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

Duncan Eddy [1]

1 Stanford University

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 (Kepler, 1619) and Newton (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). 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, predicting 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)

for sat in starlink:
    sat.propagate_to(sat.epoch + 86400.0)  # Propagate one orbit (24 hours)
```

The above routine can propagate orbits for all ~9000 Starlink satellites in approximately 5 minutes 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

# Create a state vector from orbital elements
state_kep = np.array([a, e, i, raan, arg_periapsis, mean_anomaly])

# Convert Keplerian state to ECI coordinates
state_eci = bh.state_osculating_to_cartesian(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_cartesian_to_osculating(state_eci_2, bh.AngleFormat.DEGREES)
```

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, creating two major drawbacks. First, the licensing costs can be prohibitive for researchers, individuals, small organizations, and start-ups. Even for larger organizations, per-node licensing cost can make large-scale deployment prohibitive. Second, 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 (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. Finally, there are academic libraries such as poliastro (Cano Rodriguez & Martínez Garrido, 2022) which are not actively maintained. Other 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, academic work often has limited documentation and usage examples, making it difficult for new users to get started.

brahe seeks to address these challenges by providing a modern, open-source astrodynamics library following design principles of the Zen of Python (Peters, 2004). 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 permissively licensed under an MIT License to encourage adoption and enable individuals and organizations to integrate and extend the software without worrying about licensing or cost restrictions. To further promote adoption, the library is extensively documented following the Diátaxis framework (Procida, 2024) to aid learning—every Rust and Python function documented with arguments, returns, and usage examples, there is a user guide that explains the major concepts of the library, and there is a set of longer-form examples that demonstrate how to accomplish common tasks. To maintain high code quality, the library has a comprehensive test suite for both Rust and Python code. Additionally, all code samples in the documentation are automatically tested to ensure they remain up-to-date and functional, and that the documentation accurately reflects the library's capabilities.

brahe has already been used in a number of scientific publications Kim et al. (2025). It has also been used by aerospace companies such as Northwood Space, Xona Space (Reid et al., 2020), and Kongsberg Satellite Services for mission analysis and planning. The Earth Observation satellite imaging prediction and task planning algorithms have been used by Capella Space and demonstrated on-orbit with their synthetic aperture radar (SAR) constellation (Stringham et al., 2019).

Acknowledgements

References

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

88 Analytic Graphics, Inc. (AGI). (2023). *Systems Tool Kit (STK)* (Version 12.7). https:
89 //www.agi.com/products/stk

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

93 Cano Rodriguez, J. L., & Martínez Garrido, J. (2022). *poliastro* (Version v0.17.0). https:
94 //github.com/poliastro/poliastro/

95 Eddy, D., Ho, M., & Kochenderfer, M. J. (2025). Optimal Ground Station Selection for Low-
96 Earth Orbiting Satellites. *IEEE Aerospace Conference*. https://arxiv.org/abs/2410.16282

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

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

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

105 Kelso, T. S. (2025). *Celestrak Active Satellite Database*.

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

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

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

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

116 Montenbruck, O., & Gill, E. (2000). *Satellite Orbits: Models, Methods and Applications*.
117 Springer. ISBN: 978-3-540-67280-7

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

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

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

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

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

126 Stringham, C., Farquharson, G., Castelletti, D., Quist, E., Riggi, L., Eddy, D., & Soenen, S.
127 (2019). The Capella X-band SAR Constellation for Rapid Imaging. *IEEE International*
128 *Geoscience and Remote Sensing Symposium*.

129 Vallado, D. A. (2001). *Fundamentals of Astrodynamics and Applications*. Springer Science &
130 Business Media.

131 Vallado, D., Crawford, P., Hujsak, R., & Kelso, T. S. (2006). Revisiting Spacetrack Report
132 #3. *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*.