The Journal of Open Source Software

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

**Duncan Eddy** [1] and **Mykel J. Kochenderfer** [1]

**1** Stanford University

**DOI:** 10.xxxxxx/draft

**Software**

- Review
- Repository
- Archive

**Editor:** Open Journals

**Reviewers:**

- @openjournals

**Submitted:** 01 January 1970
**Published:** unpublished

## Summary

brahe is a modern astrodynamics dynamics library for research and engineering applications. The representation and prediction of satellite motion is the fundamental problem of astrodynamics. The motion of celestial bodies has been studied for centuries with initial 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). It also provides standard orbit propagation algorithms, including the Simplified General Perturbations (SGP) Model (D. Vallado et al., 2006). Finally, it implements recent algorithms for fast, parallelized computation of ground station and imaging-target visibility (Eddy & Kochenderfer, 2021), a foundational problem in satellite scheduling and mission planning.

With `brahe`, predicing upcoming satellite passes over ground stations or imaging targets can be accomplished in seconds and three lines of code.

```python
import brahe as bh
bh.initialize_eop()
passes = bh.location_accesses(
    bh.PointLocation(-122.4194, 37.7749, 0.0),  # San Francisco
    bh.celestrak.get_tle_by_id_as_propagator(25544, 60.0, "active"),  # 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 access Two-Line Element (TLE) data from Celestrak (Kelso, T. S., 2025) and propagate orbits using the SGP4 dynamics model. This can be used to perform space situational awareness tasks such as predicting the orbits of all Starlink satellites over the next 24 hours.

```python
import brahe as bh
bh.initialize_eop()
starlink = bh.datasets.celestrak.get_tles_as_propagators("starlink", 60.0)
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.

```python
import brahe as bh
import numpy as np

# Initialize Earth Orientation Parameter data
bh.initialize_eop()

# Define orbital elements
a = bh.constants.R_EARTH + 700e3  # Semi-major axis in meters (700 km altitude)
e = 0.001                         # Eccentricity
i = 98.7                          # Inclination in radians
raan = 15.0                       # Right Ascension of Ascending Node in radians
arg_periapsis = 30.0              # Argument of Periapsis in radians
mean_anomaly = 45.0               # Mean Anomaly
state_kep = np.array([a, e, i, raan, arg_periapsis, mean_anomaly])

# Convert Keplerian state to ECI coordinates
state_eci = bh.state_koe_to_eci(state_kep, bh.AngleFormat.DEGREES)

# Define a time epoch
epoch = bh.Epoch(2024, 6, 1, 12, 0, 0.0, time_system=bh.TimeSystem.UTC)

# Convert ECI coordinates to ECEF coordinates at the given epoch
state_ecef = bh.state_eci_to_ecef(epoch, state_eci)

# Convert back from ECEF to ECI coordinates
state_eci_2 = bh.state_ecef_to_eci(epoch, state_ecef)

# Convert back from ECI to Keplerian elements
state_kep_2 = bh.state_eci_to_koe(state_eci_2, bh.AngleFormat.DEGREES)
```

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).

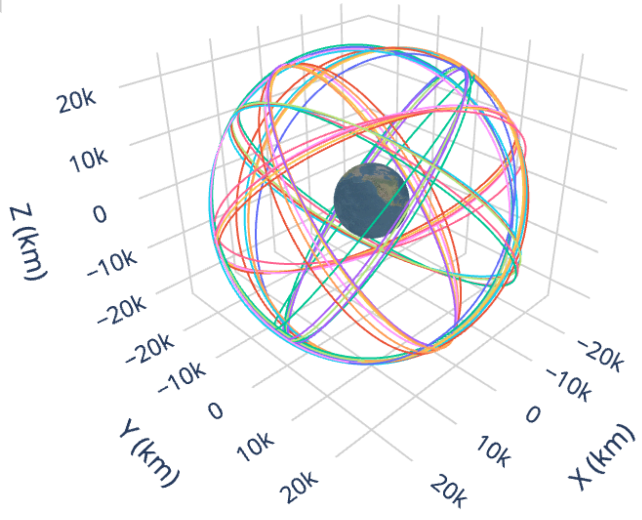

**Figure 1:** Visualization of all GPS Satellite Orbits

## 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 for individuals and organizations looking to develop astrodynamics applications, and often leads to duplicated and redundant effort as researchers and engineers are forced to re-implement foundational algorithms.

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. The closed-source nature of these packages 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). 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 can be a barrier to adoption in the current scientific ecosystem with users who are more familiar with Python. GMAT uses a domain-specific scripting language and has limited documentation and examples, making it difficult for new users to get started. 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 predefined modeling frameworks. Additionally, poliastro is no longer actively maintained and OSTk only supports Linux environments and requires a specialized Docker environment to run. Other academic tools like Basilisk (Kenneally et al., 2020), provide high-fidelity modeling capabilities for full spacecraft guidance, navigation, and control (GNC) simulations, but are not directly distributed through standard package managers like PyPI and must be compiled from source to be used. Finally, these works often have limited documentation and usage examples, making it difficult for new users to get started.

## 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 evaluated contributing to existing open-source astrodynamics libraries, but ultimately decided that the implementation choices and architecture of existing libraries made introduced too many layers of abstraction and complexity to make it easy for new users to understand, validate, or extend the core algorithms. Therefore we designed `brahe` from the ground-up to be modular, yet highly-composable that emphasizes simple functions that can be easily chained together to build more complex functionality. Given the breadth and complexity of astrodynamics modeling choices and, we adopt a design philosophy of "Do the Rightest Thing"—always provide a reasonable default choice for modeling decisions (e.g., time systems, reference frames, perturbation models) that is modern, current, and accurate, but allow users to override or extend these choices when needed. This allows new users to get started quickly without needing to understand all the nuances of astrodynamics modeling, while still both ensuring the correctness of their results and allowing advanced users to customize the library to their specific needs.

The core functionality is implemented in Rust for performance and safety, with Python bindings for ease-of-use and integration with the scientific Python ecosystem. `brahe` is provided under an MIT License to encourage adoption and facilitate integration and extensibility. To further promote adoption and aid user learning, the library is extensively documented following the Diátaxis framework (Procida, 2024)—every Rust and Python function documented with types and usage examples, there is a user guide that explains the major concepts of the library, and set of longer-form examples demonstrating how to accomplish common tasks. To maintain high code quality, the library has a comprehensive test suite for both Rust and Python. Additionally, all code samples in the documentation are automatically tested on commit to ensure they remain functional, and that the documentation accurately reflects the library's capabilities. The package testing and distribution is fully automated using GitHub Actions to run tests on multiple platforms (Linux, macOS, Windows) and Python versions (3.10+) on every code change.

## 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. They have used it for mission analysis and planning, and in some cases supporting on-orbit mission operations. One particularly significant contribution is that satellite imaging prediction and task planning algorithms of Brahe were used by the first US synthetic aperture radar (SAR) satellite constellation, operated by Capella Space (Stringham et al., 2019) to predict communications and imaging opportunities. It has also been used in a number of scientific publications (Eddy et al., 2025; Kim et al., 2025). It has a small, but growing, user base in both academia and industry.

## AI Usage Disclosure

The core development of brahe did not involve the use of AI tools, and in-fact, predated them. However, AI tools were used to unblock continued development in 2025 due to a breaking change in how PyO3 (the Rust-to-Python bindings library) API worked. Specifically, Claude was used to help understand the new API and refactor existing code to be compatible with the new version. Additionally, Github Copilot and Claude code were used to help implement a few long-outstanding features, specifically, the addition of trajectory data structures, numerical orbit propagators, and space weather data ingestion. AI tools have also been used to help improve test coverage and documentation. In all cases, the generated code was carefully reviewed, tested, and modified by the authors prior to merging to ensure correctness and maintainability. AI tools were not used in the writing of this paper.

## Acknowledgments

We also 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. Finally, we would like to thank the Stanford Institute for Human-Centered AI for funding in part this work.

## References

a.i. Solutions, Inc. (2025). *FreeFlyer: Spacecraft Mission Analysis and Operations Software* (Version 7.10). https://www.ai-solutions.com/freeflyer

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

Eddy, D., & Kochenderfer, M. J. (2021). A Maximum Independent Set Method for Scheduling Earth-Observing Satellite Constellations. *Journal of Spacecraft and Rockets*, *58*(5), 1416–1429.

Hohenkerk, C. (2017). IAU Standards of Fundamental Astronomy (SOFA): Time and Date. In *The Science of Time 2016: Time in Astronomy & Society, Past, Present and Future*. Springer.

Hughes, S. P., Qureshi, R. H., Cooley, S. D., & Parker, J. J. (2014). Verification and Validation of the General Mission Analysis Tool (GMAT). *AIAA/AAS Astrodynamics Specialist Conference*.

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Kelso, T. S. (2025). *Celestrak Active Satellite Database*. https://celestrak.com/

Kenneally, P. W., Piggott, S., & Schaub, H. (2020). Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework. *Journal of Aerospace Information Systems*, *17*(9), 496–507. https://doi.org/10.2514/1.I010762

Kepler, J. (1619). *Epitome Astronomiae Copernicanae*.

Kim, G. R., Eddy, D., Srinivas, V., & Kochenderfer, M. J. (2025). Scalable Ground Station Selection for Large LEO Constellations. *arXiv Preprint arXiv:2510.03438*. https://arxiv.org/abs/2510.03438

Maisonobe, L., Pommier, V., & Parraud, P. (2010). Orekit: An Open Source Library for Operational Flight Dynamics Applications. *International Conference on Astrodynamics Tools and Techniques*.

Mars Climate Orbiter Mishap Investigation Board. (1999). *Mars Climate Orbiter Mishap Investigation Board Phase I Report* [Tech. Report]. Jet Propulsion Laboratory / National Aeronautics and Space Administration. https://llis.nasa.gov/llis_lib/pdf/1009464main1_0641-mr.pdf

Montenbruck, O., & Gill, E. (2000). *Satellite Orbits: Models, Methods and Applications*. Springer.

Newton, I. (1687). *Philosophiae Naturalis Principia Mathematica*.

Open Space Collective. (2025). *Open Space Toolkit*. https://github.com/open-space-collective/open-space-toolkit

Peters, T. (2004). The Zen of Python. In *Pro Python*. Springer.

Petit, G., & Luzum, B. (2010). *IERS Conventions, Technical Note 36*.

Plotly Technologies Inc. (2015). *Collaborative Data Science*. Plotly Technologies Inc. https://plot.ly

Procida, D. (2024). *Diátaxis: A Systematic Approach to Technical Documentation Authoring*. https://diataxis.fr/.

Reid, T. G., Chan, B., Goel, A., Gunning, K., Manning, B., Martin, J., Neish, A., Perkins, A., & Tarantino, P. (2020). Satellite Navigation for the Age of Autonomy. *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*.

Stringham, C., Farquharson, G., Castelletti, D., Quist, E., Riggi, L., Eddy, D., & Soenen, S. (2019). The Capella X-band SAR Constellation for Rapid Imaging. *IEEE International Geoscience and Remote Sensing Symposium*.

Vallado, D. A. (2001). *Fundamentals of Astrodynamics and Applications*.

Vallado, D., Crawford, P., Hujsak, R., & Kelso, T. S. (2006). Revisiting Spacetrack Report

#3. *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*.