The Journal of Open Source Software

rowan: A Python package for working with quaternions

Vyas Ramasubramani[1] **and Sharon C. Glotzer**[1, 2, 3]

1 Department of Chemical Engineering, University of Michigan **2** Department of Materials Science and Engineering, University of Michigan **3** Biointerfaces Institute, University of Michigan

DOI: 10.21105/joss.00787

Software

- Review
- Repository
- Archive

Submitted: 29 May 2018
Published: 30 July 2018

Summary

Numerous fields in science and engineering require methods for rotating objects. Quaternions are perhaps the most popular formalism for representing spatial rotations due to their natural parameterization of the space of rotations $SO(3)$ and the relative efficiency of computing quaternion-based rotation operations. A simple, uniform, and efficient implementation of quaternion operations is therefore critical to developing code to solve domain-specific problems in areas such as particle simulation and attitude determination. Python implementations of quaternion operations do exist, but they suffer from various drawbacks. Some tools are performance limited due to, *e.g.*, having limited or no support for NumPy style array broadcasting (Wynn 2015, Brett and Gohlke (2009)). Since NumPy is a *de facto* standard in scientific computing applications, such support is both a prerequisite for a package to be easily incorporated into existing code bases and a Pythonic way to achieve a performant solution. Other packages that do support NumPy may have complex dependencies for accessing their full features or require conversion into some internal format, making them cumbersome to incorporate into existing code bases that need to operate on raw arrays (Boyle 2017).

The `rowan` package, named for William Rowan Hamilton, is a quaternion package that addresses these issues. By operating directly on NumPy arrays and offering first-class support for broadcasting for all modules and functions in the package, `rowan` ensures high efficiency for operating on the large arrays common in computer graphics or scientific applications. We quantify performance in Figure 1 by comparison to `pyquaternion` (Wynn 2015) and `numpy-quaternion` (Boyle 2017), two well-known alternatives to `rowan`. For small arrays, the performance benefits of `numpy-quaternion` and `rowan` are somewhat muted since the Python function calls dominate the total run time. In fact, at this scale `rowan` performs quite similarly to `pyquaternion`, which is a pure Python solution, while `numpy-quaternion`, a hybrid Python-C solution, performs faster than both. For large arrays (*e.g.* $N > 10000$) where performance limitations become significant, however, `rowan` outstrips `pyquaternion` by roughly two orders of magnitude and approaches the performance of `numpy-quaternion`. Although a typical function call with `rowan` is, on average, roughly 4 times slower than `numpy-quaternion`, this performance difference is offset by `rowan`'s relative ease of installation and incorporation. The package avoids any hard dependencies other than NumPy itself and directly uses NumPy arrays, making `rowan` an unobtrusive dependency with essentially zero barrier for introduction into existing code bases.

A full-featured quaternion library, `rowan` has extensive capabilities in addition to basic quaternion arithmetic operations. These functions include: methods for point set registration, including some that are specialized for solving the Procrustes problem of superimposing corresponding sets of points; functions for quaternion calculus and interpolation; the ability to sample random rotation quaternions from $SO(3)$; and functions to compute various distance metrics on the quaternion manifold. For applications focused on rotations, `rowan` provides the ability to convert between numerous common rotation

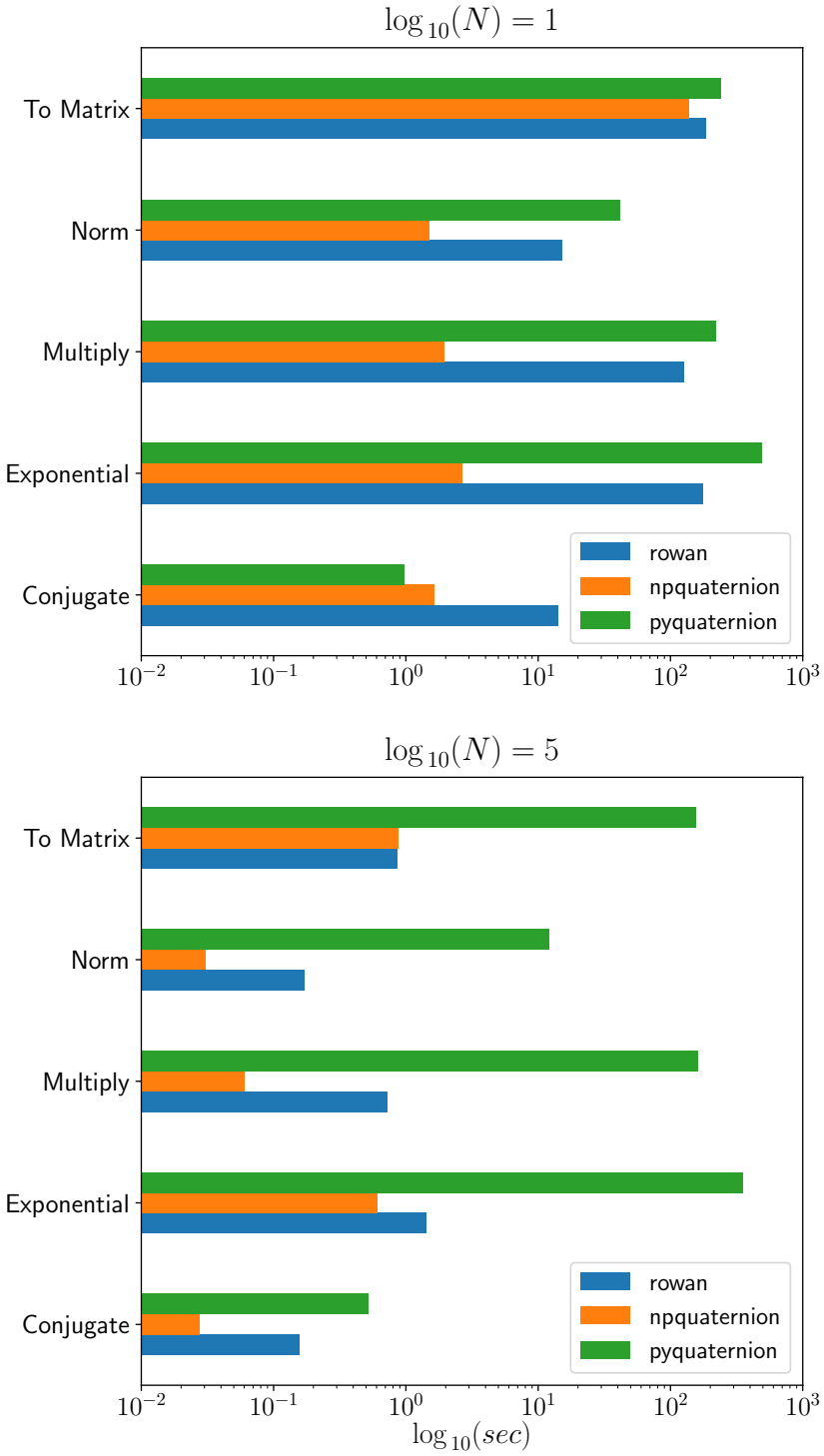

Figure 1: Performance Comparison

formalisms, including full support for all Euler angle conventions, which is not found in other Python quaternion packages.

This package arose due to the need to represent anisotropic particle orientations in Monte Carlo simulations in the Glotzer Group at the University of Michigan. Unlike configurations of spherical particles, which are entirely described by their positions alone, configurations of anisotropic particles also contain information on particle orientations that change over the course of the simulation. Our simulation software HOOMD-blue (Anderson, Lorenz, and Travesset 2008, Glaser et al. (2015)) uses quaternions to represent particle orientations, as do many of the packages we write for analyzing these simulations. Although some packages require C++ implementations, a large number are pure Python code bases that each originally contained their own independent implementation of quaternion operations with slightly different features and levels of generality. The resulting code fragmentation made code maintenance much more challenging and failed to provide a standard implementation of quaternion operations for more ad hoc analysis tasks. `rowan` addressed these needs by providing a unified, high-performance, and easily utilized solution. The package was incorporated into the open-source plato (Spellings 2018) simulation visualization tool as well as some internal packages that have not yet been open-sourced. Going forward, `rowan` will not only simplify the maintenance of our existing code bases, it will also facilitate future code development involving quaternion operations both within and outside our group.

Acknowledgements

This work was partially supported by a Simons Investigator award from the Simons Foundation to Sharon Glotzer. We would like to acknowledge Bradley D. Dice, Carl S. Adorf, and Matthew P. Spellings for helpful suggestions and discussions during the development of this package.

References

Anderson, Joshua A., Chris D. Lorenz, and A. Travesset. 2008. "General purpose molecular dynamics simulations fully implemented on graphics processing units." *Journal of Computational Physics* 227 (10). Academic Press:5342–59. https://doi.org/10.1016/j.jcp.2008.01.047.

Boyle, Michael. 2017. "Numpy-Quaternion." https://github.com/moble/quaternion.

Brett, Matthew, and Christoph Gohlke. 2009. "Transforms3d." https://github.com/matthew-brett/transforms3d.

Glaser, Jens, Trung Dac Nguyen, Joshua A. Anderson, Pak Lui, Filippo Spiga, Jaime A. Millan, David C. Morse, and Sharon C. Glotzer. 2015. "Strong scaling of general-purpose molecular dynamics simulations on GPUs." *Computer Physics Communications* 192 (July). North-Holland:97–107. https://doi.org/10.1016/j.cpc.2015.02.028.

Spellings, Matthew. 2018. "Plato." https://bitbucket.org/glotzer/plato.

Wynn, Kieran. 2015. "Pyquaternion." https://github.com/KieranWynn/pyquaternion.