unxt: A Python package for unit-aware computing with JAX

Nathaniel Starkman [1¶], **Adrian M. Price-Whelan** [2], and **Jake Nibauer** [3]

1 Brinson Prize Fellow at Kavli Institute for Astrophysics and Space Research, Massachusetts Institute of Technology, USA **2** Center for Computational Astrophysics, Flatiron Institute, USA **3** Department of Physics, Princeton University, USA ¶ Corresponding author

DOI: 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Published: unpublished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).

Summary

unxt is a Python package for unit-aware computing with JAX (Bradbury et al., 2018), which is a high-performance numerical computing library that enables automatic differentiation and just-in-time compilation to accelerate code execution on multiple compute architectures. unxt is built on top of quax (Kidger, 2023), which provides a framework for building array-like objects that can be used with JAX. unxt extends quax to provide support for unit-aware computing using the astropy.units package (Astropy Collaboration et al., 2013, 2022) as a units backend. unxt provides seamless integration of physical units into high performance numerical computations, significantly enhancing the capabilities of JAX for scientific applications.

The primary purpose of unxt is to facilitate unit-aware computations in JAX, ensuring that operations involving physical quantities are handled correctly and consistently. This is crucial for avoiding errors in scientific calculations, such as those that could lead to significant consequences like the infamous Mars Climate Orbiter incident (NASA). unxt is designed to be intuitive, easy to use, and performant, allowing for a straightforward implementation of units into existing JAX codebases.

unxt is accessible to researchers and developers, providing a user-friendly interface for defining and working with units and unit systems. It supports both static and dynamic definitions of unit systems, allowing for flexibility in various computational environments. Additionally, unxt leverages multiple dispatch to enable deep interoperability with other libraries, currently astropy, and to support custom array-like objects in JAX. This extensibility makes unxt a powerful tool for a wide range of scientific and engineering applications, where unit-aware computations are essential.

Statement of Need

JAX is a powerful tool for high-performance numerical computing, offering features such as automatic differentiation, just-in-time compilation, and support for sharding computations across multiple devices. It excels in providing unified interfaces to various compute architectures, including CPUs, GPUs, and TPUs, to accelerate code execution (Bradbury et al., 2018). However, JAX operates primarily on "pure" arrays, which means it lacks support to define custom array-like objects, including those that can handle units, and to use those use those objects in within the JAX ecosystem. While JAX can handle PyTrees with some pre-programmed support and the ability to register additional support, the operations it performs are still fundamentally array-based. This limitation poses a challenge for scientific applications that require handling of physical units.

Astropy has been an invaluable resource for the scientific community, with over 10,000 citations to its initial paper and more than 2,000 citations to its 2022 paper (Astropy Collaboration et al., 2013, 2022). One of the foundational sub-packages within Astropy is `astropy.units`, which provides robust support for units and quantities, enabling the propagation of units through NumPy functions. This functionality ensures that scientific calculations involving physical quantities are handled correctly and consistently. However, despite JAX's numpy-like API, it does not support the same level of extensibility, and `astropy.units` cannot be directly extended to work with JAX. This gap highlights the need for a solution that integrates the powerful unit-handling capabilities of Astropy with the high-performance computing features of JAX.

`unxt` addresses this gap by providing a function-oriented framework—consistent with the style of JAX—for handling units and dimensions, with an object-oriented front-end that will be familiar to users of `astropy.units`. By leveraging `quax`, `unxt` defines a `Quantity` class that seamlessly integrates with JAX functions. This integration is achieved by providing a comprehensive set of overrides for JAX primitives, ensuring that users can utilize the `Quantity` class without needing to worry about the underlying JAX interfacing. This design allows users to perform unit-aware computations effortlessly, maintaining the high performance and flexibility that JAX offers while ensuring the correctness and consistency of operations involving physical quantities.

Related Works

`unxt` is designed to be extensible to other unitful-computation libraries. The `unxt` package is not intended to replace these libraries, but rather to provide a JAX-optimized frontend. Some prominent libraries include:

- `astropy.units` (Astropy Collaboration et al., 2013, 2022). The `unxt` package currently uses the unit conversion framework from `astropy.units` package in its backend, providing a more flexible front-end interface and particularly JAX-compatible Quantity classes for doing array computations with units.
- `unyt` (Goldbaum et al., 2018). The `unyt` library is a popular Python package for unit-aware computations. It provides `Quantity` classes that work with (at time of writing) `numpy` (Harris et al., 2020) and `dask` (Dask Development Team, 2016) arrays.
- `pint` (Grecco, 2012). The `pint` library is a popular Python package for unit-aware computations. It provides `Quantity` classes that work with many array types, but not `jax` (at time of writing).

Acknowledgements

Support for this work was provided by The Brinson Foundation through a Brinson Prize Fellowship grant.

The authors thank the Astropy collaboration and many contributors for their work on `astropy`, which has been invaluable to the scientific community. Members of the `unxt` development team are also core developers and maintainers of the `astropy.units` package, and we had `astropy.units` as our guiding star while developing `unxt`. The authors also thank Dan Foreman-Mackey for useful discussions, and the attendees of the 2024 JAXtronomy workshop at the Center for Computational Astrophysics at the Flatiron Institute. We also extend our gratitude to Patrick Kidger for his valuable communications and guidance on using `quax` to ensure seamless integration of `unxt` with `jax`.

References

Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., Earl, N., Starkman, N., Bradley, L., Shupe, D. L., Patil, A. A., Corrales, L., Brasseur, C. E., Nöthe, M., Donath, A., Tollerud, E., Morris, B. M., Ginsburg, A., Vaher, E., Weaver, B. A., Tocknell, J., Jamieson, W., … Astropy Project Contributors. (2022). The Astropy project: Sustaining and growing a community-oriented open-source project and the latest major release (v5.0) of the core package. *935*(2), 167. https://doi.org/10.3847/1538-4357/ac7c74

Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A. M., Kerzendorf, W. E., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M. M., Nair, P. H., … Streicher, O. (2013). Astropy: A community Python package for astronomy. *558*, A33. https://doi.org/10.1051/0004-6361/201322068

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.4.35). http://github.com/jax-ml/jax

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. http://dask.pydata.org

Goldbaum, N. J., ZuHone, J. A., Turk, M. J., Kowalik, K., & Rosen, A. L. (2018). unyt: Handle, manipulate, and convert data with units in Python. *Journal of Open Source Software*, *3*(28), 809. https://doi.org/10.21105/joss.00809

Grecco, H. (2012). *Pint: Makes units easy*. https://github.com/hgrecco/pint

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Kidger, P. (2023). *Quax: JAX + multiple dispatch + custom array-ish objects*.

NASA. *Mars Climate Orbiter - NASA Science — science.nasa.gov*. https://science.nasa.gov/mission/mars-climate-orbiter/

DRAFT