

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Published: unpublished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](#)).

⁵ Summary

⁶ `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. ⁷ The motion of celestial bodies has been studied for centuries with initial equations of motion ⁸ dating back to Kepler (1619) and Newton (1687). Current research and applications in space ⁹ situational awareness, satellite task planning, and space mission operations require accurate ¹⁰ and efficient numerical tools to perform coordinate transformations, model perturbations, and ¹¹ propagate orbits. `brahe` incorporates the latest conventions and models for time systems and ¹² reference frame transformations from the International Astronomical Union (IAU) (Hohenkerk, ¹³ 2017) and International Earth Rotation and Reference Systems Service (IERS) (Petit & Luzum, ¹⁴ 2010). It implements force models for Earth-orbiting satellites including atmospheric drag, ¹⁵ 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, ¹⁶ including the Simplified General Perturbations (SGP) Model (D. Vallado et al., 2006). Finally, ¹⁷ it implements recent algorithms for fast, parallelized computation of ground station and ¹⁸ imaging-target visibility (Eddy & Kochenderfer, 2021), a foundational problem in satellite ¹⁹ scheduling and mission planning.

²⁰ With `brahe`, predicting upcoming satellite passes over ground stations or imaging targets can ²¹ 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
```

³⁷ `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 ³⁸ space situational awareness tasks such as predicting the orbits of all Starlink satellites over the ³⁹ 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
87
88

```

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](#)).

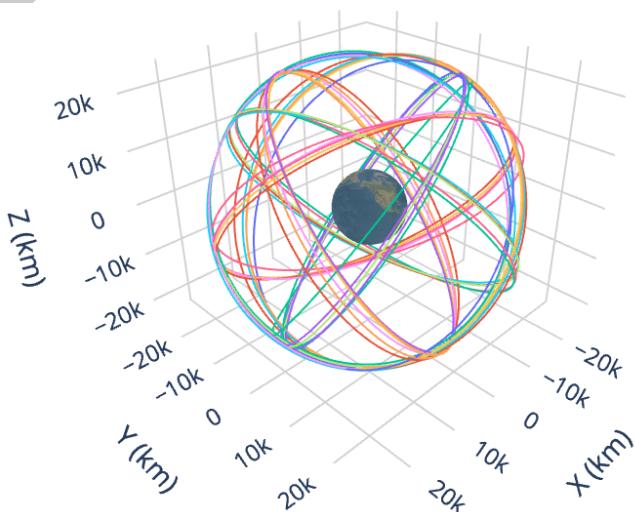


Figure 1: Visualization of all GPS Satellite Orbits

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 ([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
102 curves, making quick-adoption and integration into projects difficult. Furthermore, Orekit is
103 implemented in Java, which can be a barrier to adoption in the current scientific ecosystem
104 with users who are more familiar with Python. GMAT uses a domain-specific scripting language
105 and has limited documentation and examples, making it difficult for new users to get started.
106 Libraries such as poliastro ([Cano Rodriguez & Martínez Garrido, 2022](#)) and Open Space Toolkit
107 (OSTk) ([Open Space Collective, 2025](#)) provides Python interfaces, but their object-oriented
108 architecture adds layers of abstraction that can make it difficult to adapt them to problems
109 that outside their predefined modeling frameworks. Additionally, poliastro is no longer actively
110 maintained and OSTk only supports Linux environments and requires a specialized Docker
111 environment to run. Other academic tools like Basilisk ([Kenneally et al., 2020](#)), provide
112 high-fidelity modeling capabilities for full spacecraft guidance, navigation, and control (GNC)
113 simulations, but are not directly distributed through standard package managers like PyPI and
114 must be compiled from source to be used. Finally, these works often have limited documentation
115 and usage examples, making it difficult for new users to get started.

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

128 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
129 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](#)).

137 Acknowledgments

138 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.

142 References

- 143 a.i. Solutions, Inc. (2025). *FreeFlyer: Spacecraft Mission Analysis and Operations Software*
144 (Version 7.10). <https://www.ai-solutions.com/freeflyer>
- 145 Analytic Graphics, Inc. (AGI). (2023). *Systems Tool Kit (STK)* (Version 12.7). <https://www.agi.com/products/stk>
- 146 Cano Rodriguez, J. L., & Martínez Garrido, J. (2022). *poliastro* (Version v0.17.0). <https://github.com/poliastro/poliastro/>
- 147 Eddy, D., Ho, M., & Kochenderfer, M. J. (2025). Optimal Ground Station Selection for Low-
148 Earth Orbiting Satellites. *IEEE Aerospace Conference*. <https://arxiv.org/abs/2410.16282>
- 149 Eddy, D., & Kochenderfer, M. J. (2021). A Maximum Independent Set Method for Schedul-
150 ing Earth-Observing Satellite Constellations. *Journal of Spacecraft and Rockets*, 58(5),
151 1416–1429.
- 152 Hohenkerk, C. (2017). IAU Standards of Fundamental Astronomy (SOFA): Time and Date.
153 In *The Science of Time 2016: Time in Astronomy & Society, Past, Present and Future*.
154 Springer.
- 155 Hughes, S. P., Qureshi, R. H., Cooley, S. D., & Parker, J. J. (2014). Verification and
156 Validation of the General Mission Analysis Tool (GMAT). *AIAA/AAS Astrodynamics
Specialist Conference*.
- 157 Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science &
158 Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 159 Kelso, T. S. (2025). *Celestrak Active Satellite Database*. <https://celestrak.com/>
- 160 Kenneally, P. W., Piggott, S., & Schaub, H. (2020). Basilisk: A Flexible, Scalable and Modular
161 Astrodynamics Simulation Framework. *Journal of Aerospace Information Systems*, 17(9),
162 496–507. <https://doi.org/10.2514/1.I010762>
- 163 Kepler, J. (1619). *Epitome Astronomiae Copernicanae*.
- 164 Kim, G. R., Eddy, D., Srinivas, V., & Kochenderfer, M. J. (2025). Scalable Ground Station
165 Selection for Large LEO Constellations. *arXiv Preprint arXiv:2510.03438*. <https://arxiv.org/abs/2510.03438>
- 166 Maisonobe, L., Pommier, V., & Parraud, P. (2010). Orekit: An Open Source Library for
167 Operational Flight Dynamics Applications. *International Conference on Astrodynamics
168 Tools and Techniques*.
- 169 Mars Climate Orbiter Mishap Investigation Board. (1999). *Mars Climate Orbiter Mishap
Investigation Board Phase I Report* [Tech. Report]. Jet Propulsion Laboratory / National
170 Aeronautics and Space Administration. https://llis.nasa.gov/llis_lib/pdf/1009464main1_0641-mr.pdf
- 171 Montenbruck, O., & Gill, E. (2000). *Satellite Orbits: Models, Methods and Applications*.
172 Springer.
- 173 Newton, I. (1687). *Philosophiae Naturalis Principia Mathematica*.

- 180 Open Space Collective. (2025). *Open Space Toolkit*. [https://github.com/open-space-collective/
open-space-toolkit](https://github.com/open-space-collective/open-space-toolkit)
- 182 Peters, T. (2004). The Zen of Python. In *Pro Python*. Springer.
- 183 Petit, G., & Luzum, B. (2010). *IERS Conventions, Technical Note 36*.
- 184 Plotly Technologies Inc. (2015). *Collaborative Data Science*. Plotly Technologies Inc.
- 185 <https://plot.ly>
- 186 Procida, D. (2024). *Diátaxis: A Systematic Approach to Technical Documentation Authoring*.
- 187 <https://diataxis.fr/>.
- 188 Reid, T. G., Chan, B., Goel, A., Gunning, K., Manning, B., Martin, J., Neish, A., Perkins, A.,
189 & Tarantino, P. (2020). Satellite Navigation for the Age of Autonomy. *2020 IEEE/ION
190 Position, Location and Navigation Symposium (PLANS)*.
- 191 Stringham, C., Farquharson, G., Castelletti, D., Quist, E., Riggi, L., Eddy, D., & Soenen, S.
192 (2019). The Capella X-band SAR Constellation for Rapid Imaging. *IEEE International
193 Geoscience and Remote Sensing Symposium*.
- 194 Vallado, D. A. (2001). *Fundamentals of Astrodynamics and Applications*.
- 195 Vallado, D., Crawford, P., Hujasak, R., & Kelso, T. S. (2006). Revisiting Spacetrack Report
196 #3. *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*.