

¹ Brahe: A Modern Astrodynamics Dynamics Library for Research and Engineering Applications

³ Duncan Eddy  ¹ and Mykel J. Kochenderfer  ¹

⁴ 1 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Published: unpublished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](#)).
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.

⁵ Summary

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](#); [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, predicting upcoming satellite passes over ground stations or imaging targets can be accomplished in seconds and three lines of code.

```
24
25 import brahe as bh
26 bh.initialize_eop()
27 passes = bh.location_accesses(
28     bh.PointLocation(-122.4194, 37.7749, 0.0), # San Francisco
29     bh.celestrak.get_tle_by_id_as_propagator(25544, 60.0, "active"), # ISS
30     bh.Epoch.now(),
31     bh.Epoch.now() + 24 * 3600.0, # Next 24 hours
32     bh.ElevationConstraint(min_elevation_deg=10.0)
33 )
34
35
36
```

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.

```
41
42 import brahe as bh
43 bh.initialize_eop()
44 starlink = bh.datasets.celestrak.get_tles_as_propagators("starlink", 60.0)
45 bh.par_propagate_to(starlink, bh.Epoch.now() + 86400.0) # Predict next 24 hours
46
47
48
```

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

```

53
54 import brahe as bh
55 import numpy as np
56
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_osculating_to_cartesian(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_cartesian_to_osculating(state_eci_2, bh.AngleFormat.DEGREES)
84
85
86 Another example application of brahe is visualizing the positions of GPS satellites in Earth
87 orbit. The package provides built-in functions for generating 3D visualizations of satellite
88 constellations using Plotly (Inc., 2015).
```

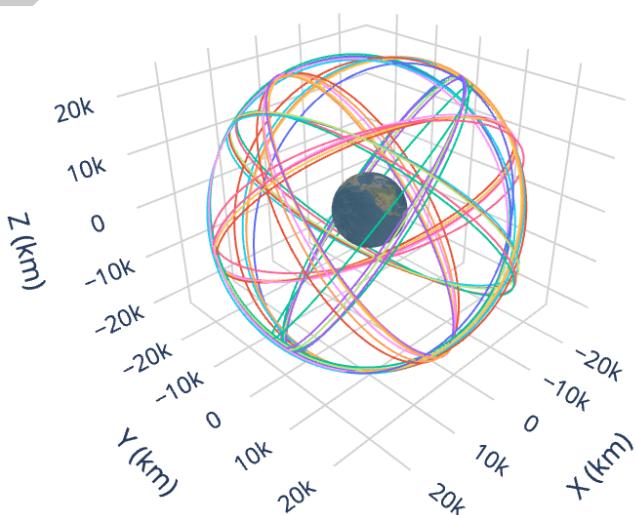


Figure 1: Visualization of all GPS Satellite Orbits

89 Statement of Need

90 While the core algorithms for predicting and modeling satellite motion have been known for
91 decades, there is a lack of modern, open-source software that implements these algorithms
92 in a way that is accessible to researchers and engineers. Generally, existing astrodynamics
93 software packages have one or more barriers to entry for individuals and organizations looking
94 to develop astrodynamics applications, and often leads to duplicated and redundant effort as
95 researchers and engineers are forced to re-implement foundational algorithms.

96 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
97 licensing costs can be prohibitive for researchers, individuals, small organizations, and start-ups.
98 Even for larger organizations, the per-node licensing cost can make large-scale deployment
99 prohibitive. The closed-source nature of these packages makes it difficult to understand
100 and verify the exact algorithms and model implementations, which is critical for high-stakes
101 applications like space mission operations ([Board, 1999](#)). Major open-source projects like Orekit
102 ([Maisonne et al., 2010](#)) and GMAT ([Hughes et al., 2014](#)) provide extensive functionality,
103 but are large codebases with steep learning curves, making quick-adoption and integration
104 into projects difficult. Furthermore, Orekit is implemented in Java, which can be a barrier to
105 adoption in the current scientific ecosystem with users who are more familiar with Python.
106 GMAT uses a domain-specific scripting language and has limited documentation and examples,
107 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](#))
108 provides Python interfaces, but their object-oriented architecture adds layers of abstraction
109 that can make it difficult to adapt them to problems that outside their predefined modeling
110 frameworks. Additionally, poliastro is no longer actively maintained and OSTk only supports
111 Linux environments and requires a specialized Docker environment to run. Other academic
112 tools like Basilisk ([Kenneally et al., 2020](#)), provide high-fidelity modeling capabilities for full
113 spacecraft guidance, navigation, and control (GNC) simulations, but are not directly distributed
114 through standard package managers like PyPI and must be compiled from source to be used.
115 Finally, these works often have limited documentation and usage examples, making it difficult
116 for new users to get started.

117 brahe seeks to address these challenges by providing a modern, open-source astrodynamics
118 library following design principles of the *Zen of Python* ([Peters, 2004](#)). The core functionality
119 is implemented in Rust for performance and safety, with Python bindings for ease-of-use and
120 integration with the scientific Python ecosystem. brahe is provided under an MIT License to
121 encourage adoption and facilitate integration and extensibility. To further promote adoption
122 and aid user learning, the library is extensively documented following the Diátaxis framework
123 ([Procula, 2024](#))—every Rust and Python function documented with types and usage examples,
124 there is a user guide that explains the major concepts of the library, and set of longer-form
125 examples demonstrating how to accomplish common tasks. To maintain high code quality,
126 the library has a comprehensive test suite for both Rust and Python. Additionally, all code
127 samples in the documentation are automatically tested to ensure they remain functional, and
128 that the documentation accurately reflects the library's capabilities.

129 brahe has already been used in a number of scientific publications ([Eddy et al., 2025; 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.
130 The Earth Observation satellite imaging prediction and task planning algorithms have been
131 used by Capella Space and demonstrated on-orbit with their synthetic aperture radar (SAR)
132 constellation ([Stringham et al., 2019](#)).

Acknowledgments

¹³⁷ We also want to acknowledge Shaurya Luthra, Adrien Perkins, and Arthur Kvalheim Merlin for
¹³⁹ supporting the adoption of the project in their organizations and providing valuable feedback.
¹⁴⁰ Finally, we would like to thank the Stanford Institute for Human-Centered AI for funding in
¹⁴¹ part this work.

References

- ¹⁴² a.i. Solutions, Inc. (2025). *FreeFlyer: Spacecraft Mission Analysis and Operations Software* (Version 7.10). <https://www.ai-solutions.com/freeflyer>
- ¹⁴³ Analytic Graphics, Inc. (AGI). (2023). *Systems Tool Kit (STK)* (Version 12.7). <https://www.agi.com/products/stk>
- ¹⁴⁴ Board, M. C. O. M. I. (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
- ¹⁴⁵ Cano Rodriguez, J. L., & Martínez Garrido, J. (2022). *poliastro* (Version v0.17.0). <https://github.com/poliastro/poliastro/>
- ¹⁴⁶ 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>
- ¹⁴⁷ 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.
- ¹⁴⁸ 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.
- ¹⁴⁹ 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*.
- ¹⁵⁰ Inc., P. T. (2015). *Collaborative data science*. Plotly Technologies Inc. <https://plot.ly>
- ¹⁵¹ Kelso, T. S. (2025). *Celestrak Active Satellite Database*. <https://celestrak.com/>
- ¹⁵² 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>
- ¹⁵³ Kepler, J. (1619). *Epitome Astronomiae Copernicanae*.
- ¹⁵⁴ 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>
- ¹⁵⁵ Maisonobe, L., Pommier, V., & Parraud, P. (2010). Orekit: An Open Source Library for Operational Flight Dynamics Applications. *International Conference on Astrodynamics Tools and Techniques*.
- ¹⁵⁶ Montenbruck, O., & Gill, E. (2000). *Satellite Orbits: Models, Methods and Applications*. Springer.
- ¹⁵⁷ Newton, I. (1687). *Philosophiae Naturalis Principia Mathematica*.
- ¹⁵⁸ Open Space Collective. (2025). *Open Space Toolkit*. <https://github.com/open-space-collective/open-space-toolkit>

- ¹⁸⁰ Peters, T. (2004). The Zen of Python. In *Pro Python*. Springer.
- ¹⁸¹ Petit, G., & Luzum, B. (2010). *IERS Conventions, Technical Note 36*.
- ¹⁸² Procida, D. (2024). *Diátaxis: A Systematic Approach to Technical Documentation Authoring*.
- ¹⁸³ <https://diataxis.fr/>.
- ¹⁸⁴ 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)*.
- ¹⁸⁵ 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*.
- ¹⁸⁶ Vallado, D. A. (2001). *Fundamentals of Astrodynamics and Applications*.
- ¹⁸⁷ Vallado, D., Crawford, P., Hujasak, R., & Kelso, T. S. (2006). Revisiting Spacetrack Report #3. *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*.
- ¹⁸⁸
- ¹⁸⁹
- ¹⁹⁰
- ¹⁹¹
- ¹⁹²