view on average are shown in Figure 5.2. Each point in Figure 5.2 corresponds to one shell. Shells within the same constellation share the same colour and are connected by a dashed line. Since the interest lies in the lowest number of satellites, all layouts with $N > 9500$ sats have been filtered out.

From Figure 5.2, it can be observed that the inclination range is selected correctly, and the optimiser does not tend to exceed the selected range. However, with the number of satellites, it can be seen that the black line lies on the boundary of the range for N. Thus, it would be of interest to see the behaviour of N if the bounds of the range for N are extended.

Figure 5.3 provides an illustration of the visibility for two-shell layouts that meet the specified requirement. From this, several observations can be made.

Firstly, the visibility curves intersect the requirement line at a latitude significantly higher than the upper bound $\varphi = 70°$. This suggests that enhancing the resolution for the inclination range could improve the current solution, as this would cause the visibility curves to shift to the left.

Secondly, it is noticeable that the visibility curve only touches the requirement line at the upper and lower latitude bounds. In between these bounds, it significantly exceeds the visibility, sometimes by more than twice the requirement. This indicates that the constellation is overdesigned. Therefore, increasing the resolution and expanding the search bounds could help reduce this overdesign, consequently

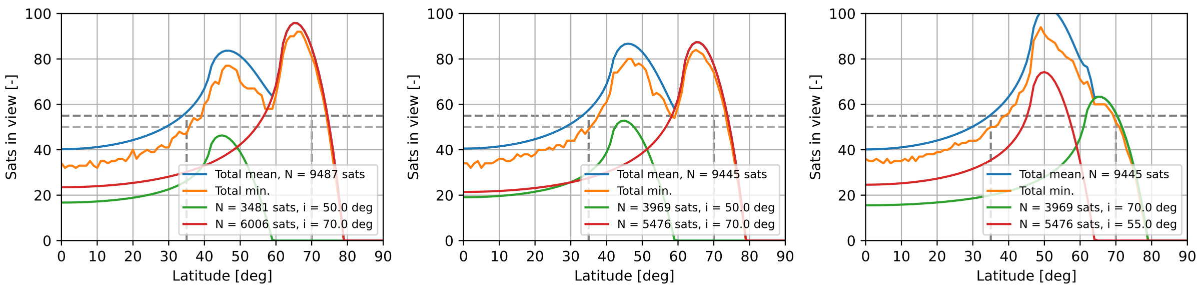


Figure 5.3: Visibility of two-shell constellation layouts satisfying requirements. Dark-grey line dashed line represents the requirement for average visibility and light-grey for minimum visibility.

decreasing the number of satellites in the constellation.

Refining and broadening the search range results in the two-shell layouts shown in Figure 5.4. The search range for this figure is presented in Table 5.4. This resulted in the total of $5.33 \cdot 10^6$ two-shell layouts, which could not be fully analysed within a reasonable timeframe if fully propagated. Therefore, a simplification of longitude averaging was applied for the design.

Table 5.4: Setup for two-shell constellation design with one meridian line simplification.

Parameter for each shell	Min. value	Max. value	Increment	Total values
Inclination i	35 deg	80 deg	1 deg	46
Number of sats N	1000 sats	8000 sats	100 sats	71
Total two-shell layouts				$5,331,745$ layouts

- **Number of steps:** $T = 1441$
  Propagation period of $t = 24$ hours with an increment $\Delta t = 60$ seconds.
- **Number of points:** $M = 91$
  91 points in latitudinal direction (1 deg resolution) lying on a single meridian line.

If all layouts with the number of satellites $N > 8600$ are excluded, a clear trend emerges: shells with higher inclinations are populated with a greater number of satellites, while shells with lower inclinations have fewer satellites. This can be attributed to the fact that shells at higher inclinations also cover lower latitudes, while the "bell" of the visibility curve remains approximately the same, i.e., the width and height are more or less consistent. This is clearly illustrated in Figure 4.25, where the only variable parameter is i. Constellations at inclinations $30° \leq i \leq 70°$ produce very similar visibility curves, except that the "tail" becomes lower (on y-axis) with increasing inclination.

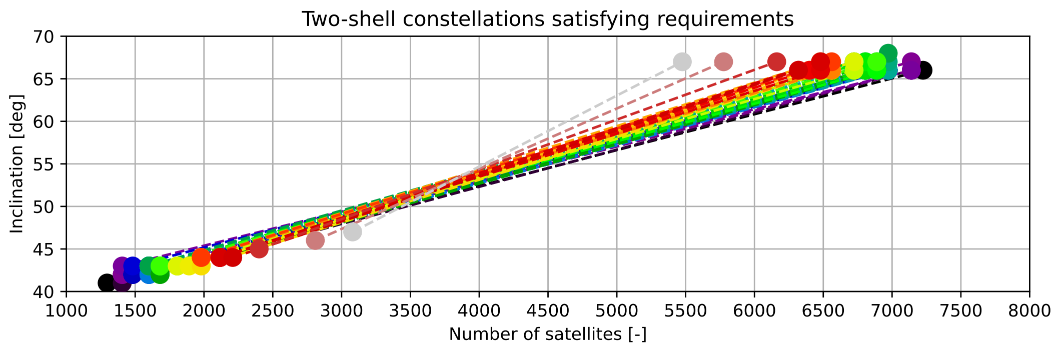


Figure 5.4: Two-shell constellation layouts satisfying requirements. Labels with total numbers of satellites removed for readability purposes. The layouts shown have the total number of satellites $N_{\text{tot}} < 8600$.

Figure 5.5 presents six mega-constellation layouts with the minimum N_{tot}. Now, the minimum number of satellites is $N_{\text{tot}} = 8,242$ satellites, which is 1959 satellites fewer than the single-shell layout. This is already a very significant decrease in N_{tot} which highlights the advantage of multi-shell design.

It can be observed that the distribution of satellites within shells, as well as the inclinations of shells, are very similar, resulting in a small difference in N_{tot}. From this, it can be noted that the visibility curve flattens out at latitudes lower than 50 degrees, indicating a more efficient design. The higher latitudes are still largely overserviced, leading us to assume that adding a third shell could be beneficial in further reducing the number of satellites. This will be analysed in the subsequent section.

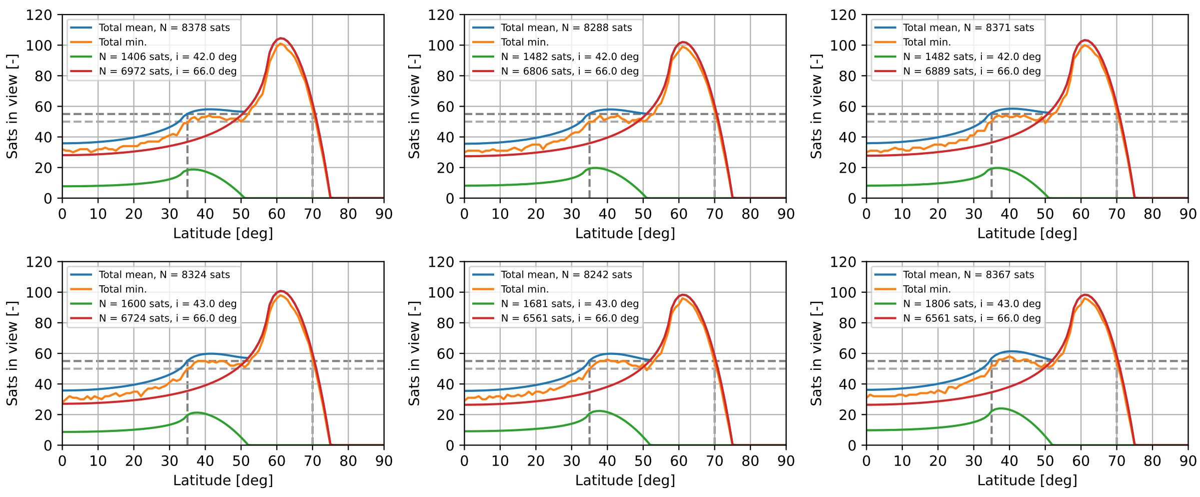


Figure 5.5: Visibility of two-shell constellation layouts satisfying requirements. Dark-grey line dashed line represents the requirement for average visibility and light-grey for minimum visibility.

5.1.3. Three shells

Building on the results from the two-shell constellation design, the next step is to proceed to three-shell constellations. As demonstrated earlier, the number of all possible layouts grows very rapidly if the number of shells is increased. If for $r = 2$ shells, the number of combinations was $C \approx 4.96 \cdot 10^6$, then for $r = 3$ with the same resolution, this number becomes $C \approx 5.80 \cdot 10^9$, i.e., three orders of magnitude larger. Therefore, the steps of $\Delta i = 1°$ and $\Delta N = 100$ sats must be increased to achieve a reasonable computational time. Due to the fact that the interest is in layouts with $N_{\text{tot}} < 8242$ satellites (as the aim is to improve two-shell design), all layouts with the total number of satellites $N_{\text{tot}} > 8300$ were discarded. The search range provided in Table 5.5 for three-shell design yields a total of $C \approx 81.3 \cdot 10^6$ various shell combinations to be iterated over.

Due to the exceedingly high number of layouts that required analysis, the "snapshot" simplification was applied. This means that only the initial states of mega-constellations were analysed. It was assumed that the accuracy is sufficient. Additionally, the final results are verified with a full-time simulation with 24 hours of propagation time.

Table 5.5: Setup for three-shell constellation design with snapshot simplification.

Parameter for each shell	**Min. value**	**Max. value**	**Increment**	**Total values**
Inclination i	35 deg	80 deg	1 deg	46
Number of sats N	1000 sats	8000 sats	200 sats	36
Total three-shell layouts			$755,514,120$ layouts	
Only with $N_{\text{tot}} \leq 8300$ sats			$81.3 \cdot 10^6$ layouts	

- **Number of steps:** $T = 1$
  Snapshot simplification applied, considering only initial states of satellites.
- **Number of points:** $M = 10,920$
  Grid of points with 120 points in longitudinal direction (3 deg resolution) and 91 points in latitudinal direction (1 deg resolution).

The results are presented in Figure 5.6. It can be observed that the trend closely resembles the two-shell design: a greater number of satellites are allocated to higher inclinations, while the number of satellites per shell decreases with decreasing inclination. The reason for this is the same as that of the two-shell layout: higher-inclination shells cover lower latitudes, while lower-inclination shells leave higher latitudes uncovered.

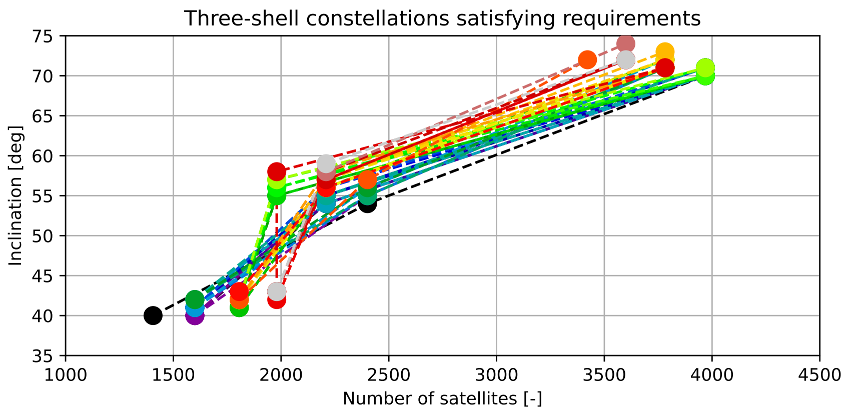


Figure 5.6: Three-shell constellation layouts satisfying requirements. Labels with total numbers of satellites removed for readability purposes. The layouts shown have the total number of satellites $N_{\text{tot}} < 7800$.

Most importantly, the total number of satellites is now significantly reduced. Specifically, the minimum is now $N_{\text{tot}} = 7629$ sats, while the two-shell layouts shown in Figure 5.5 have the lowest $N_{\text{tot}} = 8242$ sats, which is 613 satellites more compared to the three-shell design. It was not possible to obtain a lower N_{tot} with a step of $\Delta N = 200$ sats. However, Figure 5.7 illustrates how flat the visibility curve becomes for the lowest obtained number of satellites. This indicates that there is little to no overdesign in the layout and leads to the conclusion that the total number of satellites cannot be further decreased.

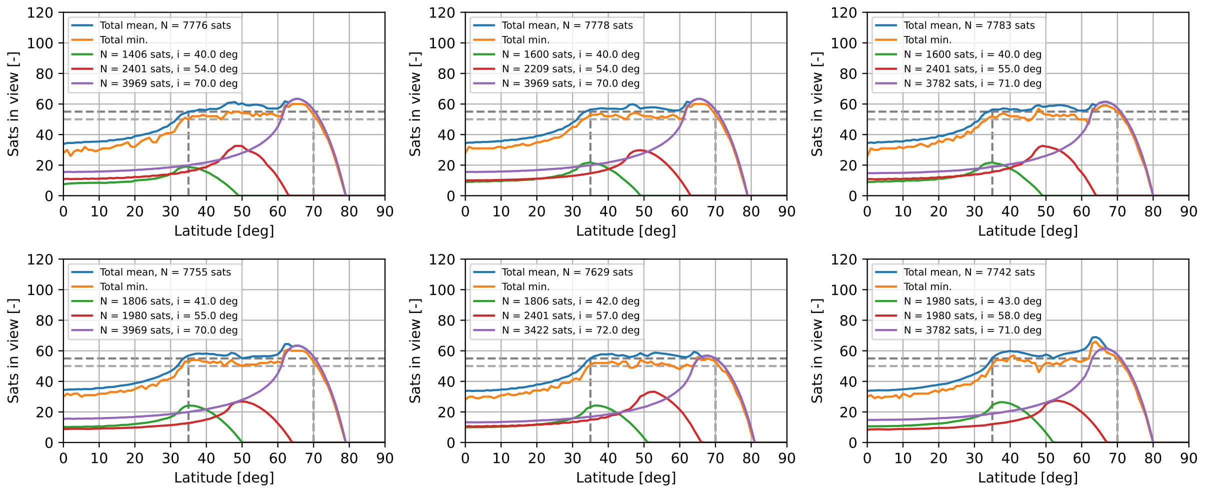


Figure 5.7: Visibility of three-shell constellation layouts satisfying requirements. Dark-grey line dashed line represents the requirement for average visibility and light-grey for minimum visibility.

5.1.4. Verification

This subsection provides a brief verification of the "winner" layouts for both two- and three-shell constellation layouts. For verification purposes, a full propagation period of $T = 24$ hours was applied. The grid size was set at 91x120, corresponding to a 1-degree resolution in the latitudinal direction and a 3-degree resolution in the longitudinal direction.

For the two-shell design, Figure 5.8 depicts that the distribution remains consistent, with minor differences in minimum visibility which align with the expectations for the employed method.

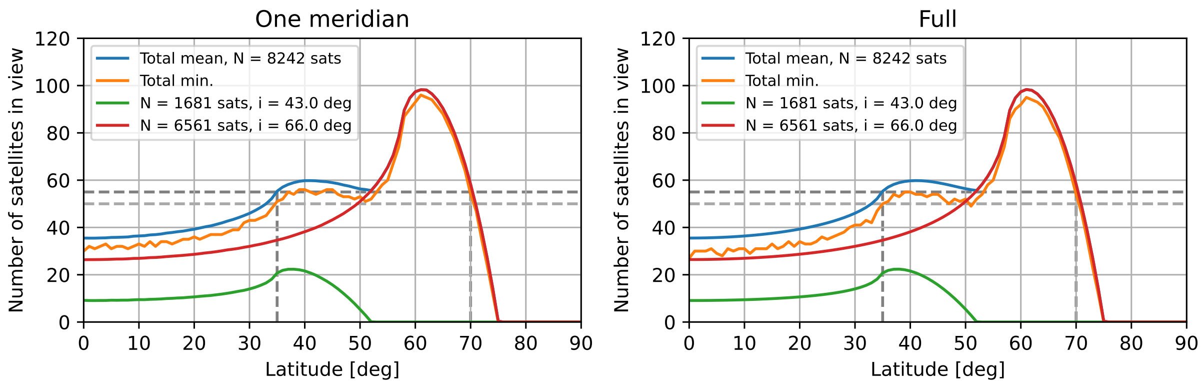


Figure 5.8: Verification of two-shell mega-constellation design. Visibility on the left is computed using one meridian line simplification, visibility on the right is computed with a full simulation ($T = 24$ hr, grid resolution is 1x3 deg).

In the case of the three-constellation design, as shown in Figure 5.9, the mean visibility of the fully propagated model aligns with that of the design obtained using snapshot simplification. However, the minimum visibility is slightly but noticeably lower. This discrepancy can be attributed to the reduced accuracy resulting from the use of simplifications. Nonetheless, this is not a significant issue as the minimum visibility can be improved later by adjusting P and F for all shells as was previously demonstrated in Section 4.5.1.

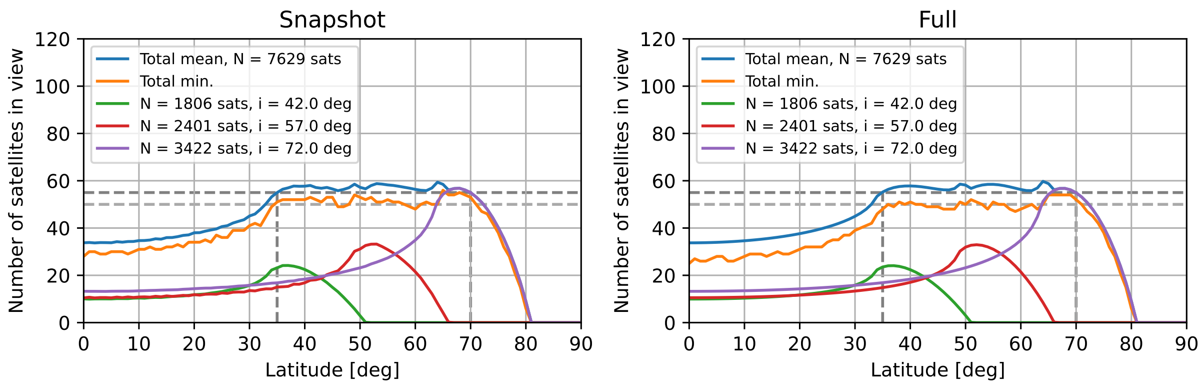


Figure 5.9: Verification of three-shell mega-constellation design. Visibility on the left is computed using snapshot simplification, visibility on the right is computed with a full simulation ($T = 24$ hr, grid resolution is 1x3 deg).

Therefore, it can be confirmed that the two- and three-shell layout optimisation produces layouts that are sufficiently accurate for study purposes, verifying the use of these methods for multi-shell design optimisation.

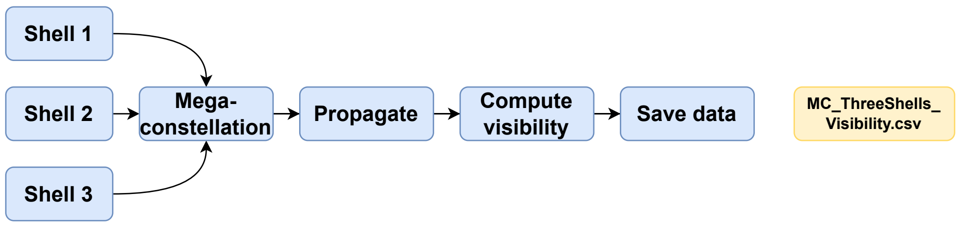
5.2. Permutation Acceleration

This section is dedicated to methods for accelerating the constellation design process, which was detailed in the Section 5.1, encompassing the approach, requirements, parameters, and optimiser behaviour. The objective of this section is to introduce methods that reduce computational time without significantly sacrificing accuracy.

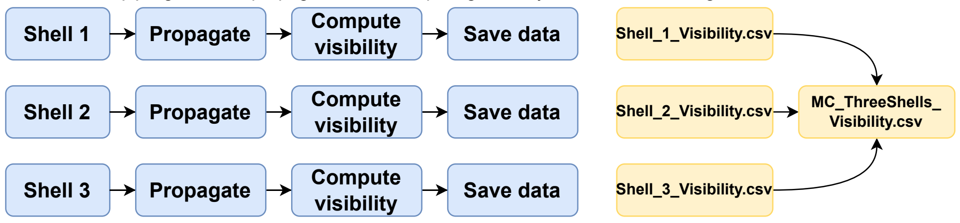
It should be noted that some of the methods outlined in this section have already been employed in Section 5.1 to speed up the constellation design process. This was a conscious decision, as the aim of that section was to examine the impact of the number of shells on the total number of satellites, as well as the distribution of satellites among shells.

5.2.1. Propagate each shell separately

Permutation can be significantly accelerated by propagating each shell of a mega-constellation individually, rather than collectively, and subsequently combining them. This method's effectiveness stems from the computation of visibility. For a given set of grid points, visibility can be computed separately for each constellation shell, and then summed up. This approach yields the same result as simultaneous propagation and visibility computation for all elements. A visual example for a three-shell mega-constellation is shown in Figure 5.10. Figure 5.10a shows the setup for computing visibility when three shells are combined initially and then propagated while Figure 5.10b shows how the same result can be obtained if three shells are propagated separately and only final results are superimposed.



(a) Algorithm of propagation and computing visibility of a three-shell mega-constellation.



(b) Algorithm of propagation and computing visibility of three shells separately.

Figure 5.10: Comparison of two methods of computing multi-shell mega-constellation visibility.

This approach greatly reduces the number of propagations and visibility computations. Each shell can be propagated separately for a given range of parameters, and the visibility files can be stored and reopened later to compute the mean and minimum visibility for each mega-constellation. This method is much more efficient than propagating each mega-constellation individually, as reading pre-computed data is faster than propagating and computing the visibility of constellations.

To enhance computational efficiency, a database can be established to store visibility files for each shell. These files can be readily retrieved by the tool for subsequent use, resulting in significant time savings through file reuse. Over time, the database will accumulate visibility files, leading to a reduction in the number of new computations required as pre-computed data becomes readily available. However, it is crucial to ensure that visibility arrays for addition or comparison correspond to the same set of grid points, simulation period T, and time step Δt. As demonstrated in Figure 4.6c, these parameters define the output array. Furthermore, memory limitations must be considered, as individual visibility arrays can consume significant memory resources (e.g., a visibility array with $M = 32,760$ points with $T = 24$ hours and $\Delta t = 60$ seconds would require approximately 180 MB).

An example of three-shell design is shown in Section 5.1.3. Given the ranges, only $n = 1656$ shells need to be propagated and stored, instead of propagating $81 \cdot 10^6$ layouts.

If the focus is only on the mean visibility, pre-computing and storing these mean values can significantly reduce computational time. This allows for the efficient analysis of a larger number of shells or finer value ranges. However, there is a caveat: computing the mean visibility in advance implies that the minimum visibility would be less accurate.

As shown previously, for a constellation, the mean of a sum is the sum of means. This does not hold true for the minimum value, because the minimum of a sum is not the same as the sum of minimums. Despite this, an acceptable estimate of the minimum visibility can still be obtained, albeit with lower accuracy. The expected difference between mean and minimum visibility was described in Section 4.5.1.

5.2.2. Apply simplifications

Section 4.4 discussed the simplifications that can be applied to accelerate the computation of a Large LEO Walker constellation's visibility and how it affects the accuracy of computations compared to non-simplified calculations. These simplifications can also be applied to multi-shell constellation design. However, it should be noted that they can decrease the accuracy of computation, especially if the focus is on the minimum visibility rather than the mean. The mission analyst would need to decide to what extent the accuracy can be lowered in exchange for a decrease in computational time.

Once the simplifications are applied and optimal layouts are found, the mega-constellation design can be verified by propagating a few layouts without simplifications to see whether the visibility results differ from those of the simplified assessment. As shown previously, the mean visibility is not significantly affected by the simplifications if the simulation setup is chosen appropriately (e.g. a sufficiently long propagation period for the single meridian simplification or a sufficiently large number of grid points for the snapshot simplification). However, the minimum visibility may differ by a few satellites. For example, for the layout shown in Figure 4.16 with $N = 4620$ satellites, this difference is 4 satellites.

It should also be kept in mind that the minimum visibility can be improved later when optimal values for the number of planes P and the phasing parameter F are found. More optimal P and F values will smooth out the minimum visibility curve and prevent very large drops in minimum visibility.

5.2.3. Scale with N

As demonstrated in Section 4.5, visibility scales linearly with the number of satellites N. This provides an alternative method to accelerate constellation design by scaling visibility curves, rather than propagating constellations and computing those. For given inclination and latitude ranges, mean visibilities can be loaded into an array. This array, representing mean visibility as a function of inclination and latitude, contains mean visibility for a fixed altitude h and a reference number of satellites N_{ref}.

The process can proceed in the same way with permuting shells. However, instead of propagating them or loading files with visibility, mean visibility can be accessed from the array for the required inclination. This visibility is then multiplied by a factor of the satellite ratio κ, which represents the number of satellites in a shell N_{shell} divided by the reference number of satellites in the array N_{ref}:

$$\kappa = \frac{N_{\text{shell}}}{N_{\text{ref}}} \tag{5.3}$$

These mean visibilities are then summed to obtain the figure of merit. This process must be performed for a fixed altitude h and minimum elevation angle ε. If shells at different altitudes are desired, multiple arrays can be used.

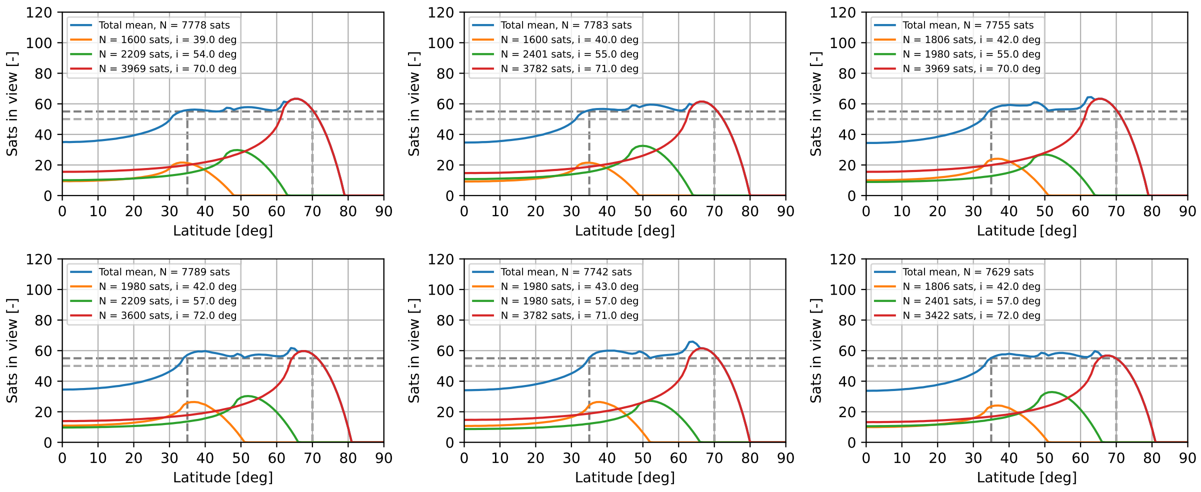


Figure 5.11: Constellation design with the method of scaling N.

This method is applied to a three-shell design shown in Section 5.1.3, and the best layouts are presented in Figure 5.11. It can be seen that the "winner" layout with $N_{\text{tot}} = 7629$ sats is precisely like that in Figure 5.6 obtained with applied "snapshot" simplification. Shells are placed almost at the same inclinations with the same number of satellites.

This method significantly accelerates the process due to the reduced amount of data that needs to be loaded and stored. The compact size of the visibility array allows it to be stored in the Random Access Memory (RAM) rather than disk memory (such as a hard drive or SSD) which is considerably faster. However, a limitation exists in the form of the size of the RAM, which does not allow full visibility arrays to be loaded into memory. Therefore, only the mean visibility is considered to maintain the size of the array at a sufficiently low level.

The downside of this method is that the minimum visibility is not computed and one can only assume that the minimum visibility will be approximately 90% of the mean visibility. This method would not be suitable if the primary requirement is the minimum visibility. However, as mentioned before, the minimum visibility also depends on P and F and this method would work for large constellations.

Additional improvements can be made by having continuous values for inclination i and latitude φ, for which an interpolator could be applied. This is illustrated in Figure 5.12, where the mean visibility is plotted as a function of latitude and inclination. Continuous values also facilitate the application of a numerical optimiser, which usually performs better with continuous functions. Therefore, by using an interpolator, mean visibility becomes a continuous function that can be used for numerical optimisation.

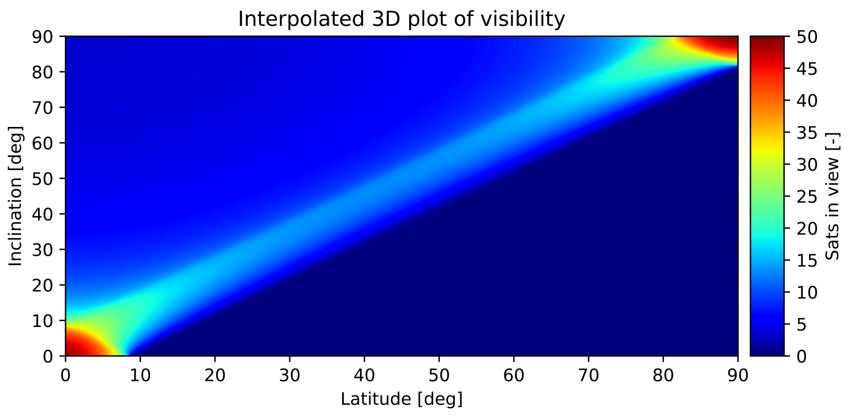


Figure 5.12: Interpolated mean visibility as a function of latitude and inclination.
Simulation setup: $N = 992$ sats, $h = 700$ km, $\varepsilon_{\text{min}} = 30$ deg
Note: P and F are not specified because they are not relevant for mean visibility.

5.3. Shells Building Blocks

Section 5.1 demonstrated the necessity of maintaining the visibility curve as close as possible to the requirement in order to minimise the number of satellites. Deviating from this approach results in an overdesign, leading to increased costs and decreased efficiency. As illustrated in Figure 5.7, expanding the target latitude band would no longer fulfil the requirement with the current N, necessitating additional shells to maintain a flat visibility curve that satisfies the requirement. Adding another shell drastically increases the number of combinations and, hence, computational time, as was demonstrated by Equation (5.2). Therefore, there is a need to either increase the step in the number of satellites and inclination or devise a different method for designing with more than three shells.

An approach called *building blocks* was developed where the target latitude band can be divided into sub-bands, each of a given width and containing two or three shells. These parameters, width and number of shells, would serve as design parameters defined by the mission analyst. Each band would be filled up to satisfy the requirement. The design process begins with higher latitudes as it was observed that in multi-shell designs lower inclination shells have no impact on higher latitudes. The approach is summarised below, and Figure 5.13 will be used as an example.

Step 1: Select width of latitude sub-ranges $\Delta\varphi$ and number of shells r_i per sub-range i

As shown in Section 5.1 and Section 5.2, depending on the target latitude band width, the optimal number of shells can be different. Thus, it would be up to a mission analyst to determine these parameters, based on requirements and computational resources available. In the example shown in Figure 5.13, latitude sub-range width is $\Delta\varphi = 12$ degrees, and the number of shells is $r_i = 2$. This results in 3 sub-ranges for the target latitude band of 35-70 degrees.

Step 2: Start with the sub-range at highest latitude
It was demonstrated that the higher inclination shells are not affected by lower inclination shells as the latter cannot satisfy requirements for higher latitudes. This is why the highest sub-range should be optimised first.

Step 3: Permute shells and find a layout with lowest N
In each sub-range the optimisation process would be the same as shown in Section 5.1 and any acceleration techniques from Section 5.2 could be applied. Visibility from higher sub-ranges would be added as those shells contribute as well.

In case there are multiple layouts with the same N, one can apply the following approach:

Step 3a: Compute area under visibility curve for lower sub-ranges
This is a quantitative metric to estimate how much each layout contributes to the lower latitude sub-ranges.

Step 3b: Select the layout with highest area
Since all layouts in question satisfy the requirements and have the same N, i.e. the same value for the cost function, we can select the one that contributes the most to visibility at lower latitudes making it require fewer satellites.

Step 4: Repeat for all sub-ranges
The process continues from higher inclination shells to lower inclinations, filling each sub-band until completion as shown in Figure 5.13.

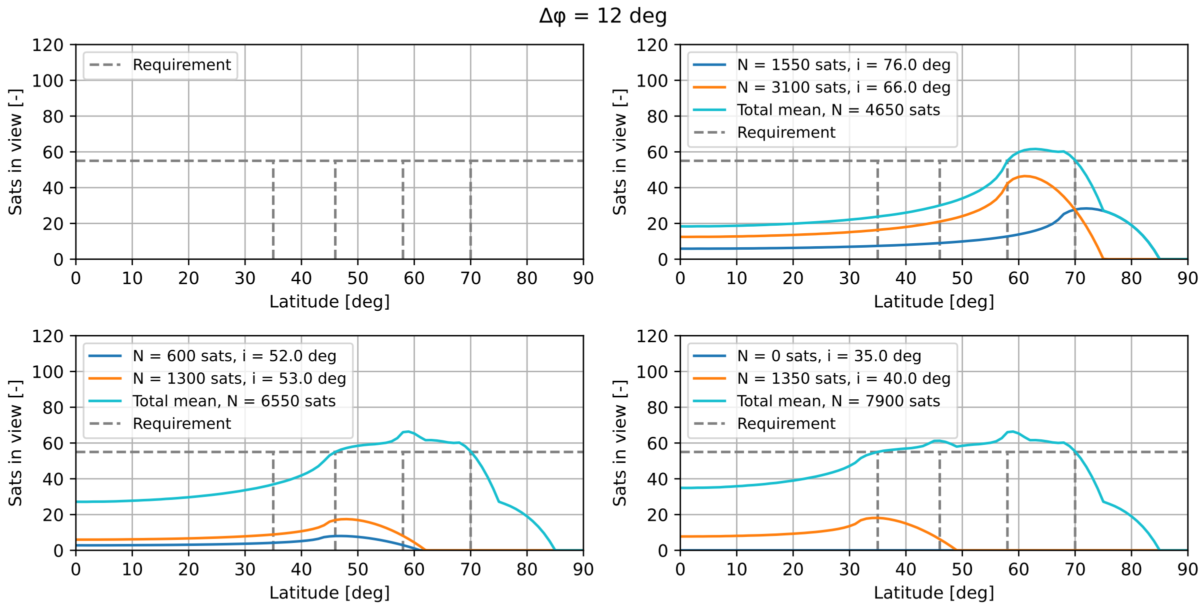


Figure 5.13: Shells building blocks example. Note that this example does not demonstrate the optimal layout as parameters were arbitrarily selected for demonstration purposes.

In Figure 5.14, it can be observed that the total number of satellites is very close to $N_{\text{tot}} = 7629$ sats obtained with simple permutation of 3 shells (see Figure 5.7 and Figure 5.11). Also, N and i of each shell are close to those obtained with previous methods. This method may require some iterations and tuning of $\Delta\varphi$ and r_i. However, given the savings in computational time, this method proves to be suitable for multi-shell mega-constellation design.

It can be noticed that for $\Delta\varphi = 19$ deg and $\Delta\varphi = 23$ deg, there are “shells” with zero satellites. Even though, the input was to have $r_i = 2$ shells per sub-range, which would lead to 4 shells in total, the algorithm found that 3 shells is more optimal than 4.

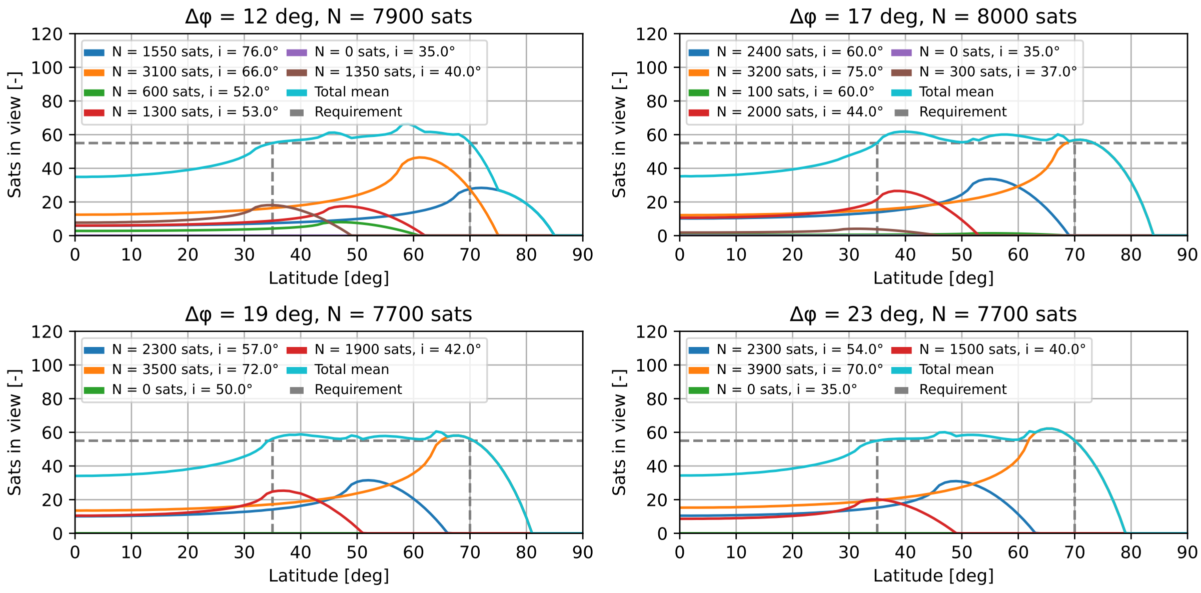


Figure 5.14: Shells building blocks with different $\Delta\varphi$.

This method was applied to improve Jia's mega-constellation design [51] that was used for verification in Section 4.3 and shown in Figure 4.10. For the same propagation period, the building blocks method was used with the final result provided in Table 5.6 with $N_{\text{tot}} = 5700$ satellites as compared to Jia's $N_{\text{tot}} = 5686$ satellites.

Figure 5.15 illustrates a comparison with Jia's design, which consists of 45 shells. Although their design clearly provides more homogeneous mean visibility, the design obtained with the building blocks method achieves better minimum visibility due to its more uniform distribution of satellites. Moreover, this can serve as a good first-order estimate for the total number of satellites. Jia's layout, consisting of 45 shells each comprising multiple planes, could pose significant challenges for real-life implementation. Thus, it is more feasible to have only 8 shells.

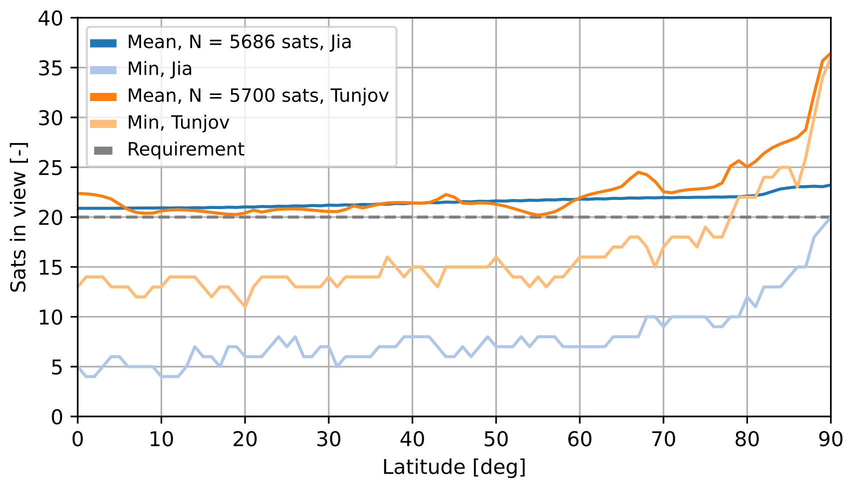


Figure 5.15: Comparison with Jia's design [51].

The current method suggests more uniform distribution of satellites which means that the minimum visibility is significantly higher compare to that of Jia's. Thus, this method can be used as a good initial estimate for the number of satellites required in a constellation. This method is also relatively fast allowing for a high number of shells. A more detailed analysis of computational time will be presented in Section 5.4.

Table 5.6: Jia's design improvement. $N = 5700$ sats

Inclination [deg]	Num. of sats	Num. of planes
85.0	1300	26
73.0	1000	25
63.0	900	25
50.0	900	25
39.0	700	25
27.0	500	20
15.0	300	15
2.0	100	10

5.4. Computational Time Comparison

This section will summarise the gain in computational performance achieved through the proposed methods. The computational time was determined by executing the tool ten times consecutively with identical input, and subsequently calculating the average. A preliminary run was performed for the visibility computation to exclude the GPU initialisation time, which can take a few seconds.

5.4.1. Constellation Propagation

The comparison of computational time solely for propagation is depicted in Figure 5.16. The vectorised implementation of analytical propagation outperforms `BallisticPropagator` from Godot by nearly a factor of 100x. Compared to the non-vectorised version of `mcdo` (analytical), the speed-up is less pronounced, but still is around a factor of 20x. As the number of satellites increases significantly, the computational time growth goes from linear to non-linear after $N = 8,000$ satellites. This is likely due to the memory requirements surpassing the available RAM, necessitating access to the considerably slower disk memory. It is expected that similar effects would be observed with a lower N value if the number of steps were increased. However, this was not tested due to the time constraints of the thesis.

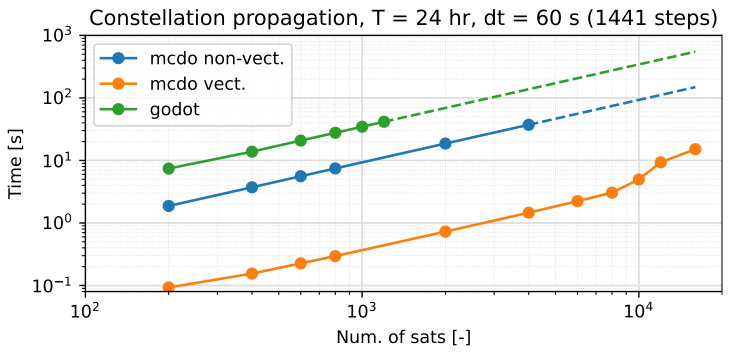


Figure 5.16: Computational time comparison for various propagation methods. `BallisticPropagator` object from Godot was used for comparison

5.4.2. Visibility Computation

In the computation of visibility, several factors influence the computational time. These factors can be broadly categorised into three groups: the computational method, the hardware, and the input parameters.

Three methods for visibility computation will be compared in terms of computational time: Godot, `mcdo` vectorised, and `mcdo` parallelised (i.e. accelerated by a GPU). Among these, the emphasis is on parallelised computation as it offers the maximum reduction in computational time.

Godot does not have direct functionality for global visibility computation. Thus, as a workaround one can define ground points manually and distribute them over the globe using `uni.frames.addPoint`[1]. The elevation angles of satellites defined and propagated with `BallisticPropagator` are computed and compared against the minimum elevation angle ε_{min}. This process, however, is highly inefficient, which will be shown later in the section.

Given that hardware is typically a fixed factor, the methods can be varied and the following input parameters can be examined to see how they affect computational time:

- Number of satellites (N)
- Number of time epochs (T)
- Number of grid points (M)
- Floating point precision (single vs. double)

The grid point method essentially involves a for-loop that iterates through all values of N, T, and M. The total number of elevation evaluations, which would be equal to $N \cdot T \cdot M$, can be examined. However, to understand the effect of a given parameter on computational time, only one parameter will be varied at a time.

Table 5.7: Setup for comparison of visibility computational time

Method	Num. of steps	Num. of points	Num. of sats
Godot	$T = 1441$	$M = 31 \cdot 12 = 372$ (resolution 3 deg in lat. direction, 30 deg. in long.)	N ranging $1 - 40$
mcdo vect.	$T = 1441$ (full propagation) and $T = 1$ (snapshot simplification)	$M = 10920$ (full propagation) and $M = 91$ (one meridian line simplification)	N ranging $25 - 1600$
mcdo paral.	T ranging $1 - 1441$	M ranging $132 - 32760$	N ranging $5 - 6400$

The setup for visibility computation, as shown in Table 5.7, was used. It is important to note that if only one parameter is varied, the other two parameters are set to their maximum values. For instance, if T is varied, then M and N are set to 32760 and 6400, respectively. This approach was adopted to push the hardware to its limits and observe the effect on computational time.

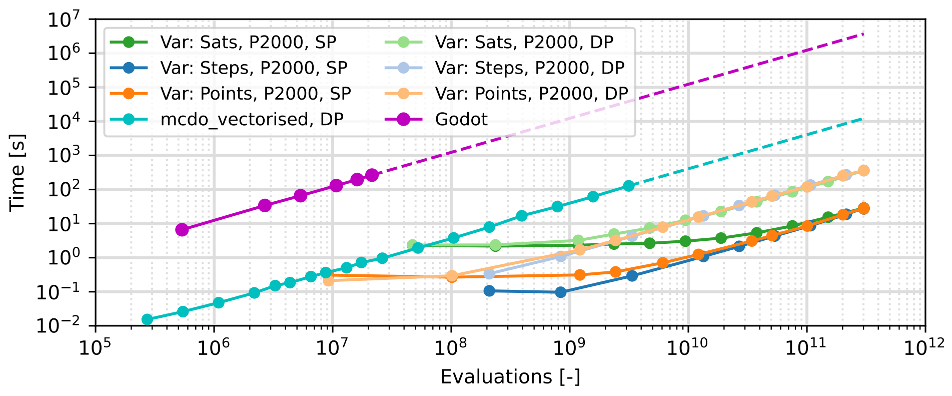
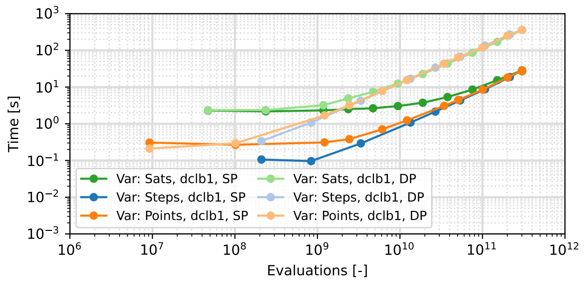


Figure 5.17: Comparison of computational time for various visibility computation methods.

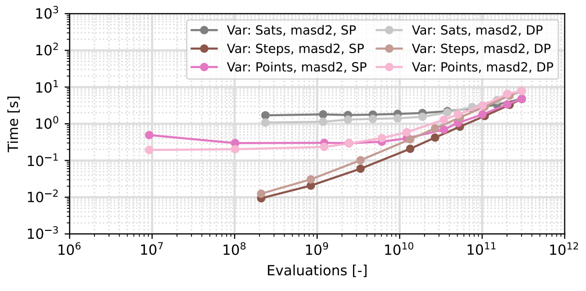
[1] https://godot.io.esa.int/godotpy/api_reference/generated/generated/godot.model.frames.Frames.html#godot.model.frames.Frames.addPoint, accessed 14 May 2024

Figure 5.17 presents a comparison of various visibility computation methods in terms of computational time. It is evident that even with a low-end GPU installed on `dclb1` (see Table 4.1) , the computational time can be reduced by two to three orders of magnitude compared to non-parallelised computation, i.e., computation handled solely by a CPU. Furthermore, it is observed that variations in N, T, and M influence the computational time, even when the number of evaluations remains constant. Possible reasons for this difference will be discussed later in this section.

Furthermore, a comparison of computational time between double precision (DP) and single precision (SP) calculations is presented in Figure 5.18, focusing on two devices as outlined in Table 4.2. The computational time differs between the two types of floating precision. However, the difference is less pronounced for the high-end GPU compared to the low-end GPU. This can be attributed to the fact that the Tesla V100 is better optimised for calculations with double floating precision [104] compared to the Quadro P2000 [105]. According to their specifications, the number of floating operations per second (FLOPS) for the P2000 is 32 times lower for DP, while for the V100, it is only two times lower.



(a) dclb1



(b) masd2

Figure 5.18: Computational time comparison with different hardware setups

As depicted in Figure 5.18b, for `masd2`, for a low number of evaluations with varying satellites and points, DP calculations are observed to be faster than SP. This may seem counter-intuitive at first. However, this is likely due to the fact that not all operations are handled by the GPU, as illustrated in Figure 4.8. Certain operations, such as data copying and floating point conversion, are managed by the CPU. The CPU paired with the V100 is significantly less powerful than the one working with the P2000 (see Table 4.2), which leads to an overhead for a low number of evaluations. Consequently, for a low number of evaluations, using DP would be faster on `masd2`, albeit not significantly, because the acceleration is negated by the DP-to-SP conversion. As the number of evaluations increases, the computational time for SP becomes lower than that for DP.

It was shown that the computational time scales differently if we vary different parameters, namely the number of steps T, satellites N or points M. The explanations for this can lie in the amount of memory allocated for the visibility computation, which can be expressed by Equation (5.4)

$$\text{Mem.} = T \cdot (3 \cdot N \cdot s_{\text{float}} + 3 \cdot M \cdot s_{\text{float}} + P \cdot s_{\text{uint}}) \tag{5.4}$$

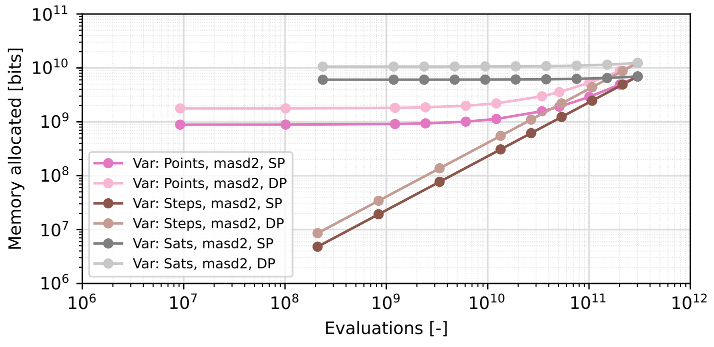


Figure 5.19: Allocated memory for visibility computation.

where s is the size of a single data value with subscript `float` corresponding to floating point size (32 or 64 bits) and `uint` to unsigned integer size (32 bits).

Since it was mentioned that SP is sufficient for this type of calculation, only SP, which has the same size as `uint`, i.e. $s = s_{\text{float}} = s_{\text{uint}}$, can be considered. This allows Equation (5.4) to be reduced to the following:

$$\text{Mem.} = T \cdot s \cdot (3\,N + 4\,M) \tag{5.5}$$

It can be observed that the memory allocated scales linearly with T, which also coincides with computational time (especially that of `masd2`). The same can be seen for points and satellites. This is especially clearly seen when looking at the plot shown in Figure 5.19 for memory allocated and comparing it to computational time on `masd2` shown in Figure 5.18b. Due to the high latency of GPUs (for memory access), the amount of memory and size of arrays affect the computational time.

Another explanation for the observed scaling and dependence on the number of evaluations is that memory access is not ideally coalesced. This increases computational time as GPU threads do not access memory in the most efficient way. Some threads in the warp have to wait for others to finish computation before they can access new memory cells. However, this was not optimised in this project and is left as a recommendation for future work.

5.4.3. Mega-Constellation Design

In contrast to the previous subsections that focused on individual computational tasks, specifically propagation and visibility computation, this section examines the combined computational times of these tasks. Additionally, this section explores the multi-shell design and the performance increase due to the application of methods that were discussed in the previous sections of this chapter.

Single Iteration of Visibility Computation

This section begins with a single iteration of the visibility computation of a large single-shell constellation. This iteration includes:

- Analytical propagation (point-mass + J2)
- Visibility computation

The following setup is used:

- **CPU:** Intel(R) Xeon(R) W-2125 CPU @ 4.00GHz
- **GPU:** NVIDIA Quadro P2000
- **Input:** $N = 6400$ sats, $T = 1441$ steps, $M = 32760$ points
- **Precision:** DP for propagation, SP for visibility computation

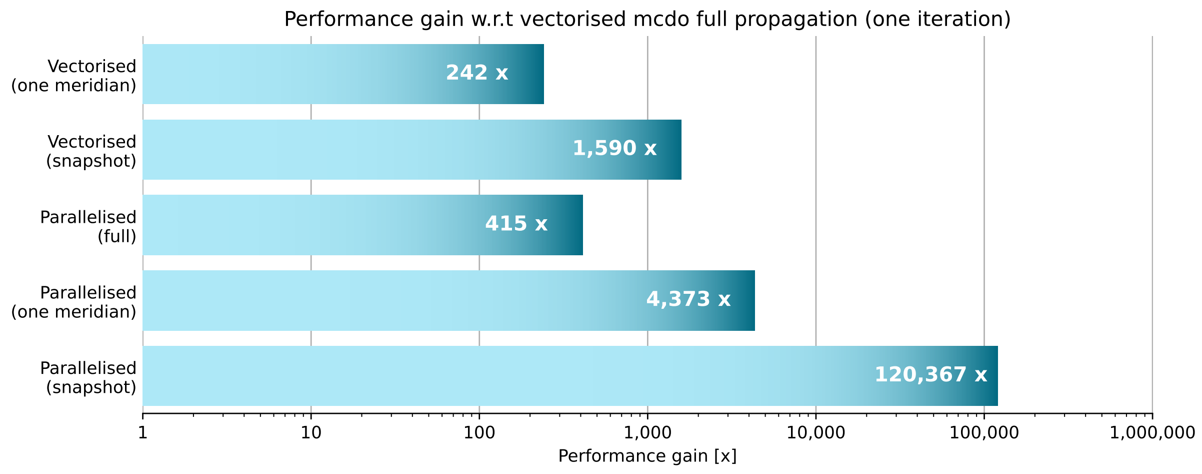


Figure 5.20: Performance gain for various methods.

Performance comparison will be shown with respect to vectorised `mcdo` implementation[2]. This reference simulation includes the full propagation time, all propagation points, and all satellites. Subsequent simplifications and accelerations are applied to assess their impact on computational time. The results presented in Figure 5.20 show that the GPU-accelerated computation with snapshot simplification demonstrates a considerable reduction in computational time, up to five orders of magnitude, without compromising accuracy significantly.

Figure 5.20 illustrates the performance gains achieved through the methods developed in this project. Simplifications prove effective in reducing calculation time, even for CPU-based computations. However, the utilisation of GPUs further enhances computational efficiency, yielding substantial time savings.

Snapshot simplification is an important contributor to computational speed, benefiting both vectorised computation and CPU-based calculations. This method saves a significant amount of time not only on visibility computation but also on propagation, which becomes quite time-consuming when the number of satellites and steps are high.

Multi-shell Design: Permutation

Furthermore, it is of interest to compare the aforementioned methods for multi-shell constellation design, which can involve thousands or even millions of iterations. Section 5.2 already showed that multi-shell design can be done by saving files with visibility for individual shells and then combining them. This subsection compares computational time for *reopening files* and *propagation* (which includes constellation propagation and visibility computation), along with other methods discussed in Section 5.2 and Section 5.3. The parameters used in comparison are presented in Table 5.8.

Table 5.8: Setup for comparison of visibility computational time

Parameter	**Min. value**	**Max. value**	**Increment**	**Total values**
Inclination i	30 deg	80 deg	2 deg	26
Number of sats N	$2,000$ sats	$8,000$ sats	300 sats	21
Total two-shell layouts			$148,785$ layouts	

The result shows that there will be 546 unique shells that will be combined into two-shell constellations, which then will be evaluated against the requirement. It also implies that only 546 files will be saved (as they are reused for both shells), enabling an estimate of memory requirements for each method used. Although not directly related to constellation design, considering memory limits might be important as each hardware setup has its own limitations.

It is crucial to note that the goal here is not to find the optimal layout, as it was thoroughly addressed in Section 5.1.

Table 5.9 shows the comparison and corresponding speed-up factors between propagation and file reopening. It can be seen that the propagation and visibility computation take very long even with one meridian simplification. For this reason, a smaller range was considered for full propagation and propagation with one meridian simplification applied, resulting in a total of $2,145$ two-shell layouts ($\Delta i = 10$ deg and $\Delta N = 500$ sats). It can be seen, however, that file reopening is much faster for all simplifications considered. File reopening offers a very significant gain. Thus, it is recommended to use this approach for mega-constellation design.

It is important to remember that file reopening still requires propagations to compute data. However, this is not a big issue because the number of single-shell layouts that need to be computed is much lower than two-shell layouts.

[2]Godot is not used in performance analysis anymore because it is not optimised for mega-constellation design, and almost any self-written software for this purpose generally outperforms Godot.

There are some notable observations with regard to one meridian simplification shown in Table 5.9. The meridian simplification is only 2x faster than full propagation, while file reopening is faster by a factor of 166x. This was not expected initially, but possible reasons are provided below.

1. Constellation propagation time (without visibility computation) is the driving factor of total computational time. The two methods require the same constellation propagations, which become the driving factor of computational time. Visibility computation, in this setup, is not significantly affected by reducing the number of points, especially with a low number of satellites. This aligns with Figure 5.18, which shows that varying the number of points may not always reduce the computational time by a large margin.
2. Output files are considerably smaller (by a factor of 360x), resulting in faster opening and processing. While snapshot files are smaller by a factor of 4x, computational time differs by a factor of 2.5x, likely due to overhead when opening files from disk memory.

Table 5.9: Setup for comparison of visibility computational time.

Simplification	Propagation	File reopening	Speed-up
Full	$3.44 \cdot 10^6$ s (*projected time based on smaller range*)	$30,102$ s	114 x
One meridian	$1.55 \cdot 10^6$ s (*projected time based on smaller range*)	181.0 s	$8,577$ x
Snapshot	$10,919$ s	72.22 s	151 x
Scaling with N	3.18 s		

Also, Table 5.9 illustrates that scaling with N offers a significant gain in computational time, as expected due to how CPU memory is handled, as was previously discussed in Section 5.2.

Multi-shell Design: Building Blocks

Finally, this subsection compares the building blocks method with the permutation method. The focus of this analysis is on the number of combinations. Given that the same methods can be employed for visibility computation in both the building blocks method and permutation methods, the relative difference in computational time remains consistent regardless of the chosen method.

The setup shown in Table 5.10 was used for comparison.

Table 5.10: Setup for comparison of computational time using the building blocks method.

Parameter	Values	Increment
Subrange 1		
Target latitude	51 to 70 deg	-
Inclination	51 to 80 deg	1 deg
N per shell	0 to 4000 sats	100 sats
Total two-shell layouts		$706,266$ layouts
Subrange 2		
Target latitude	35 to 51 deg	-
Inclination	35 to 61 deg	1 deg
N per shell	0 to 4000 sats	100 sats

Total two-shell layouts		$524,800$ layouts
Total		
Target latitude	35 to 70 deg	19 deg
Total two-shell layouts		$1,231,066$ layouts

Applying the same resolution for the search grid, namely the same Δi and ΔN, for a three-shell layout over the entire target latitude range, the total number of layouts is computed as shown in Table 5.11.

Table 5.11: Setup for comparison of computational time using the permutation method for three-shell design.

Parameter	Values	Increment
Target latitude	35 to 70 deg	-
Inclination	35 to 80 deg	1 deg
N per shell	0 to 4000 sats	100 sats
Total three-shell layouts		$1,116,304,540$ layouts
Only with $7000 \leq N_{\text{tot}} \leq 8000$ sats		$182,849,954$ layouts

The result shows that the number of layouts that have to be iterated over is much higher for a three-shell permutation due to its lower efficiency (e.g. all 3 shells can be at $i < 40$ deg). In this particular example, the computational time is reduced by a factor of 190 regardless of the method used for visibility computation. This reduction is already based on the assumption that the approximate total number of satellites is known, and bounds for lower and upper values are set. If this assumption is not considered, the difference becomes even more pronounced, growing to a factor of $1,161$.

Lastly, it is compared how Jia's design improvement was achieved through the application of the building blocks method, resulting in the configuration of 8 shells at various inclinations (refer to Table 5.6).

Table 5.12 shows that the building blocks method can allow for a multi-shell design with a relatively large number (more than three) of shells, while keeping the computational time reasonably low. Simply permuting such a high number of shells would have been impractical or necessitated unreasonably extensive computational resources, which would have made it impossible to find such a design. This confirms the usefulness of the building blocks method.

Table 5.12: Comparison of building blocks and permutation methods for improving Jia's design [51].

Parameter	Values	Increment
Building blocks		
Target latitude	0 to 90 deg	23 deg
Number of subranges	4	-
Shells per subrange	2	-
Inclination	see Appendix D	1 deg
N per shell	0 to 2000 sats	100 sats
Total eight-shell layouts		$765,450$ layouts
Computational time		8.18 seconds
Permutation		

Target latitude	0 to 90 deg	-
Inclination	0 to 90 deg	1 deg
N per shell	0 to 2000 sats	100 sats
Total eight-shell layouts		$4.347 \cdot 10^{21}$ layouts
Estimated computational time		$4.647 \cdot 10^{16}$ seconds

As shown in Table 5.12, the improvement of Jia's design with the building blocks method takes only a few seconds. This compares to a non-optimised where a simple permutation method would be applied. The permutation method would likely require an unreasonable amount of time or computational resources, making the search for the optimal layout impractical. Fortunately, the building blocks method, combined with the previously described visibility computation methods, offers a framework for time-efficient and straightforward mega-constellation optimisation.

5.5. Summary and Conclusions

This chapter analysed multi-shell mega-constellation design for a defined study case requiring several thousands of satellites which allowed to push the `mcdo` tool and to its limits. Two- and three-shell layouts were compared to a single-shell layout in terms of visibility distribution and the minimum number of satellites required. The analysis showed that multiple shells distributed the visibility more uniformly than a single shell by flattening the visibility curve and reducing the total number of satellites.

Furthermore, this chapter analysed how the multi-shell design can be accelerated. Three approaches were identified: storing visibility arrays for individual shells, applying simplifications, and scaling the mean visibility of a reference constellation with N.

A novel method was proposed for mega-constellations requiring more than three shells. The method works by dividing the target latitude range into sub-ranges and applying permutation method to each sub-range starting from the highest latitude sub-range.

Finally, a comprehensive analysis of computational performance was carried out for the `mcdo` tool and newly developed methods. It showed that even with a low-end GPU visibility computation time can be decreased by a factor up to 120,000x with respect to non-simplified CPU-only computation while maintaining acceptable accuracy. Also, it was shown that multi-shell design can be accelerated to complete in a few seconds.

Key Takeaways

- Higher-inclination shells cover both higher and lower latitudes and thus tend to have more satellites than lower-inclination ones.
- Multiple shells at different inclinations allow to bring the visibility curve closer to the requirement making the target latitudes less overserviced.
- Storing visibility files for each shell in a database allows to reduce the number of simulations (and hence computational time) in the course of the use of the software.
- The building blocks method can be applied to any visibility requirement and any latitude band making it a very efficient tool for multi-shell mega-constellation design.

Limitations and Recommendations

- If the method of scaling the visibility curve with N is applied, then the minimum visibility cannot be computed directly, and an assumption for that must be applied.
- P and F were not subject to optimisation in this thesis, but it would make an interesting topic for the future research.

6 Conclusions

This thesis significantly advances the analysis and design of large satellite constellations. The novel tool `mcdo` developed within the thesis allows the optimisation of mega-constellations. It achieves this via several methods:

a) Utilisation of vectorisation for analytical constellation propagation which leads to an acceleration by a factor of up to 100x with respect to `BallisticPropagator` in Godot.

b) Accelerated visibility computation, specifically utilising GPUs for parallelising visibility computations and simplifications derived from the analysis of large Walker constellations (these come on top of the vectorisation effect). This leads to an acceleration by a factor of up to 120,000x.

Utilising these tools, the thesis thoroughly examines large Walker constellations, revealing valuable insights into their visibility characteristics. Subsequently, multi-shell constellations are optimised, minimising the number of satellites needed for specific visibility requirements. Furthermore, a novel design method is proposed specifically for complex multi-shell constellations, expanding their design potential.

The thesis makes significant contributions to the field of large satellite constellation design and analysis. This work allows for building efficient, high-performing constellations that can effectively meet the growing demands of the space industry. It paves the way for future advancements, laying the groundwork for a future interconnected with reliable space networks.

This thesis was conducted with the primary objective of developing optimisation methods for the orbital layout of mega-constellations, with a particular emphasis on the complexity of multi-shell constellation design, that is, how to combine shells with different orbital parameters for the given mission objective and requirements.

This is formulated by the **main research question**:

How does selection of the mission objective affect the optimal design of mega-constellations?

There are three sub-questions to the main question:

Subquestion 1: *What are the figures of merit for evaluating mega-constellation design for different mission objectives?*

In Chapter 2, a comprehensive review of existing systems and research outlined the key objectives of mega-constellations, including satellite communications (Satcom), satellite navigation (Satnav), remote sensing (RS), and the internet of things (IoT). While certain constellations, such as Planet Labs or Spire, cater to remote sensing applications, the majority aim at global navigation or communication services. Satcom constellations, in particular, operating at low altitudes (up to 1,200 km) with a high number of satellites (more than 1,000 sats), were of interest.

The thesis identified figures of merit (FoMs) most commonly used in the design of the mega-constellation orbital layouts. These are Visibility, Revisit Time, Dilution of Precision, Precise Point Positioning, Latency, and Signal-to-Noise Ratio.

Visibility was chosen as the primary focus among all the listed FoMs, because it plays a crucial role in most mega-constellation designs, except for remote sensing where revisit time is more important. Two related FoMs that are commonly used in industry and scientific literature to measure the visibility are mean and minimum visibility which show the mean and minimum numbers of satellites, respectively, that are visible from a given grid point or a range of points on the ground over time.

However, visibility is also the main bottleneck in designing mega-constellations, because it involves a large number of satellites and ground grid points. Therefore, finding ways to speed up the visibility computation was of interest.

Subquestion 2: *How can the computational time of the optimisation problem be reduced?*

To estimate the selected FoMs, the constellation needs to be propagated over time. Chapter 3 showed that the propagation of a large number of satellites was improved using an analytical approach and vectorisation. This achieved a 100x speed-up compared to Godot (`BallisticPropagator` with point-mass and J2 dynamics) and a 20x speed-up compared to a non-vectorised implementation of the analytical model. The positional accuracy was up to 25 km per 24 hr propagation period in LEO, which was acceptable for this study.

The visibility computation was also optimised in Chapter 4. Because of its "embarrassingly parallel" nature, GPUs could be used to speed up the visibility computation by two to three orders of magnitude, i.e. 100x-1000x faster (depending on the hardware used). It was found that using single floating precision (SP) was enough for the visibility computation and had the probability of an error of 10^{-5} compared to double floating precision (DP). This was beneficial because GPUs, especially low-end ones, are optimised for operations with SP values. However, the CPU working with the GPU can affect the computational time because it handles some operations such as data copying or DP-SP conversion.

From the survey of the mega-constellation systems, it was found that almost all of them used Walker constellation layouts. Therefore, some simplifications derived from Walker constellation symmetry and dynamics could be applied. North-South symmetry simplification is the assumption that satellites in Northern and Southern hemisphere are distributed in the same manner. This simplification was usually applied by default. Longitude averaging reduced the computational time but still required a full propagation. Snapshot simplification was an observation made during the analysis that allowed to avoid propagating satellites and just look at their initial states. Applying these simplifications together with using a GPU could accelerate the visibility computation by a remarkable factor of 120,000x while maintaining a sufficient accuracy.

A parametric analysis was performed and it revealed that the mean visibility did not depend on P and F. However, they could affect the minimum visibility significantly because satellites could form a pattern that left gaps with very low visibility, i.e. they were distributed non-uniformly. Therefore, P and F were removed from the optimisation process if the FoM was the mean visibility which reduced the number of design variables and accelerated optimisation process.

It was also shown that the visibility curve scaled linearly with *N*, which allowed to propagate a smaller constellation and then just multiply it by a desired factor. It was observed that the minimum visibility was typically around 10% lower than the mean visibility if satellites were distributed uniformly. This assumption could be used during the constellation design.

All of this showed that the visibility computation could be accelerated by using vectorisation (software acceleration), GPUs (hardware acceleration), and knowledge of Walker constellation properties (simplifications). These methods could be used later in the constellation design.

Subquestion 3: *How can the defined figures of merit be optimised for a multi-shell constellation layout?*

The new methods for the faster visibility computation, that can be used for multi-shell constellation design, are presented in Chapter 5. A case study for a broadband mega-constellation over Europe latitudes (35 to 70 deg) was conducted where the number of shells was varied from one to three. It showed that higher-inclination shells had more satellites if all shells were nearly at the same altitude, in case of two- and three-shell layouts. The number of satellites per shell decreased with lower inclination.

Propagating each multi-shell constellation layout was much less efficient than propagating individual shells, storing them on disk memory, and then adding them by reopening stored arrays. This was much faster because reopening files required much less computational effort than complex simulations involving both CPU and GPU resources. This operation alone increased the performance by a factor of 100x-8,500x. However, this required enough disk memory space for data storage.

Scaling the visibility with N increased the performance even more because the data would be uploaded to RAM, which was much faster than disk memory. This method was 20x faster than the fastest method for shell permutation (reopening files + snapshot simplification). However, this method cannot be used to compute the minimum visibility. Therefore, this method can be used for the mean visibility computation, with the possibility of heuristically estimating the minimum visibility based on the mean value.

A new method called *building blocks* was developed. The main principle is that for mega-constellations that need coverage of a wide latitude band, we can split the target band into smaller sub-bands. In each sub-band, the permutation method with a selected number of shells can be applied. The design should start from higher inclinations and move towards the equator. This is because higher-inclination shells have higher visibility at higher latitudes, but they also cover lower latitudes, while lower-inclination shells do not cover higher latitudes.

This method is much faster than the permutation method: a factor of 150x compared to a three-shell design using the permutation method. For the study case, the final results are also consistent with the permutation method. This confirms that this approach can be used for multi-shell mega-constellation design.

This method was also used to improve the visibility of a mega-constellation presented by Jia et al. [51] that has 5,686 satellites distributed over 45 shells covering the whole globe from 0 to 90 deg latitudes. Their average visibility is very uniform over time, but the variations in visibility are very large, resulting in very low minimum visibility. The building blocks method was used to propose an alternative design with the same number of satellites. Without any optimisation of *P* and *F*, it already increased the minimum visibility by 5-10 satellites for a constellation with 5,700 satellites distributed over 8 shells.

Although prior research has been conducted and various mega-constellation systems have been both deployed and proposed, there remains a notable absence of a comprehensive methodology for their design optimisation in public domain. This thesis addresses this gap by providing a framework for mega-constellation design optimisation. Furthermore, this thesis presents the development of a software tool that can be used for such studies by engineers and researchers. This tool, in conjunction with the methodologies proposed in this report, allows for efficient mega-constellation design, significantly reducing computational time to mere seconds.

Recommendations for the future work are highlighted in the end of each chapter. The two most relevant improvements that can be applied to the current framework are following:

a) Find optimal values for the number of planes P and phasing parameter F (for each shell) to maximise minimum visibility
b) Use Earth's central angle instead of altitude h and minimum elevation angle ε_{min}

In conclusion, this report summarises results of an extensive research on characteristics of mega-constellations and their behaviour when the visibility as a figure of merit is considered. The `mcdo` tool, observations made and methods developed have been already applied to internal ESA projects indicating the relevance and demand for the outcomes of this research project.

References

[1] Mark Roberts, Christoph Beischl, and Sa'id Mosteshar. “Small satellite constellations: national security implications”. In: *Handbook of Small Satellites: Technology, Design, Manufacture, Applications, Economics and Regulation*. Springer, 2020, pp. 863–883.

[2] BryceTech. *Smallsats by the Numbers*. accessed 14 May 2024. July 2023. URL: https://brycetech.com/reports/report-documents/Bryce_Smallsats_2023.pdf.

[3] R David Luders. “Satellite networks for continuous zonal coverage”. In: *ARS Journal* 31.2 (1961), pp. 179–184.

[4] MH Ullock and AH Schoen. “Optimum polar satellite networks for continuous earth coverage”. In: *AIAA Journal* 1.1 (1963), pp. 69–72.

[5] David C Beste. “Design of satellite constellations for optimal continuous coverage”. In: *IEEE Transactions on Aerospace and Electronic Systems* 3 (1978), pp. 466–473.

[6] L Rider. “Analytic design of satellite constellations for zonal earth coverage using inclined circular orbits”. In: *Journal of the Astronautical Sciences* 34 (1986), pp. 31–64.

[7] John Gerard Walker. *Circular orbit patterns providing continuous whole earth coverage*. Tech. rep. Royal Aircraft Establishment Farnborough (United Kingdom), 1970.

[8] John Gerard Walker. *Coverage Predictions and Selection Criteria for Satellite Constellations.* Tech. rep. ROYAL AIRCRAFT ESTABLISHMENT FARNBOROUGH (ENGLAND), 1982.

[9] John G Walker. “Satellite constellations”. In: *Journal of the British Interplanetary Society* 37 (1984), p. 559.

[10] Arthur H Ballard. “Rosette constellations of earth satellites”. In: *IEEE transactions on aerospace and electronic systems* 5 (1980), pp. 656–673.

[11] WS Adams and L Rider. “Circular polar constellations providing continuous single or multiple coverage above a specified latitude”. In: *Journal of the Astronautical Sciences* 35 (1987), pp. 155–192.

[12] Thomas J Lang. “Symmetric circular orbit satellite constellations for continuous global coverage”. In: *Astrodynamics 1987* (1988), pp. 1111–1132.

[13] John M Hanson and William Burley Higgins. “Designing good geosynchronous constellations”. In: *Journal of the Astronautical Sciences* 38 (1990), pp. 143–159.

[14] John E Draim. “A common-period four-satellite continuous global coverage constellation”. In: *Journal of Guidance, Control, and Dynamics* 10.5 (1987), pp. 492–499.

[15] G Rondinelli, A Cramarossa, and F Graziani. “Orbit control strategy for a constellation of three satellites in TUNDRA orbits”. In: *Astrodynamics 1989* (1990), pp. 697–708.

[16] GV Mozhaev. “Problem of continuous coverage of the Earth and kinematically regular satellite systems”. In: *Kosm. Issled.* 11.1 (1973), pp. 59–69.

[17] Yuri Ulybyshev. “Near-polar satellite constellations for continuous global coverage”. In: *Journal of spacecraft and rockets* 36.1 (1999), pp. 92–99.

[18] Yuri Ulybyshev. “Satellite constellation design for continuous coverage: short historical survey, current status and new solutions”. In: *Proceedings of Moscow Aviation Institute* 13.34 (2009), pp. 1–25.

[19] Yuri Ulybyshev. “Geometric analysis and design method for discontinuous coverage satellite constellations”. In: *Journal of Guidance, Control, and Dynamics* 37.2 (2014), pp. 549–557.

[20] Yury N Razoumny. “Fundamentals of the route theory for satellite constellation design for Earth discontinuous coverage. Part 1: Analytic emulation of the Earth coverage”. In: *Acta Astronautica* 128 (2016), pp. 722–740.

[21] Daniele Mortari, Matthew P Wilkins, and Christian Bruccoleri. “The flower constellations”. In: *Journal of Astronautical Sciences* 52.1 (2004), pp. 107–127.

[22] Marina Ruggieri et al. “The flower constellation set and its possible applications”. In: *ACT Final report, Aridana* 5 (2006), p. 4108.

[23] Martín E Avendaño, Jeremy J Davis, and Daniele Mortari. “The 2-D lattice theory of Flower Constellations”. In: *Celestial Mechanics and Dynamical Astronomy* 116 (2013), pp. 325–337.

[24] Jeremy J Davis, Martín E Avendaño, and Daniele Mortari. “The 3-D lattice theory of flower constellations”. In: *Celestial Mechanics and Dynamical Astronomy* 116 (2013), pp. 339–356.

[25] David Arnas, Daniel Casanova, and Eva Tresaco. “4D Lattice flower constellations”. In: *Advances in Space Research* 67.11 (2021), pp. 3683–3695.

[26] Gerard Maral et al. “Low earth orbit satellite systems for communications”. In: *International Journal of satellite communications* 9.4 (1991), pp. 209–225.

[27] Erick Lansard, Eric Frayssinhes, and Jean-Luc Palmade. “Global design of satellite constellations: a multi-criteria performance comparison of classical walker patterns and new design patterns”. In: *Acta Astronautica* 42.9 (1998), pp. 555–564.

[28] F Dufour et al. “Constellation design optimization with a DOP based criterion”. In: *14th Int. Symposium On Space Fligt Dynamics*. 1995.

[29] William A. Crossley and Edwin A. Williams. “Simulated annealing and genetic algorithm approaches for discontinuous coverage satellite constellation design”. In: *Engineering Optimization* 32 (3 2000), pp. 353–371. ISSN: 0305215X. DOI: 10.1080/03052150008941304.

[30] T A Ely, W A Crossley, and E A Williams. *Satellite Constellation Design for Zonal Coverage Using Genetic Algorithms 1*, pp. 207–228.

[31] Kai Xie et al. “LEO space-based radar constellation design using a genetic algorithm”. In: Institute of Electrical and Electronics Engineers Inc., 2006. ISBN: 0780395824. DOI: 10.1109/ICR.2006.343500.

[32] Miguel A Nunes. “Satellite constellation optimization method for future earth observation missions using small satellites”. In: *Advances in the Astronautical Sciences* 146 (2013), pp. 159–179.

[33] Hongrae Kim and Young Keun Chang. “Mission scheduling optimization of SAR satellite constellation for minimizing system response time”. In: *Aerospace Science and Technology* 40 (Jan. 2015), pp. 17–32. ISSN: 12709638. DOI: 10.1016/j.ast.2014.10.006.

[34] Yunjoong Kim et al. “Optimum design of an SAR satellite constellation considering the revisit time using a genetic algorithm”. In: *International Journal of Aeronautical and Space Sciences* 18 (2 June 2017), pp. 334–343. ISSN: 20932480. DOI: 10.5139/IJASS.2017.18.2.334.

[35] Xin Luo et al. “Constellation design for earth observation based on the characteristics of the satellite ground track”. In: *Advances in Space Research* 59 (7 Apr. 2017), pp. 1740–1750. ISSN: 18791948. DOI: 10.1016/j.asr.2017.01.010.

[36] Nicholas H. Crisp, Sabrina Livadiotti, and Peter C. E. Roberts. “A Semi-Analytical Method for Calculating Revisit Time for Satellite Constellations with Discontinuous Coverage”. In: (July 2018). URL: http://arxiv.org/abs/1807.02021.

[37] Douglas J Pegher and Jason A Parish. *Optimizing coverage and revisit time in sparse military satellite constellations: A comparison of traditional approaches and genetic algorithms*. Tech. rep. NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 2004.

[38] A T Takano and B G Marchand. *AAS 11-543 OPTIMAL CONSTELLATION DESIGN FOR SPACE BASED SITUATIONAL AWARENESS APPLICATIONS*. 2011.

[39] Zhicheng Qu et al. “LEO Satellite Constellation for Internet of Things”. In: *IEEE Access* 5 (Aug. 2017), pp. 18391–18401. ISSN: 21693536. DOI: 10.1109/ACCESS.2017.2735988.

[40] M. Asvial, R. Tafazolli, and B. G. Evans. “Genetic hybrid Satellite Constellation design”. In: American Institute of Aeronautics and Astronautics Inc., 2003. ISBN: 9781624100932. DOI: 10.2514/6.2003-2283.

[41] Irene A. Budianto and John R. Olds. “Design and deployment of a satellite constellation using collaborative optimization”. In: *Journal of Spacecraft and Rockets* 41 (6 2004), pp. 956–963. ISSN: 15336794. DOI: 10.2514/1.14254.

[42] Matthew P. Ferringer and David B. Spencer. “Satellite constellation design tradeoffs using multiple-objective evolutionary computation”. In: *Journal of Spacecraft and Rockets* 43 (6 2006), pp. 1404–1411. ISSN: 15336794. DOI: 10.2514/1.18788.

[43] Matthew P. Ferringer, Ronald S. Clifton, and Timothy G. Thompson. “Efficient and accurate evolutionary multi-objective optimization paradigms for satellite constellation design”. In: *Journal of Spacecraft and Rockets* 44 (3 2007), pp. 682–691. ISSN: 15336794. DOI: 10.2514/1.26747.

[44] I. Meziane-Tani et al. “Optimization of small satellite constellation design for continuous mutual regional coverage with multi-objective genetic algorithm”. In: *International Journal of Computational Intelligence Systems* 9 (4 July 2016), pp. 627–637. ISSN: 18756883. DOI: 10.1080/18756891.2016.1204112.

[45] Xiaohui Wang et al. “Design of agile satellite constellation based on hybrid-resampling particle swarm optimization method”. In: *Acta Astronautica* 178 (Jan. 2021), pp. 595–605. ISSN: 00945765. DOI: 10.1016/j.actaastro.2020.09.040.

[46] Xingchi He and Urs Hugentobler. “Design of Mega-Constellations of LEO Satellites for Positioning”. In: vol. 497. Springer Verlag, 2018, pp. 663–673. ISBN: 9789811300042. DOI: 10.1007/978-981-13-0005-9_54.

[47] Haibo Ge et al. “LEO constellation optimization for LEO enhanced global navigation satellite system (LeGNSS)”. In: *Advances in Space Research* 66 (3 Aug. 2020), pp. 520–532. ISSN: 18791948. DOI: 10.1016/j.asr.2020.04.031.

[48] Fujian Ma et al. “Hybrid constellation design using a genetic algorithm for a LEO-based navigation augmentation system”. In: *GPS Solutions* 24 (2 Apr. 2020). ISSN: 15211886. DOI: 10.1007/s10291-020-00977-0.

[49] Jing Liu et al. “Design optimisation of low earth orbit constellation based on BeiDou Satellite Navigation System precise point positioning”. In: *IET Radar, Sonar and Navigation* 16 (8 Aug. 2022), pp. 1241–1252. ISSN: 17518792. DOI: 10.1049/rsn2.12257.

[50] Ahan Kak and Ian F. Akyildiz. “Designing Large-Scale Constellations for the Internet of Space Things with CubeSats”. In: *IEEE Internet of Things Journal* 8 (3 Feb. 2021), pp. 1749–1768. ISSN: 23274662. DOI: 10.1109/JIOT.2020.3016889.

[51] Lu Jia et al. “Design of Mega-Constellations for Global Uniform Coverage with Inter-Satellite Links”. In: *Aerospace* 9 (5 May 2022). ISSN: 22264310. DOI: 10.3390/aerospace9050234.

[52] Lloyd Wood. “Satellite Constellation Networks: The path from orbital geometry through network topology to autonomous systems”. In: *Internetworking and Computing over Satellite Networks* (2003), pp. 13–34.

[53] Arthur K Lacombe. “Mega-Constellations: Technical Aspects”. In: *Promoting Productive Cooperation Between Space Lawyers and Engineers*. IGI Global, 2019, pp. 114–140.

[54] Kuiper Systems LLC. *TECHNICAL APPENDIX Application of Kuiper Systems LLC for Authority to Launch and Operate a Non-Geostationary Satellite Orbit System in V-band and Ku-band Frequencies*. SAT-LOA-20211104-00145. FCC, 2021.

[55] INC. SPIRE GLOBAL. *ITU Satellite Network Filing: LEMUR-2-3*. Notice ID: 120545231, Submission Reference Number: USA2020-33727. 2020. URL: https://www.itu.int/ITU-R/space/asreceived/Publication/DisplayPublication/23797.

[56] Israel Leyva-Mayorga et al. “NGSO Constellation Design for Global Connectivity”. In: (Mar. 2022). URL: http://arxiv.org/abs/2203.16597.

[57] S. Huang, C. Colombo, and F. Bernelli-Zazzera. “Multi-criteria design of continuous global coverage Walker and Street-of-Coverage constellations through property assessment”. In: *Acta Astronautica* 188 (Nov. 2021), pp. 151–170. ISSN: 00945765. DOI: 10.1016/j.actaastro.2021.07.002.

[58] Kris Maine, Carrie Devieux, and Pete Swan. “Overview of IRIDIUM satellite network”. In: *Proceedings of WESCON’95*. IEEE. 1995, p. 483.

[59] Carl E Fossa et al. “An overview of the IRIDIUM (R) low Earth orbit (LEO) satellite system”. In: *Proceedings of the IEEE 1998 National Aerospace and Electronics Conference. NAECON 1998. Celebrating 50 Years (Cat. No. 98CH36185)*. IEEE. 1998, pp. 152–159.

[60] Iridium Communications. *Iridium NEXT: In Review*. accessed 14 May 2024. 2023. URL: https://www.iridium.com/blog/iridium-next-review/.

[61] Jonathan McDowelll. *Jonathan’s Space Pages, Starlink Statistics*. accessed 14 May 2024. 2023. URL: https://planet4589.org/space/con/star/stats.html.

[62] Space Exploration Holdings LLC. *SpaceX Non-Geostationary Satellite System. Attachment A: Technical Information To Supplement Schedule S*. SAT-MOD-20190830-00087. FCC, 2019.

[63] Space Exploration Holdings LLC. *SpaceX V-band Non-Geostationary Satellite System. Attachment A: Technical Information To Supplement Schedule S*. SAT-LOA-20170726-00110. FCC, 2017.

[64] Space Exploration Holdings LLC. *SpaceX Gen2 Non-Geostationary Satellite System*. SAT-LOA-20200526-00055. FCC, 2020.

[65] MARVEL SPACE COMMUNICATIONS CO. *USASAT-NGSO-MVLS*. Notice ID: 123520186, Submission Reference Number: USA2023-65024. Dec. 2023.

[66] OneWeb. *OneWeb Non-Geostationary Satellite System (LEO) Phase 1: Modification to Authorized System*. SAT-MPL-20200526-00062, A Technical Information to Supplement Schedule S. FCC, 2020.

[67] OneWeb. *OneWeb Satellites and partners OneWeb and Airbus transform space industry with world’s first high-volume satellite production facility in Florida*. accessed 14 May 2024. 2020. URL: https://oneweb.net/resources/oneweb-satellites-and-partners-oneweb-and-airbus-transform-space-industry-worlds-first.

[68] OneWeb. *ONEWEB NON-GEOSTATIONARY SATELLITE SYSTEM (LEO) PHASE 2: AMENDED MODIFICATION TO AUTHORIZED SYSTEM ATTACHMENT B Technical Information to Supplement Schedule S*. SAT-MPL-20210112-00007. FCC, 2021.

[69] SN Space Systems Ltd. *SN Space Systems Limited Non-Geostationary Orbit Satellite System*. SAT-PDR-20211104-00147. FCC, 2021.

[70] Kepler Communications Inc. *Schedule S Technical Report. Phase 1*. SAT-PDR-20161115-00114. FCC, 2016.

[71] Kepler Communications Inc. *Application for Market Access Authority for a Non-Geostationary Satellite Orbit System in Ka- and Ku-band Frequencies. Phase 2*. SAT-LOA-20200526-00059. FCC, 2020.

[72] MARVEL SPACE COMMUNICATIONS CO. *ITU Satellite Network Filing: CINNAMON-217*. Notice ID: 121545225, Submission Reference Number: RRW2021-42538. 2021.

[73] MARVEL SPACE COMMUNICATIONS CO. *ITU Satellite Network Filing: CINNAMON-937*. Notice ID: 121545224, Submission Reference Number: RRW2021-42537. 2021.

[74] Cuiqin Dai, Guimin Zheng, and Qianbin Chen. “Satellite constellation design with multi-objective genetic algorithm for regional terrestrial satellite network”. In: *China Communications* 15 (8 Aug. 2018), pp. 1–10. ISSN: 16735447. DOI: 10.1109/CC.2018.8438269.

[75] Nozomi Hitomi and Daniel Selva. “Constellation optimization using an evolutionary algorithm with a variable-length chromosome”. In: vol. 2018-March. IEEE Computer Society, June 2018, pp. 1–12. ISBN: 9781538620144. DOI: 10.1109/AERO.2018.8396743.

[76] Chen Zhang et al. “LEO constellation design methodology for observing multi-targets”. In: *Astrodynamics* 2 (2 June 2018), pp. 121–131. ISSN: 2522-008X. DOI: 10.1007/s42064-017-0015-4.

[77] Deng Pan et al. “LEO Constellation Optimization Model with Non-uniformly Distributed RAAN for Global Navigation Enhancement”. In: vol. 497. Springer Verlag, 2018, pp. 769–778. ISBN: 9789811300042. DOI: 10.1007/978-981-13-0005-9_63.

[78] Tian Jiao Zhang et al. “Restricted constellation design for regional navigation augmentation”. In: *Acta Astronautica* 150 (Sept. 2018), pp. 231–239. ISSN: 00945765. DOI: 10.1016/j.actaastro.2018.04.044.

[79] Hang Woon Lee et al. “Satellite constellation pattern optimization for complex regional coverage”. In: *Journal of Spacecraft and Rockets* 57 (6 2020), pp. 1309–1327. ISSN: 15336794. DOI: 10.2514/1.A34657.

[80] Huijiang Wang and Shengzhou Bai. “A versatile method for target area coverage analysis with arbitrary satellite attitude maneuver paths”. In: *Acta Astronautica* 194 (May 2022), pp. 242–254. ISSN: 00945765. DOI: 10.1016/j.actaastro.2022.02.008.

[81] Tania Savitri et al. “Satellite Constellation Orbit Design Optimization with Combined Genetic Algorithm and Semianalytical Approach”. In: *International Journal of Aerospace Engineering* 2017 (2017). ISSN: 16875974. DOI: 10.1155/2017/1235692.

[82] Jaume Sanz Subirana, José Miguel Juan Zornoza, and Manuel Hernández-Pajares. *GNSS Data Processing. Vol. 1: Fundamentals and Algorithms (ESA TM-23/1, May 2013)*. ESA Communications, 2013. ISBN: 978-92-9221-886-7.

[83] Yuehao Teng, Xiaolin Jia, and Ge Peng. “LEO navigation augmentation constellation design and precise point positioning performance analysis based on BDS-3”. In: *Advances in Space Research* (2023).

[84] Ju Hong et al. “GNSS rapid precise point positioning enhanced by low Earth orbit satellites”. In: *Satellite Navigation* 4.1 (2023), pp. 1–13.

[85] F ea Poletti et al. “Towards high-capacity fibre-optic communications at the speed of light in vacuum”. In: *Nature Photonics* 7.4 (2013), pp. 279–284.

[86] Zeqi Lai, Hewu Li, and Jihao Li. “Starperf: Characterizing network performance for emerging mega-constellations”. In: *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE. 2020, pp. 1–11.

[87] Zeqi Lai et al. “Spacertc: Unleashing the low-latency potential of mega-constellations for real-time communications”. In: *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE. 2022, pp. 1339–1348.

[88] Mark Handley. “Using ground relays for low-latency wide-area routing in megaconstellations”. In: *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*. 2019, pp. 125–132.

[89] Gaofeng Pan et al. “Latency versus reliability in LEO mega-constellations: Terrestrial, aerial, or space relay”. In: *IEEE Transactions on Mobile Computing* (2022).

[90] Ali J Alqaraghuli, Hussam Abdellatif, and Josep M Jornet. “Performance analysis of a dual terahertz/ka band communication system for satellite mega-constellations”. In: *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE. 2021, pp. 316–322.

[91] Channasandra Ravishankar et al. “Next-generation global satellite system with mega-constellations”. In: *International journal of satellite communications and networking* 39.1 (2021), pp. 6–28.

[92] Haoge Jia et al. “Uplink interference and performance analysis for megasatellite constellation”. In: *IEEE Internet of Things Journal* 9.6 (2021), pp. 4318–4329.

[93] Niloofar Okati and Taneli Riihonen. “Coverage and rate analysis of mega-constellations under generalized serving satellite selection”. In: *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2022, pp. 2214–2219.

[94] J.R. Wertz. *Mission Geometry: Orbit and Constellation Design and Management. Spacecraft Orbit and Attitude Systems*. Springer Dordrecht, 2002. ISBN: 978-0-7923-7148-9.

[95] Karel F Wakker. *Fundamentals of astrodynamics*. TU Delft Library, 2015.

[96] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[97] Márton Geda, Florian Renk, and Ron Noomen. “Massive Parallelization of Trajectory Propagations Using GPUs”. In: (2019).

[98] Yupeng Gong, Shijie Zhang, and Xuan Peng. “Quick coverage analysis of mega Walker Constellation based on 2D map”. In: *Acta Astronautica* 188 (2021), pp. 99–109.

[99] Maocai Wang et al. “Application of latitude stripe division in satellite constellation coverage to ground”. In: *International Journal of Aerospace Engineering* 2016 (2016).

[100] Guangming Dai et al. “Analysis of satellite constellations for the continuous coverage of ground regions”. In: *Journal of Spacecraft and Rockets* 54.6 (2017), pp. 1294–1303.

[101] Maurice Herlihy et al. *The Art of Multiprocessor Programming*. Morgan Kaufmann Publishers Inc., 2020. ISBN: 9780124159501.

[102] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. *pybind11 – Seamless operability between C++11 and Python*. https://github.com/pybind/pybind11. 2017.

[103] Osoro B Ogutu and Edward J Oughton. “A techno-economic cost framework for satellite networks applied to low earth orbit constellations: Assessing starlink, oneweb and kuiper”. In: *arXiv preprint arXiv:2108.10834* (2021).

[104] Nvidia. *NVIDIA TESLA V100 GPU ACCELERATOR. The Most Advanced Data Center GPU Ever Built.* accessed 14 May 2024. 2018. URL: https://images.nvidia.com/content/technologies/volta/pdf/tesla-volta-v100-datasheet-letter-fnl-web.pdf.

[105] TechPowerUp. *NVIDIA Quadro P2000. Quadro P2000 Specs.* accessed 14 May 2024. 2024. URL: https://www.techpowerup.com/gpu-specs/quadro-p2000.c2931.

[106] *Massive GPU Parallelisation for Cislunar Debris Mitigation Analyses*. IAC-22,A6,IP,12,x68521. 73rd International Astronautical Congress (IAC), 2022.

[107] David Culler, Jaswinder Pal Singh, and Anoop Gupta. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann Publishers Inc., 1999. ISBN: 978-0-08-057307-6.

[108] Milind Girkar and Constantine D Polychronopoulos. “Extracting task-level parallelism”. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 17.4 (1995), pp. 600–634.

[109] Roger Espasa and Mateo Valero. “Exploiting instruction-and data-level parallelism”. In: *IEEE micro* 17.5 (1997), pp. 20–27.

[110] Michael J Flynn. “Very high-speed computing systems”. In: *Proceedings of the IEEE* 54.12 (1966), pp. 1901–1909.

[111] Alfred Spector and David Gifford. “The space shuttle primary computer system”. In: *Communications of the ACM* 27.9 (1984), pp. 872–900.

[112] Iddo Hanniel and Kirk Haller. “Direct rendering of solid cad models on the gpu”. In: *2011 12th International Conference on Computer-Aided Design and Computer Graphics*. IEEE. 2011, pp. 25–32.

[113] Dennis C Jespersen. “Acceleration of a CFD code with a GPU”. In: *Scientific Programming* 18.3-4 (2010), pp. 193–201.

[114] Tai-Sung Lee et al. “GPU-accelerated molecular dynamics and free energy methods in Amber18: performance enhancements and new features”. In: *Journal of chemical information and modeling* 58.10 (2018), pp. 2043–2050.

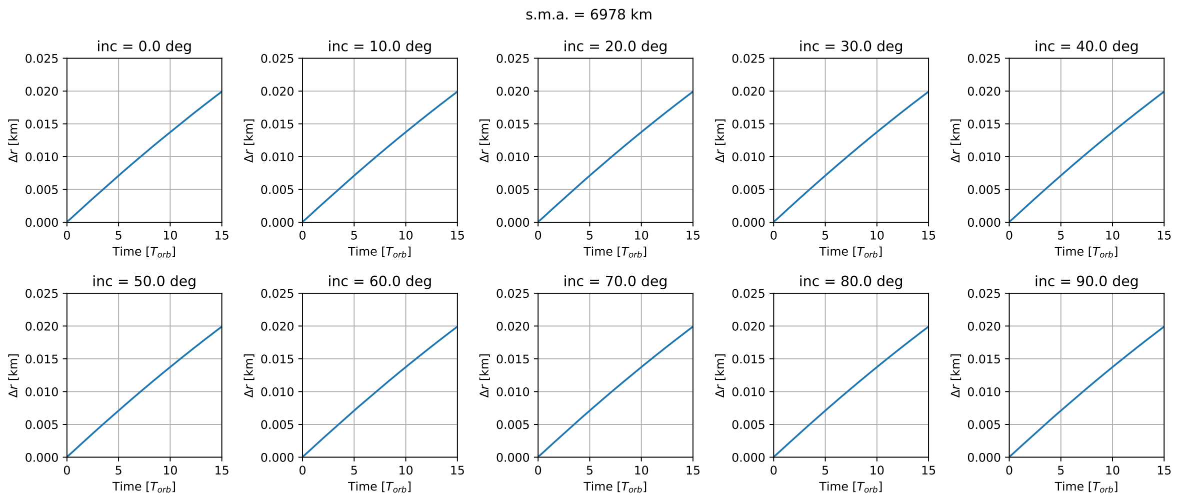
[115] Wikipedia. *Flynn’s taxonomy*. accessed 14 May 2024. 2023. URL: https://en.wikipedia.org/wiki/Flynn%27s_taxonomy.

[116] Ryosuke 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. URL: http://learningsys.org/nips17/assets/papers/paper_16.pdf.

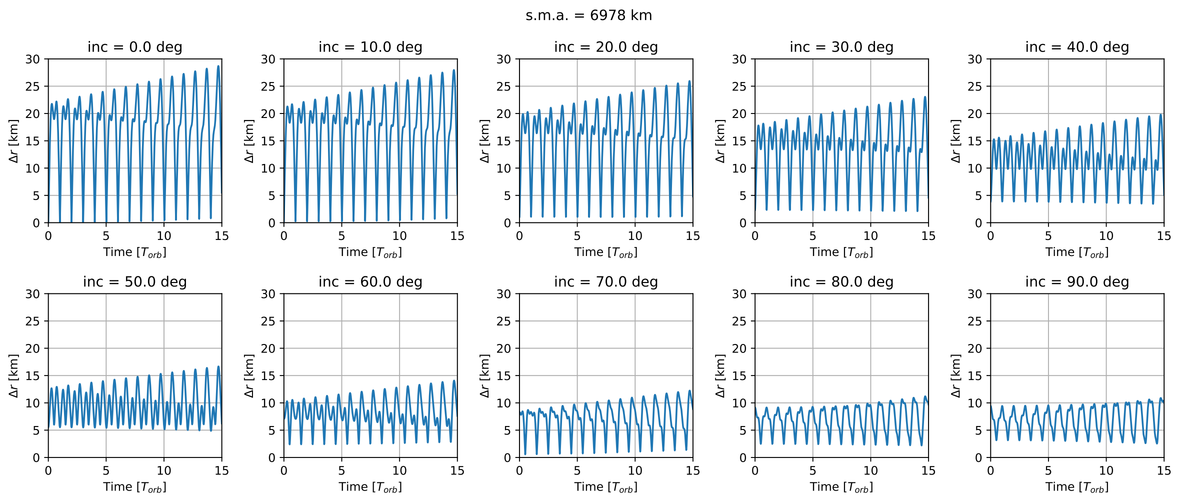
[117] Robert A. Dalrymple. *560.602 GPU/CPU Programming for Engineers. Class 13. GPU Memory.* John Hopkins University. 2014.

[118] John Nickolls et al. “Scalable Parallel Programming with CUDA: Is CUDA the parallel programming model that application developers have been waiting for?” In: *Queue* 6.2 (2008), pp. 40–53.

A
Propagation Verification

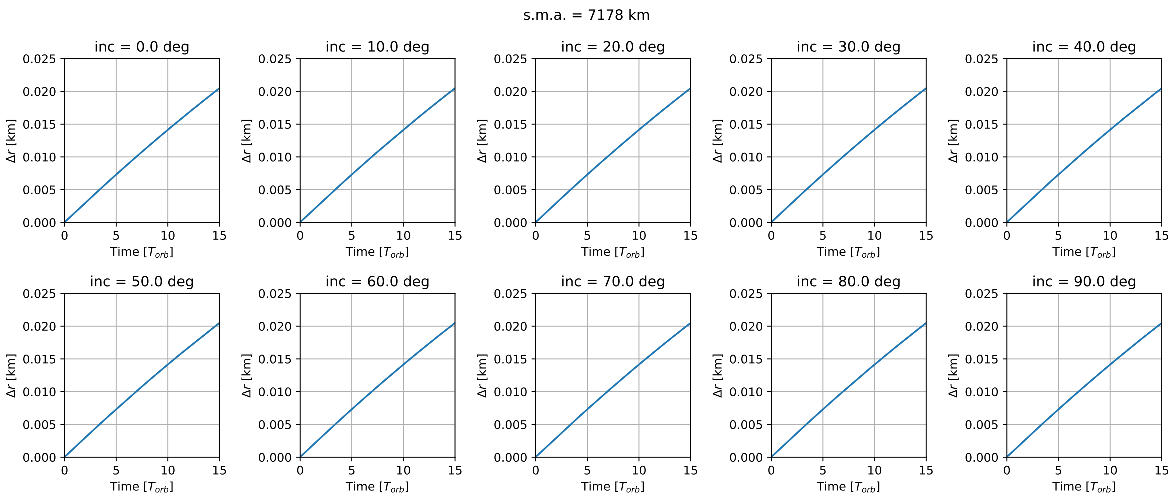


(a) Point-mass only.

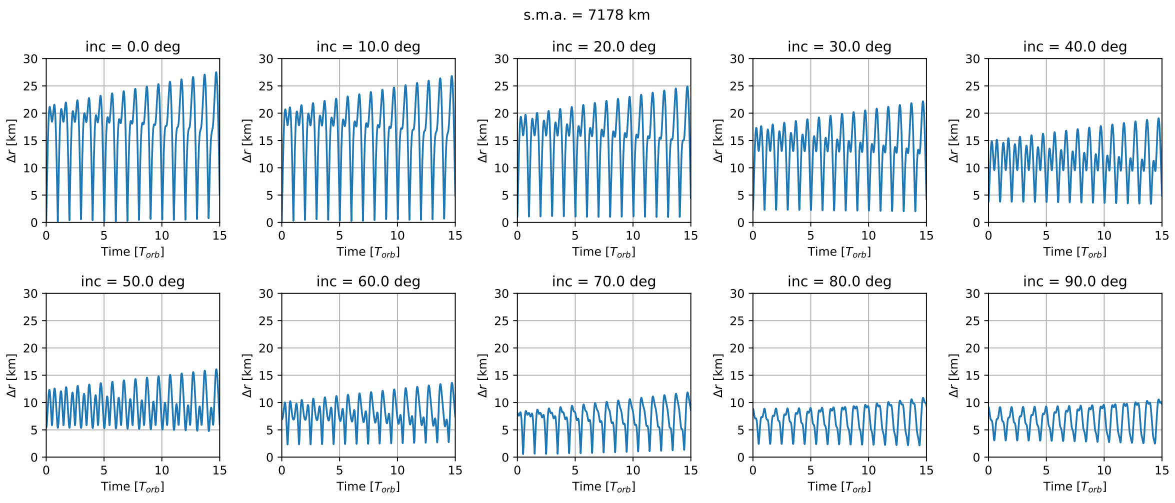


(b) Point-mass + J2.

Figure A.1: Positional error of analytical model with respect to numerical model for $a = 6978$ km.

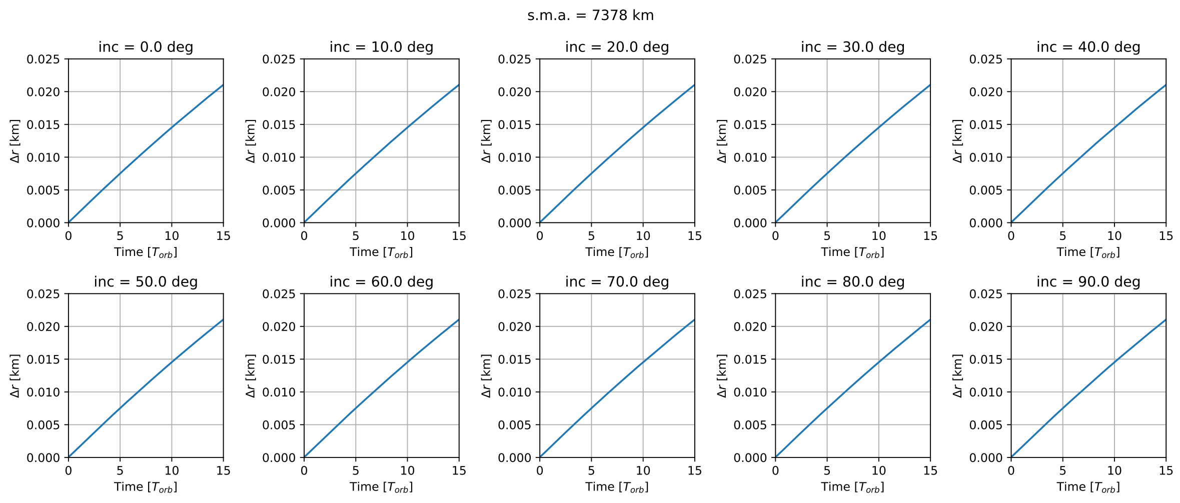


(a) Point-mass only.

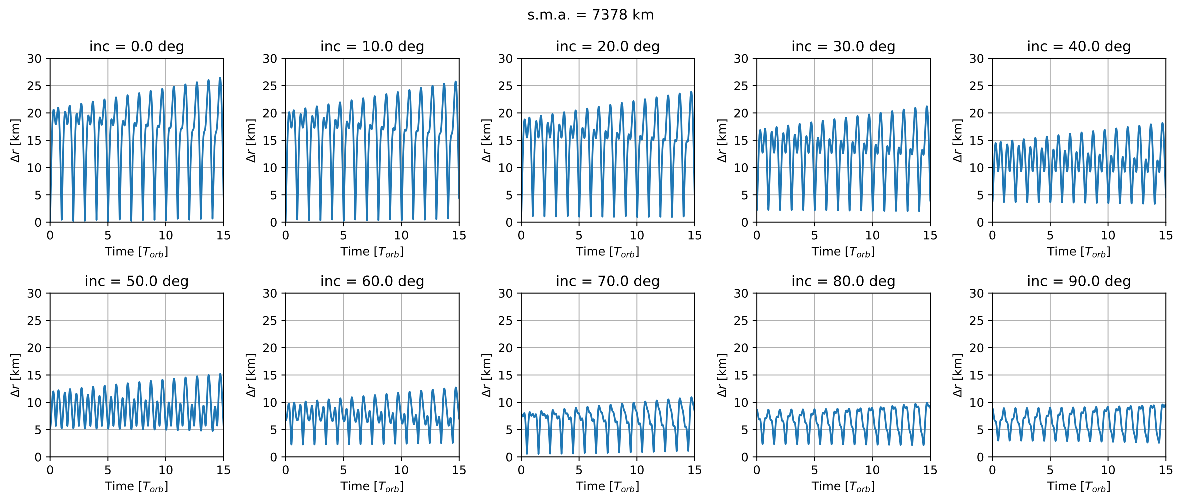


(b) Point-mass + J2.

Figure A.2: Positional error of analytical model with respect to numerical model for $a = 7178$ km.

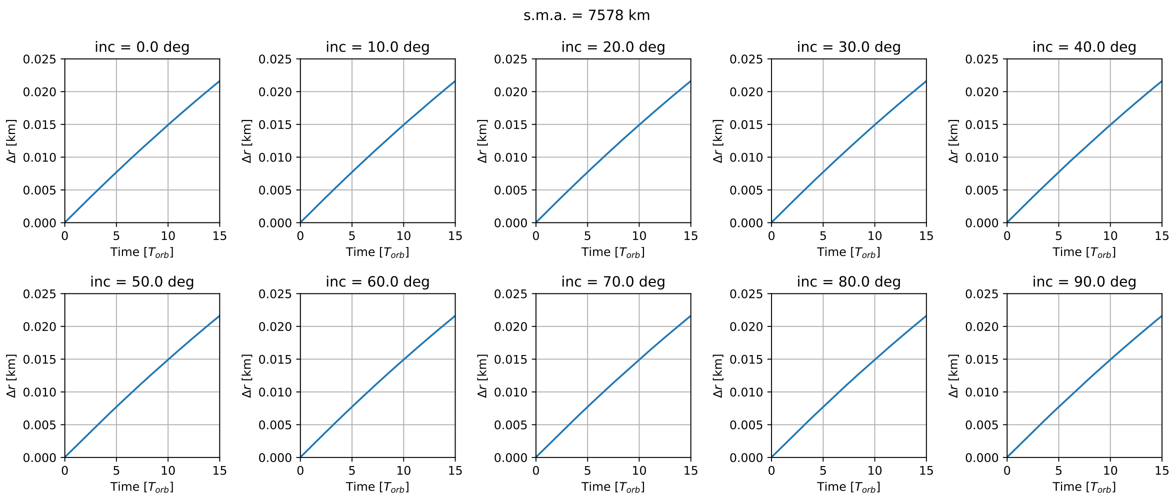


(a) Point-mass only.

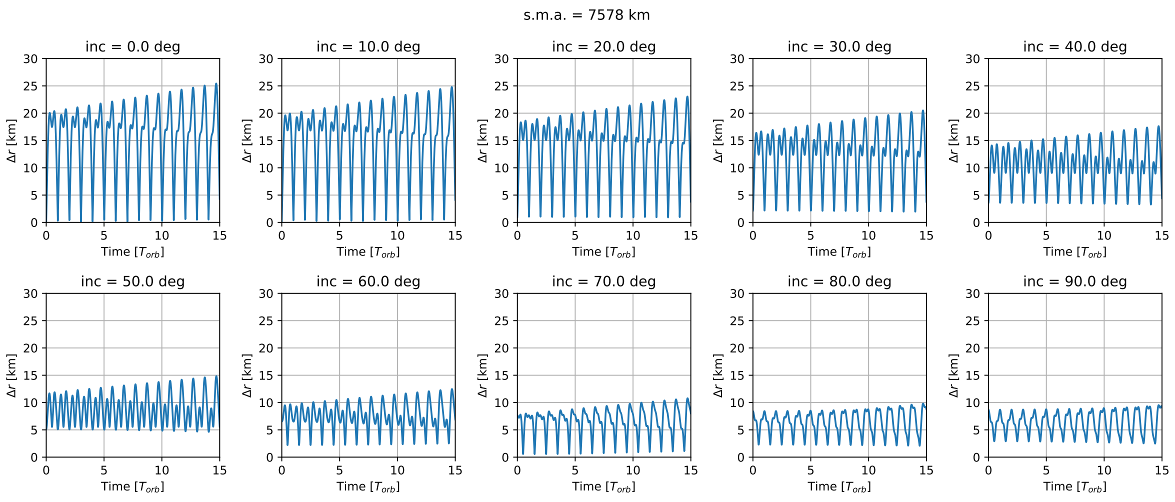


(b) Point-mass + J2.

Figure A.3: Positional error of analytical model with respect to numerical model for $a = 7378$ km.

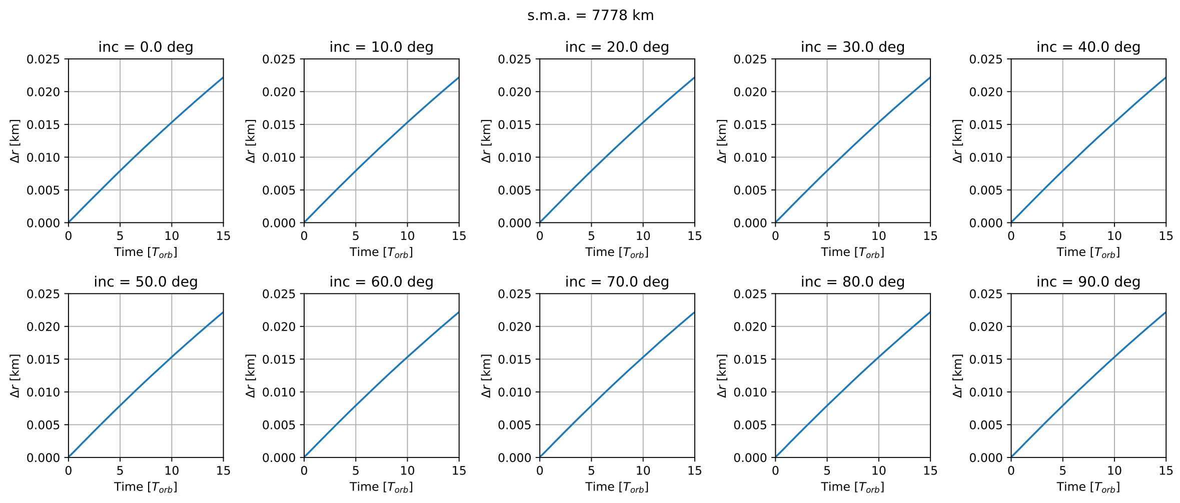


(a) Point-mass only.

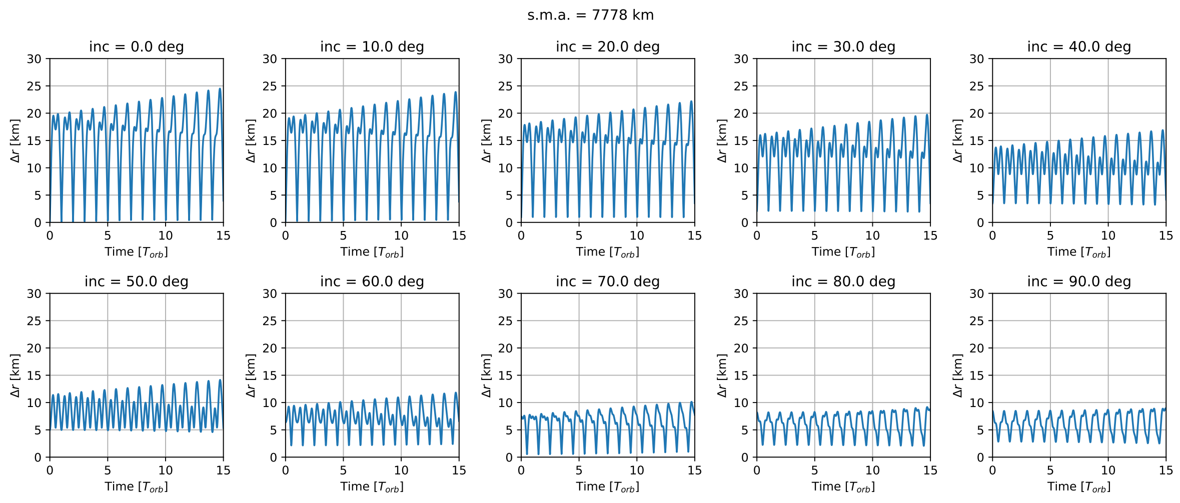


(b) Point-mass + J2.

Figure A.4: Positional error of analytical model with respect to numerical model for $a = 7578$ km.

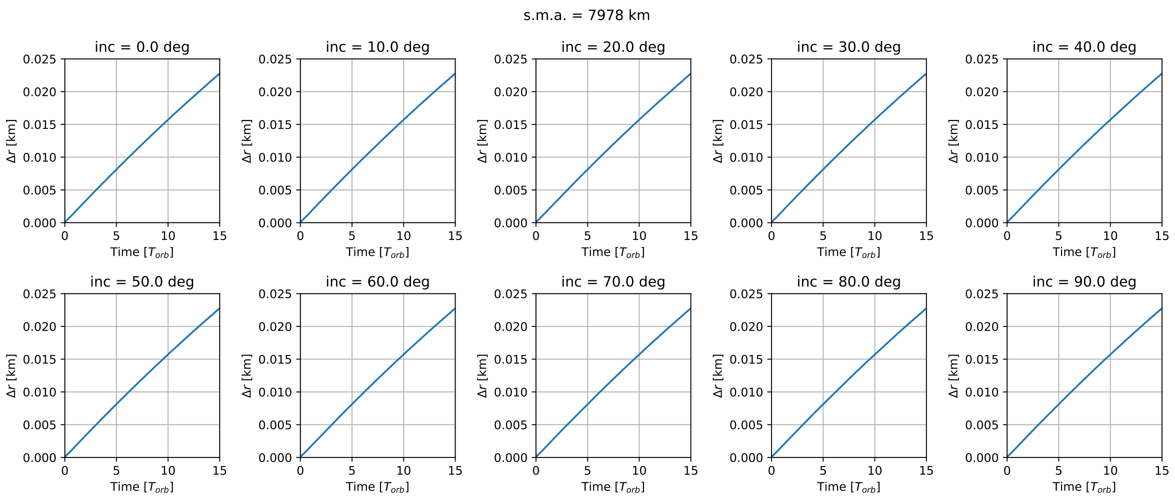


(a) Point-mass only.

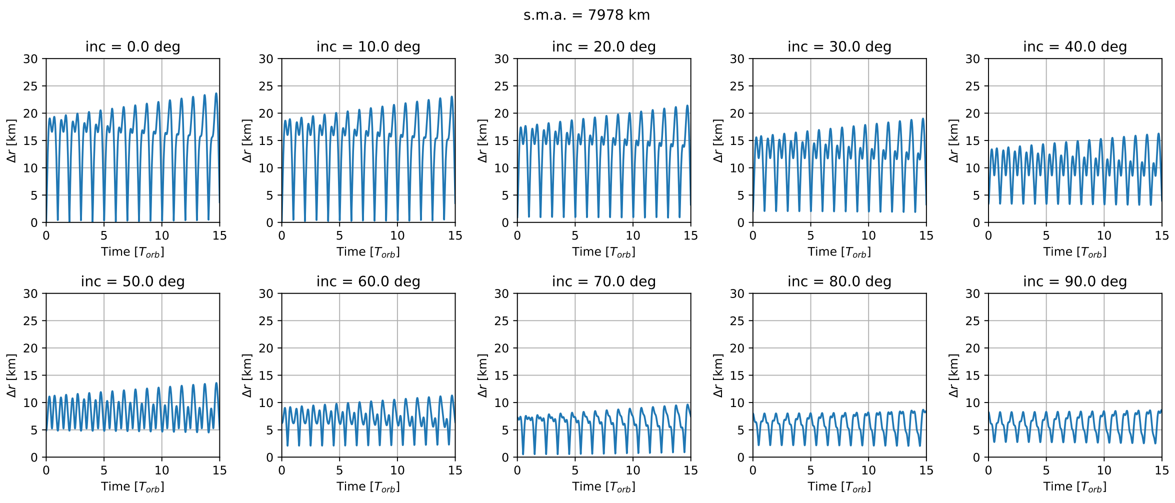


(b) Point-mass + J2.

Figure A.5: Positional error of analytical model with respect to numerical model for $a = 7778$ km.

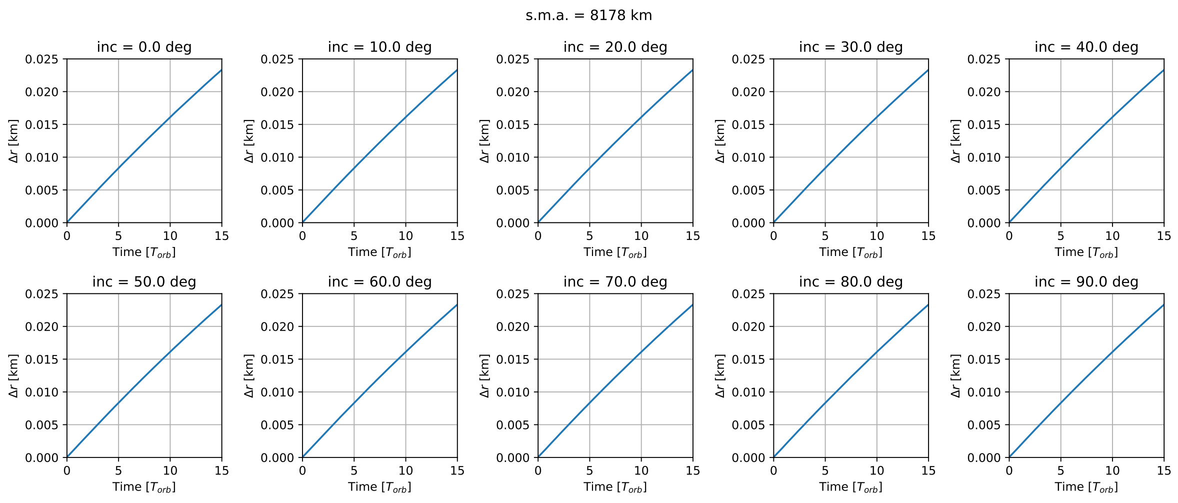


(a) Point-mass only.

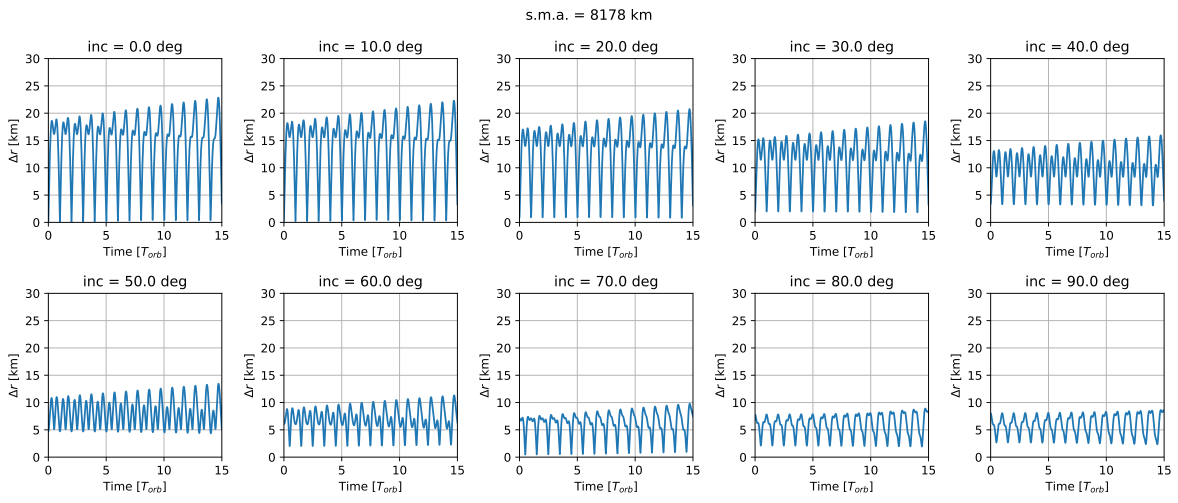


(b) Point-mass + J2.

Figure A.6: Positional error of analytical model with respect to numerical model for $a = 7978$ km.

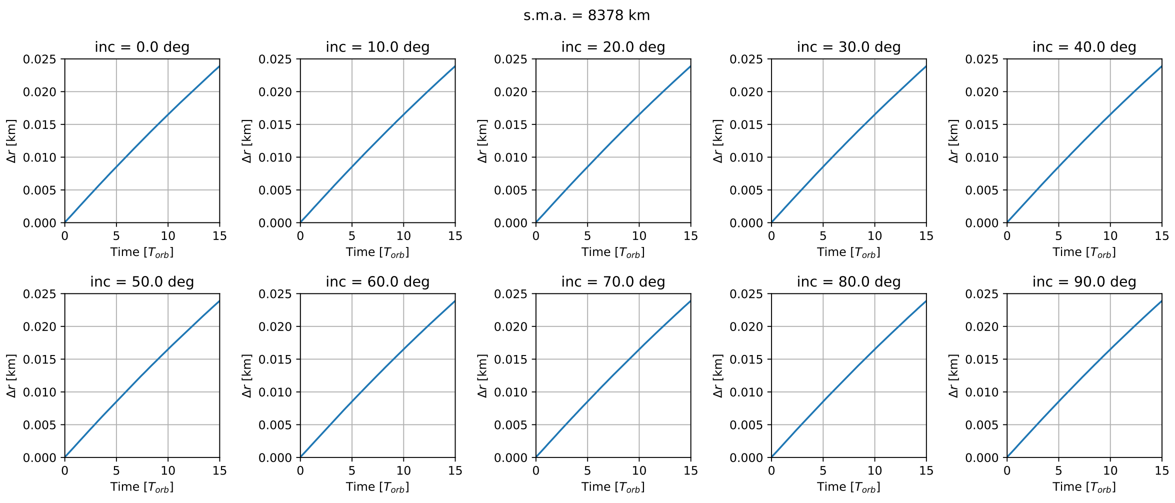


(a) Point-mass only.

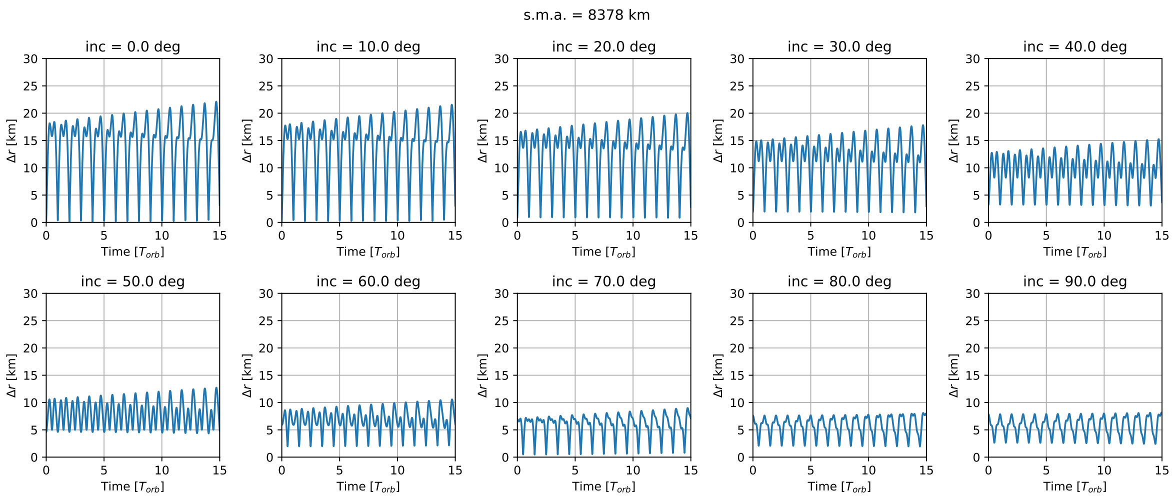


(b) Point-mass + J2.

Figure A.7: Positional error of analytical model with respect to numerical model for $a = 8178$ km.



(a) Point-mass only.



(b) Point-mass + J2.

Figure A.8: Positional error of analytical model with respect to numerical model for $a = 8378$ km.

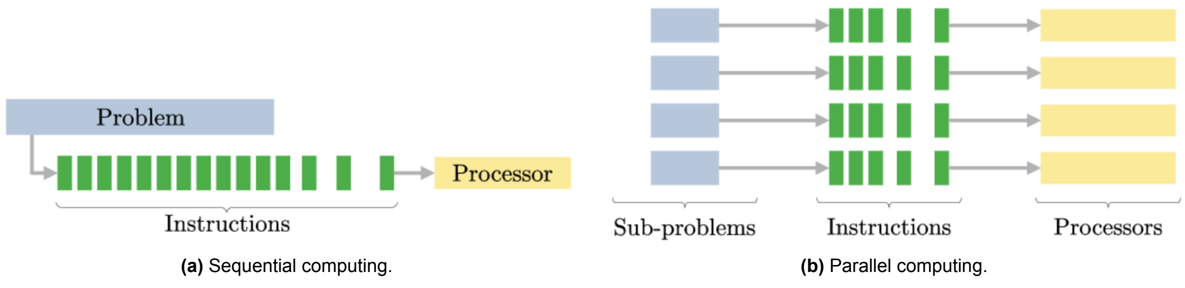
B

Parallel Computing

This section provides a brief introduction to the concepts of parallel computing and their application to visibility computation. Additionally, a discussion on the implementation and comparison with non-parallelised computation is presented. Finally, the limitations of this implementation are outlined, along with potential resolutions and workarounds.

B.1. Sequential and Parallel Computing

Sequential computing involves the decomposition of tasks into a series of instructions that are executed sequentially. While this approach is straightforward and easy to implement, it can be inefficient for a large number of tasks, leading to long computation times. Figure B.1a illustrates the fundamental principle of sequential computing. Instructions or problems can be reiterated, but the underlying principle remains constant: the execution of new tasks commences only upon the completion of previous tasks.



(a) Sequential computing. **(b)** Parallel computing.

Figure B.1: Comparison of sequential and parallel computing [106].

In contrast, parallel computing enables the simultaneous execution of tasks, significantly enhancing throughput, particularly for large tasks or datasets. The basic principle is shown in Figure B.1b. However, it necessitates meticulous task management and coordination. Implementation can be more complex as certain tasks may be interdependent or require shared resources. While parallel computing may appear better than sequential computing, this is not always the case, as not all tasks are parallelisable and may necessitate sequential implementation regardless. It is important to understand whether a task can be parallelised and, if so, how it can be subdivided into multiple sub-problems or sub-instructions for simultaneous execution.

There are different types of parallel computing and also different types of parallelism which are discussed in the next section.

B.2. Types of Parallelism and Parallel Processing

This subsection provides a brief overview of the types of parallelism and parallel processing. While the terms *parallelism* and *parallel processing* are often used interchangeably, this report distinguishes between the two.

Parallelism refers to the level at which computation is performed and includes 4 categories:

- **Bit-level parallelism**
  This form of parallel computing achieves parallelism by increasing the processor's size [107]. For instance, the addition of two 16-bit integers would need two instructions from an 8-bit processor, whereas a 16-bit processor would only require one instruction.
- **Instruction-level parallelism**
  This form of parallel computing executes multiple instruction sequences concurrently [107]. For example, modern CPUs often employ this type of parallelism to improve performance.
- **Task-level parallelism**
  This form of parallel computing executes tasks (also referred to as *instructions* or *functions*), which can be part of a larger application or program, in parallel [108]. These tasks can be executed independently or at least partially independently with synchronisation at some point. An example of this would be a web server handling multiple requests.
- **Data-level parallelism**
  This form of parallel computing executes instructions on multiple data points simultaneously [109]. This is often used in operations such as matrix multiplication where the same operation (multiplication and addition) is performed on different data points.

Parallel processing is referred to the categorisation introduced by Flynn's taxonomy [110]. This is a classification of architectural models for achieving parallelism. This includes 4 categories which are listed below and visually represented in Figure B.2.

- **SISD (Single Instruction stream, Single Data stream)**
  SISD is the simplest computer architecture where a single instruction is executed for a single data point. These operations are performed sequentially. Multiple instructions or data points would imply that those would have to be broken down in individual parts, and then operations would be performed one after other.
- **SIMD (Single Instruction stream, Multiple Data streams)**
  This architecture model implies a single instruction performed on multiple data points simultaneously. If there is more than one instruction, then they would have to be executed sequentially. This architecture is commonly used in vector processing as well as graphics.
- **MISD (Multiple Instruction streams, Single Data stream)**
  In a MISD architecture model multiple instructions are applied to the same data point simultaneously. This is a lesser used type of parallel computing architecture, compared to SIMD and MIMD, as there are not many applications requiring such a model. Typically, MISD is used for redundancy and fault tolerance in systems where a single error can be critical (e.g. Space Shuttle mission [111]).
- **MIMD (Multiple Instruction streams, Multiple Data streams)**
  MIMD architecture allows to execute multiple instructions on multiple data streams simultaneously. This is another common type of architecture used in parallel computing for such things as CAD [112], CFD [113], molecular dynamics [114], and others. This, however, also requires sophisticated programming techniques allowing to coordinate and synchronise all sub-tasks.

From this taxonomy, it is evident that the visibility computation with the Grid Point method is a SIMD-like operation, with a single instruction to compute the elevation angle applied to multiple data points, namely observation points and satellites. The elevation angle between each observation point and satellite can be computed independently, which facilitates the parallelisation of this task.

Typical hardware utilising SIMD architecture model are multi-core central processing units (CPU) and graphics processing units (GPU). Both types of hardware can be employed for visibility computation. However, each comes with its own set of advantages and disadvantages, which will be explored in the subsequent subsections.

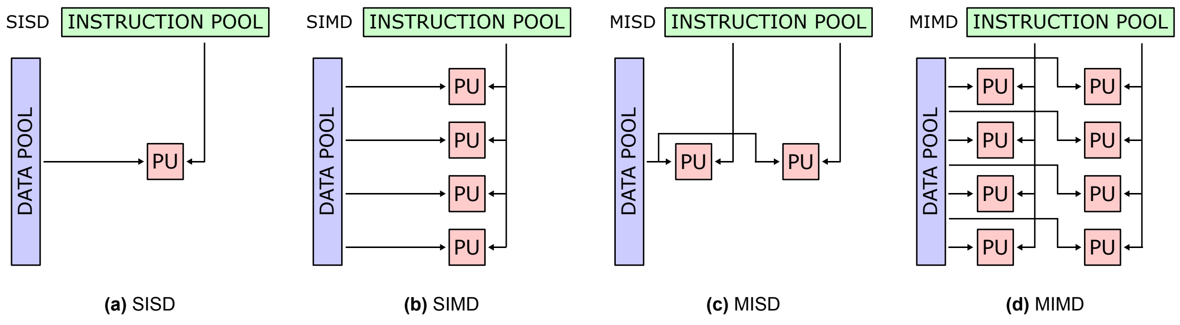


Figure B.2: Visual representation of Flynn's taxonomy [115]. *PU* stands for *processing unit*.

B.3. Multi-core Central Processing Units (CPU)

Unlike single-core CPUs, multi-core CPUs comprise multiple cores capable of executing operations in parallel. The primary advantage of multi-core CPUs lies in their architecture, characterized by a large cache and low latency as shown in Figure B.3[1]. This architecture facilitates the distribution of parallel processes across cores, as each core can execute code previously run on a single core. This not only simplifies the implementation of parallel computing but also reduces development time, which can be particularly advantageous for short-term projects like this MSc thesis project.

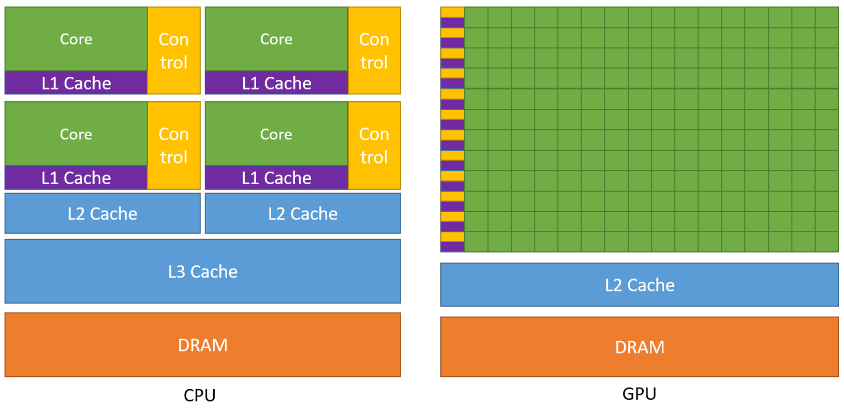


Figure B.3: CPU and GPU architectures schematic.

In terms of cost, the price range for computers with multi-core CPUs can vary widely depending on the number of cores and other specifications. As of 2023, the price for computers with 50+ cores typically falls within the high-end range with prices reaching thousands of euros[2].

B.4. Graphics processing units (GPU)

A Graphics Processing Unit (GPU) is a specialised electronic circuit that was initially designed for image processing and graphics acceleration. GPUs are especially useful for operations that can be parallelised very easily with little or no effort [101]. The architecture of GPUs designed for parallel processing also found its use in scientific computing in operations that can be done in parallel such as e.g. vector or matrix operations [116]. They can also be used in astrodynamics for tasks such as planetary protection and defense, with tools like CUDAjectory being developed for this purpose [97].

GPU architecture

GPUs and CPUs are designed with different goals in mind, which is reflected in their architecture. Compared to CPUs, GPUs, with their hundreds or thousands of smaller, specialised cores, are designed

[1] https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html, accessed 14 May 2024
[2] https://www.tomshardware.com/reviews/best-performance-cpus,5683.html, accessed 14 May 2024

for high throughput as seen in Figure B.3. They are better in performing multiple parallel computing operations, particularly when the same operation is being performed on many data elements simultaneously. This makes them ideal for tasks that involve large datasets and require the same operations to be performed on each data element, such as image or video processing, scientific simulations, and machine learning.

GPU memory types

GPUs have various types of memory, including register, shared, constant, local, and global. The size and latency of each memory type can vary depending on the specific GPU model and architecture. These memory types can be seen in Figure B.4 showing them within a typical GPU architecture.

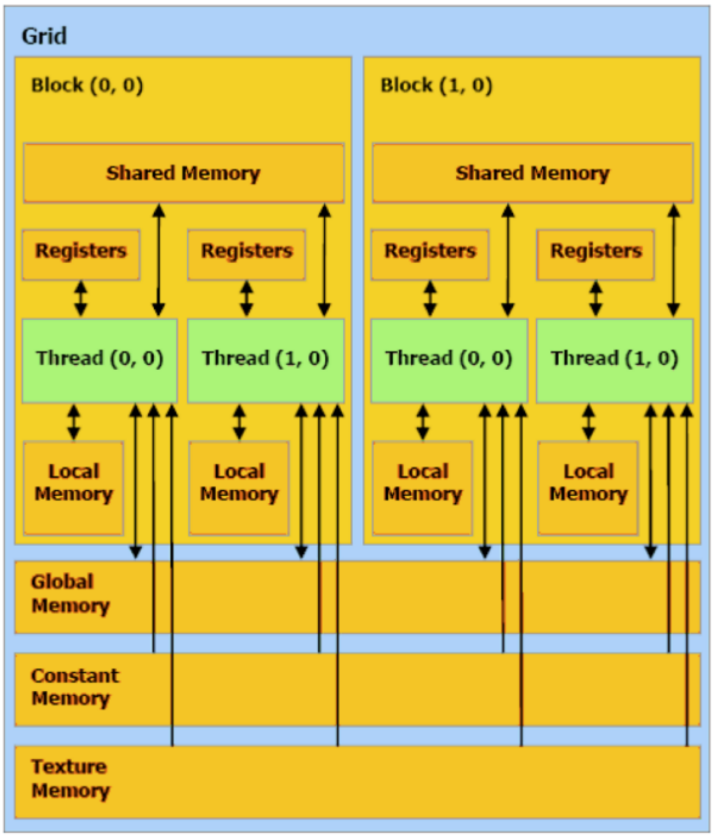


Figure B.4: Typical GPU memory types [117].

From smallest to largest in size and lowest to highest in latency, they are listed below [117].

- **Register**
  The smallest and fastest memory type, used to hold data being processed by the GPU.
- **Shared**
  Slightly larger and slower than register memory, shared among threads in a warp.
- **Constant**
  Larger and slower than shared memory, it is read-only and holds data that does not change during kernel execution.
- **Texture**
  Texture memory is read-only memory type which holds the texture data that used for 3D models. It is optimised for efficient access to 2D arrays while accessing 1D arrays sometimes can be more efficient with global memory.
- **Local**
  Local memory is private to each thread and used when there is not enough register memory. It is

larger in size than register and shared memory but it comes at a cost of being significantly slower.

- **Global**
  The largest and slowest memory type, it is the main memory for the GPU and holds data being processed and the computation results.

In the context of visibility computation, one has to deal with large data arrays, which have to be stored in global memory as this is the only memory type that can accommodate such arrays. Register memory will be used for storing data of intermediate calculations such as vector difference, dot product of two vectors, cosine of angle between two vectors, etc.

CUDA API

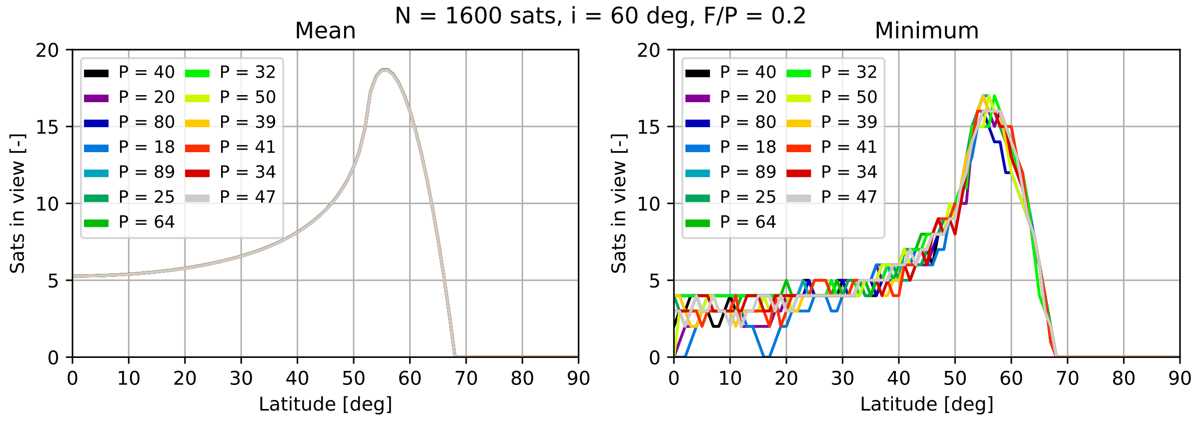
The CUDA API is a software layer provided by Nvidia that allows developers to use the power of Nvidia GPUs directly [118]. It supports development in C/C++, providing very low-level control of GPUs, which is crucial for optimising performance and efficiency. The CUDA API provides a comprehensive set of programming constructs, libraries, and runtime support to help developers achieve high performance with their applications.

CUDA is available in Python as well. However, CUDA programming in C/C++ is more common, and much more information and documentation is available on these languages. Thus, C/C++ interface will be used for this project.

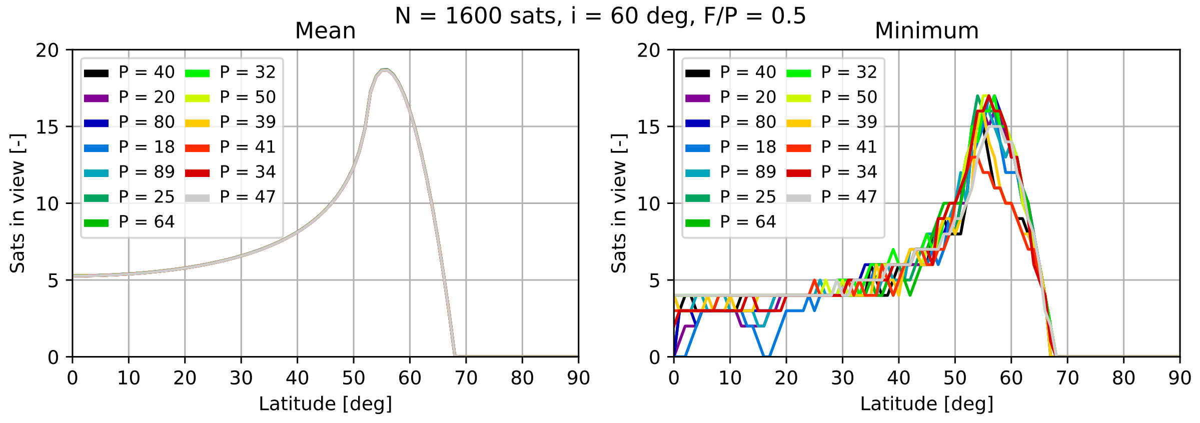
C

Parametric Analysis

C.1. Number of Planes and Phasing Parameter

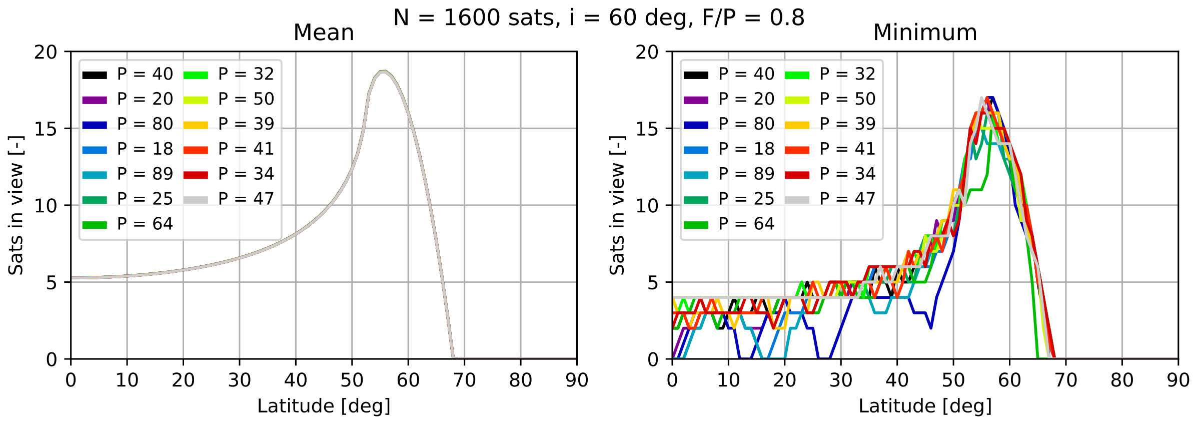


(a)

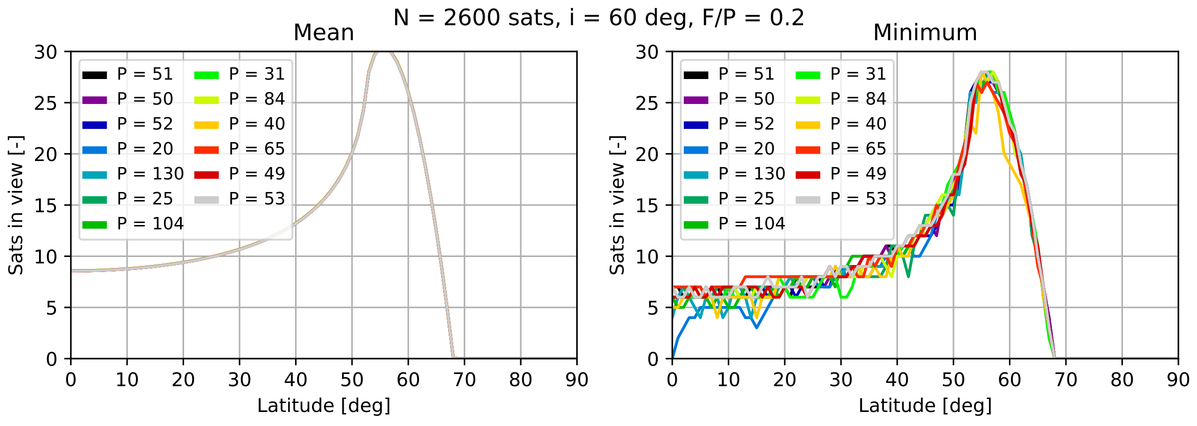


(b)

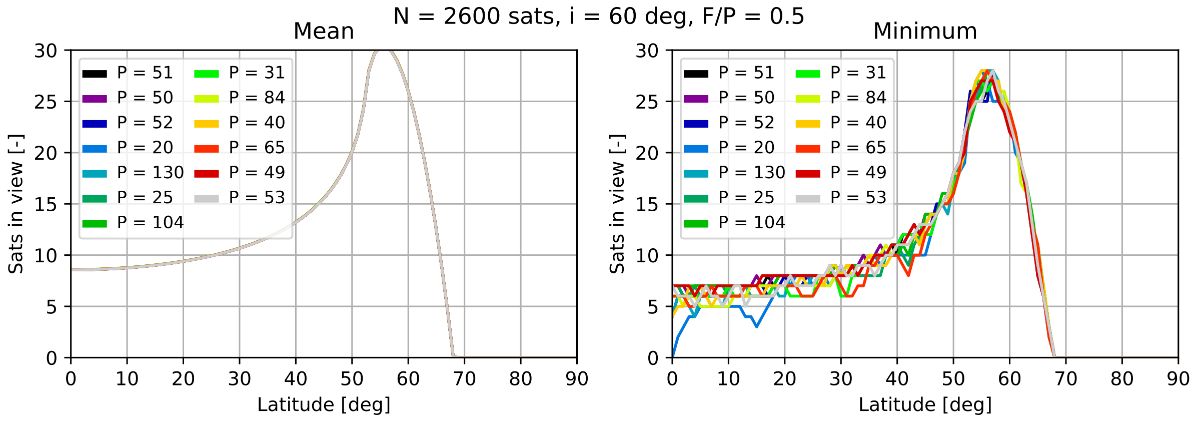
Figure C.1: Mean and minimum visibility for given constellations as function of P with a fixed F/P.



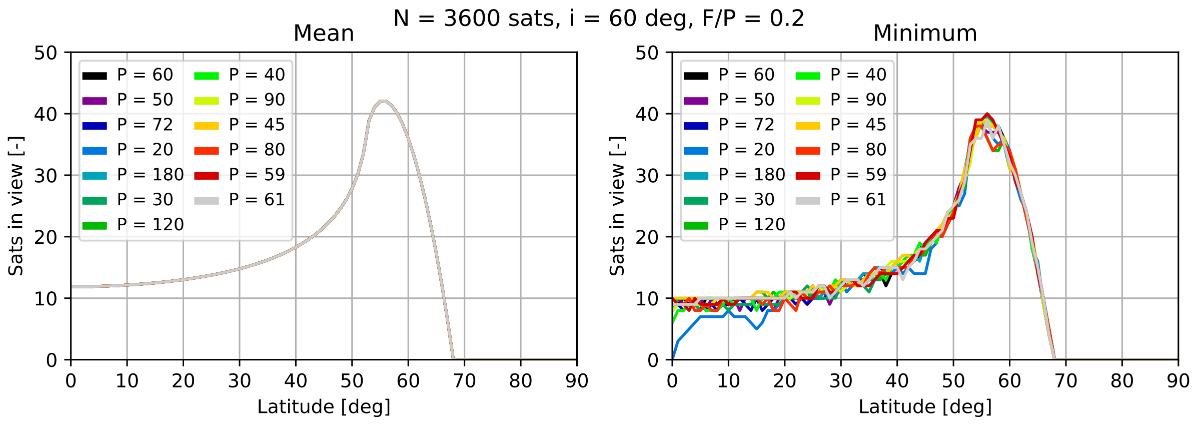
(c)



(d)

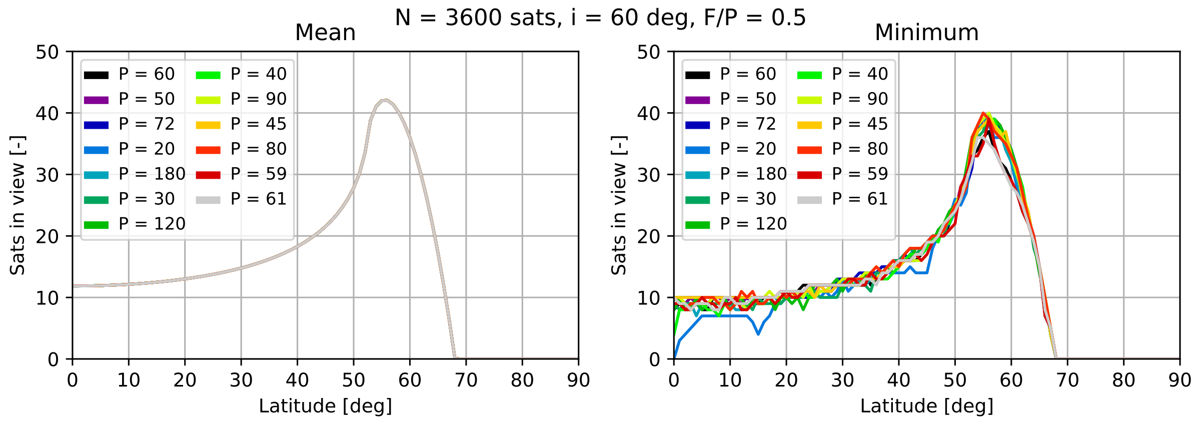


(e)

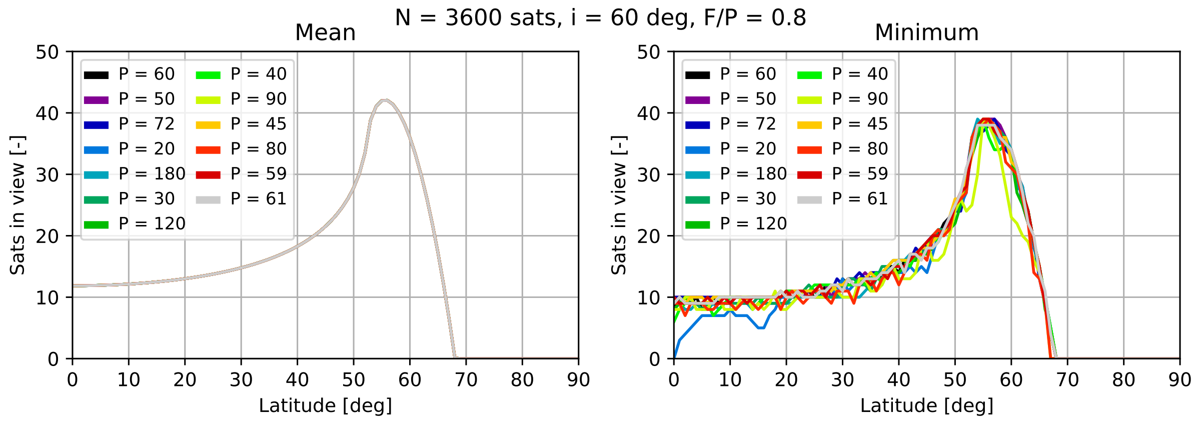


(f)

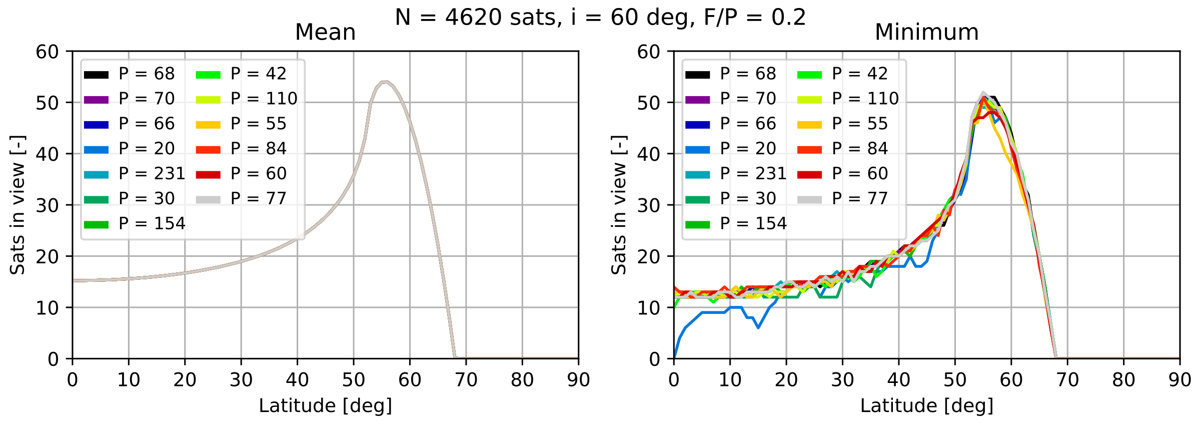
Figure C.1: Mean and minimum visibility for given constellations as function of P with a fixed F/P.



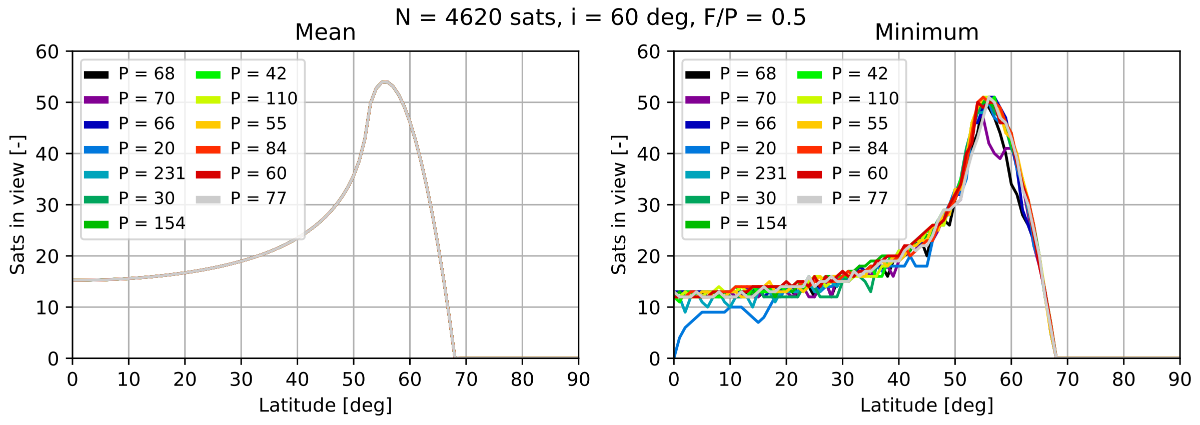
(g)



(h)

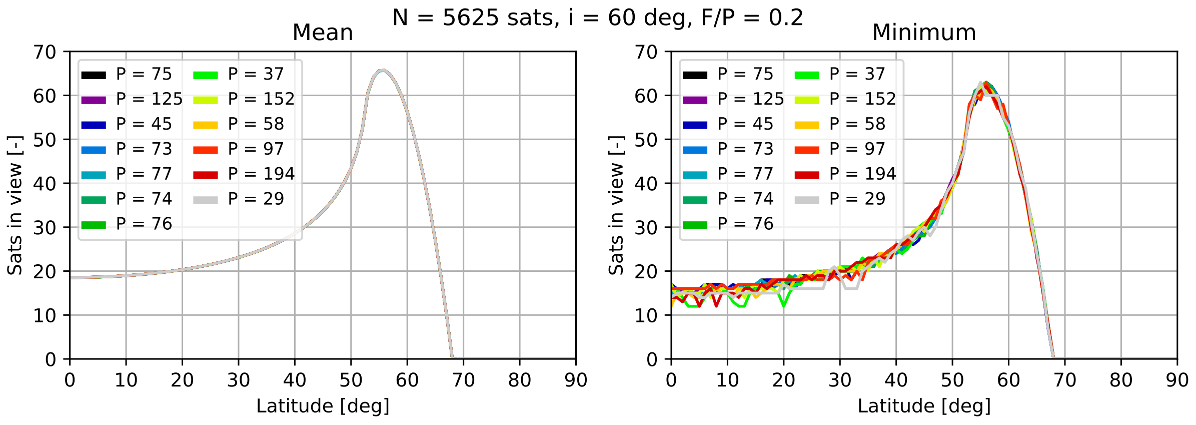


(i)

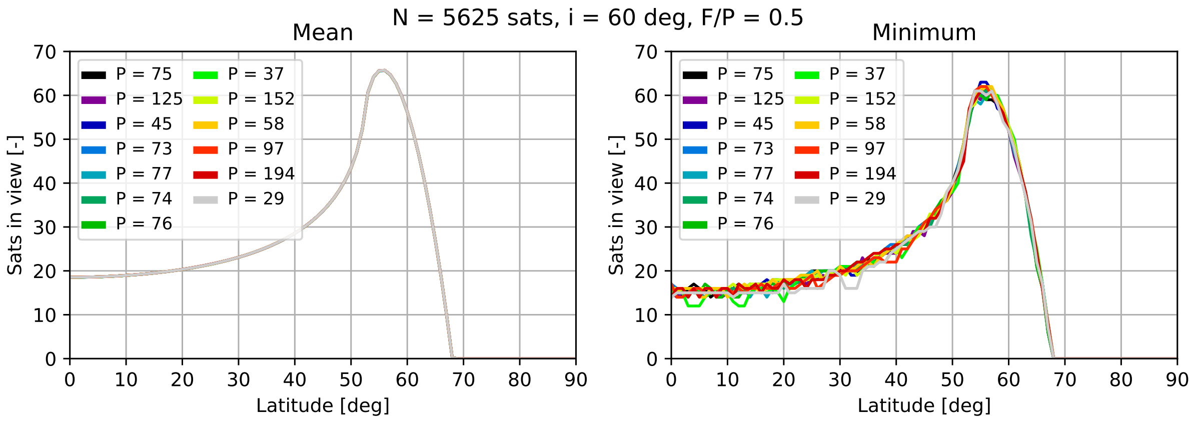


(j)

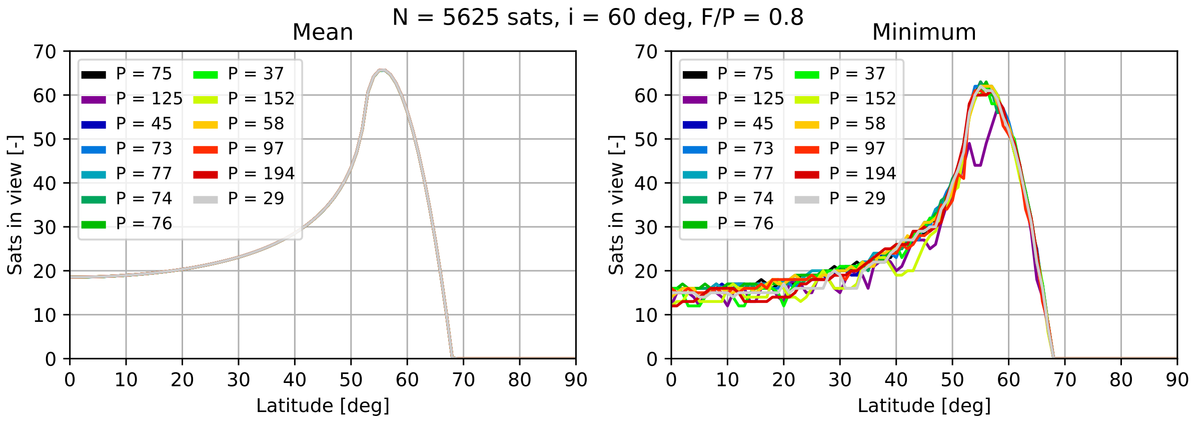
Figure C.1: Mean and minimum visibility for given constellations as function of P with a fixed F/P.



(k)

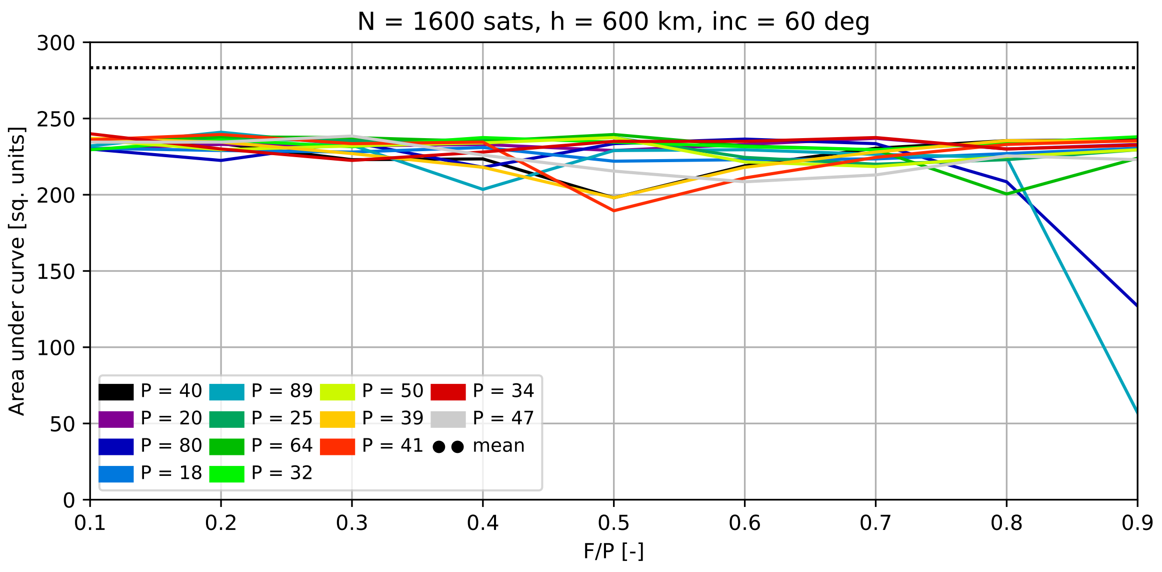


(l)

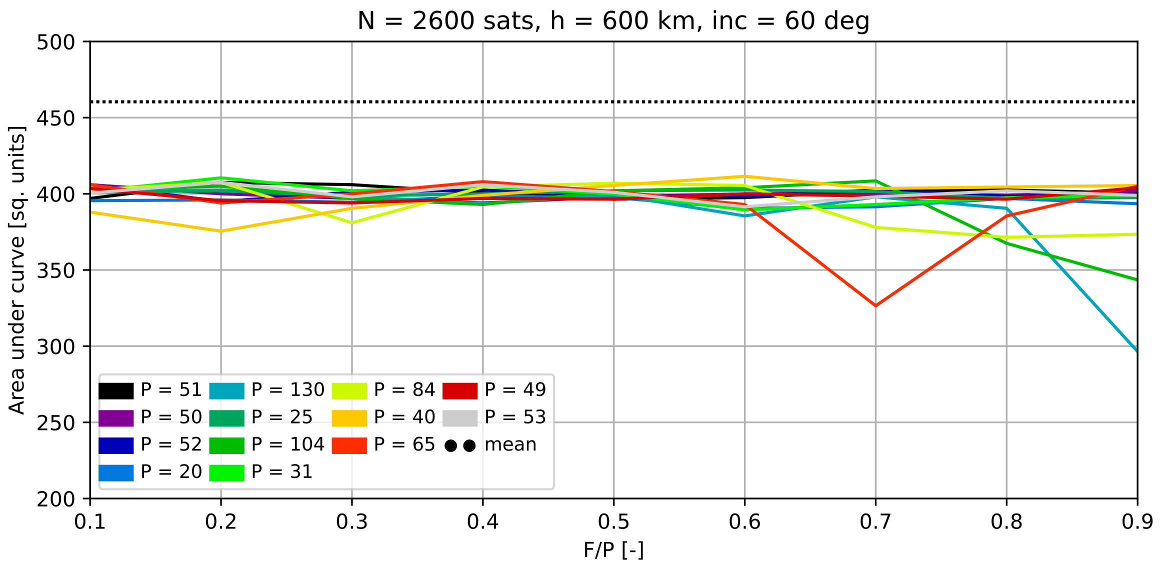


(m)

Figure C.1: Mean and minimum visibility for given constellations as function of P with a fixed F/P.

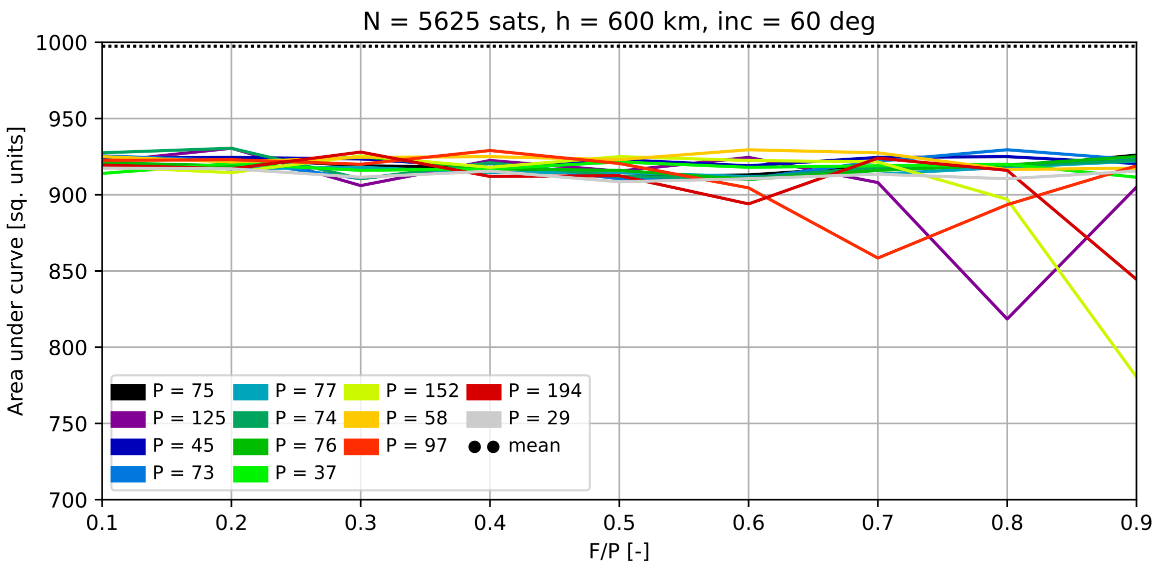


(a)

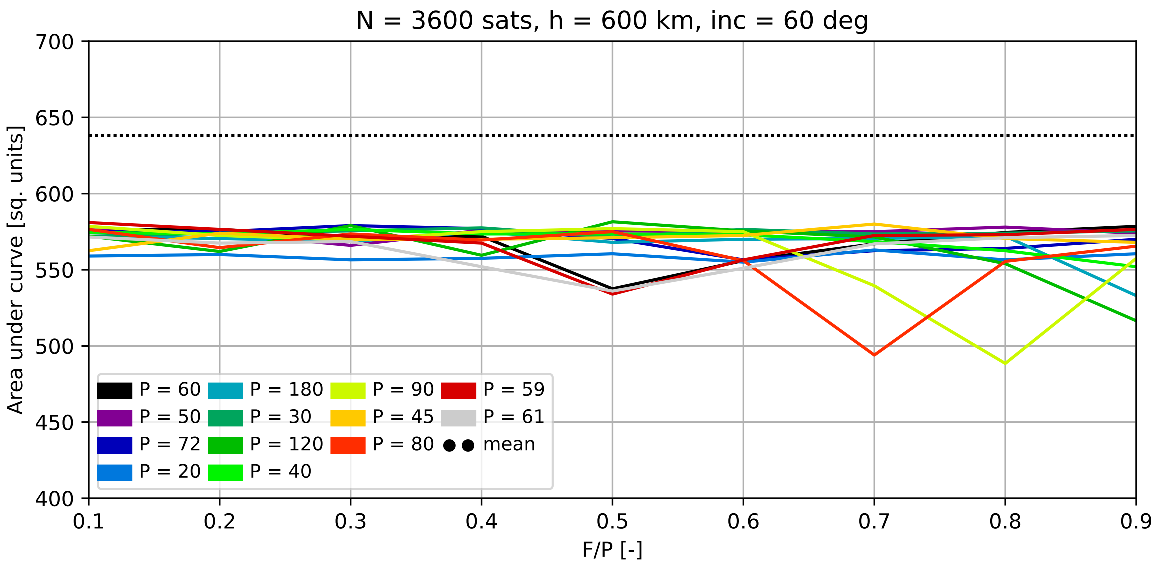


(b)

Figure C.2: Area under curve for different layouts.



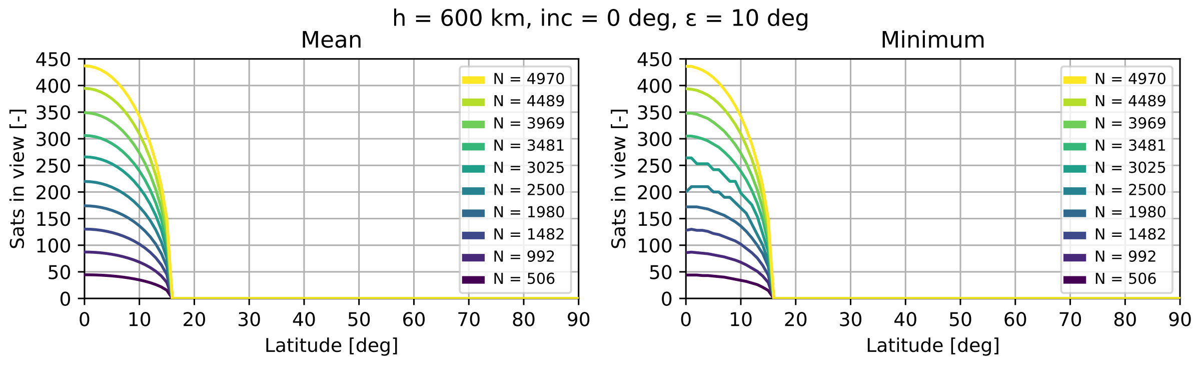
(c)



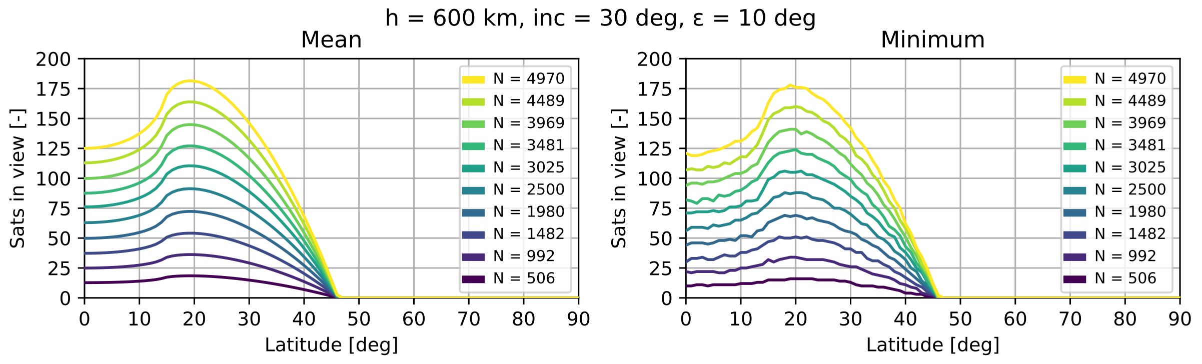
(d)

Figure C.2: Area under curve for different layouts.

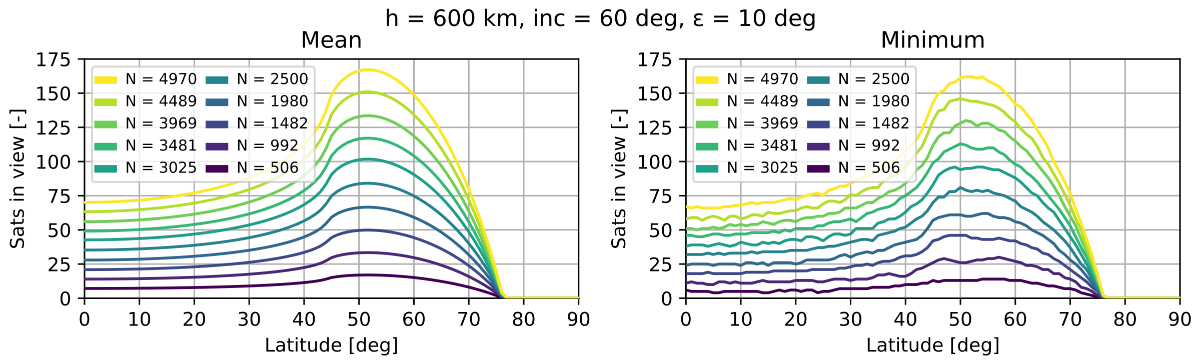
C.2. Number of Satellites



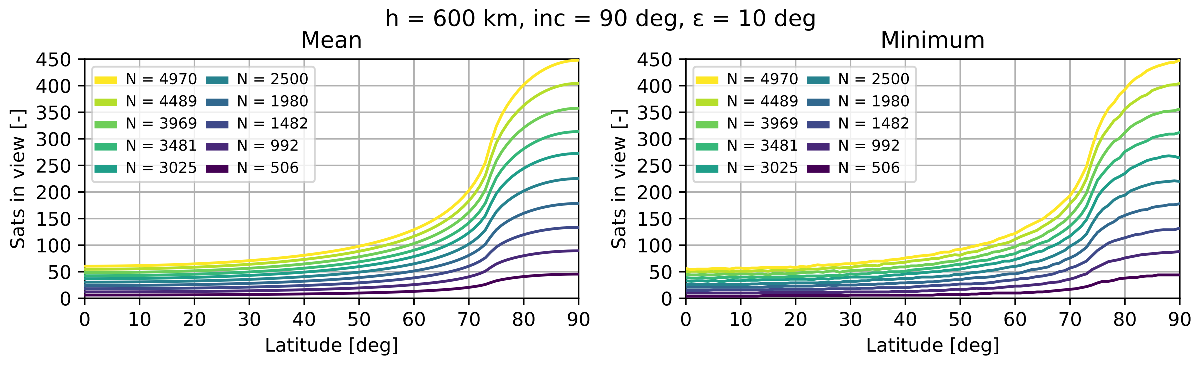
(a)



(b)

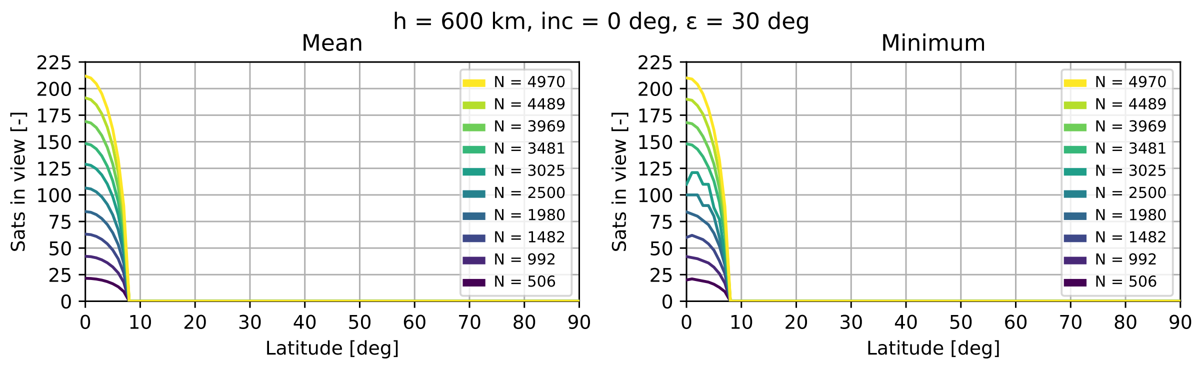


(c)

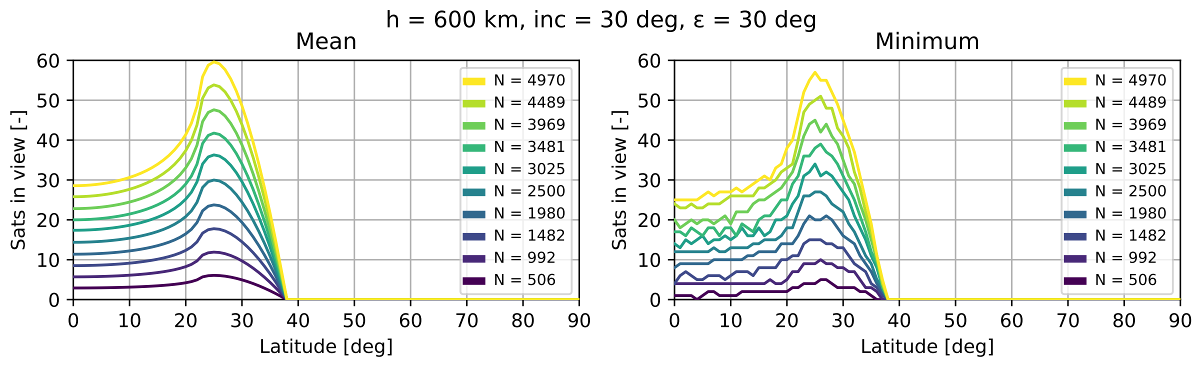


(d)

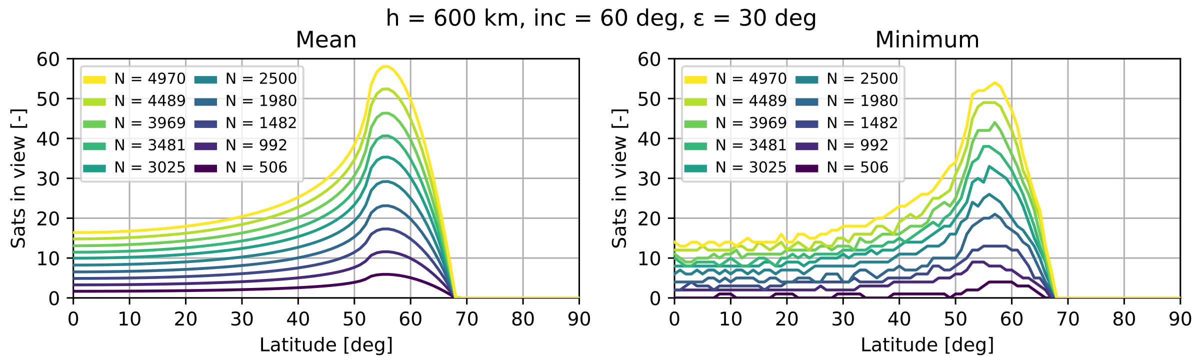
Figure C.3: Parametric analysis, varying N.



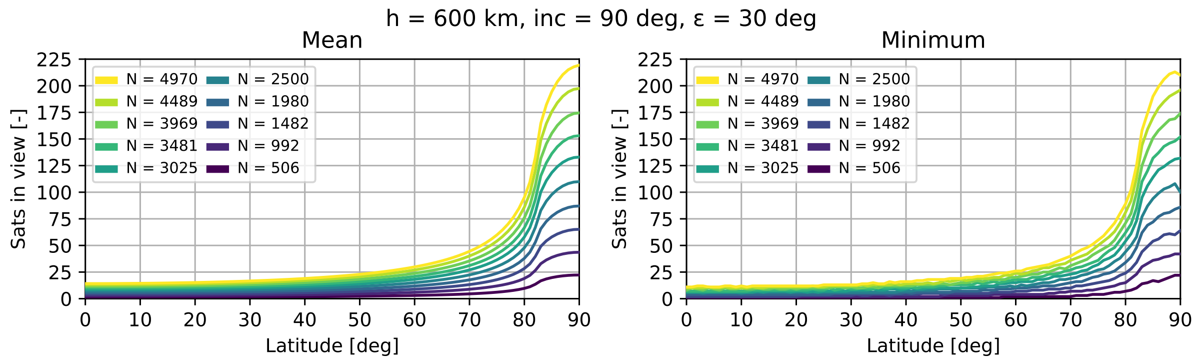
(e)



(f)

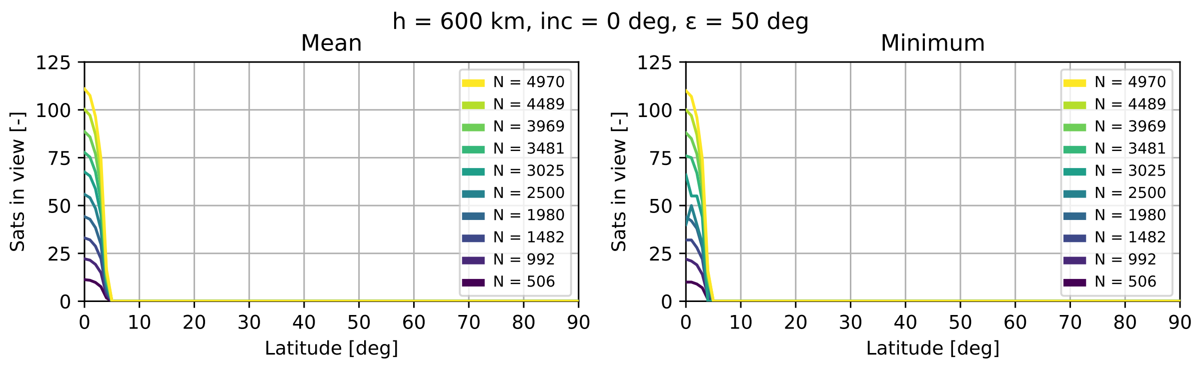


(g)

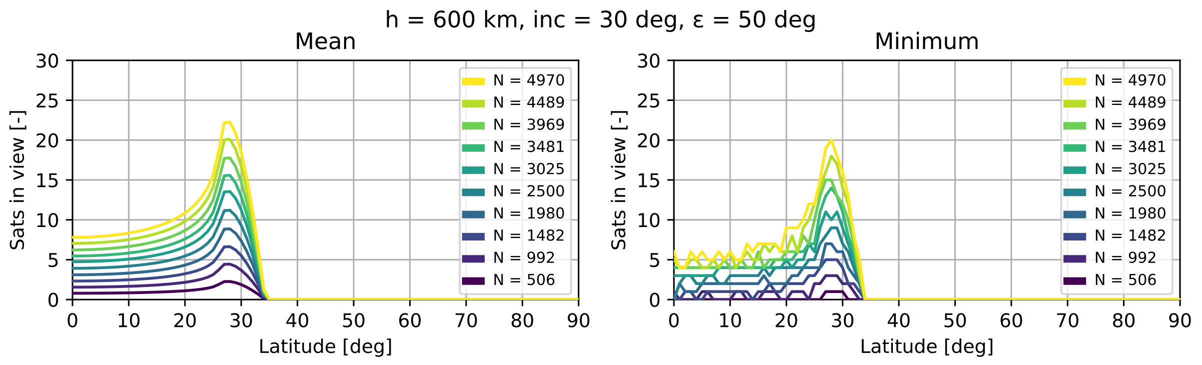


(h)

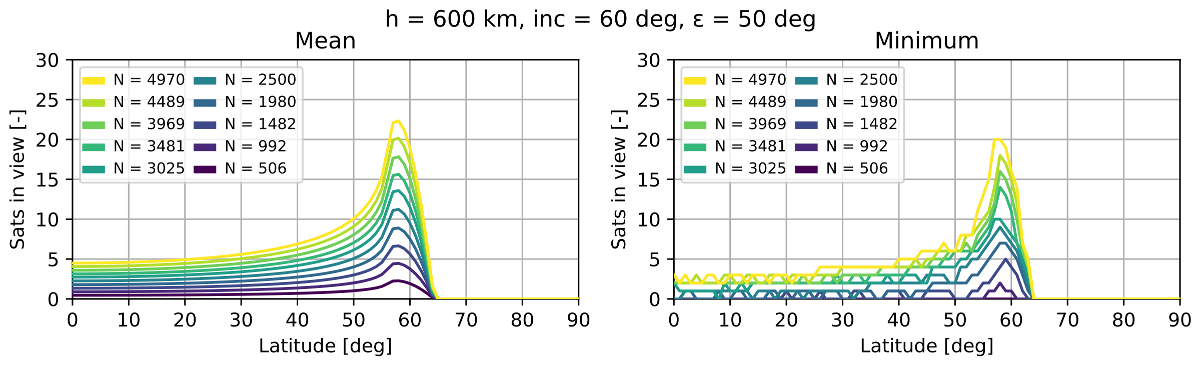
Figure C.3: Parametric analysis, varying N.



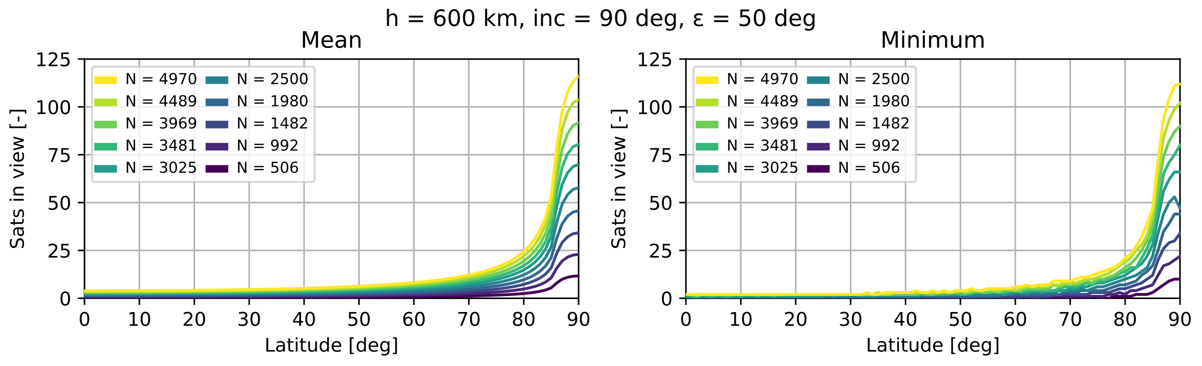
(i)



(j)

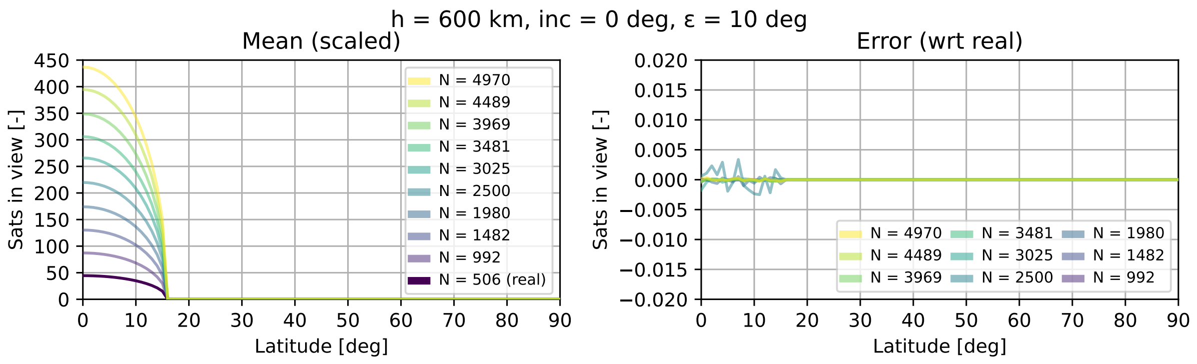


(k)

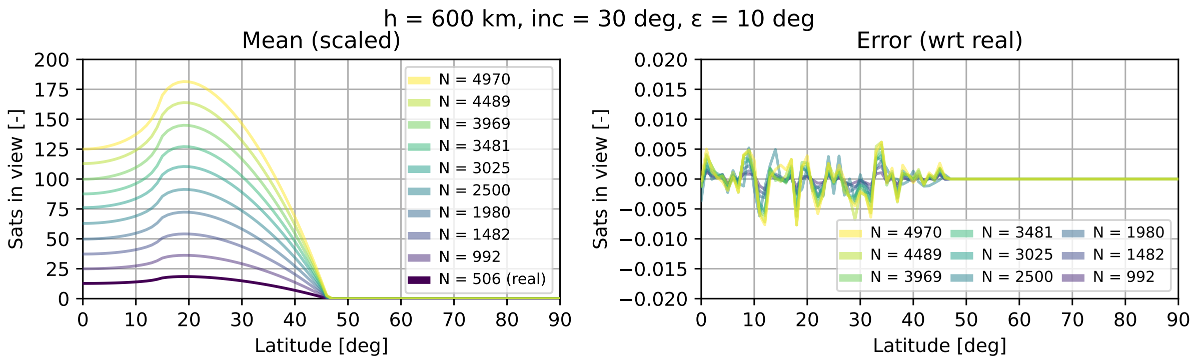


(l)

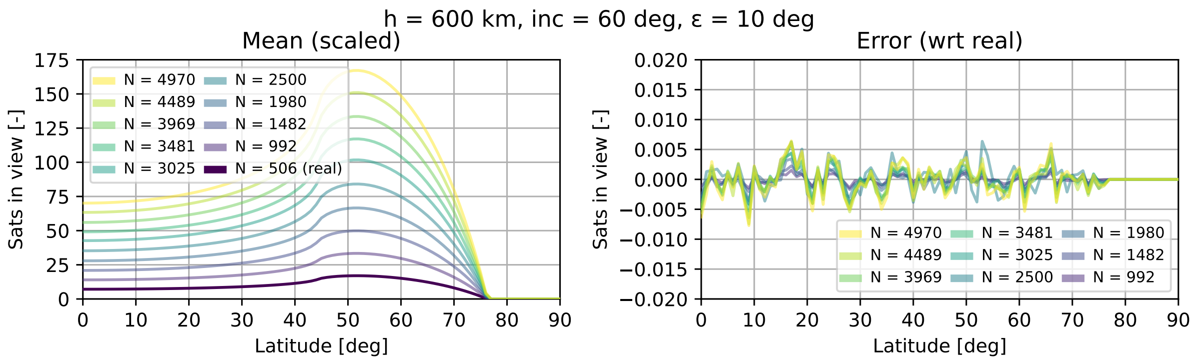
Figure C.3: Parametric analysis, varying N.



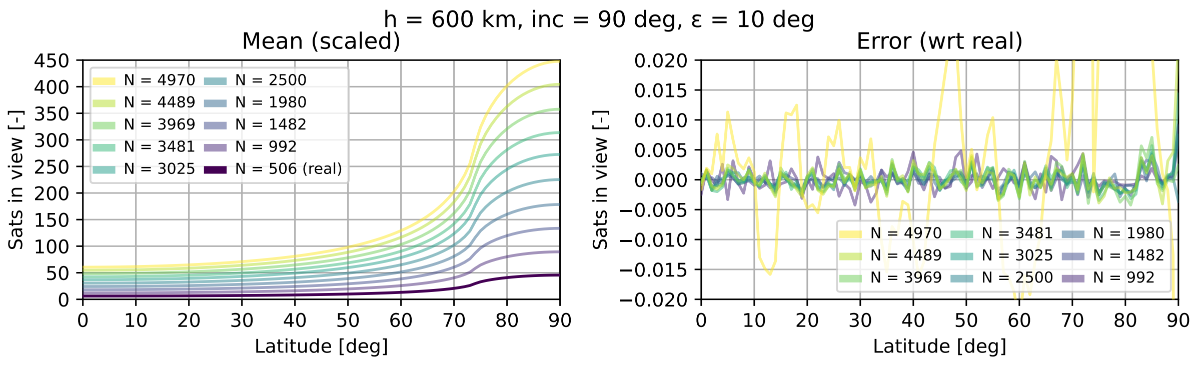
(a)



(b)

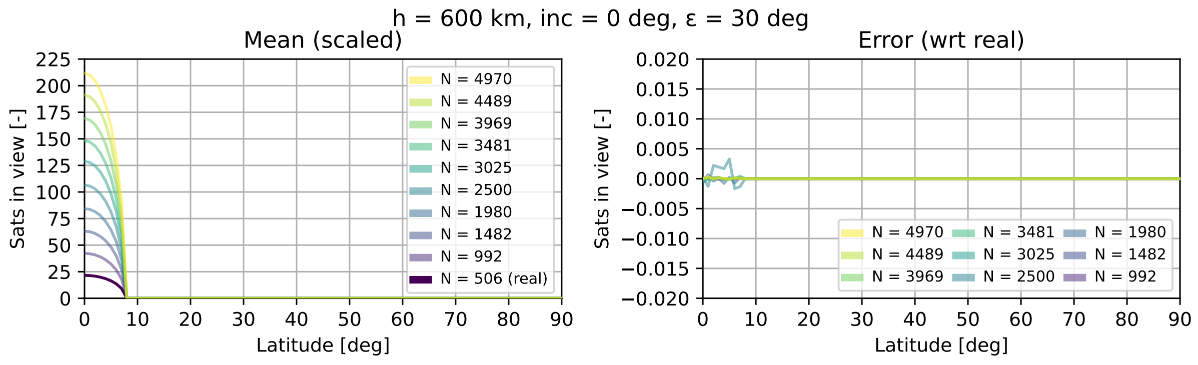


(c)

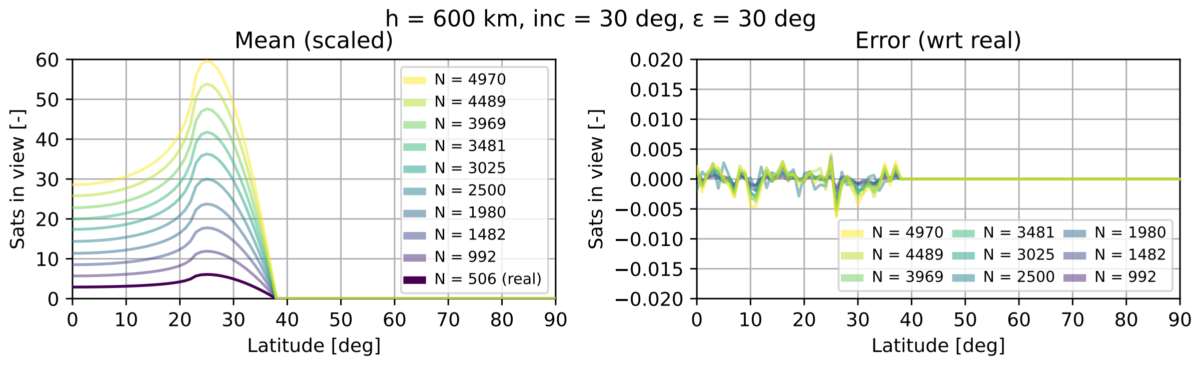


(d)

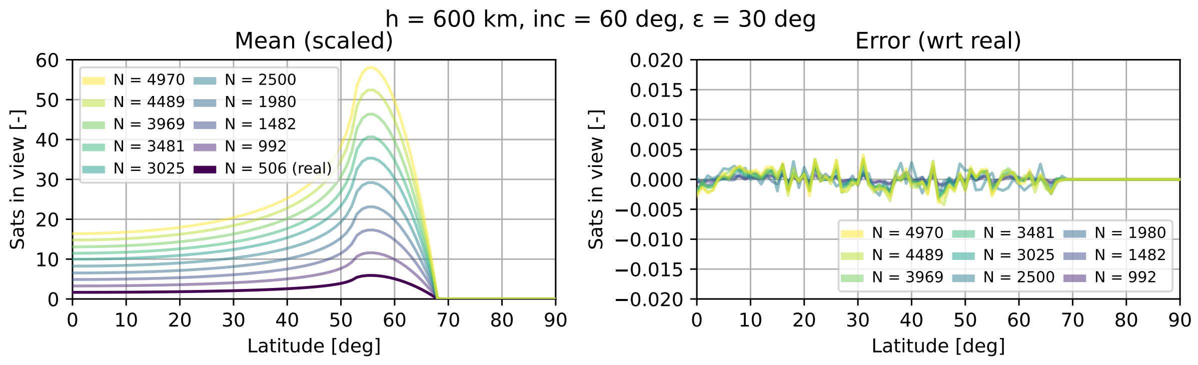
Figure C.4: Scaled visibility, varying N.



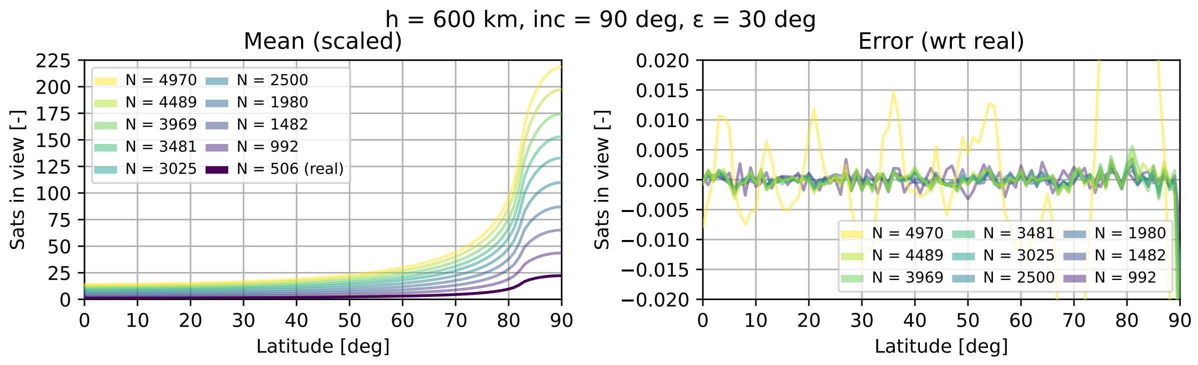
(e)



(f)

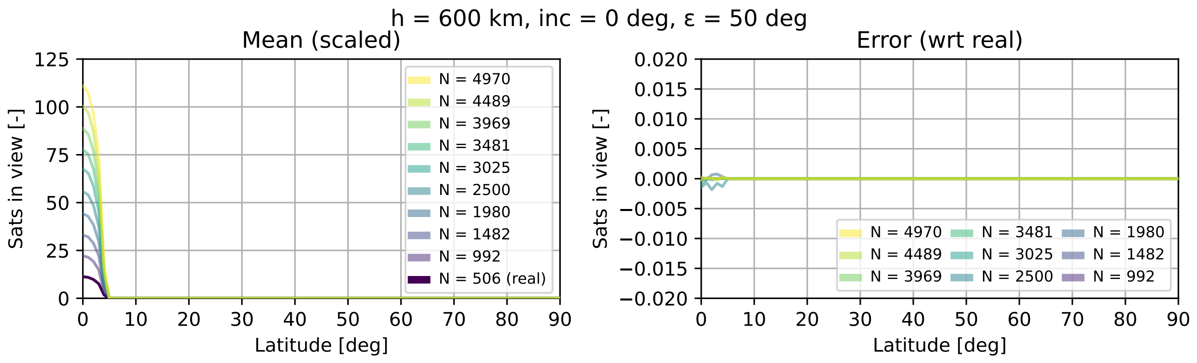


(g)

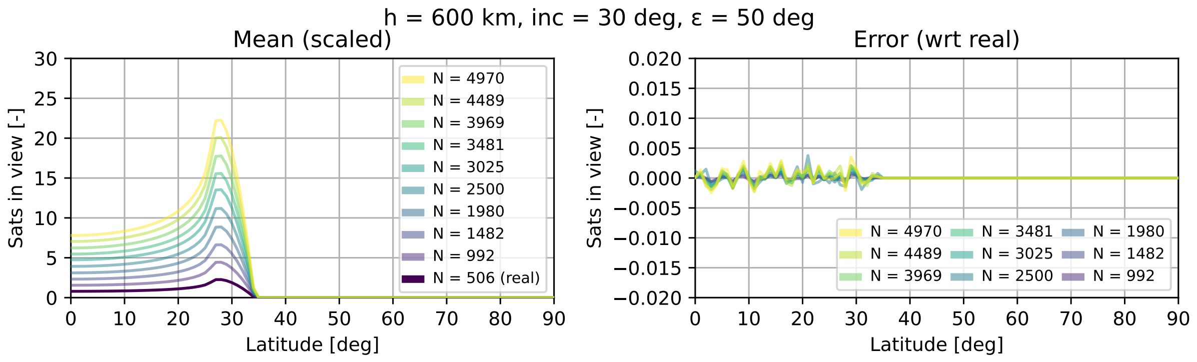


(h)

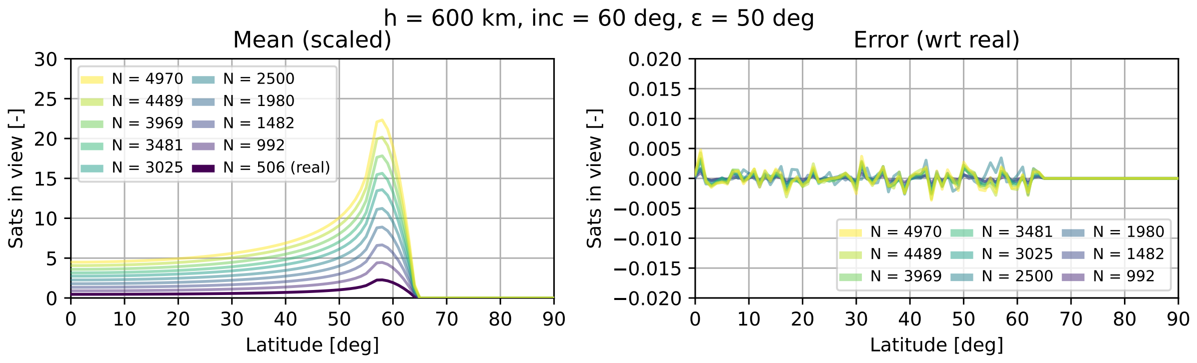
Figure C.4: Scaled visibility, varying N.



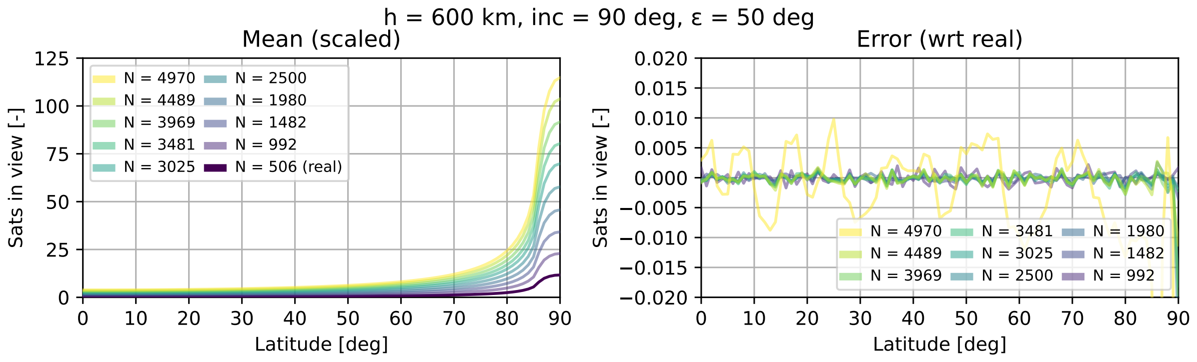
(i)



(j)



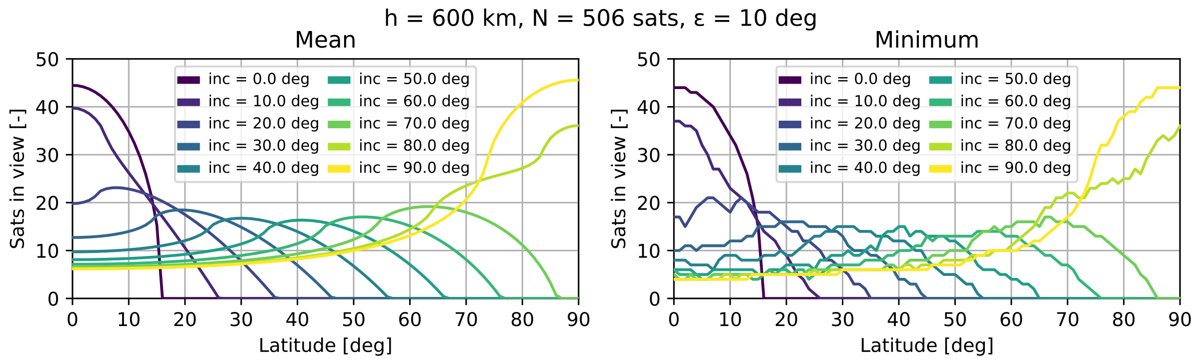
(k)



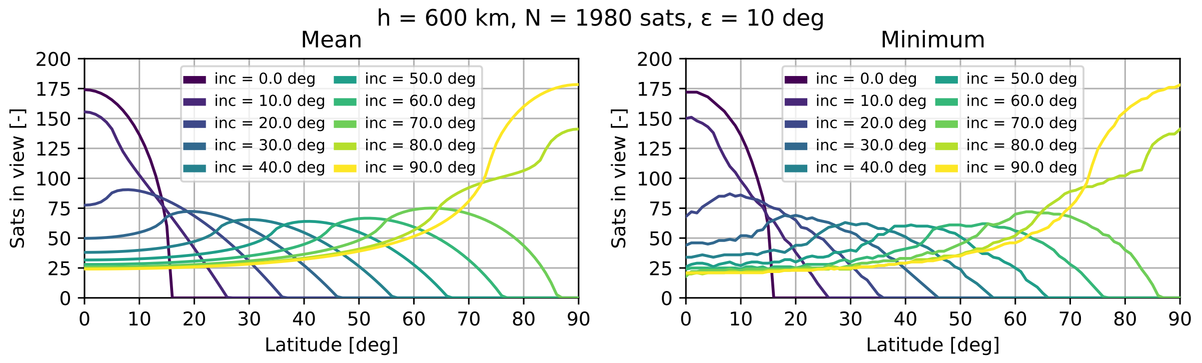
(l)

Figure C.4: Scaled visibility, varying N.

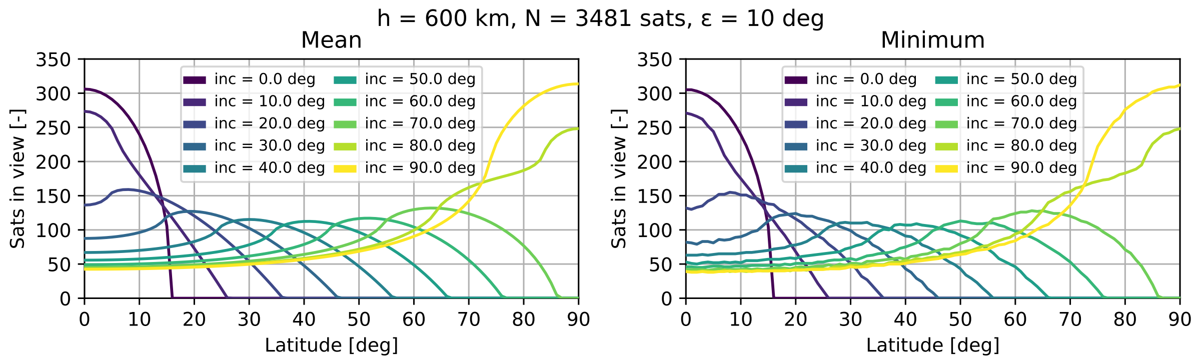
C.3. Inclination



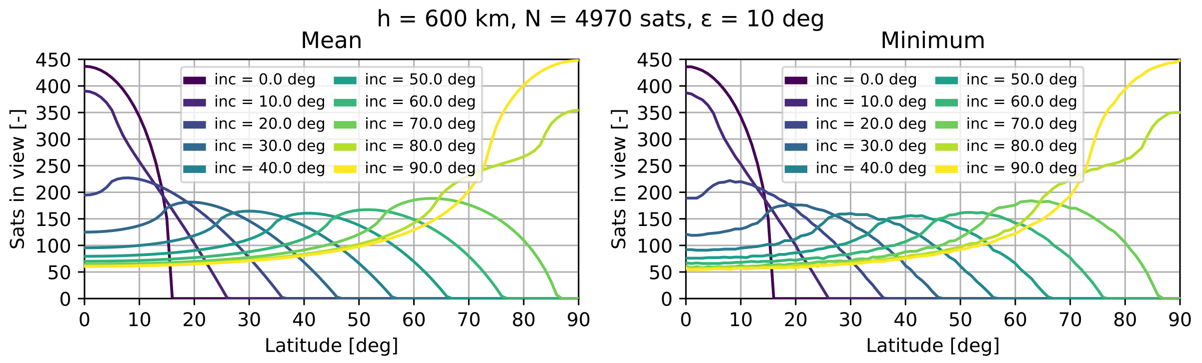
(a)



(b)

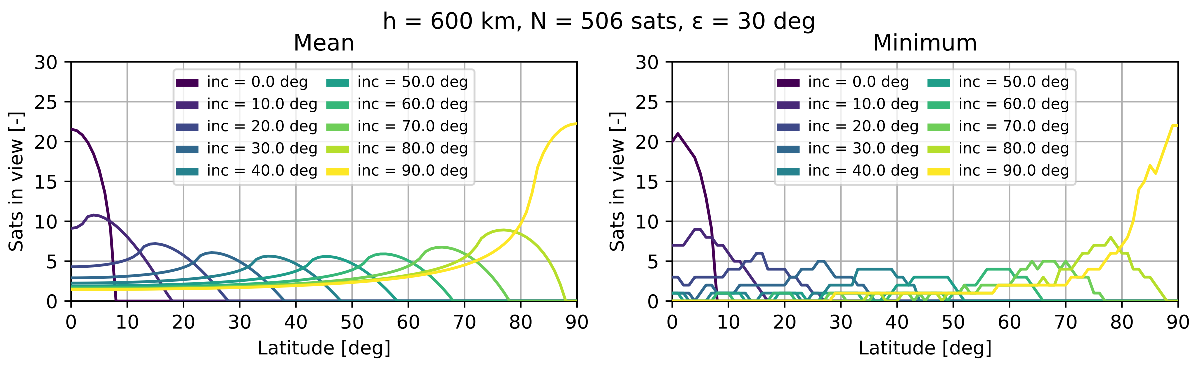


(c)

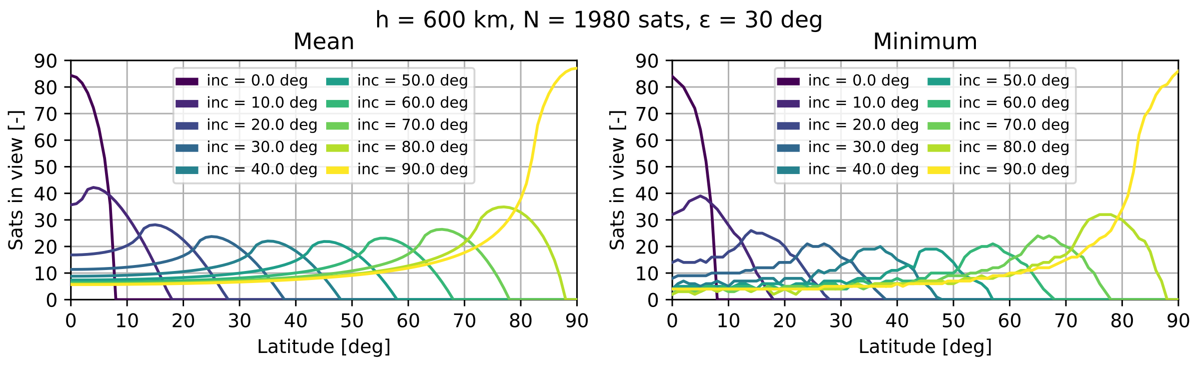


(d)

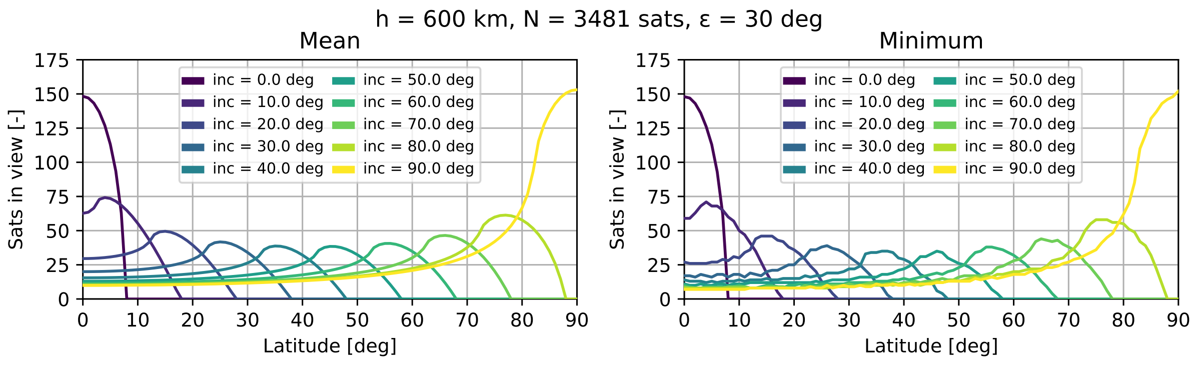
Figure C.5: Parametric analysis, varying i.



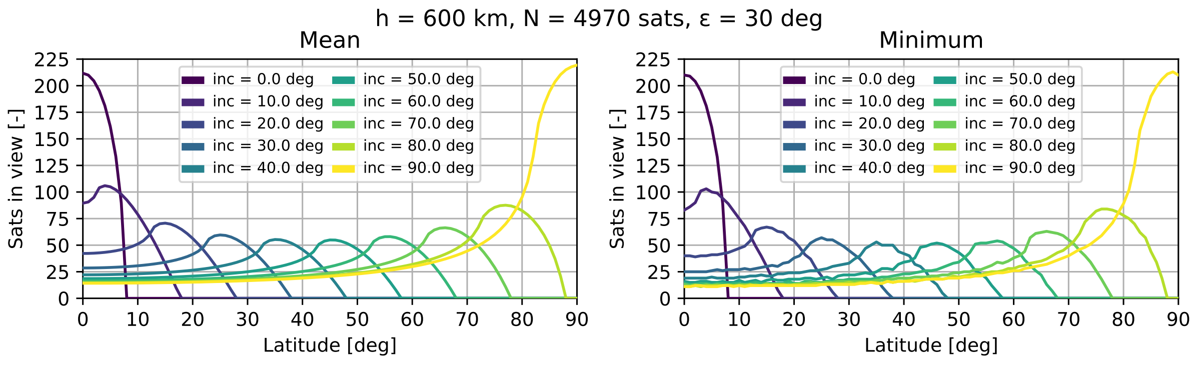
(e)



(f)

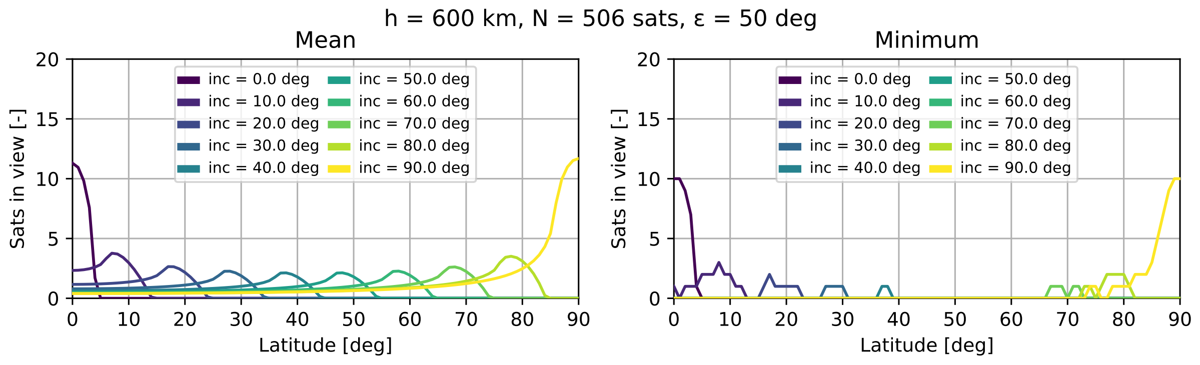


(g)

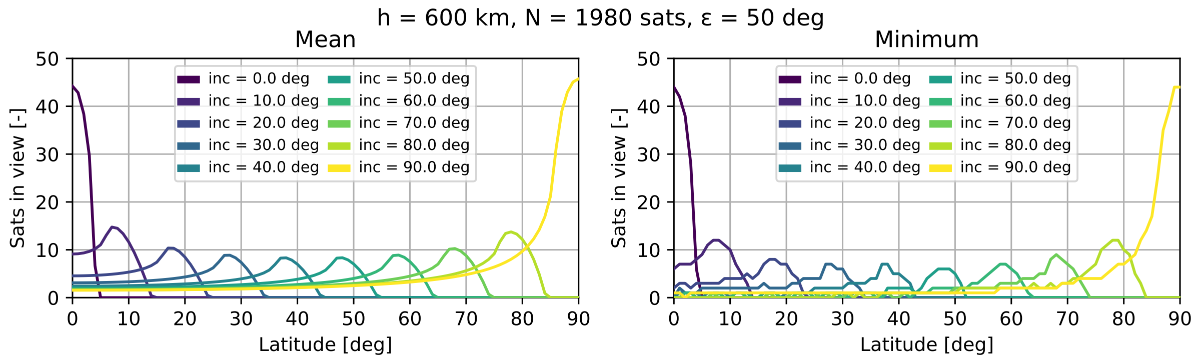


(h)

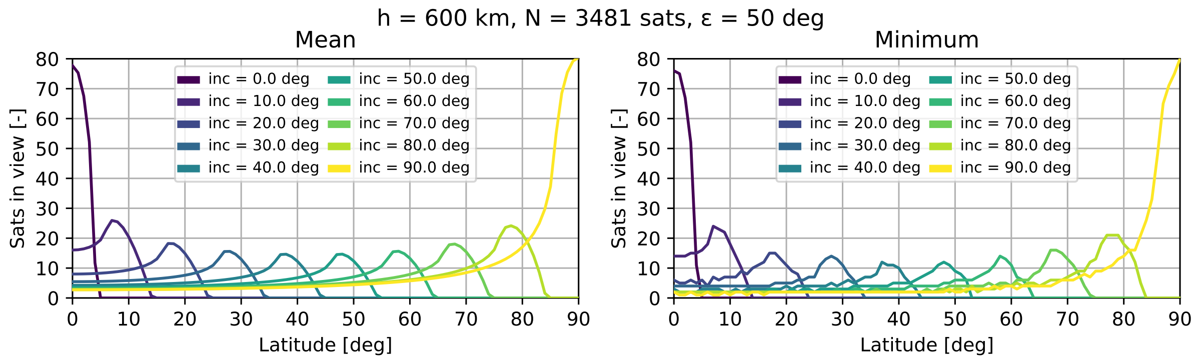
Figure C.5: Parametric analysis, varying i.



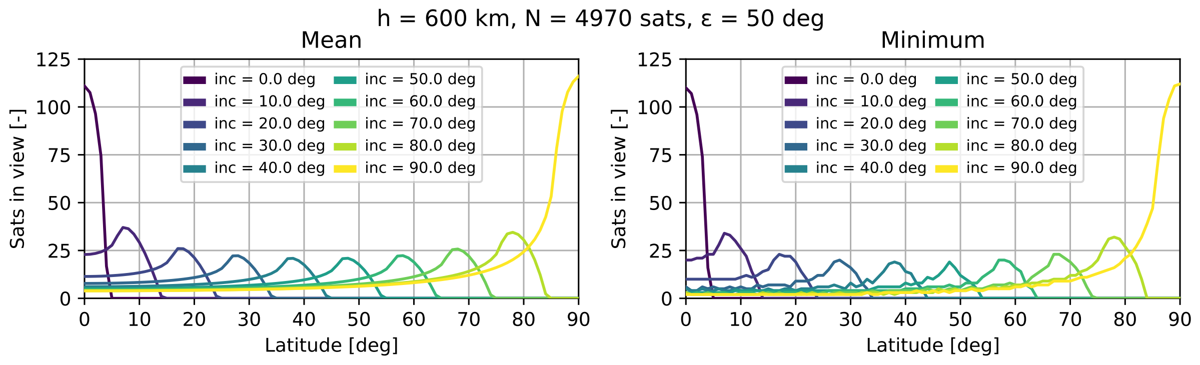
(i)



(j)



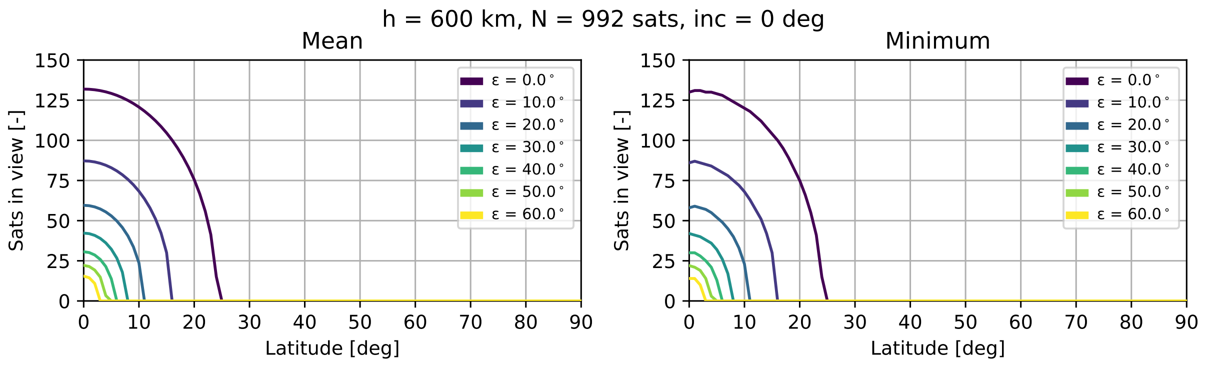
(k)



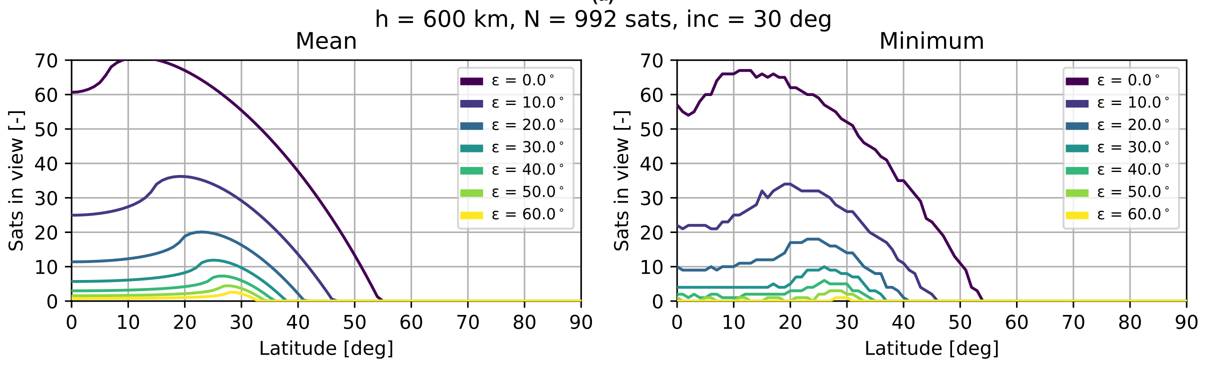
(l)

Figure C.5: Parametric analysis, varying i.

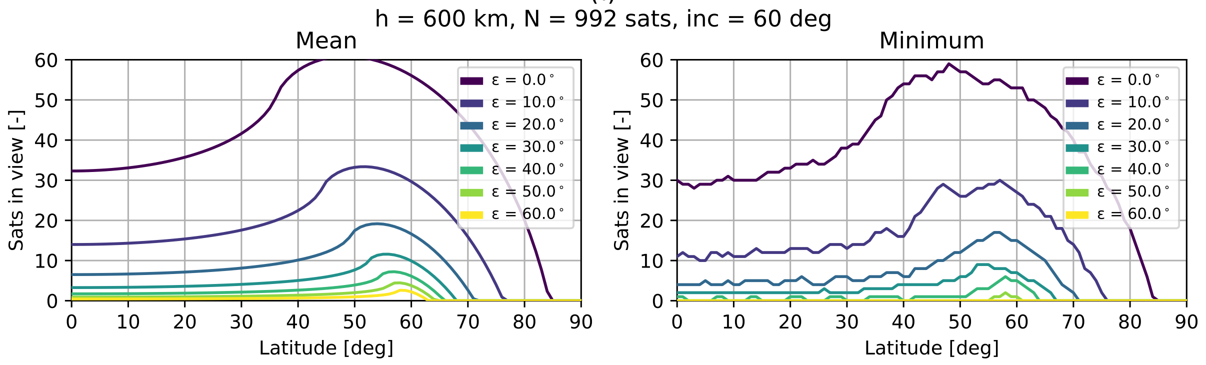
C.4. Minimum Elevation Angle and Altitude



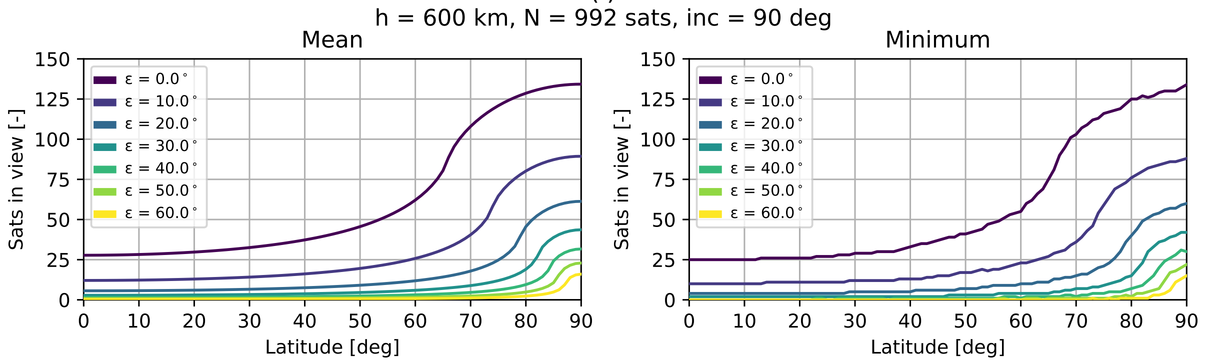
(a)



(b)

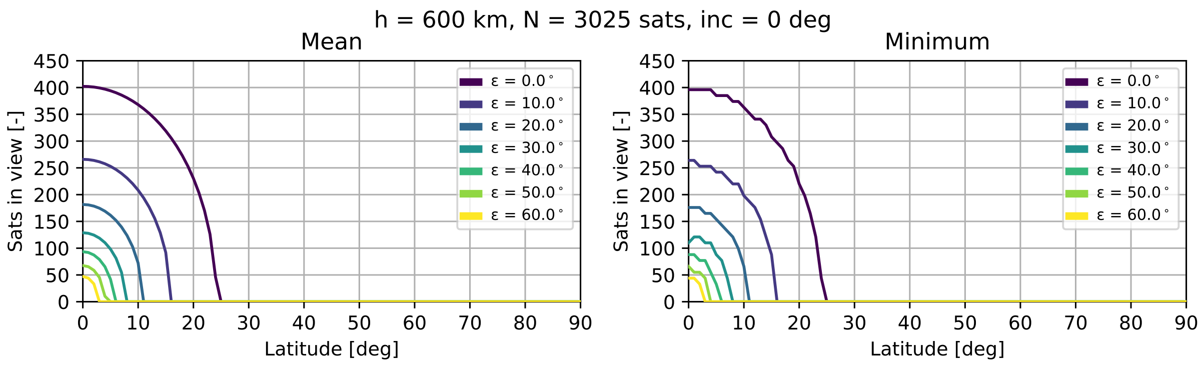


(c)

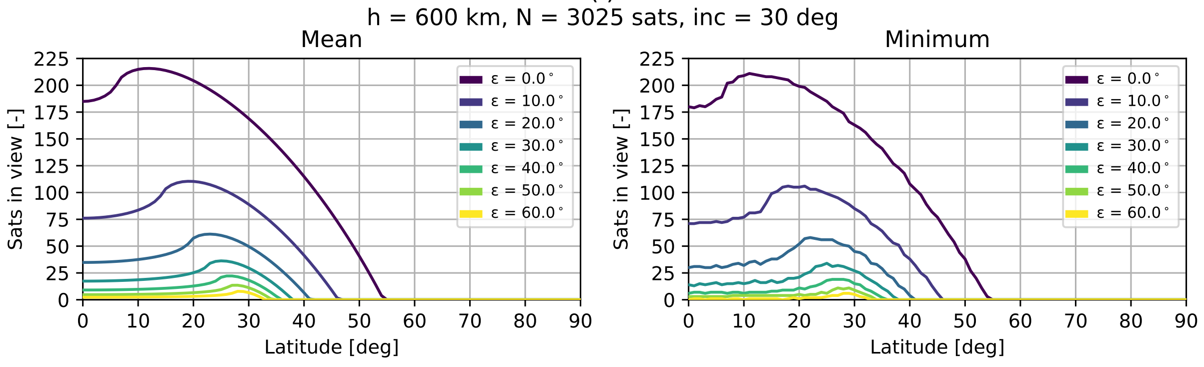


(d)

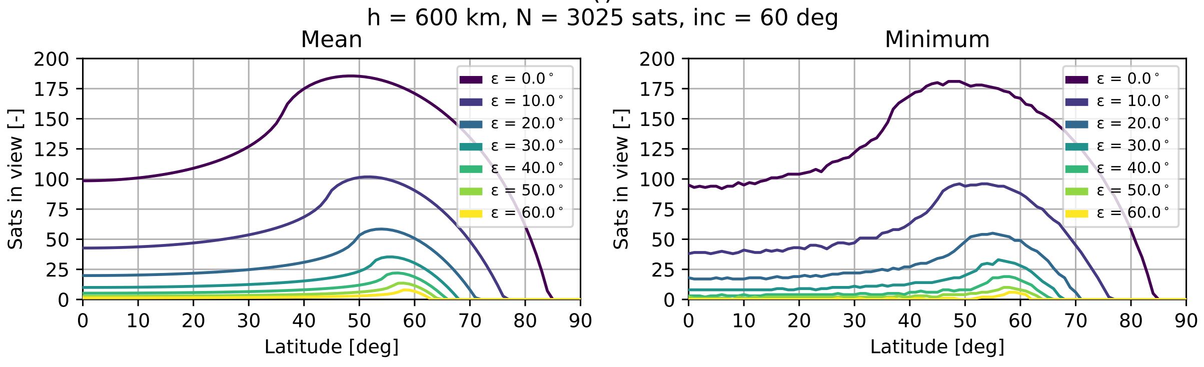
Figure C.6: Parametric analysis, varying ε_{min}.



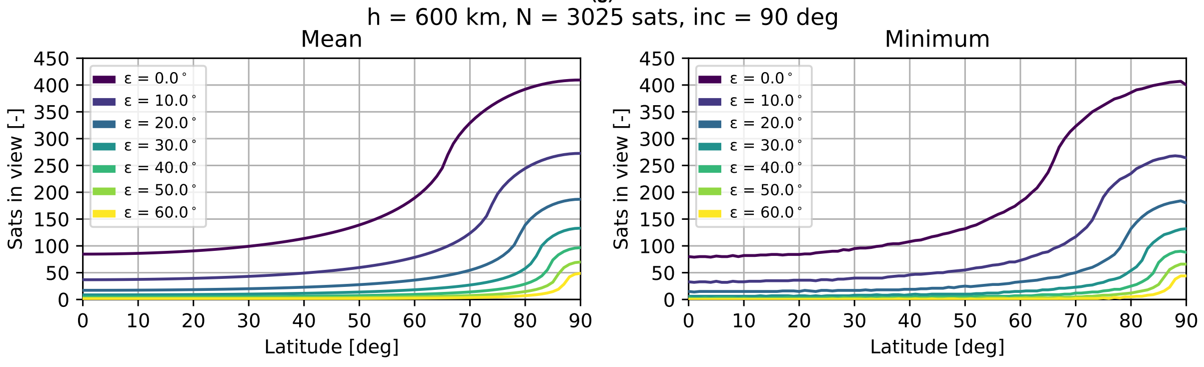
(e)



(f)

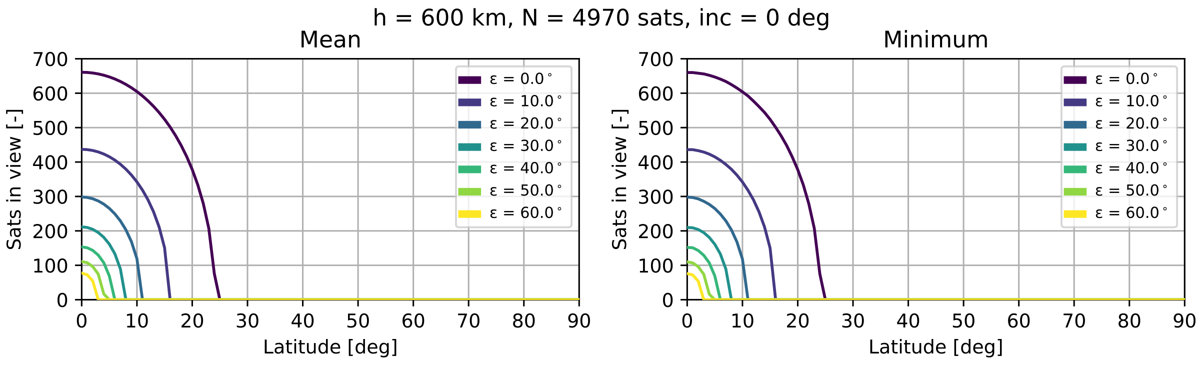


(g)

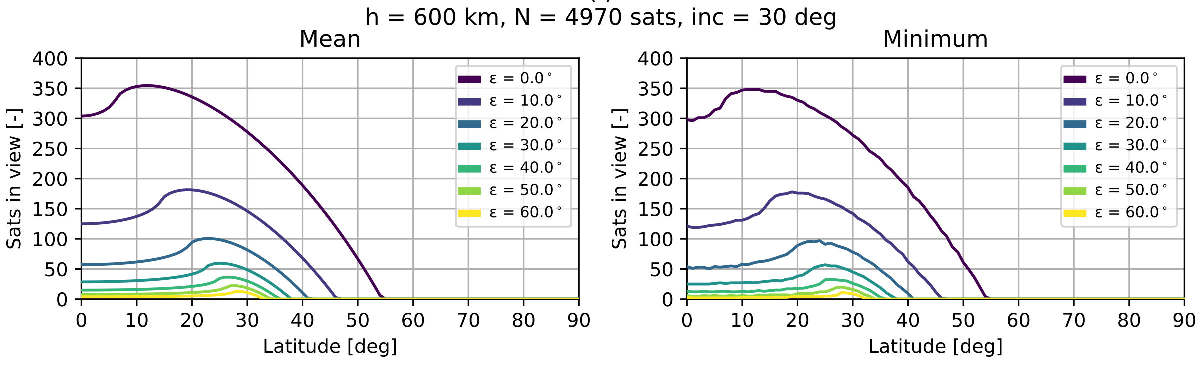


(h)

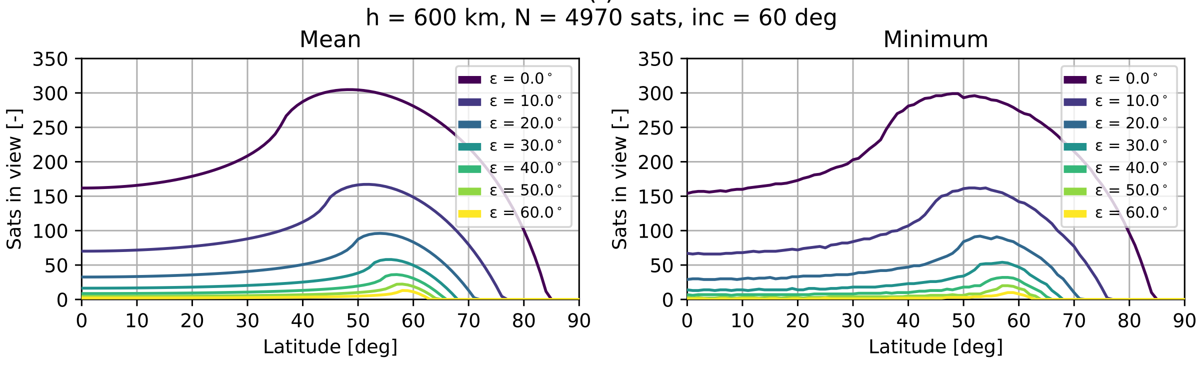
Figure C.6: Parametric analysis, varying ε_{min}.



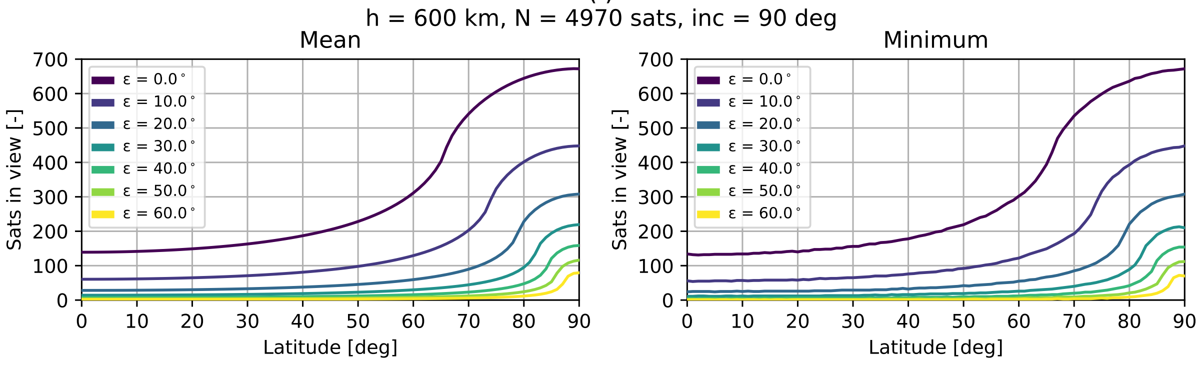
(a)



(b)



(c)



(d)

Figure C.7: Parametric analysis, varying ε_{min}.

D Building Blocks

Table D.1: Setup for comparison of computational time using the building blocks method.

Parameter	**Values**	**Increment**
Subrange 1		
Target latitude	67 to 90 deg	-
Inclination	67 to 90 deg	1 deg
N per shell	0 to 2000 sats	100 sats
Total two-shell layouts		116, 403 layouts
Subrange 2		
Target latitude	44 to 67 deg	-
Inclination	44 to 76 deg	1 deg
N per shell	0 to 2000 sats	100 sats
Total two-shell layouts		225, 456 layouts
Subrange 3		
Target latitude	21 to 44 deg	-
Inclination	21 to 53 deg	1 deg
N per shell	0 to 2000 sats	100 sats
Total two-shell layouts		225, 456 layouts
Subrange 4		
Target latitude	0 to 21 deg	-
Inclination	0 to 30 deg	1 deg
N per shell	0 to 2000 sats	100 sats
Total two-shell layouts		198, 135 layouts
Total		
Target latitude	0 to 90 deg	23 deg
Total two-shell layouts		765, 450 layouts