The Journal of Open Source Software

`lhorizon`: geometry and targeting via JPL *Horizons*

Michael St. Clair[1] **and Matthew Siegler**[2]

1 Chief Technical Officer, Million Concepts **2** Research Scientist, Planetary Science Institute

DOI: 10.21105/joss.03495

Software

- Review
- Repository
- Archive

Editor: Arfon Smith

Reviewers:

- @malmans2
- @steo85it

Submitted: 06 July 2021
Published: 15 July 2021

Summary

`lhorizon` helps you find where things are in the solar system. It is built around a thick Python wrapper for the Jet Propulsion Laboratory (JPL) Solar System Dynamics Group *Horizons* service (Giorgini, 2015). *Horizons* is the only system in the world that provides ready-to-go, no-assembly-required access to geometry data for almost every object in the solar system. Other interfaces to *Horizons* exist, but `lhorizon` is particularly optimized for stability and time-to-value for large queries and for processing results with reference to arbitrary topocentric coordinate systems.

`lhorizon` offers a flexible, idiomatic, highly performant pseudo-API to *Horizons* that returns data as standard scientific Python objects (*NumPy* `ndarrays` and *pandas* DataFrames). It provides special handling functions for bulk, chunked, and body-listing queries; it also includes an ancillary module, `lhorizon.targeter`, for finding the footprint of an observer's boresight or field of view in topocentric coordinates on target bodies.

We wrote `lhorizon` in support of a research effort to use Earth-based radio telescopes, including Arecibo and the Very Large Array (VLA), to perform heat flow mapping of the Moon. We needed to coregister these data with existing models, so we were specifically interested in answering questions like: "in lunar latitude and longitude, where is Arecibo pointing?" These facilities were not primarily designed to answer questions about nearby bodies, so their processing pipelines did not make lunar maps. Because we were using these instruments in unusual ways, there were many uncertainties in the measurements, and we wanted to minimize geometric uncertainty. *Horizons* is high-precision and has the unique capability to produce corrected positions relative to arbitrary topocentric points, so it was an appealing source of ephemerides.

However, we had millions of data points, widely dispersed across times and observing locations, so we needed a highly performant programmatic interface. We were pleased to discover that the *astroquery* (Ginsburg et al., 2019) project included a module for querying *Horizons* called `jplhorizons`. This module, written primarily by Michael Mommert around 2016, is tightly integrated with *astroquery*. It uses *astroquery*'s session handlers and parsing system, and returns results in *astropy* tables. Unfortunately, we discovered that `jplhorizons` had experienced bitrot due to server-side changes in *Horizons* that broke functionality we specifically needed. We implemented workarounds, but discovered that the performance of `jplhorizons` was inadequate for our use case. *astroquery*'s parsers and *astropy* tables are powerful, but this power comes at a performance cost. The cost is irrelevant for many applications, but quite relevant for use cases with tens to hundreds of thousands of data points per analysis. We wrote an entirely new response parser using only builtins, *NumPy*, and *pandas*, resulting in performance improvements of 10-100x. We submitted minimal workarounds for the API issues to *astroquery*, but the changes we made in our fork were too extensive to be folded into *astroquery* via a PR – especially because removing *astropy* objects and idioms was one of our major design goals. We named this fork `lhorizon` and have continued developing it as a distinct project.

Statement of Need

JPL is the most authoritative producer of solar system ep'hemerides. Its geometry products are essential elements of academic and industrial work in planetary science, astronomy, geosciences, and many other fields. They are useful for any application that makes use of artificial satellites or needs to know about the position of solar system bodies (even "simple" quantities like the solar angle at an arbitrary Earth location). Their value as public resources is incalculable. Unfortunately, they are not always easy to use.

JPL offers two automated interfaces to its geometry products: the SPICE toolkit (NASA NAIF, 2021a) (developed by NAIF, NASA's Navigation and Ancillary Information Facility) and *Horizons*. SPICE is very powerful but has an extremely high barrier to entry. Using SPICE requires not only acquiring and configuring software, but also collecting the appropriate data files, called "kernels." Not counting automatically-generated kernels for minor bodies like asteroids, there are tens of thousands of kernels; counting them, there are tens of millions. There is no central repository for kernels – NAIF's website comes closest, but crucial kernels are scattered across hundreds of other Planetary Data System (PDS) archives. Making kernels work together is challenging and requires scripting in a domain-specific markup language. The SPICE "required reading" (NASA NAIF, 2021b) is dozens of chapters and hundreds of pages long. SPICE is difficult for planetary scientists and specialized engineers, let alone nonspecialists who need quick access to geometry data. SPICE implementations exist in several languages, and excellent wrappers exist (notably the idiomatic Python wrapper *SpiceyPy* (Annex et al., 2020)), but they do not solve the conceptual and data access difficulties of SPICE.

Horizons is, by comparison, user-friendly. While it does not implement all of the utilities of the SPICE toolkit, it covers more sites and offers flexibility SPICE does not. *Horizons* offers several interface methods: interactive web, telnet, email, and web CGI. A wrapper that builds URLs for the CGI interface and parses returned text is an obvious architecture for a *Horizons* pseudo-API. Because bulk queries are not easy to compose and parsing the returned text is not straightforward, this goes a long way towards improving access to solar system geometry data. An official REST API to *Horizons* is forthcoming but not yet available, and the details of its capabilities have not been publicly released (Giorgini, 2020). It is likely that high-level wrappers for this API will be useful, and we plan to update `lhorizon` to fill this role.

Other Related Work

Many wrappers, helpers, and interfaces for *Horizons* have been developed, though most are incomplete, defunct, or encapsulated in other applications. They include:

- *py-NASA-horizons*, a vectors query wrapper; abandoned since 2013 and no longer functional (tpltnt (pseudonymous), 2013)
- Mihok's *HORIZON-JPL*, a REST API; abandoned since 2014 and no longer functional (Mihok, 2014)
- Fejes' *JS-HORIZONS*, a js library focused on physical rather than geometry data (Fejes, 2020)

More broadly, libraries like `astropy.coordinates` (A. M. Price-Whelan et al., 2018) and *Skyfield* (Rhodes, 2019) that perform calculations on JPL ephemerides are similar in application to `lhorizon` and should be considered by potential users.

Acknowledgements

This work was supported by a NASA Solar System Workings grant, #NNX16AQ10G, and a NASA Solar System Observations grant, #NNX17AF12G.

References

A. M. Price-Whelan, and, Sipőcz, B. M., Günther, H. M., Lim, P. L., Crawford, S. M., Conseil, S., Shupe, D. L., Craig, M. W., Dencheva, N., Ginsburg, A., VanderPlas, J. T., Bradley, L. D., Pérez-Suárez, D., Val-Borro, M. de, Aldcroft, T. L., Cruz, K. L., Robitaille, T. P., Tollerud, E. J., Ardelean, C., … and, and. (2018). The astropy project: Building an open-science project and status of the v2.0 core package. *The Astronomical Journal*, *156*(3), 123. https://doi.org/10.3847/1538-3881/aabc4f

Annex, A., Pearson, B., Seignovert, B., Carcich, B., Eichhorn, H., Mapel, J., Forstner, J. von, McAuliffe, J., Rio, J. del, Berry, K., Aye, K.-M., Stefko, M., Val-Borro, M. de, Kulumani, S., & Murakami, S. (2020). SpiceyPy: A pythonic wrapper for the SPICE toolkit. *Journal of Open Source Software*, *5*(46), 2050. https://doi.org/doi.org/10.21105/joss.02050

Fejes, Z. (2020). *JS-HORIZONS*. GitHub. https://github.com/zachfejes/js-horizons

Ginsburg, A., Sipőcz, B. M., Brasseur, C. E., Cowperthwaite, P. S., Craig, M. W., Deil, C., Guillochon, J., Guzman, G., Liedtke, S., Lim, P. L., Lockhart, K. E., Mommert, M., Morris, B. M., Norman, H., Parikh, M., Persson, M. V., Robitaille, T. P., Segovia, J.-C., Singer, L. P., … and, J. W. (2019). Astroquery: An astronomical web-querying package in python. *The Astronomical Journal*, *157*(3), 98. https://doi.org/10.3847/1538-3881/aafc33

Giorgini, J. D. (2015). Status of the JPL horizons ephemeris system. *IAU General Assembly*.

Giorgini, J. D. (2020). *JPL horizons overview and future plans*. https://lsst-sssc.github.io/Sprint2020/talks/Day-2_Giorgini_Horizons_LSST20200617.pdf

Mihok, M. (2014). *Horizon-jpl*. GitHub. https://github.com/mihok/horizon-jpl

NASA NAIF. (2021a). *NAIF spice data*. https://naif.jpl.nasa.gov/naif/data.html

NASA NAIF. (2021b). *Spice required reading*. https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/C/req/index.html

Rhodes, B. (2019). *Skyfield: High precision research-grade positions for planets and earth satellites generator*. Astrophysics Source Code Library. https://ui.adsabs.harvard.edu/abs/2019ascl.soft07024R/abstract

tpltnt (pseudonymous). (2013). *Py-NASA-horizons*. GitHub. https://github.com/tpltnt/py-NASA-horizons