Exploring the Use of Molecular Dynamics Simulations for High-Performance Space Debris Collision Modelling

Final report

Authors: Pablo Gómez[1], Fabio Gratl[2]
Affiliation: [1]ESA ACT, [2]Technical University Munich

Date: 18/06/2022

ACT research category: Informatics

Contacts:

Fabio Gratl
Tel: +49-89-289-18-618
E-mail: f.gratl@tum.de

Leopold Summerer (Technical Officer)
Tel: +31(0)715654192
Fax: +31(0)715658018
e-mail: act@esa.int

Available on the ACT website
http://www.esa.int/act

Ariadna ID: 21-5105
Ariadna study type: Standard
Contract Number: 4000134654/21/NL/AS

Contents

Abstract

Space debris has become a central topic for all space actors given the increased need for collision avoidance maneuvers, the impact on mission design, and the significant increase in the number of launched payloads. ESA and other space agencies have developed a range of sophisticated tools to analyze and predict the evolution of space debris as well as probabilities of conjunction events between space debris and active spacecraft.

For long-term simulations, numerical simulations are at the center of predicting the space debris environment of the upcoming decades. In light of debris generating events, such as continued anti-satellite weapon tests and planned mega-constellations, accurate predictions of the space debris environment are critical to ensure the long-term sustainability of critical satellite orbits. Given the computational complexity of accurate long-term trajectory propagation paired with conjunction tracking for a large number of particles, numerical models usually rely on Monte-Carlo approaches for stochastic conjunction assessment. On the other hand, deterministic methods bear the promise of higher accuracy and can serve to validate stochastic approaches. However, they pose a substantial challenge in terms of computational feasibility.

This report describes the outcomes of the Ariadna study No. 21-5105 exploring the adaptation of highly optimized software created for molecular dynamics simulations to efficiently perform long-term space debris simulation. In the scope of the study, we present the architecture and proof-of-concept results for a numerical simulation capable of modeling the long-term debris evolution over decades with a deterministic conjunction tracking model. For the simulation, an efficient propagator in modern C++ accounting for Earth's gravitational anomalies, solar radiation pressure, and atmospheric drag was developed. We utilized *AutoPas*, a sophisticated particle container, which automatically selects the most efficient data structures and algorithms.

First, we present results from a simulation of 16 024 particles in low-Earth orbit over 20 years. Overall, conjunctions are tracked for both, predicted collisions and close encounters, to allow a detailed study of potentially catastrophic events. We analyze the runtime and computational cost of the simulation in detail. Secondly, a larger-scale simulation featuring an additional 598 491 small debris particles on a supercomputing cluster was conducted using 3584 cores for a

shorter simulation timeframe to investigate the scaling potential. Results show that the problem is very scalable, and already a simple MPI parallelization can achieve a speed of 0.2 seconds per timestep or roughly 50x real time.

In summary, the results show that modern computational tools finally enable deterministic conjunction tracking in long-term simulations and can serve to validate prior results and build higher-fidelity numerical simulations of the long-term debris environment. All code of the study is available open-source online and we provide detailed next steps to further increase the fidelity of the simulation.

Parts of this report are based on publications accompanying the conducted research. [1, 2, 3, 4]

1 Introduction

1.1 Background

In recent decades the problem of space debris has become a central factor in mission design, spacecraft operations, and the question of the long-term sustainability of important orbits. Already, more than 5-10% of total mission costs for low-Earth orbit (LEO) satellites are protective and mitigation costs [5]. With the sharp increase in the number of launched payloads in recent years [6] (see Figure 1), the challenges caused by space debris are poised to increase further. As seen in Figure 2, current modeling efforts indicate that continuing the current trend of launches and practices will lead to an increasingly exponential development in the number of expected catastrophic collisions in orbit.

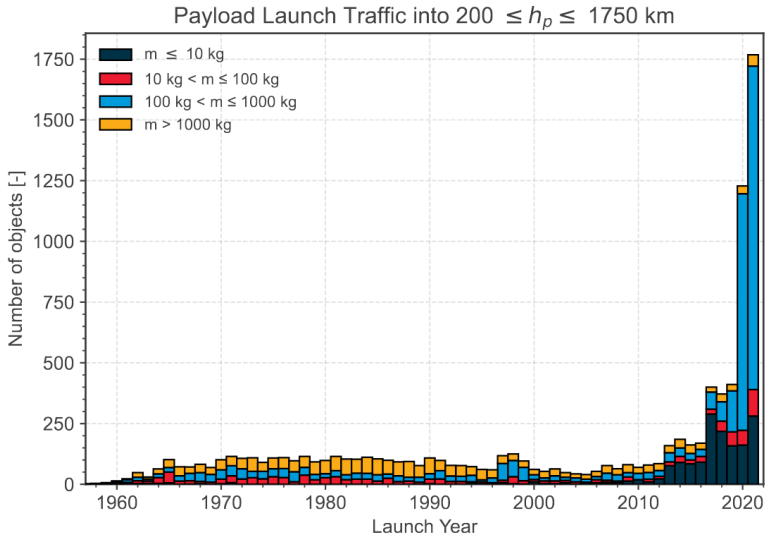


Figure 1: Launches into low earth orbit per year [6]

Consequently, modeling efforts of the space debris environment over a long time frame have become increasingly relevant in the last decades. The most well-known tool for long-term debris environment simulation is NASA's *LEGEND* software [7] but there are other tools such as ESA's *DELTA* software [8]. A central challenge in long-term modeling lies in dealing with the need to simulate ever-growing numbers of particles (satellites and debris) with a fine

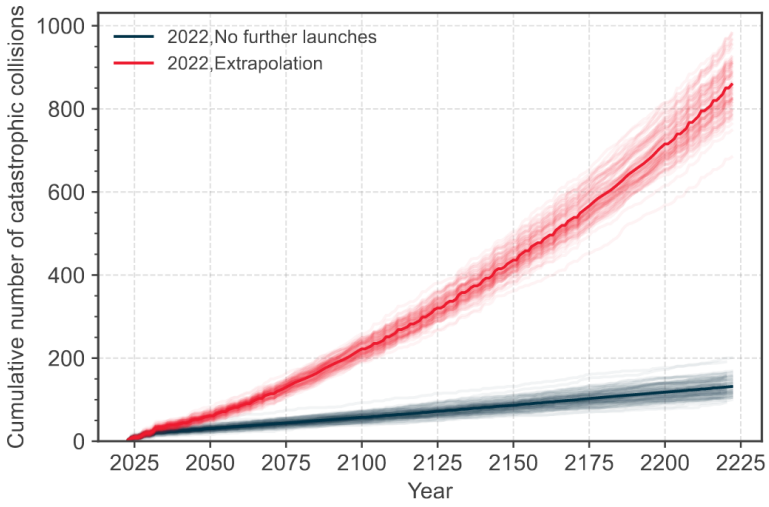


Figure 2: Expected number of collisions in long-term space debris simulations [6]

temporal resolution for accurate propagation and conjunction tracking – with timesteps on the scale of seconds or less – while the overall simulation time spans decades, requiring millions if not billions of necessary iterations depending on the setup. Furthermore, the question of whether two particles collide during a timestep is, fundamentally, a pairwise interaction requiring exhaustive checks of distances between the particles' positions. These two factors have made the deterministic simulation of the debris environment over a long time frame prohibitively expensive in terms of computational cost. Therefore, previous long-term models, such as *LEGEND* [7], rely on stochastic analyses of the dynamics and occurring collisions.

In another field concerned with large-scale particle simulation, computational molecular dynamics, similar challenges have been dealt with. Here, particles are propagated by computing short-ranged pairwise potentials and solving Newton's equations of motion for all particles in every iteration. Due to the fast decay of the employed potentials with growing distance, efficiency optimizations are possible for the algorithms applied. They resolve the pairwise interaction, which is prohibitively computationally expensive in $O(N^2)$, in $O(N)$ by introducing a spatial cutoff beyond which forces are too small to contribute in a significant way [9]. Long simulation times are common,

needing millions of timesteps for the equilibration of systems, or sampling of evolving properties [10].
Based on these advances, ESA and TU Munich have been exploring these algorithms in this study to use them to model the long-term space debris environment in a deterministic fashion. State of the art tools like GROMACS[1] or LAMMPS[2] were deemed to be to specialized to the molecular dynamics domain to be suitable for the simulation problem at hand. With its highly customizable plugin approach LAMMPS looks promising at a first glance, however as the tool is tailored to molecular dynamics using it would have meant writing new plugins for almost every aspect of the simulation while trying to fit everything in the LAMMPS framework.
This reports presents findings obtained from the newly created *Large-scale Deterministic Debris Simulation* (*LADDS*) implemented in `C++17`. In particular, we study the viability of modeling objects in LEO, which given their speed are most challenging, over a period of 20 years. Detailed descriptions of *LADDS* together with an extensive analysis of the observed conjunctions and single node, as well as multi node hybrid-parallelized performance follow. All code for this project is available online under an open-source license.[3]

Study Objectives In summary, the following objectives were identified as central to the study:

1. Investigate the feasibility of the performing long-term space debris simulations without relying on stochastic / Monte Carlo methods

2. Demonstrate the obtainable performance gains and scalability when applying state-of-the-art molecular dynamics software to space debris simulations

3. With the expected performance improvements, demonstrate feasibility of the simulation of the small debris particles ($\geqslant$ 1cm) which has typically been neglected. This is also necessary to model the increasing number of large objects in orbits.

4. Develop modular, open-source software solutions to enable comparative

[1] https://www.gromacs.org/
[2] https://www.lammps.org/
[3] https://github.com/esa/LADDS/ Accessed: 2022-06-18

studies between components (e.g. different breakup models or propagators) and support future follow-up studies

2 Methods

The following components were identified as critical parts of the software implementation:

- **Propagator**: Required to compute the trajectories of orbiting particles (i.e. debris and satellites). Has to provide an efficient trade-off between long-term accuracy and computational cost.
- **Conjunction Tracking**: Conjunctions during the simulation need to be identified and tracked efficiently. In particular, the number of pairwise comparisons needs to be minimized and discretization errors due to timestepping compensated or accounted for.
- **Breakup Model**: To correctly model potential cascading effects such as the infamous *Kessler Syndrome* [11] it is important to efficiently account for debris created during collisions.

These components and the software implementation are described in detail in the following.

2.1 Propagator

Overall, the implemented propagator aims to be computationally cheap while modeling all essential perturbations, which affect the particles' trajectories in LEO. Efficient parallelization is a critical factor for high performance. Hence, integrators which use adaptive timesteps are challenging to parallelize as individual particles don't share a timestep size. Thus, while higher accuracy integrators are available [12] they were in their native form not suitable for this work. Instead, we implemented a fully parallel propagator relying on a fixed timestep. The modeled perturbations are based on Keplerian orbits combined with spherical harmonics (J2, S22, C22), solar radiation pressure, and atmospheric drag. Since we target an LEO population, lunar and solar gravitational forces are considered negligible. They are however available to be turned on in the provided propagator. The concrete equations of motion are similar to the formulations described by Vallado [13].

In summary, we use the following descriptions:

Keplerian Motion

$$\vec{a}_{Kep} = -\frac{G(m_\oplus)}{|\vec{x}|^3}\vec{x} \tag{1}$$

with $\vec{x}$ position relative to Earth core, G the gravitation constant and $m_\oplus$ the mass of Earth.

Spherical Harmonics (inhomogeneous gravity field)

The potential U is given as

$$U = \frac{Gm_\oplus}{|\vec{x}|}\left(1 + \sum_{l=2}^{\infty}\sum_{m=0}^{l}\left(\frac{R_\oplus}{|\vec{x}|}\right)^l P_{lm}\left(\sin\left(\phi_{gc}\right)\right)\left(C_{lm}\cos\left(m\lambda_{gc}\right) + S_{lm}\sin\left(m\lambda_{gc}\right)\right)\right) \tag{2}$$

with $R_\oplus$ as the radius of Earth, ϕ_{gc} the geocentric longitude, P_{lm} the Legendre function, λ_{gc} the object's polar coordinates, and C_{lm} and S_{lm} are empirical coefficients determined from observation data of satellites orbiting the Earth [14]. The acceleration follows from the partial derivatives of the potential

$$\vec{a}_{harmonics} = \frac{\partial U}{\partial|\vec{x}|}\left(\frac{\partial|\vec{x}|}{\partial\vec{x}}\right)^T + \frac{\partial U}{\partial\phi_{gc}}\left(\frac{\partial\phi_{gc}}{\partial\vec{x}}\right)^T + \frac{\partial U}{\partial\lambda_{gc}}\left(\frac{\partial\lambda_{gc}}{\partial\vec{x}}\right)^T. \tag{3}$$

Note that this formulation also includes the zonal terms.

Solar Radiation Pressure

$$\vec{a}_{SRP} = \mathtt{AU}^2 P_{SRP}\frac{A}{m}\frac{\vec{x} - \vec{x}_\odot}{|\vec{x} - \vec{x}_\odot|^3} \tag{4}$$

with $\mathtt{AU}$ as the astronomical unit, $P_{SRP} = 4.56 \cdot 10^-6\frac{N}{m^2}$ the solar radiation pressure at 1 $\mathtt{AU}$, A object's area facing the sun, m the object's mass, and $\vec{x} - \vec{x}_\odot$ the distance between the object's and Sun's center.

Atmospheric Drag

$$\vec{a}_{Drag} = -\frac{C_d A}{2m}p(h)|\vec{v}_{rel}|^2\frac{\vec{v}_{rel}}{|\vec{v}_{rel}|} \tag{5}$$

with C_d as the drag coefficient, A the object's (estimated) cross section area, m the object's mass, and $p(h)$ being the atmospheric density at the object's altitude h above ground based on [15]. $\vec{v}_{rel}$ is the object's velocity relative to Earth's atmosphere.

2.2 Conjunction Tracking

The conjunction detection method is the second critical component for simulating and estimating the long-term occurrence of conjunctions. Given that this work aims to showcase the potential of a deterministic simulation, previous approaches with probabilistic collision estimates, such as the *Cube* approach [16, 17], are not applicable. Instead, we propose a method – similar to Alarcon *et al.* [18] – that is based on the deterministic positions, size and operational status of the particles. To keep computational costs manageable, we assume spherical particle shapes for the purposes of conjunction detection. Furthermore, we use a sub-timestep interpolation to achieve a higher temporal resolution than the timestep δt used for the propagation.
More precisely, given two particles p_1 and p_2, their locations x_1^t,x_2^t and velocities v_1^t,v_2^t at timestep $t+1$, and radii r_1,r_2, we compute their closest approach d in the interval $[\tau_t, \tau_{t+1}]$ using a linear interpolation as depicted in Figure 3. This allows an analytic formulation that is more precise than using d_t or d_{t+1} while remaining computationally cheap. Thus, the time of the closest encounter during this timestep τ_d is given by

$$\tau_d = \frac{x_1^t v_1^t - x_1^t v_2^t + x_2^t v_2^t - x_2^t v_1^t}{v_1^t v_1^t + v_2^t v_2^t - 2 v_1^t v_2^t} \tag{6}$$

and consequently, the closest encounter distance is given by

$$d = \begin{cases} ||x_1^t - x_2^t||_2 & \text{for } \tau_d < 0 \\ ||x_1^t + \tau_d v_1^t - x_2^t - \tau_d v_2^t||_2 & \text{for } 0 \leqslant \tau_d \leqslant \delta t \\ ||x_1^t + \delta t v_1^t - x_2^t - \delta t v_2^t||_2 & \text{for } \delta t < \tau_d \end{cases} . \tag{7}$$

Based on this formulation, we define a collision as an encounter where $d \leqslant (r_1 + r_2)$. In light of unavoidable numerical errors and to study the impact of the estimated particle radii, we introduce the additional parameter κ to study all conjunctions below some distance. Thus, a conjunction occurs when $d \leqslant \kappa(r_1 + r_2)$. We tracked conjunctions for $0 < \kappa \leqslant 10$.
Furthermore, the simulation distinguishes between passive objects and actively operated satellites as defined by the *CelesTrak SATCAT Operational Status*[4] codes. Conjunctions between actively operated satellites are discarded as these would presumably be evaded in daily satellite operations.

[4] https://celestrak.com/satcat/status.php Accessed: 2022-06-18

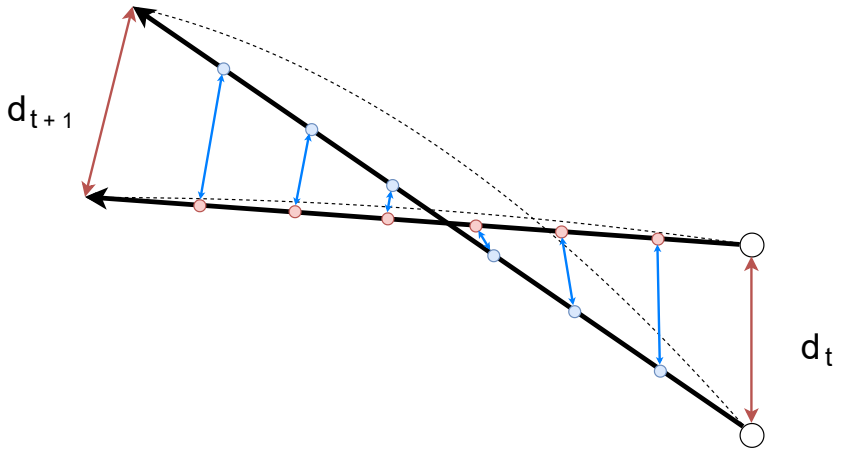


Figure 3: Schematic illustrating the actual trajectories of two particles (dashed lines), linear interpolations (thick lines) with particle positions and distances at some points in time (as dots and blue lines) as well as the naive closest encounters at τ_t and τ_{t+1} (red lines). The interpolation allows a sub-timestep estimation of the distance at closest encounter d that is more accurate.

Conjunctions between active satellites and passive objects are only considered if the passive object has a radius smaller than 10 cm to model the challenges in tracking small objects [19]. Finally, as this initial work focuses only on conjunctions and not simulating resulting breakups, satellites remain active after a conjunction. To account for this, only the closest encounter two objects had throughout the simulation is considered to avoid repeated encounters during consecutive timesteps.

2.3 Breakup Model

For the simulation of breakups due to collisions, the developed software relies on a validated reimplementation of the *NASA Breakup Model* [20]. As part of the study [2], the *NASA Breakup Model*was implemented in modern *C++17* and is available open-source online.[5] The *NASA Breakup Model* is a widely established standard in long-term debris modeling [7] although it has

[5] https://github.com/esa/NASA-breakup-model-cpp Accessed: 2022-07-06

some limitations. Most notably, satellite material compositions have changed since its experimental validation and the model does not conserve mass. The latter has been remedied in the re-implementation but the former point is an outstanding issue given a lack of experimental data.
In condensed form, the *NASA Breakup Model* describes the fragments resulting from either collision of two spacecraft or explosions of individual ones (e.g. due to leftover propellant). It has been validated on different real breakups such as anti-satellite missile tests. Based on radar cross sections the radius of a spacecraft is approximated from that mass and density. Combined with the velocities of the spacecraft, based on these data, a population of debris particles is generated. A more detailed description of the implemented model can be found in the accompanying thesis [2]. Note that the model itself is not deterministic, however, it can be seeded to create reproduce specific simulations.
An exemplary breakup of two small 50 cm satellites each weighing 10 kg into about 3000 debris fragments in a circular orbit at 380 km altitude is displayed in Figure 4. The satellites initially have opposing directions of motion which leads to two clouds similarly going in opposite directions. The corresponding debris fragments partially continue orbiting the Earth while many of them burn up in the atmosphere or escape the simulation domain ($>$ 10000 km from the center of the Earth). The debris particles quickly spread out in terms of true anomaly while mostly retaining the orbital plane in which the satellites operated.

2.4 Implementation

The implementation of *LADDS* follows a modular design, compartmentalizing any state within each component. This method increases the comprehensibility of the code and facilitates exchanging components to explore the impact of, e.g., different propagation or conjunction tracking algorithms.
The first of the major components is the storage of all simulated particles. For this, we employ *AutoPas*,[6] a `C++17` node-level high-performance library for arbitrary, short-range N-Body simulations. Here, it acts as a black-box particle container, which means that the simulation is oblivious to the actual data structure used within *AutoPas*. This allows *AutoPas* to freely choose its internal data structures and algorithms, such as Linked Cells or Verlet Lists,

[6] https://github.com/AutoPas/AutoPas Accessed: 2022-06-18

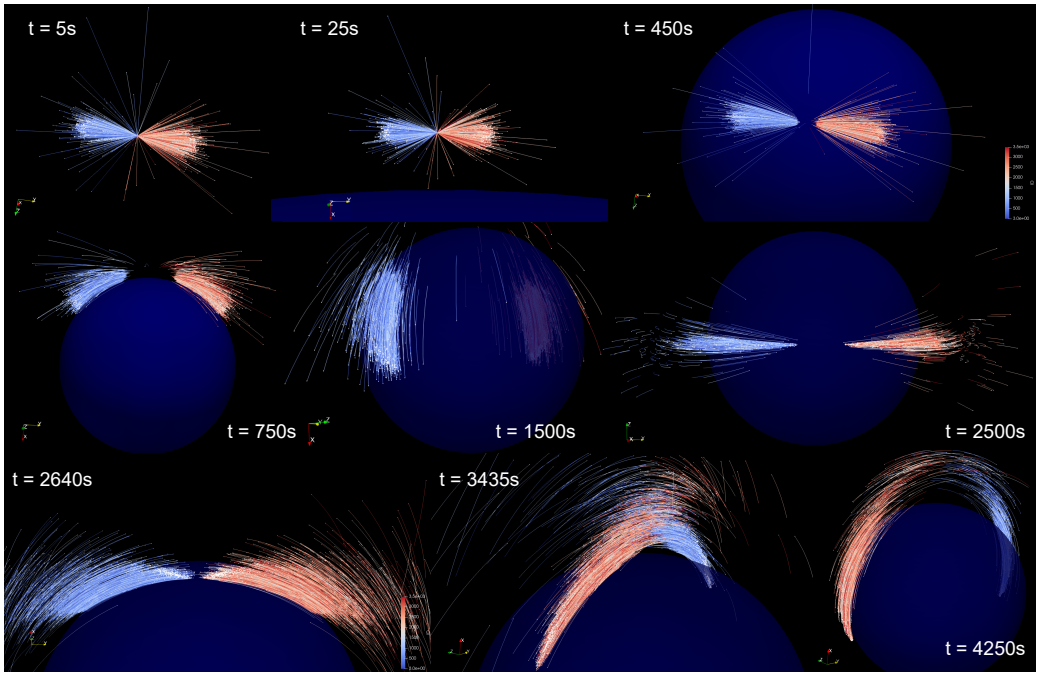


Figure 4: Visualization of a breakup event between two satellites on opposite orbits at 380km altitude. The blue sphere represents Earth, trace colors indicate to which satellite the fragment belonged. A total of 2.5 orbits is shown.

as well as algorithmic optimizations, e.g. exploiting symmetries in potentials to reduce the number of computations. For more information about the possible configurations refer to the *AutoPas* release paper [21] or the official documentation.[7] The library can even adapt this configuration at runtime multiple times over the course of the simulation – a feature called dynamic automatic algorithm selection.

Designed for N-Body simulations, *AutoPas* provides an interface for the efficient pairwise interaction of spatially close particles. We use this feature for our conjunction tracking by implementing the algorithm presented in Section 2.2 within a functor which we pass to *AutoPas*. During such a pairwise iteration step, *AutoPas* then applies the functor to at least all particles within a given cutoff distance. If they are indeed within that cutoff and at

[7] https://autopas.github.io/ Accessed: 2022-06-18

least one particle is passive and has a radius smaller than 10 cm, we compute the linear interpolation to estimate the closest sub-timestep approach and whether a conjunction has taken place.
When compiled with *OpenMP*,[8] *AutoPas* will automatically execute this pairwise interaction using shared memory parallelization, choosing the fastest algorithm while exploiting interaction symmetry and avoiding data races.
The second major component is the numerical propagator, responsible for simulating the motion of particles over time. For this we developed *Orbit-Propagator*,[9] which implements a fourth-order Yoshida integrator [22] and the perturbations described in Section 2.1. The integrator was validated against *heyoka* [12]. For every position update, all particles stored in *AutoPas* are piped through the propagator. Since all of these updates are independent of each other, this is easily parallelized in *OpenMP*.
To enable the use of vast computing resources *LADDS* employs MPI for distributed memory parallelization. The domain is divided into smaller box-shaped subdomains with one subdomain per MPI rank. Due to the short time horizon, only a regular grid decomposition was implemented, which is shown in Figure 5. From the figure, it is already apparent that given enough ranks, hence, small enough subdomains, there are boxes in the corner of the domain and the center of the earth which might contain very few to no particles. This is of course suboptimal from a resource utilization perspective, on the other hand, the near-empty ranks do not add any overhead as they also do not induce any further communication. The big advantages, which are also the reasons why this decomposition was chosen, are its ease of implementation on the one hand and its perfect load balancing for up to eight ranks due to the high symmetry of the problem on the other hand.
Figure 6 shows the general flow of the simulation. After the initialization step, the rest of the simulation consists only of the main simulation loop, primarily alternating between the integrator and the conjunction detection. The frequency in which output is written can be configured by the user.

[8] https://www.openmp.org/ Accessed: 2022-06-18
[9] https://github.com/FG-TUM/OrbitPropagator Accessed: 2022-06-18

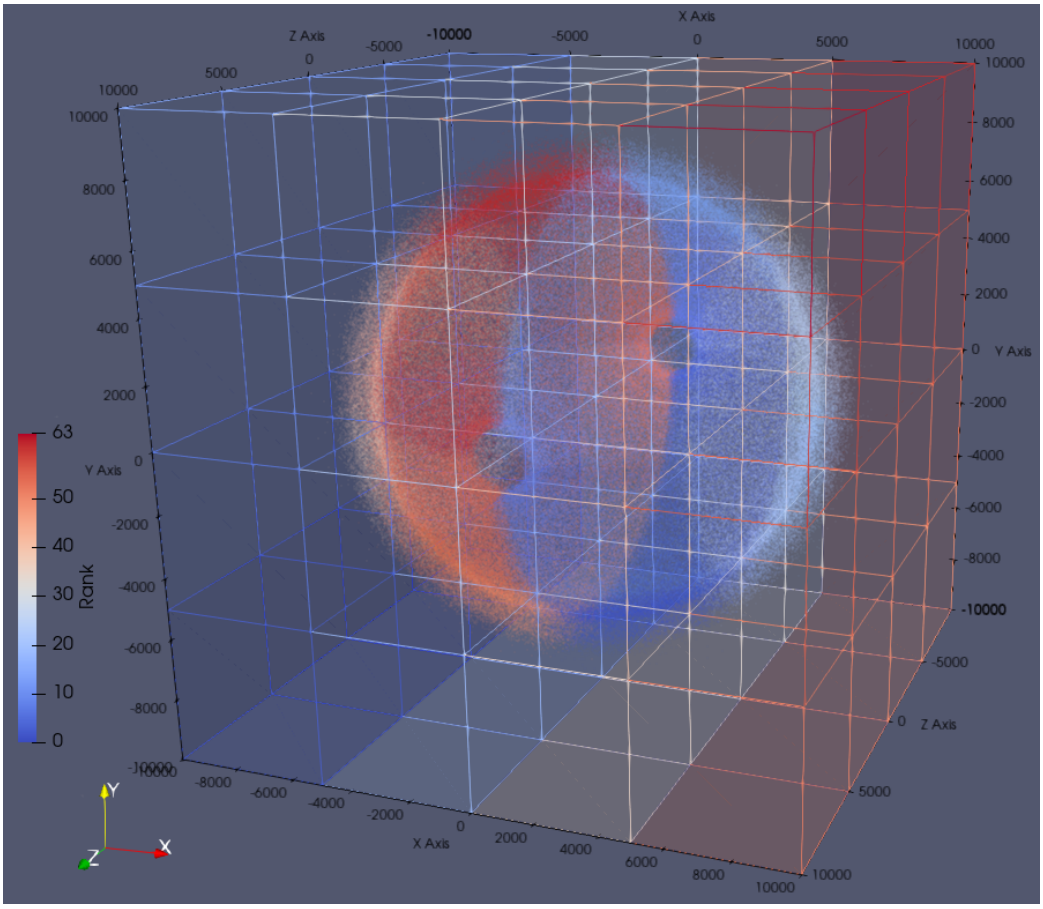


Figure 5: Visualization of the MPI decomposition with 64 ranks at timestep 0. Particles are colored by their ID. Ranks assign global IDs from local ID ranges to avoid overlapping.

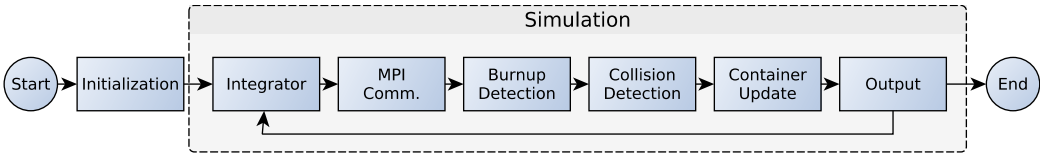


Figure 6: Flowchart of *LADDS*

3 Results

3.1 Setup

All presented results on the base population were obtained on the *CoolMUC2*[10] segment of the *Linux Cluster* at the *Leibniz Supercomputing Centre*. The code was compiled with *GCC* version 10.2.[11]. [12] For the base population we only make use of shared memory parallelization for the base population. There the simulations were executed with 28 threads, which is a full node without hyper-threading. The extended population of 600 thousand particles was executed with an MPI+*OpenMP* hybrid parallelization. On each rank, 28 *OpenMP* threads were running and there are always two ranks per node, leveraging the complete hardware.
The actual simulations were conducted with a timestep $\delta t = 10$ seconds. Particles are removed as soon as they reach an altitude of less than 150 km above ground as they are considered to be burning up. *AutoPas* is restricted to finite, box-shaped domains. Therefore, the simulation is confined to a cube with a side length of 20000 km and its center at the center of the Earth. This size was chosen so that no particles will escape throughout the simulation, so no boundary conditions need consideration.

3.2 Data

3.2.1 Base Population

The dataset for the presented results on the base population, that is all currently tracked objects, originates from the data provided by *CelesTrak*[13] and the *18th Space Control Squadron* on *space-track.org*.[14] Data from a total of 16 024 objects was used after filtering based on several criteria. The International Space Station (ISS) was removed from the data given its exceptional size (which makes collisions much more likely). Already operational satellites from the constellations by OneWeb and SpaceX (Starlink) were removed as these are being studied separately in a parallel study. Further, this work

[10] https://doku.lrz.de/display/PUBLIC/CoolMUC-2 Accessed: 2022-06-18
[11] https://gcc.gnu.org/ Accessed: 2022-06-18
[12] https://www.lrz.de/ Accessed: 2022-06-18
[13] https://celestrak.com/ Accessed: 2022-06-18
[14] https://www.space-track.org/ Accessed: 2022-06-18

focused in particular on LEO where conjunctions are – given the smaller volume – proportionally likelier, thus only satellites operating between 175 and 2000 km above the ground were investigated.
Only conjunctions between active and passive objects smaller than 10 cm radius or passive and passive objects are considered, respectively (see Section 2.2). Thus, based on the *CelesTrak SATCAT Operational Status* the dataset contains 1996 active and 14 028 passive objects. Similar to the work by Johnson *et al.* [20], radii of objects were estimated as

$$r[m] = \sqrt{\frac{RCS}{\pi}}$$

based on the radar cross section (RCS) and masses as

$$m[kg] = \begin{cases} \frac{4}{3}\,\pi\,r^3\,2698.9\frac{kg}{m^3} & \text{for } r \leqslant 0.01m \\ \frac{4}{3}\,\pi\,r^3\,92.937\frac{kg}{m^2}\,(2r)^{-0.74} & \text{for } r > 0.01m \end{cases}.$$

These values were also used to compute the area-over-mass for the solar radiation pressure and atmospheric drag. If available on *CelesTrak* the numerical RCS from there was used. Otherwise, the *space-track.org* RCS classification into *small* ($RCS < 0.1\ m^2$), *medium* ($0.1\ m^2 < RCS < 1\ m^2$) and *large* ($1\ m^2 < RCS$) was used to sample from normal distributions for these respective classes matching the data from *CelesTrak* in terms of mean and standard deviation. No objects below $r = 0.005\ m$ were included. All objects were propagated to the time 2022-01-01 00:00:00.000 using the *SGP4* propagator [23] implementation available through the Python module *pykep* [24]. BSTAR values used in the drag formulation (see Section 2.1) relied on the values provided in the two-line element set if available. Otherwise, they were approximated using above computed radius, mass and a drag coefficient C_d of 2.2 [25]. A detailed overview of the dataset's distribution of semi-major axis, mass and radius in relation to status is given in Figure 7. As expected, most active objects are larger with more mass and many of them are in LEO. Passive objects are on average smaller, lighter, and have a higher semi-major axis.

3.2.2 Small Debris Population

In addition to the base population, an extended dataset of small debris between 1 and 10 cm radius was created to accurately assess the prevalence

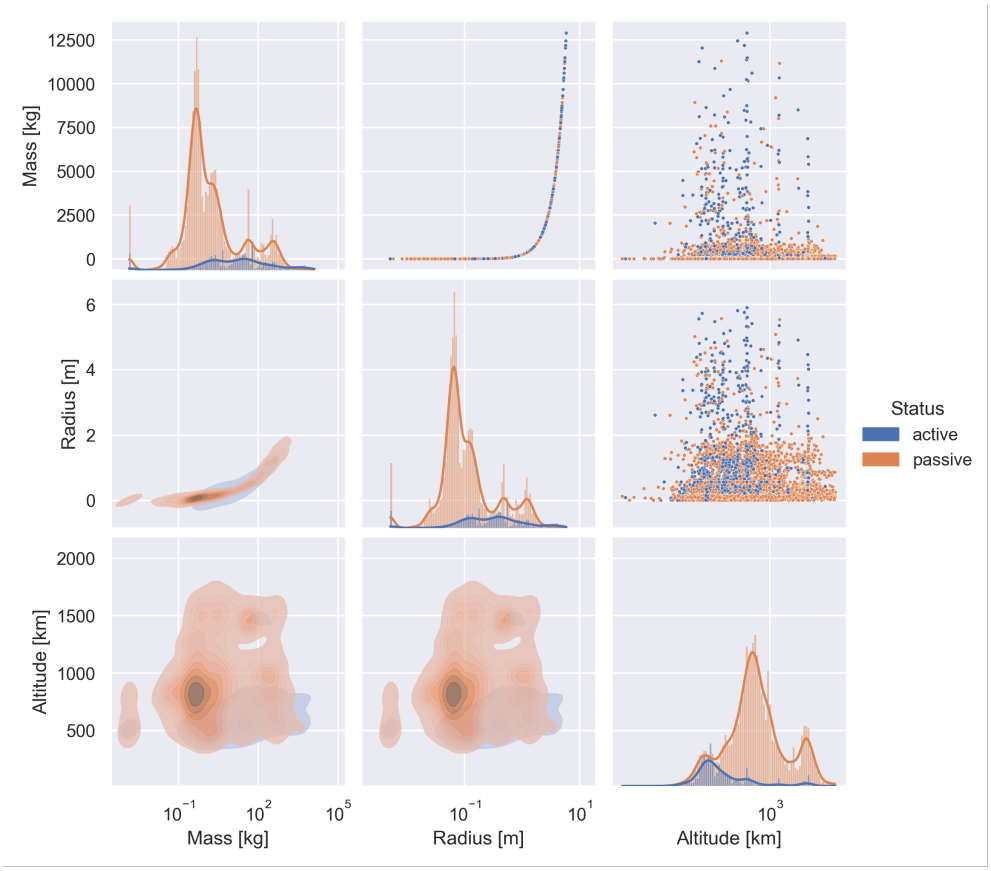


Figure 7: Detailed distributions for active and passive objects in regard to mass, radius and altitude above Earth of their orbit. Diagonal plots display histograms (with counts on the vertical axis), upper triangle scatters and the lower triangle kernel density estimates of the joint distributions.

and importance of small debris. ESA's MASTER-8 software tool [26] was used to model the spatial density of these small debris particles at the investigated altitudes between 186 km and 2000 km. For this purpose, altitudes were split logarithmically into 1000 bins. MASTER was used to sample four different sizes and debris was uniformly sampled within those bins of $[1.0cm, 2.5cm[, [2.5cm, 5.0cm[, [5.0cm, 7.5cm[, [7.5cm, 10.0cm]$. In this manner, a total of $598,491$ small debris particles were created. Their orbits

were computed using orbital elements with their semi-major axis uniformly sampled in their altitude bin. Inclination and eccentricity were sampled randomly according to the distribution in the base population using kernel density estimation. The longitude of the ascending node, the argument of periapsis, and the true anomaly were sampled uniformly at random. BSTAR was determined as described for the base population. Mass was determined from the size as for the *NASA Breakup Model* [20] and the value got down to 1.4g. Figure 8 displays the distribution of small debris by altitude. Based on MASTER, especially altitudes around 800 and 1400 km have particularly high small debris densities. In Figure 9, the full dataset combining small debris and the base population is shown. In comparison to only the base population the particles are on average much smaller, lighter, and at higher altitudes than average active satellites, which operate mostly in the lower LEO regime. Given the high incidence of small debris particles, most particles in the simulation are in fact small debris for this dataset.

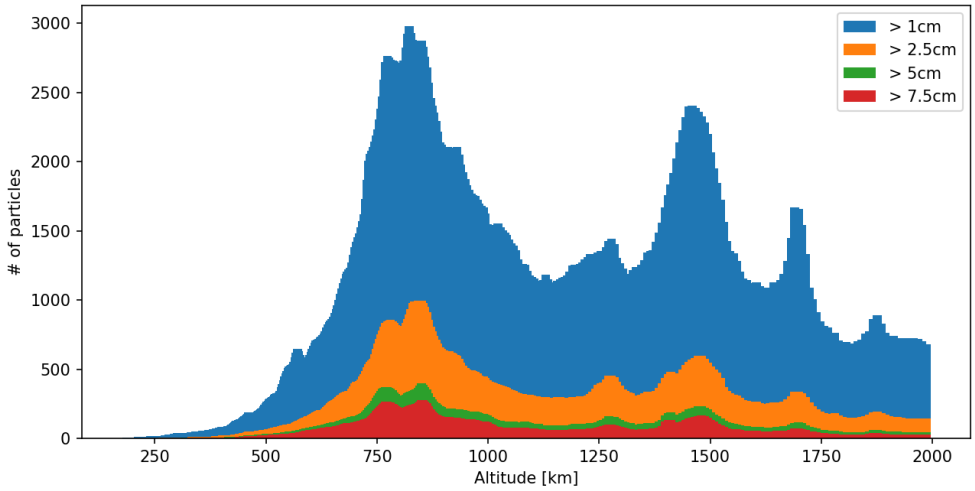


Figure 8: Number of small debris particles up to 10 cm at different altitudes above Earth by size. Especially very small debris particles between 1 and 2.5 cm are frequent.

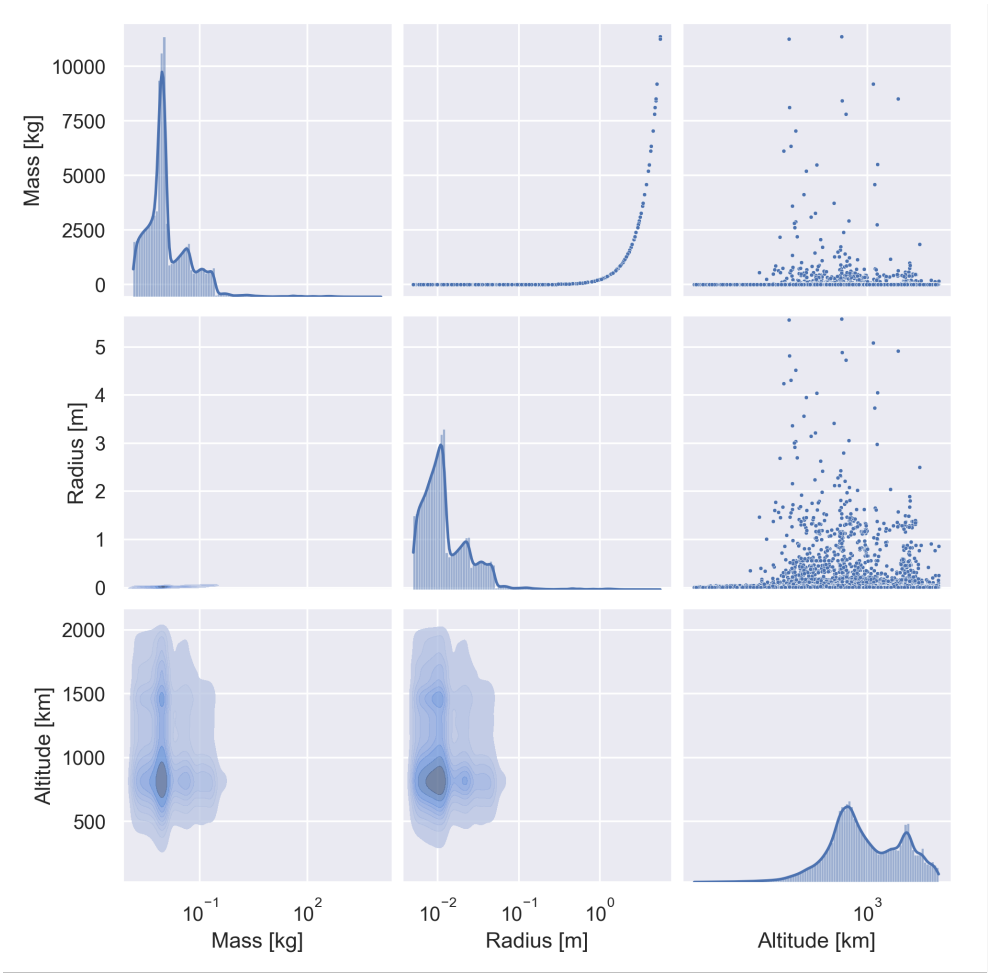


Figure 9: Distribution for the full population including small debris in regard to mass, radius and altitude above Earth of their orbit. Diagonal plots display histograms (with counts on the vertical axis), upper triangle scatters and the lower triangle kernel density estimates of the joint distributions.

3.3 Conjunctions in Long-term Modeling

This section describes results observed for the long-term study of the base population. In terms of conjunctions, several factors are noteworthy regarding the observations. First off, Figure 10 (left) displays the observed conjunc-

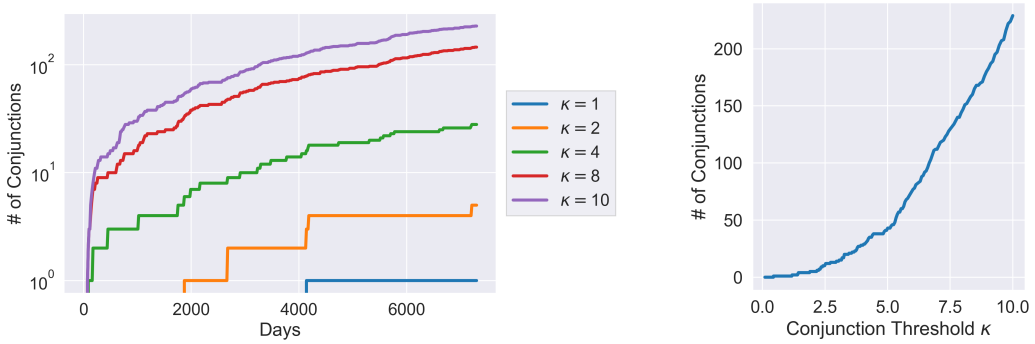


Figure 10: (left) Evolution of the number of conjunctions over time depending on the threshold κ; (right) Detailed relationship between the total number of conjunctions and κ

tions depending on κ over the course of the simulation. Given the cubical impact of κ, the number of observed conjunctions rises by two orders of magnitude. Between 1 and 229 conjunctions are detected for $\kappa = 1$ and $\kappa = 10$, respectively. In total, 70 conjunctions happened between active and passive particles while 159 conjunctions involved only passive particles (active particles evade each other). Figure 10 (right) displays the detailed relationship between the total number of observed conjunctions and κ. Overall, it is noteworthy that even though only one ($\kappa = 1$) collision was found, the much more frequent conjunctions are also a relevant quantity as they may trigger collision avoidance maneuvers due to higher collision probabilities as monitored in spacecraft operations [27, 28].

A second relationship worth monitoring is the one between size and conjunctions. As seen in Figure 11 (left), most conjunctions occur between large and small, and large and large particles. This is due to the higher likelihood of conjunctions given the objects' larger radii. The – compared to large-with-small conjunctions – small number of conjunctions between large and large particles is due to their overall lower occurrence rates and the higher ratio of active satellites with higher radii. Figure 11 (right) also confirms this as the size distribution of particles involved in conjunctions is clearly distinct from the overall population featuring a higher rate of large satellites. Note that collisions between large debris fragments are particularly critical as they will likely lead to a large number of new debris fragments. However, the very small particles below 10 cm radius also feature prominently in conjunctions

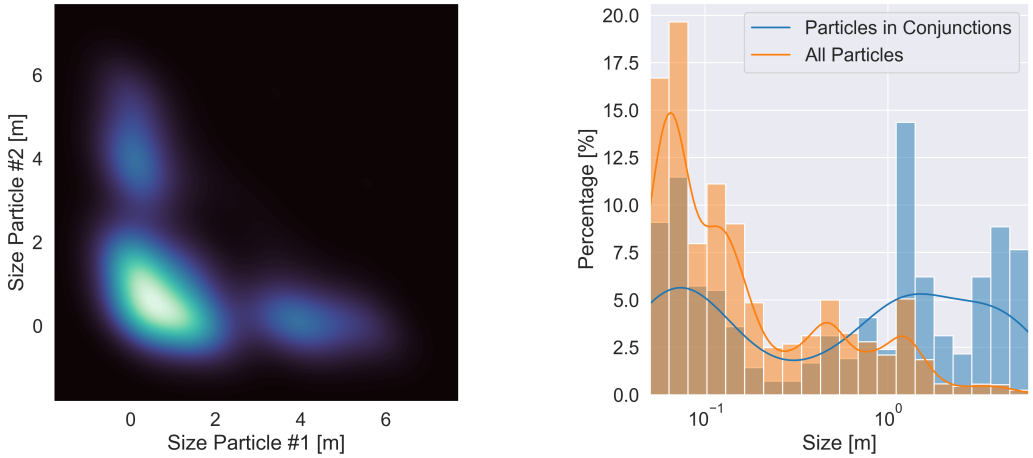


Figure 11: (left) Visualization of the sizes of objects in conjunctions with each other (commutative) with brightness indicating density; (right) Size of the particle population vs. the objects involved in conjunctions

as they are not evaded given the difficulty in tracking them (see Section 2.2).

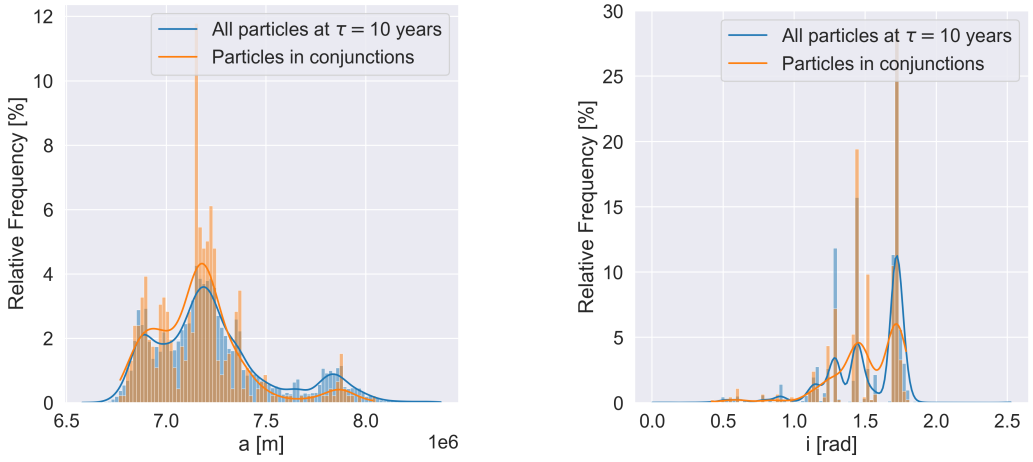


Figure 12: (left) Semi-major axes of the particle population vs. the objects involved in conjunctions; (right) Inclination of the particle population vs. the objects involved in conjunctions

Finally, a closer inspection of the orbital elements of objects in conjunc-

tions is warranted. Figure 12 (left) demonstrates that objects involved in conjunctions have a proportionally smaller semi-major axis compared to the overall population. This is particularly noteworthy since – as explored in Section 3.2.1 – especially active satellites in the population have a smaller semi-major axis, whereas most of the passive objects are at higher semi-major axes. The other orbital elements do not show similarly clear relationships. Figure 12 (right) displays the inclination of the orbits in conjunctions. There are some notable differences in the distributions, which feature different peaks, but the relationship is less clear compared to the semi-major axis.

3.4 Performance Analysis

3.4.1 Single Node

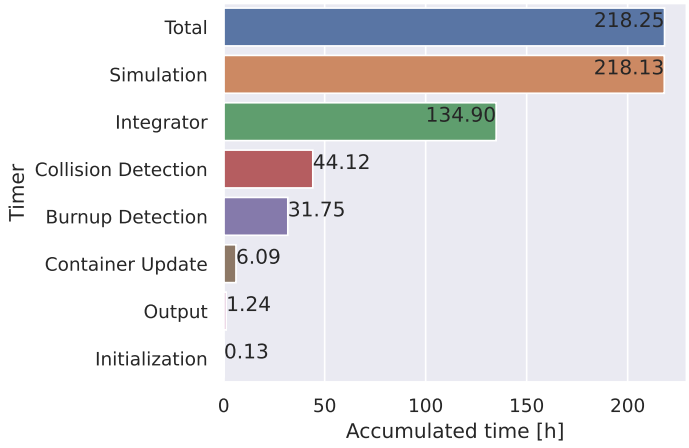


Figure 13: Accumulated runtime of steps over 20 years of the simulation. Labels correspond to the steps shown in Figure 6

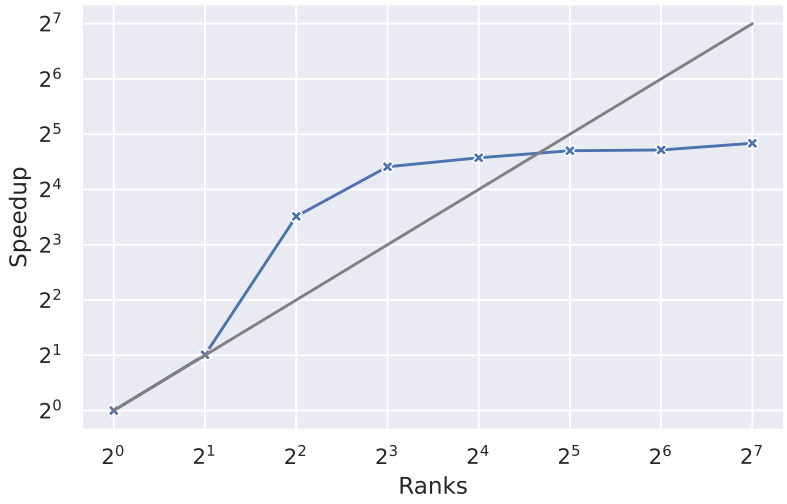
In the following, a closer look into the overall performance and performance-critical elements of the small simulation without distributed memory parallelization is taken. To get an idea of which parts of the simulation are the most expensive, Figure 13 provides a breakdown of all major steps in the initial run featuring the base population over twenty years without breakups.

Due to timeout restrictions on the utilized cluster computer, the simulation was conducted in four runs, by saving and reloading checkpoints. The `Initialization` timer covers the loading of the checkpoints. It shows an insignificant share of the total time at less than 0.1%. `Output` represents the time spent for writing the HDF5 files,[15] which track all conjunctions as well as all particle positions every 1000 iterations. Thereby, we end up with over 63 710 saved simulation states in a file of about 32 GB. The time spent on this is 0.5% of the total time and thus acceptable given the file size. Also, the `Container Update`, which tracks the time for *AutoPas*' bookkeeping to maintain the Linked Cells data structure, is reasonably quick with less than 3%. `Burnup Detection` takes longer than initially expected with almost 15% of the total time. This is due to the missing parallelization of this simulation step at the time of the long simulation. Later experiments with the same dataset and hardware showed that parallelizing the burn-up handling yields a speedup of 13 for this step. To put this into perspective, before the optimization `Burnup Detection` takes more than five times longer than the `Container Update`. After implementing the parallelization, it is more than twice as fast. Interestingly, *LADDS* spends only about 20% of the time on `Collision detection`. In molecular dynamics simulations, pairwise interactions similar to this are typically the dominating part. Here, however, the main part of the simulation was spent in the `Integrator` with over 61%. The reason for this is a combination of the integrator's higher amount of floating-point operations per particle, as well as the fact that due to interface restrictions, all particles have to be traversed 10 times per iteration, leading to suboptimal caching. Overall, the simulation speed was about 0.01 seconds per iteration which advanced the simulation by 10 seconds or 10.91 hours for one year. With this, it was possible to simulate 20 years in 218.25 hours.

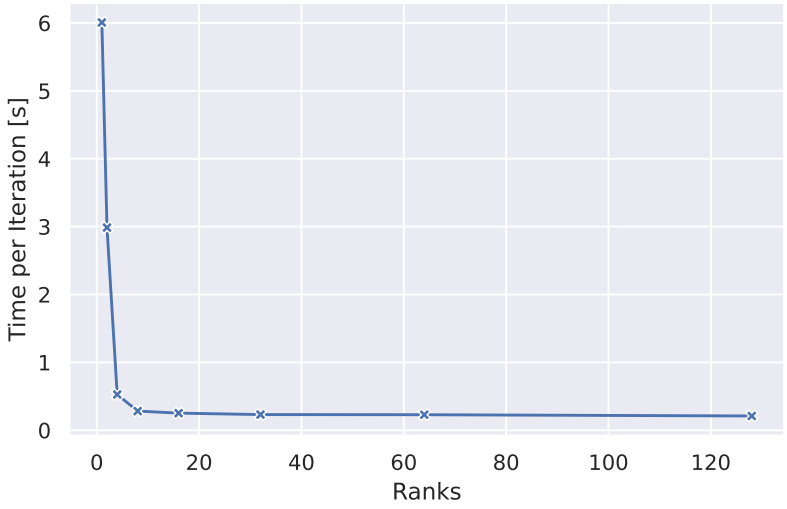
3.4.2 Multi Node

This section takes a look at the larger simulation, and how the MPI implementation performed with it.
Figure 14a shows the classical strong scaling analysis, where the same experiment is repeated with an increasing number of ranks. From two to four ranks the simulation seems to show super-linear speedup but for more than

[15] https://www.hdfgroup.org/solutions/hdf5/ Accessed: 2022-06-18



(a) Speedup relative to one rank. Gray line shows linear scaling for reference. log-log Plot.



(b) Time per iteration.

Figure 14: Strong scaling of the 600k small debris scenario for 1 to 128 MPI ranks.

eight ranks only small gains are achieved, up to a performance of 0.21 seconds per iteration for 128 ranks. However, in the context of *AutoPas*, such speedup plots can be deceiving, because at each point a different mixture of algorithms is employed, which explains the super-linear scaling. There are two reasons for the performance ceiling after eight ranks. On the one hand, the wall clock time per iteration is already down to 0.28 seconds, as can be seen in Figure 14b, so there is only so much room for improvement. On the other hand, the domain decomposition starts to become inefficient for more than eight because ranks do not get equal shares of the orbital shell anymore. This imbalance thwarts scalability.

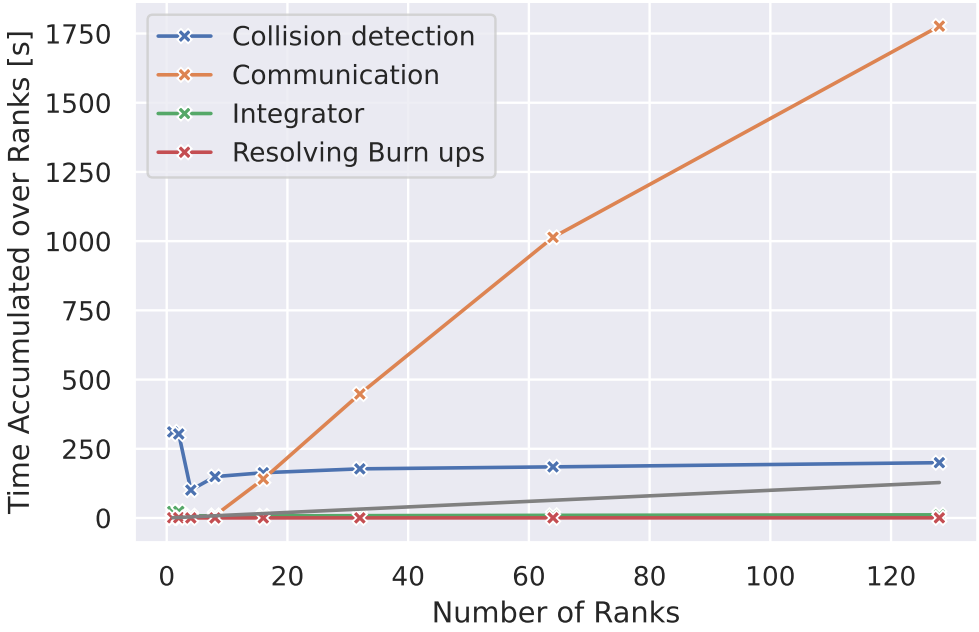


Figure 15: Component timers accumulated over all ranks for each step of the strong scaling. Gray line shows linear scaling for reference.

To better see the effect the decomposition has on the performance we take a look at the simulation component timers accumulated over all MPI ranks in Figure 15. We see that the most relevant timers presented in Section 3.4.1 have hardly changed. This is expected, as the computational effort for e.g.

the integrator stays the same with any number of ranks. The individual ranks' time decreases, however, their sum will stay the same. Only for the collision detection, there is a fluctuation for low rank numbers, this however is due to the changing in algorithms in *AutoPas*. Starting from 16 ranks, we see communication quickly taking more and more time. The scaling is linear, however with a large factor of about 15, taking up to 80% of the total time for 128 ranks. It is a defining characteristic of LEO satellite simulations, that particles travel fast with respect to the domain size. Hence, every straight boundary plane placed in the domain will induce a lot of MPI particle communication.

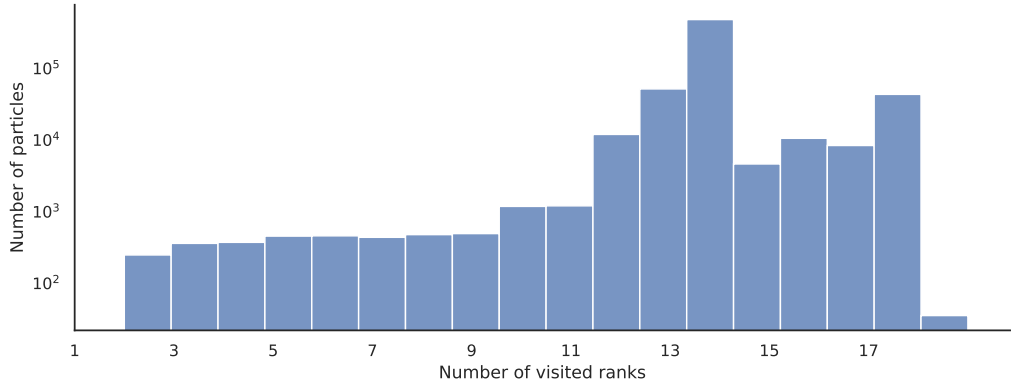


Figure 16: Number of particles that visited a given amount of ranks over 800 iterations. Notice the log y-scale.

Figure 16 shows how many particles visited how many MPI ranks over 800 iteration steps. In total 613 397 rank changes happened throughout this very short simulation with the vast majority ($> 78\%$) of particles visiting 14 ranks. When looking at the number of particles per rank over the course of the simulation, as displayed in Figure 17, we see a very stable distribution. In the Figure, error bars show the variation of the number of particles, and the fact that they are reduced to points shows how little movement is in these numbers. This figure also demonstrates the immense load imbalances within the simulation with some ranks holding over 35 000 particles while others hold zero.

To summarize, we have demonstrated very good potential for scaling of the overall simulation with already good scaling behavior for all components except for MPI communication. The current implementation leaves room for

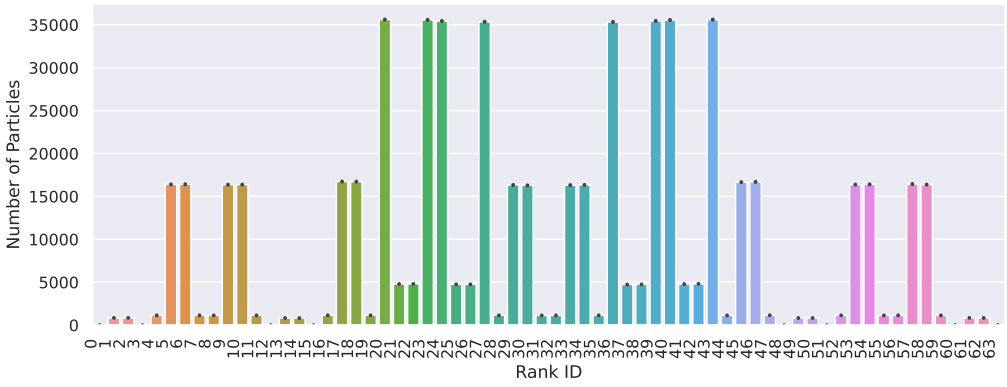


Figure 17: Number of particles per rank. Error bars show the variation over the whole simulation.

improvement. For example, MPI ranks could model layers of orbital shells instead of cuboid boxes of the simulation domain. Then, only particles with a certain eccentricity would change between ranks. Since most stable particles in LEO tend to be on near-circular orbits, this would significantly reduce the rank changes, and thus communication.

3.4.3 Distributed Auto Tuning

As mentioned previously, each MPI rank maintains its own *AutoPas* instance, which tunes its internal algorithmic configuration independently from all others. This leads to situations, where different subdomains of the simulation are processed with different algorithms. Qualitatively all algorithms achieve the same result (within floating point precision). *AutoPas* ensures a unified interface for the seamless interaction between any combinations.
In Figure 18 the choices for the particle container of each rank is shown for the larger simulation on 64 ranks.
We can see, that the majority of ranks opted for `VarVerletLists`, which is a very versatile Verlet Lists based container. It excels at medium particle densities with inhomogeneous but stable particle distributions. This is exactly what is present throughout most parts of the domain. Ranks in the corner which contain no particles at all opted for `LinkedCells`. An explanation is that this container has less bookkeeping overhead compared to the Verlet List based containers. Since there are no particles in these ranks only

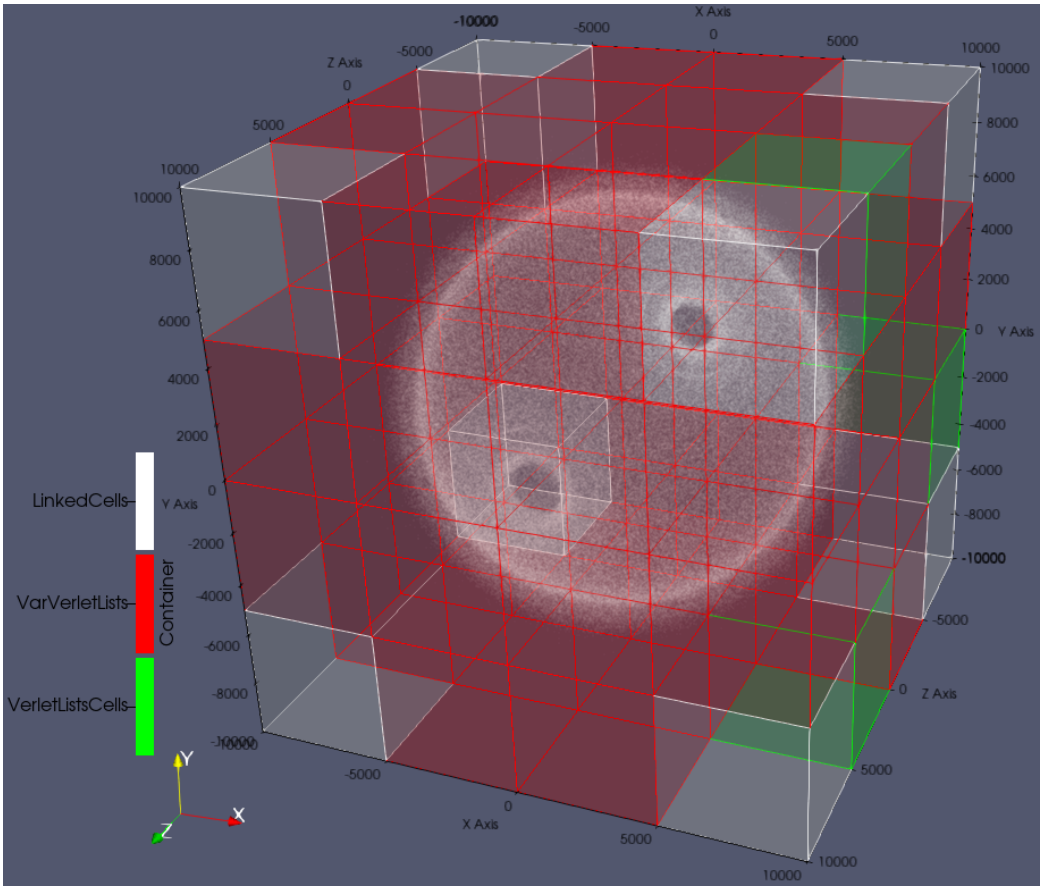


Figure 18: Particle container choices per rank. Time step 800 of the big simulation on 64 ranks.

the overhead matters, hence, `LinkedCells` have the edge. The remaining three border ranks opted for the hybrid algorithm `VerletListsCells`. They contain only a few particles, therefore benefitting from the low overhead of the `LinkedCells` aspects and the quicker evaluation thanks to `VerletLists` traits.

4 Conclusion

4.1 Main Findings

This study demonstrated that deterministic conjunction tracking of space debris over a long time is feasible and scalable. Relying on state-of-the-art particle simulation methods combined with efficient choices for trajectory propagation and conjunction tracking, the population of actively tracked objects was successfully modeled over 20 years. Thus, this model can serve to validate the well-established stochastic methods [7, 17, 16] and bears the promise to provide a deterministic model of the future debris environment. In comparison with previous studies [6], the number of observed conjunctions is slightly smaller at present. However, this finding will require additional studies to validate as there are several critical choices, such as estimates of objects' sizes, used propagator, and conjunction logic, which will need further study to ascertain reasonable choices. Only a simple MPI implementation for large-scale runs was demonstrated, however, this was already enough to calculate one timestep with over 600 000 particles in 0.21 seconds.
The code for all involved software components can be found online. [16] [17] [18]

4.2 Proposed Follow-up Studies

In the future, there are several natural continuations and extensions of this work. First, a larger scale, long-term run using the small debris population including breakups is indicated to investigate the possible impact on critical orbits through exponential effects such as Kessler syndrome [11]. This run should also incorporate better information on the sizes and masses of objects as the introduced κ parameter from this study highlights the critical nature of these data.
Secondly, additional accuracy should be obtainable by employing a higher-order interpolation during the conjunction tracking (see Section 2.2). Ideally, already computed positions and velocities at additional sub-timesteps from the integrator used for the propagation can serve to increase the interpolation order. A more sophisticated integrator targeted specifically at astrodynamics may improve results further [29, 12].

[16] https://github.com/esa/LADDS/ Accessed: 2022-06-18
[17] https://github.com/esa/NASA-breakup-model-cpp Accessed: 2022-06-18
[18] https://github.com/FG-TUM/OrbitPropagator/ Accessed: 2022-06-18

Furthermore, the current domain splitting technique in LADDS is suboptimal for the orbital regime as the rectangular splits introduce frequent transitions from one rank to another. Conceivably, moving the entire simulation to utilize a polar coordinate system where individual ranks then cover specific altitudes bears the promise to provide a better split.
Finally, preparations are underway to include planned upcoming mega-constellations in the simulation which can serve to study a more complex and realistic scenario [4]. Planned constellations already range in the tens of thousands of satellites with some of them (Starlink, OneWeb) having launched a considerable number of satellites already.

References

[1] Pablo Gómez, Fabio Gratl, Oliver Bösing, and Dario Izzo. Deterministic conjunction tracking in long-term space debris simulations. *arXiv preprint arXiv:2203.06957*, 2022.

[2] Jonas Schuhmacher. Efficient implementation and evaluation of the nasa breakup model in modern c++. 2021.

[3] Oliver Bösing. Efficient trajectory modelling for space debris evolution. 2021.

[4] Albert Noswitz. Modeling upcoming megaconstellations in space debris environment simulations. 2022.

[5] Marit Undseth, Claire Jolly, and Mattia Olivari. Space sustainability: The economics of space debris in perspective. 2020.

[6] Stijn Lemmens and Francesca Letizia. Esa's annual space environment report. Technical report, Technical Report GEN-DB-LOG-00288-OPS-SD, ESA Space Debris Office, 2021.

[7] J-C Liou, DT Hall, PH Krisko, and JN Opiela. Legend–a three-dimensional leo-to-geo debris evolutionary model. *Advances in Space Research*, 34(5):981–986, 2004.

[8] B Bastida Virgili. Delta debris environment long-term analysis. In *Proceedings of the 6th International Conference on Astrodynamics Tools and Techniques (ICATT)*, 2016.

[9] Dennis C Rapaport and Dennis C Rapaport Rapaport. *The art of molecular dynamics simulation*. Cambridge university press, 2004.

[10] Matthias Heinen, Jadran Vrabec, and Johann Fischer. Communication: Evaporation: Influence of heat transport in the liquid on the interface temperature and the particle flux. *The Journal of Chemical Physics*, 145(8):081101, 2016.

[11] Donald J Kessler and Burton G Cour-Palais. Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research: Space Physics*, 83(A6):2637–2646, 1978.

[12] Francesco Biscani and Dario Izzo. Revisiting high-order taylor methods for astrodynamics and celestial mechanics. *Monthly Notices of the Royal Astronomical Society*, 504(2):2614–2628, 2021.

[13] David A Vallado. *Fundamentals of astrodynamics and applications*, volume 12. Springer Science & Business Media, 2001.

[14] A.C. Long, R. Pajerski, R. Luczak, United States. National Aeronautics, Space Administration, and Goddard Space Flight Center. *Goddard Trajectory Determination System (GTDS): Mathematical Theory*. Goddard Space Flight Center, 1989.

[15] JM Picone, AE Hedin, D Pj Drob, and AC Aikin. Nrlmsise-00 empirical model of the atmosphere: Statistical comparisons and scientific issues. *Journal of Geophysical Research: Space Physics*, 107(A12):SIA–15, 2002.

[16] J-C Liou, DJ Kessler, M Matney, and G Stansbery. A new approach to evaluate collision probabilities among asteroids, comets, and kuiper belt objects. In *Lunar and Planetary Science Conference*, page 1828, 2003.

[17] Hugh G Lewis, Adam E White, Richard Crowther, and Hedley Stokes. Synergy of debris mitigation and removal. *Acta Astronautica*, 81(1):62–68, 2012.

[18] JR Alarcón Rodríguez, F Martínez Fadrique, and Heiner Klinkrad. Collision risk assessment with "smart sieve" method. In *Joint ESA-NASA Space-Flight Safety Conference*, volume 486, page 159, 2002.

[19] Inés Alonso Gómez, Sara Ansorena Vildarraz, Carlos García, JMH Garnica, C Perez Hernandez, MAR Prada, JU Carrazo, GM Pinna, S Moulin, P Besso, et al. Description of the architecture of the spanish space surveillance and tracking system. In *Proceedings of the 7th European Conference on Space Debris*, 2017.

[20] Nicholas L Johnson, Paula H Krisko, J-C Liou, and Phillip D Anz-Meador. Nasa's new breakup model of evolve 4.0. *Advances in Space Research*, 28(9):1377–1384, 2001.

[21] Fabio Alexander Gratl, Steffen Seckler, Hans-Joachim Bungartz, and Philipp Neumann. N ways to simulate short-range particle systems: Automated algorithm selection with the node-level library autopas. *Computer Physics Communications*, 273:108262, 2021.

[22] Haruo Yoshida. Construction of higher order symplectic integrators. *Physics letters A*, 150(5-7):262–268, 1990.

[23] Felix R Hoots, Paul W Schumacher Jr, and Robert A Glover. History of analytical orbit modeling in the us space surveillance system. *Journal of Guidance, Control, and Dynamics*, 27(2):174–185, 2004.

[24] Dario Izzo. Pygmo and pykep: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization). In *Proceedings of the fifth international conference on astrodynamics tools and techniques, ICATT*. sn, 2012.

[25] GE Cook. Satellite drag coefficients. *Planetary and Space Science*, 13(10):929–946, 1965.

[26] V Braun, A Horstmann, S Lemmens, C Wiedemann, and L Böttcher. Recent developments in space debris environment modelling, verification and validation with master. In *8th European Conference on Space Debris*, 2021.

[27] John Bacon and JC Liou. The 2019 us government orbital debris mitigation standard practices. 2020.

[28] K Merz, B Bastida Virgili, V Braun, T Flohrer, Q Funke, H Krag, S Lemmens, and J Siminski. Current collision avoidance service by ESA's space debris office. In *Proceedings 7th European Conference on Space Debris, Darmstadt, Germany*, pages 18–21, 2017.

[29] Sergio Blanes, Fernando Casas, Ariadna Farres, Jacques Laskar, Joseba Makazaga, and Ander Murua. New families of symplectic splitting methods for numerical integration in dynamical astronomy. *Applied Numerical Mathematics*, 68:58–72, 2013.