The Journal of Open Source Software

coxeter: A Python package for working with shapes

Vyas Ramasubramani[1], **Bradley D. Dice**[2], **Tobias T. Dwyer**[1], **and Sharon C. Glotzer**[1, 2, 3]

1 Department of Chemical Engineering, University of Michigan **2** Department of Physics, University of Michigan **3** Biointerfaces Institute, University of Michigan

DOI: 10.21105/joss.03098

Software

- Review
- Repository
- Archive

Editor: Juan Nunez-Iglesias

Reviewers:

- @wolfv
- @evetion

Submitted: 25 February 2021
Published: 17 July 2021

Package Overview

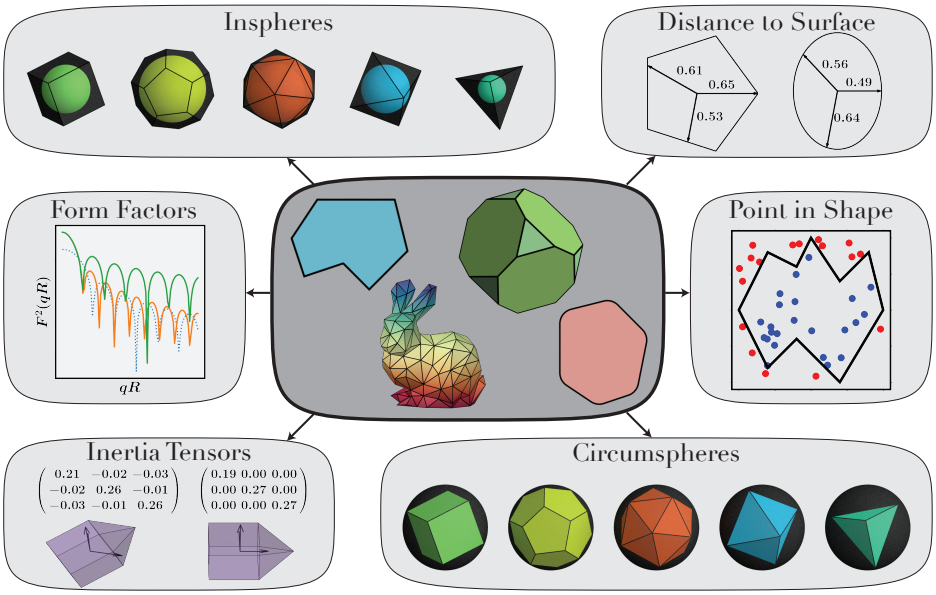

Figure 1: The coxeter package supports calculating a wide range of properties on shapes in two and three dimensions. The central bubble in the figure contains a subset the shapes supported by coxeter, which includes simple polygons in 2D and arbitrary 3D polyhedral meshes. The surrounding bubbles represent a sampling of what coxeter can do with a shape. From the top-left going clockwise, these are: determining the inspheres of polytopes (as well as minimal bounding spheres for all shapes); calculating the distance from the centroid of a shape to its boundary; checking whether a point lies inside or outside a shape; determining the circumspheres of polytopes (as well as maximal bounded spheres for all shapes); calculating moments of inertia in 2D and inertia tensors in 3D (including support for reorienting a shape along its principal axes); and anisotropic form factors for arbitrary shapes, which are important for scattering calculations.

The coxeter Python package provides tools to represent, generate, and compute properties of shapes in two and three dimensions. The package emphasizes simplicity and flexibility, using a common set of abstractions to present a largely uniform interface across various shapes and allowing easy mutation of almost all of their geometric attributes. The package also serves as a repository for specific groups of shapes, exposing an easy-to-use API for shape generation that users can extend to make particular geometric objects collectively accessible.

Statement of Need

Considerations of shape are becoming increasingly important in materials science as improved synthetic capabilities have allowed the creation of a wide range of anisotropic particles (Glotzer & Solomon, 2007). Colloidal science in particular has seen immense growth in this area, and numerous studies have shown that particle shape is an important handle for controlling the self-assembly (Damasceno et al., 2012) and packing (Chen et al., 2014) of colloidal crystals. Precise modeling of these systems requires reproducible methods for generating shapes and calculating their properties (Allen & Germano, 2006; Anderson et al., 2020). An important aspect of achieving this reproducibility is making canonical definitions of shapes used in particular studies readily available to other researchers. Furthermore, since these shapes may be used in physics-based simulations, any calculations must be robust enough to handle any numerical issues that may arise across a wide range of different geometries. Some of the applications of coxeter to date include: the development of equations of state for polyhedral particles (Irrgang et al., 2017); the calculation of physical properties for dynamical simulation of anisotropic particles (Ramasubramani et al., 2020); and the orientational ordering of ellipsoidal colloids in a magnetic field (Kao et al., 2019).

Summary of Features

Computing Geometric and Physical Properties

The central elements in coxeter are the shape classes, which encode the features of particular types of shapes. In order to enforce a highly uniform API and ensure conceptual clarity, all shape classes inherit from a small set of abstract base classes that encode specific subsets of properties: for instance, the standard properties of all shapes in two dimensions are embedded in the `Shape2D` class. In addition to standard types of shapes such as ellipsoids or polygons, coxeter also includes more esoteric shape classes like spheropolyhedra, which are important for modeling the imperfect rounded polyhedra frequently synthesized at the nano- and micron scales (Rossi et al., 2015; Zhang et al., 2011). Even simple properties like surface areas are generally nontrivial to compute for such shapes, but using coxeter spheropolyhedra are no more difficult to work with than any other 3D shape. Working with convex polygons and polyhedra using coxeter is greatly simplified via internal use of SciPy's convex hull calculations (Virtanen et al., 2020), allowing the user to simply provide a set of vertices while coxeter performs the appropriate determination of facets and plane equations based on the simplices of the convex hull.

The shape classes transparently expose many geometric attributes in the form of settable Python properties, allowing on-the-fly rescaling or transformation of the shape. This aspect of coxeter is designed to fill a common gap in most computational geometry libraries, which typically focus on solving more complex problems like finding convex hulls, Voronoi tessellations, and Delaunay triangulations (The CGAL Project, 2020); coxeter aims to provide a standard implementation for simpler calculations such as surface areas and bounding spheres for which formulas are generally well-known but often require careful consideration to calculate robustly and efficiently. These properties range from standard calculations like volumes and surface areas to less common metrics like mean curvatures and asphericities that are relevant for computing equations of state for polyhedral particles (Irrgang et al., 2017). The package also provides various types of bounding and bounded spheres of shapes, which are measures of the extent of polygons and polyhedra within crystal structures. To simplify interoperability with other packages in the scientific computing ecosystem, non-scalar properties are generally provided as NumPy arrays (Harris et al., 2020).

In addition to purely geometric properties, shapes in coxeter also expose various physically relevant quantities in order to support a wide range of applications for shapes of constant

density. Some examples of such properties are inertia tensors, which are integral to the equations of motion for anisotropic bodies, and scattering form factors, which are Fourier transforms of the shape volume that help characterize structure in condensed matter physics (Als-Nielsen & McMorrow, 2011). Since physical equations and observables can be highly sensitive to inputs like inertia tensors, coxeter emphasizes robust methods for their evaluation (Kallay, 2006). Two dimensional shapes like polygons are embedded in three dimensions rather than in the plane, so coxeter uses the rowan library (Ramasubramani & Glotzer, 2018) to rotate them into the plane and then compute various properties to avoid complications and numerical instabilities that arise from performing integrals over planar lamina embedded in 3D Euclidean space.

Shape Generation

The library also serves as a repository for the generation of shapes. While simple classes of shapes like spheres and ellipsoids can be described via a small fixed set of parameters, the definitions of polygons and polyhedra can be arbitrarily long depending on the number of vertices of these shapes. The shape family API in coxeter provides a flexible way to define and work with collections of related shapes, ranging from enumerable sets like the Platonic solids to continuously defined sets of shapes (Chen et al., 2014). These different types of shape families are handled using identical APIs, so users can easily switch between shapes that have completely different mathematical definitions using a single line of code. Shape families generate coxeter shape classes from input parameters, simplifying access to computed geometric and physical properties.

A number of such families are bundled into coxeter, but just as importantly, the framework allows users to work with arbitrary lists of shapes provided as dictionaries of attributes. This dictionary-based definition can be simply converted to JSON, making it trivial to share representations of shapes. The library also stores mappings from digital object identifiers (DOIs) to families, so that any user can contribute families associated with published research to make them collectively accessible. We anticipate that the set of shape families in coxeter will grow over time as users generate and contribute their shape families to coxeter, with the goal of providing a centralized repository for use in reproducing and extending prior research, particularly in the field of shape-driven nanoparticle self-assembly. Currently coxeter primarily supports the schema proposed by the GSD library (Glotzer Lab, 2021), making it directly compatible with the HOOMD-blue molecular simulation tool (Anderson et al., 2020), but other schema can be implemented as needed.

Acknowledgements

This research was supported in part by the National Science Foundation, Division of Materials Research Award No. DMR-1808342. V. R. also acknowledges the 2019-2020 J. Robert Beyster Computational Innovation Graduate Fellowship from the College of Engineering, University of Michigan. B. D. acknowledges fellowship support from the National Science Foundation under ACI-1547580, S212: Impl: The Molecular Sciences Software Institute (Krylov et al., 2018; Wilkins-Diehr & Crawford, 2018) and an earlier National Science Foundation Graduate Research Fellowship Grant DGE-1256260 (2016–2019). T. D. is supported by a National Science Foundation Graduate Research Fellowship Grant DGE-1256260.

We would like to acknowledge M. Eric Irrgang for prototype implementations of various parts of this code, as well as Bryan VanSaders and James Proctor for collecting the various early prototypes and relevant methods into a single code base. We additionally thank all code contributors. In addition to the authors and aforementioned contributors, the list of code contributors includes Brandon Butler, Thomas Waltmann, Timothy Moore, Corwin Kerr, Eric Harper, Jens Glaser, William Zygmunt, and Mariano Semelman.

Finally, we would like to acknowledge the following authors of external open-source tools that are used in coxeter:

- Mark Dickinson, who wrote the **polyhedron** module used for point-in-polygon and point-in-polyhedron checks in coxeter.
- David Björkevik, who wrote the **polytri** package used for polygon triangulation in coxeter.
- Campbell Barton, who wrote the **isect_segments-bentley_ottman** package used to validate that polygons have no crossings (i.e. that polygons are simple) in coxeter.

References

Allen, M. P., & Germano, G. (2006). Expressions for forces and torques in molecular simulations using rigid bodies. *Molecular Physics*, *104*(20-21), 3225–3235. https://doi.org/10.1080/00268970601075238

Als-Nielsen, J., & McMorrow, D. (2011). *Elements of modern x-ray physics*. John Wiley & Sons.

Anderson, J. A., Glaser, J., & Glotzer, S. C. (2020). HOOMD-blue: A Python package for high-performance molecular dynamics and hard particle Monte Carlo simulations. *Computational Materials Science*, *173*, 109363. https://doi.org/10.1016/j.commatsci.2019.109363

Chen, E. R., Klotsa, D., Engel, M., Damasceno, P. F., & Glotzer, S. C. (2014). Complexity in Surfaces of Densest Packings for Families of Polyhedra. *Physical Review X*, *4*(1). https://doi.org/10.1103/PhysRevX.4.011024

Damasceno, P. F., Engel, M., & Glotzer, S. C. (2012). Predictive Self-Assembly of Polyhedra into Complex Structures. *Science*, *337*(6093), 453–457. https://doi.org/10.1126/science.1220869

Glotzer Lab. (2021). *GSD v2.4.0*. https://github.com/glotzerlab/gsd

Glotzer, S. C., & Solomon, M. J. (2007). Anisotropy of building blocks and their assembly into complex structures. *Nature Materials*, *6*(8), 557–562. https://doi.org/10.1038/nmat1949

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., 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

Irrgang, M. E., Engel, M., Schultz, A. J., Kofke, D. A., & Glotzer, S. C. (2017). Virial Coefficients and Equations of State for Hard Polyhedron Fluids. *Langmuir*, *33*(42), 11788–11796. https://doi.org/10.1021/acs.langmuir.7b02384

Kallay, M. (2006). Computing the Moment of Inertia of a Solid Defined by a Triangle Mesh. *Journal of Graphics Tools*, *11*(2), 51–57. https://doi.org/10.1080/2151237X.2006.10129220

Kao, P.-K., VanSaders, B. J., Durkin, M. D., Glotzer, S. C., & Solomon, M. J. (2019). Anisotropy effects on the kinetics of colloidal crystallization and melting: comparison of spheres and ellipsoids. *Soft Matter*, *15*, 7479–7489. https://doi.org/10.1039/C9SM00887J

Krylov, A., Windus, T. L., Barnes, T., Marin-Rimoldi, E., Nash, J. A., Pritchard, B., Smith, D. G. A., Altarawy, D., Saxe, P., Clementi, C., Crawford, T. D., Harrison, R. J., Jha, S., Pande, V. S., & Head-Gordon, T. (2018). Perspective: Computational chemistry software

and its advancement as illustrated through three grand challenge cases for molecular science. *Journal of Chemical Physics*, *149*(18), 180901. https://doi.org/10.1063/1.5052551

Ramasubramani, V., & Glotzer, S. C. (2018). rowan: A Python package for working with quaternions. *Journal of Open Source Software*, *3*(27), 787. https://doi.org/10.21105/joss.00787

Ramasubramani, V., Vo, T., Anderson, J. A., & Glotzer, S. C. (2020). A mean-field approach to simulating anisotropic particles. *Journal of Chemical Physics*, *153*(8), 084106. https://doi.org/10.1063/5.0019735

Rossi, L., Soni, V., Ashton, D. J., Pine, D. J., Philipse, A. P., Chaikin, P. M., Dijkstra, M., Sacanna, S., & Irvine, W. T. M. (2015). Shape-sensitive crystallization in colloidal superball fluids. *Proceedings of the National Academy of Sciences*, *112*(17), 5286–5290. https://doi.org/10.1073/pnas.1415467112

The CGAL Project. (2020). *CGAL User and Reference Manual* (5.1.1 ed.). CGAL Editorial Board.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wilkins-Diehr, N., & Crawford, D. T. (2018). NSF's Inaugural Software Institutes: The Science Gateways Community Institute and the Molecular Sciences Software Institute. *Computing in Science and Engineering*, *20*(5), 26–38. https://doi.org/10.1109/MCSE.2018.05329813

Zhang, Y., Lu, F., Lelie, D. van der, & Gang, O. (2011). Continuous phase transformation in nanocube assemblies. *Physical Review Letters*, *107*, 135701. https://doi.org/10.1103/PhysRevLett.107.135701