The Journal of Open Source Software

Stripy: A Python module for (constrained) triangulation in Cartesian coordinates and on a sphere.

Louis Moresi[1,3] **and Ben Mather**[2]

1 School of Earth Science, The University of Melbourne **2** School of Geoscience, The University of Sydney **3** Research School of Earth Sciences, Australian National University

DOI: 10.21105/joss.01410

Software

- Review
- Repository
- Archive

Submitted: 15 March 2019
Published: 13 June 2019

Summary

The triangulation of scattered points is a common problem in science and engineering when local neighbourhood information is required for computation. Typical applications include the calculation of neighbour relationships, interpolants, derivatives, and smoothly-fitting surfaces. A more specialised problem is the triangulation of points distributed on the surface of a sphere, nevertheless, this is important in most global geographical applications where data are registered in longitude and latitude and the Earth's ellipticity can be considered as a second order effect.

Our package, `stripy`, is a python, `numpy`-based interface to the TRIPACK and STRIPACK Fortran code for (constrained) triangulation in Cartesian coordinates and on a sphere, respectively (Renka, 1996a, 1997a) and includes routines from SRFPACK and SSRFPACK for interpolation (nearest neighbour, linear, and hermite cubic) and to evaluate derivatives (Renka, 1996b, 1997b). Our focus in developing an augmented set of wrappers for a venerable and widely used set of fortran subroutines has been on ease of use so that triangulated or scattered data on the sphere can be brought quickly into interactive jupyter notebooks for analysis and interrogation.

`stripy` is an object-oriented package that extends the functionality of the original collection of subroutine by adding a number of useful tools such as the construction of regular meshes on the sphere (icosahedral and octahedral each with face-centre-point variants, a triangulated cube and a truncated icosahedron with face-centre points that produces a C60 / soccerball mesh). `stripy` also includes functionality to refine meshes globally and locally by triangle or edge subdivision. We have taken some care to ensure that queries on the triangulations are vectorised within numpy queries. `stripy` offers a consistent API between the two coordinate

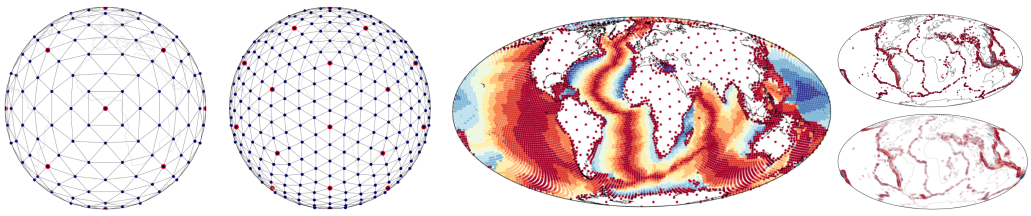

Figure 1: Samples meshes on the sphere produced by `stripy` based on the octahedron with added points at the face centroids (far left), and the icosahedron with with face points (left). An icosahedron grid with refinement in regions where ocean age-grid information is available (Müller, Sdrolias, Gaina, & Roest, 2008), and an icosahedral mesh refined to create a smooth heat-map of seismic activity and average earthquake depth from a global catalogue of earthquakes of magnitude > 5.5 since 1990 from the IRIS online catalogue

system so users can transfer from Cartesian to spherical coordinates with minor changes to their code.

`stripy` includes the following functionality:

- Spherical and Cartesian triangulation of scattered points.
- Construction of Cartesian and Spherical meshes.
- Nearest-neighbour, linear, and hermite cubic interpolation.
- Evaluation of derivatives.
- Smoothing operations.
- Mesh refinement on line segments / triangle centroids.
- Fast point location with k-d tree interface with angular separation metric on the sphere.

Documentation

`stripy` is bundled with a linked collection of jupyter notebooks that can act as a user guide and an introduction to the package. The notebooks are split into matching sets for spherical and Cartesian triangulations. The notebooks cover:

- Introduction to the triangulation classes
- Use of the gridding tools
- Interpolation
- Gradient operations
- Smoothing operations
- Issues that arise with highly irregular data
- Refinement of triangulations

Particularly for the spherical notebooks, global geographical examples are appropriate and we use the familiar pattern of global coastlines when we plot the global distributions of vertices and edges.

We also provide some worked examples where we mix data that come with very different gridding strategies. The *Crust 1.0* dataset (Laske, Masters, Ma, & Pasyanos, 2013) is supplied as cell-centred values on a 1x1 degree grid of points (i.e. equally spaced in longitude and latitude) with no depth information, whereas the related *litho 1.0* dataset (M. E. Pasyanos, Masters, Laske, & Ma, 2014) is supplied as columns of depth-values at points that are distributed on a seven-times-refined icosahedral mesh. The mixing and matching of these datasets in global maps was the original use-case for `stripy` (Cooper, Miller, & Moresi, 2016).

All documentation can be accessed from within the module via a python function that installs the notebooks at a filesystem location specified by the user at run time.

Installation, Dependencies and Usage

`stripy` requires `numpy` and a fortran compiler such as gfortran to compile the fortran90 versions of the (S)TRIPACK and (S)SRFPACK routines that are included with the distribution. The optional k-d tree methods on the meshes require the `scipy.spatial` module. The documentation is supplied in the form of jupyter notebooks (the jupyter system is a dependency) which also have optional dependencies for the `cartopy` mapping package and the `lavavu` embedded, 3D visualisation package. `stripy` and all python dependencies can be installed through the pypi.org `pip` package. However, the fortran compiler, and several of the dependencies for `cartopy` and `lavavu` may cause problems for inexperienced users. We therefore provided a fully build docker image and a deployment of the documentation / examples on mybinder.org

Acknowledgements

Louis Moresi would like to acknowledge the support of the CIDER 2016 summer program (NSF grant EAR-1135452) during which this project was conceived. Development of `stripy` was financially supported by AuScope (www.auscope.org.au) which is funded by the Australian Government through the National Collaborative Research Infrastructure Strategy.

References

Cooper, C. M., Miller, M., & Moresi, L. (2016). The structural evolution of the deep continental lithosphere. *Tectonophysics*, *695*, 81–89. doi:10.1016/j.tecto.2016.12.004

Laske, G., Masters, G., Ma, Z., & Pasyanos, M. (2013). Update on crust1.0 - a 1-degree global model of earth's crust. *Abstract EGU2013-2658 presented at 2013 Geophys. Res. Abstracts 15*, *15*, 2658.

Müller, R., Sdrolias, M., Gaina, C., & Roest, W. (2008). Age spreading rates and spreading asymmetry of the world's ocean crust. *Geochemistry, Geophysics, Geosystems*, *9*, Q04006. doi:10.1029/2007GC001743

Pasyanos, M. E., Masters, T. G., Laske, G., & Ma, Z. (2014). LITHO1.0: An updated crust and lithospheric model of the Earth. *Journal of Geophysical Research: Solid Earth*, *119*(3), 2153–2173. doi:10.1002/2013JB010626

Renka, R. J. (1996a). Algorithm 751; TRIPACK: a constrained two-dimensional Delaunay triangulation package. *ACM Transactions on Mathematical Software*, *22*(1), 1–8. doi:10.1145/225545.225546

Renka, R. J. (1996b). Algorithm 752: SRFPACK: Software for scattered data fitting with a constrained surface under tension. *ACM Transactions on Mathematical Software*, *22*(1), 9–17. doi:10.1145/225545.225547

Renka, R. J. (1997a). Algorithm 772: STRIPACK: Delaunay triangulation and Voronoi diagram on the surface of a sphere. *ACM Transactions on Mathematical Software*, *23*(3), 416–434. doi:10.1145/275323.275329

Renka, R. J. (1997b). Algorithm 773: SSRFPACK: interpolation of scattered data on the surface of a sphere with a surface under tension. *ACM Transactions on Mathematical Software*, *23*(3), 435–442. doi:10.1145/275323.275330