




# Brahe: A Modern Astrodynamics Library for Research and Engineering Applications

Duncan Eddy<sup>1</sup> and Mykel J. Kochenderfer<sup>1</sup>

<sup>1</sup> Stanford University

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

## Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright,  
and release the work under a  
Creative Commons Attribution 4.0  
International License ([CC BY 4.0](#)).

## Summary

`brahe` is a modern astrodynamics library for research and engineering applications. The representation and prediction of satellite motion is the fundamental problem of astrodynamics, with initial formulations of equations of motion dating back to Kepler (1619) and Newton (1687). Current research and applications in space situational awareness, satellite task planning, and space mission operations require accurate and efficient numerical tools to perform coordinate transformations, model perturbations, and propagate orbits. `brahe` incorporates the latest conventions and models for time systems and reference frame transformations from the International Astronomical Union (IAU) (Hohenkerk, 2017) and International Earth Rotation and Reference Systems Service (IERS) (Petit & Luzum, 2010). It implements force models for Earth-orbiting satellites including atmospheric drag, solar radiation pressure, and third-body perturbations from the Sun and Moon (Montenbruck & Gill, 2000; D. A. Vallado, 2001), standard orbit propagation algorithms including the Simplified General Perturbations (SGP) Model (D. Vallado et al., 2006), and recent algorithms for fast, parallelized computation of ground station and imaging-target visibility (Eddy & Kochenderfer, 2021).

With `brahe`, predicting upcoming satellite passes over ground stations or imaging targets can be quickly accomplished in three lines of code:

```
import brahe as bh
bh.initialize_eop()
passes = bh.location_accesses(
    bh.PointLocation(-122.4194, 37.7749, 0.0), # San Francisco
    bh.CelestrakClient().get_sgp_propagator(catnr=25544, step_size=60.0), #
    ISS
    bh.Epoch.now(),
    bh.Epoch.now() + 24 * 3600.0, # Next 24 hours
    bh.ElevationConstraint(min_elevation_deg=10.0)
)
```

`brahe` allows users to quickly retrieve satellite ephemeris data from Space-Track (18th Space Defense Squadron, 2026) or Celestrak (Kelso, T. S., 2025) and propagate orbits using different dynamics models. This can be used for space situational awareness tasks such as predicting the orbits of all Starlink satellites over the next 24 hours:

```
import brahe as bh
bh.initialize_eop()
gp_records = bh.CelestrakClient().get_gp(group="starlink")
starlink = [rec.to_sgp_propagator(step_size=60.0) for rec in gp_records]
bh.par_propagate_to(starlink, bh.Epoch.now() + 86400.0) # Predict next 24 hours
```

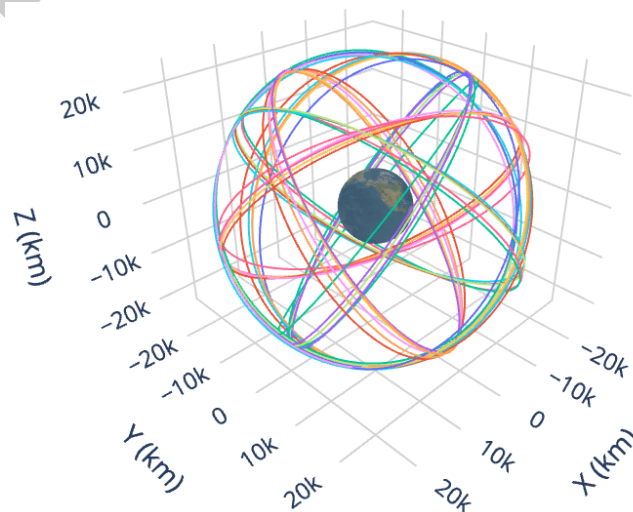
The above routine can propagate orbits for all ~9000 Starlink satellites in approximately 1 minute 30 seconds on an M1 Max MacBook Pro with 10 cores and 64 GB RAM. Finally, the package provides direct, easy-to-use functions for low-level astrodynamics routines such as Keplerian to Cartesian state conversions and reference frame transformations.

```

53
54
55 import brahe as bh
56 import numpy as np
57
58 # Initialize Earth Orientation Parameter data
59 bh.initialize_eop()
60
61 # Define orbital elements
62 a = bh.constants.R_EARTH + 700e3 # Semi-major axis in meters (700 km altitude)
63 e = 0.001 # Eccentricity
64 i = 98.7 # Inclination in radians
65 raan = 15.0 # Right Ascension of Ascending Node in radians
66 arg_periapsis = 30.0 # Argument of Periapsis in radians
67 mean_anomaly = 45.0 # Mean Anomaly
68 state_kep = np.array([a, e, i, raan, arg_periapsis, mean_anomaly])
69
70 # Convert Keplerian state to ECI coordinates
71 state_eci = bh.state_koe_to_eci(state_kep, bh.AngleFormat.DEGREES)
72
73 # Define a time epoch
74 epoch = bh.Epoch(2024, 6, 1, 12, 0, 0.0, time_system=bh.TimeSystem.UTC)
75
76 # Convert ECI coordinates to ECEF coordinates at the given epoch
77 state_ecef = bh.state_eci_to_ecef(epoch, state_eci)
78
79 # Convert back from ECEF to ECI coordinates
80 state_eci_2 = bh.state_ecef_to_eci(epoch, state_ecef)
81
82 # Convert back from ECI to Keplerian elements
83 state_kep_2 = bh.state_eci_to_koe(state_eci_2, bh.AngleFormat.DEGREES)
84
85

```

Another example application of brahe is predicting and visualizing GPS satellite orbits. The package provides built-in functions for generating 2D and 3D visualizations of satellite constellations using Plotly (Plotly Technologies Inc., 2015) and matplotlib (Hunter, 2007).



**Figure 1:** Visualization of all GPS Satellite Orbits

## Statement of Need

While the core algorithms for predicting and modeling satellite motion have been known for decades, there is a lack of modern, open-source software that implements these algorithms in a way that is accessible to researchers and engineers. Generally, existing astrodynamics software packages have one or more barriers to entry, which leads to duplicated effort as researchers frequently choose to re-implement and validate common algorithms to avoid issue with other software. Flagship commercial astrodynamics software like Systems Tool Kit (STK) (Analytic Graphics, 2023) and FreeFlyer (a.i. Solutions, Inc., 2025) are individually licensed and closed-source. The licensing costs can be prohibitive for researchers, individuals, small organizations, and start-ups. Even for larger organizations, the per-node licensing cost can make large-scale deployment prohibitive. Their closed-source nature makes it difficult to understand and verify the exact algorithms and model implementations, which is critical for high-stakes applications like space mission operations (Mars Climate Orbiter Mishap Investigation Board, 1999). brahe targets students, researchers, engineers, and organizations who need a well-documented, easily-installed astrodynamics library that integrates with the Python scientific ecosystem and provides transparent, verifiable implementations of standard algorithms.

## State of the Field

Major open-source projects like Orekit (Maisonobe et al., 2010) and GMAT (Hughes et al., 2014) provide extensive functionality, but are large codebases with steep learning curves, making quick-adoption and integration into projects difficult. Furthermore, Orekit is implemented in Java, which adds friction when integrating with the current, Python-focused, scientific software ecosystem. Libraries such as poliastro (Cano Rodriguez & Martínez Garrido, 2022) and Open Space Toolkit (OSTk) (Open Space Collective, 2025) provides Python interfaces, but their object-oriented architecture adds layers of abstraction that can make it difficult to adapt them to problems that outside their supported problem-domains. Additionally, poliastro is no longer actively maintained and OSTk requires a Linux environment or specialized Docker container to run, limiting integration with other applications. Academic tools like Basilisk (Kenneally et al., 2020), provide high-fidelity modeling capabilities for full spacecraft guidance, navigation, and control (GNC) simulations, but are primarily focused on spacecraft-system modeling.

We evaluated contributing to these existing libraries but concluded that their architectural choices—deep class hierarchies, framework-specific abstractions, or platform constraints—introduced too many layers of complexity for users to easily understand, validate, extend, or adopt when seeking solutions to simple, fundamental astrodynamics problems. brahe was built from the ground up to provide a composable, function-oriented alternative distributed through standard channels (crates.io and PyPI) with full cross-platform support.

## Software Design

brahe addresses these challenges by providing a modern, open-source astrodynamics library following design principles of the *Zen of Python* (Peters, 2004). We designed brahe to be modular, yet highly-composable—emphasizing simple functions that can be chained together to build more complex functionality. To address decision-fatigue from astrodynamics modeling complexity without burdening new users, we adopt a design philosophy of “Do the Rightest Thing” philosophy: provide colloquially-named functions with reasonable default for modeling decisions (e.g., time systems, reference frames, perturbation models) that are current and accurate, but allow users to swap-out default implementations for specific models when needed. This lets beginners get reasonably accurate results quickly while preserving full control for advanced users.

The core is implemented in Rust for performance and memory safety, with Python bindings via

PyO3 for integration with the scientific Python ecosystem. Documentation follows the Diátaxis framework (Procida, 2024), with every function documented with API reference entries and usage examples. In addition, there is a conceptual user guide and long-form tutorials. All documentation code blocks are automatically tested on every commit to ensure they remain accurate and up-to-date. The library has a comprehensive test suite for both Rust and Python, and automated CI/CD for testing and distribution.

## Research Impact Statement

brahe has been used by current satellite missions after its adoption by aerospace companies such as Capella Space, Northwood Space, Xona Space (Reid et al., 2020), and Kongsberg Satellite Services. Most notably, the satellite imaging prediction and task planning algorithms in brahe were used operationally by Capella Space (Stringham et al., 2019) in the first private US synthetic aperture radar (SAR) satellite constellation to predict communications and imaging opportunities.

The library has been used in scientific publications on satellite scheduling optimization (Eddy et al., 2025) and scalable satellite constellation management (Kim et al., 2025). It is provided under an MIT License to facilitate adoption and integration.

## AI Usage Disclosure

brahe development dates back to 2014, predating the availability of generative AI tools. Core architecture, modules, and implementations were developed without their use. Recently, AI tools have been used to unblock development of a new submodules and features (e.g., trajectory data structures, advanced numerical propagators, ephemeris client moduels) and to improve test coverage and documentation. In all cases, generated code was carefully reviewed, tested, and modified by the authors prior to merging to ensure correctness and maintainability. It was also verified against reference tests or values whenever possible. AI tools were not used in the writing of original submission, but were used to assist in reformatting to fit the new JOSS template.

## Acknowledgments

We want to acknowledge Shaurya Luthra, Adrien Perkins, and Arthur Kvalheim Merlin for supporting the adoption of the project in their organizations and providing valuable feedback. We thank the Stanford Institute for Human-Centered AI for funding in part this work.

## References

- 18th Space Defense Squadron. (2026). *Space-track.org*. United States Space Command. <https://space-track.org/>
- a.i. Solutions, Inc. (2025). *FreeFlyer: Spacecraft Mission Analysis and Operations Software* (Version 7.10). <https://www.ai-solutions.com/freelyer>
- Analytic Graphics, Inc. (AGI). (2023). *Systems Tool Kit (STK)* (Version 12.7). <https://www.agi.com/products/stk>
- Cano Rodriguez, J. L., & Martínez Garrido, J. (2022). *poliastro* (Version v0.17.0). <https://github.com/poliastro/poliastro/>
- Eddy, D., Ho, M., & Kochenderfer, M. J. (2025). Optimal Ground Station Selection for Low-Earth Orbiting Satellites. *IEEE Aerospace Conference*. <https://arxiv.org/abs/2410.16282>

- 177 Eddy, D., & Kochenderfer, M. J. (2021). A Maximum Independent Set Method for Scheduling  
178 Earth-Observing Satellite Constellations. *Journal of Spacecraft and Rockets*, 58(5),  
179 1416–1429.
- 180 Hohenkerk, C. (2017). IAU Standards of Fundamental Astronomy (SOFA): Time and Date.  
181 In *The Science of Time 2016: Time in Astronomy & Society, Past, Present and Future*.  
182 Springer.
- 183 Hughes, S. P., Qureshi, R. H., Cooley, S. D., & Parker, J. J. (2014). Verification and  
184 Validation of the General Mission Analysis Tool (GMAT). *AIAA/AAS Astrodynamics*  
185 *Specialist Conference*.
- 186 Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science &*  
187 *Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 188 Kelso, T. S. (2025). *Celestrak Active Satellite Database*. <https://celestrak.com/>
- 189 Kenneally, P. W., Piggott, S., & Schaub, H. (2020). Basilisk: A Flexible, Scalable and Modular  
190 Astrodynamics Simulation Framework. *Journal of Aerospace Information Systems*, 17(9),  
191 496–507. <https://doi.org/10.2514/1.I010762>
- 192 Kepler, J. (1619). *Epitome Astronomiae Copernicanae*.
- 193 Kim, G. R., Eddy, D., Srinivas, V., & Kochenderfer, M. J. (2025). Scalable Ground Station  
194 Selection for Large LEO Constellations. *arXiv Preprint arXiv:2510.03438*. <https://arxiv.org/abs/2510.03438>
- 196 Maisonobe, L., Pommier, V., & Parraud, P. (2010). Orekit: An Open Source Library for  
197 Operational Flight Dynamics Applications. *International Conference on Astrodynamics*  
198 *Tools and Techniques*.
- 199 Mars Climate Orbiter Mishap Investigation Board. (1999). *Mars Climate Orbiter Mishap*  
200 *Investigation Board Phase I Report* [Tech. Report]. Jet Propulsion Laboratory / National  
201 Aeronautics and Space Administration. [https://llis.nasa.gov/llis\\_lib/pdf/1009464main1\\_](https://llis.nasa.gov/llis_lib/pdf/1009464main1_0641-mr.pdf)  
202 [0641-mr.pdf](https://llis.nasa.gov/llis_lib/pdf/1009464main1_0641-mr.pdf)
- 203 Montenbruck, O., & Gill, E. (2000). *Satellite Orbits: Models, Methods and Applications*.  
204 Springer.
- 205 Newton, I. (1687). *Philosophiae Naturalis Principia Mathematica*.
- 206 Open Space Collective. (2025). *Open Space Toolkit*. [https://github.com/open-space-](https://github.com/open-space-collective/open-space-toolkit)  
207 [collective/open-space-toolkit](https://github.com/open-space-collective/open-space-toolkit)
- 208 Peters, T. (2004). The Zen of Python. In *Pro Python*. Springer.
- 209 Petit, G., & Luzum, B. (2010). *IERS Conventions, Technical Note 36*.
- 210 Plotly Technologies Inc. (2015). *Collaborative Data Science*. Plotly Technologies Inc.  
211 <https://plot.ly>
- 212 Procida, D. (2024). *Diátaxis: A Systematic Approach to Technical Documentation Authoring*.  
213 <https://diataxis.fr/>.
- 214 Reid, T. G., Chan, B., Goel, A., Gunning, K., Manning, B., Martin, J., Neish, A., Perkins, A.,  
215 & Tarantino, P. (2020). Satellite Navigation for the Age of Autonomy. *2020 IEEE/ION*  
216 *Position, Location and Navigation Symposium (PLANS)*.
- 217 Stringham, C., Farquharson, G., Castelletti, D., Quist, E., Riggi, L., Eddy, D., & Soenen, S.  
218 (2019). The Capella X-band SAR Constellation for Rapid Imaging. *IEEE International*  
219 *Geoscience and Remote Sensing Symposium*.
- 220 Vallado, D. A. (2001). *Fundamentals of Astrodynamics and Applications*.
- 221 Vallado, D., Crawford, P., Hujsak, R., & Kelso, T. S. (2006). Revisiting Spacetrack Report

DRAFT