

GCM-Filters: A Python Package for Diffusion-based Spatial Filtering of Gridded Data

Nora Loose¹, Ryan Abernathey², Ian Grooms¹, Julius Busecke², Arthur Barthe³, Elizabeth Yankovsky³, Gustavo Marques⁴, Jacob Steinberg⁵, Andrew Slavin Ross³, Hemant Khatri⁶, Scott Bachman⁴, and Laure Zanna³

¹ Department of Applied Mathematics, University of Colorado Boulder, Boulder, CO, USA ² Lamont-Doherty Earth Observatory, Columbia University, New York, NY, USA ³ Courant Institute of Mathematical Sciences, New York University, New York, NY, USA ⁴ Climate and Global Dynamics Division, National Center for Atmospheric Research, Boulder, CO, USA ⁵ Woods Hole Oceanographic Institution, Woods Hole, MA, USA ⁶ Earth, Ocean and Ecological Sciences, University of Liverpool, UK

DOI: [10.21105/joss.0XXXX](https://doi.org/10.21105/joss.0XXXX)

Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Editor Name](#) ↗

Submitted: 01 January XXXX

Published: 01 January XXXX

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

GCM-Filters is a python package that allows scientists to perform spatial filtering analysis in an easy, flexible and efficient way. The package implements the filtering method based on the discrete Laplacian operator that was introduced by Grooms et al. (2021). The filtering algorithm is analogous to smoothing via diffusion; hence the name *diffusion-based filters*. GCM-Filters can be used with either gridded observational data or gridded data that is produced by General Circulation Models (GCMs) of ocean, weather, and climate. Spatial filtering of observational or GCM data is a common analysis method in the Earth Sciences, for example to study oceanic and atmospheric motions at different spatial scales or to develop subgrid-scale parameterizations for ocean models.

GCM-Filters provides filters that are highly configurable, with the goal to be useful for a wide range of scientific applications. The user has different options for selecting the filter scale and filter shape. The filter scale can be defined in several ways: a fixed length scale (e.g., 100 km), a scale tied to a model grid scale (e.g., 1°), or a scale tied to a varying dynamical scale (e.g., the Rossby radius of deformation). As an example, [Figure 1](#) shows unfiltered and filtered relative vorticity, where the filter scale is set to a model grid scale of 4°. GCM-Filters also allows for anisotropic, i.e., direction-dependent, filtering. Finally, the filter shape – currently: either Gaussian or Taper – determines how sharply the filter separates scales above and below the target filter scale.

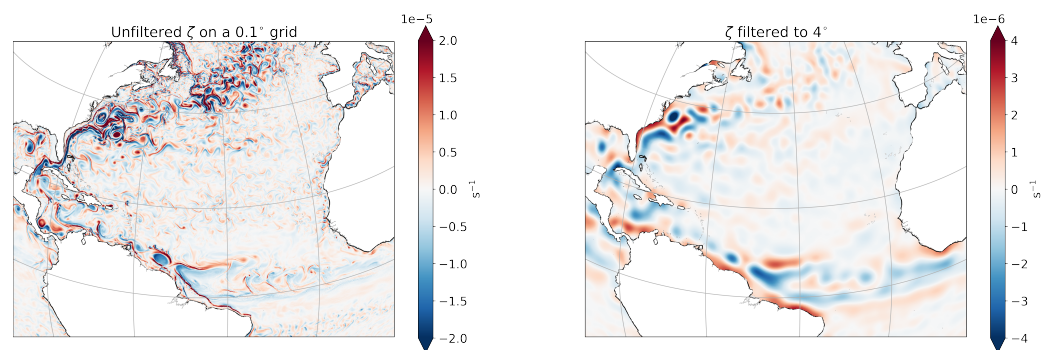


Figure 1: (Left) Snapshot of unfiltered surface relative vorticity $\zeta = \partial_x v - \partial_y u$ from a global 0.1° simulation with MOM6 (Adcroft et al., 2019). (Right) Relative vorticity filtered to 4° , obtained by applying GCM-Filters to the field ζ on the left. The plots are made with matplotlib (Hunter, 2007) and cartopy (Met Office, 2010 - 2015).

Statement of Need

Spatial filtering is commonly used as a scientific tool for analyzing gridded data. An example of an existing spatial filtering tool in python is SciPy's (Virtanen et al., 2020) `ndimage.gaussian_filter` function, implemented as a sequence of convolution filters. While being a valuable tool for image processing (or blurring), SciPy's Gaussian filter is of limited use for GCM data; it assumes a regular and rectangular Cartesian grid, employs a simple boundary condition, and the definitions of filter scale and shape have little or no flexibility. The python package GCM-Filters is specifically designed to filter GCM data, and seeks to solve a number of challenges for the user:

1. GCM data comes on irregular curvilinear grids with spatially varying grid-cell geometry.
2. Continental boundaries require careful and special treatment when filtering ocean GCM output.
3. Earth Science applications benefit from configurable filters, where the definition of filter scale and shape is flexible.
4. GCM output is often too large to process in memory, requiring distributed and / or delayed execution.

The GCM-Filters algorithm (Grooms et al., 2021) applies a discrete Laplacian to smooth a field through an iterative process that resembles diffusion. The discrete Laplacian takes into account the varying grid-cell geometry and uses a no-flux boundary condition, mimicking how diffusion is internally implemented in GCMs. The no-flux boundary conditions ensures that the filter preserves the integral: $\int_{\Omega} \bar{f}(x, y) dA = \int_{\Omega} f(x, y) dA$, where f is the original field, \bar{f} the filtered field, and Ω the ocean domain. Conservation of the integral is a desirable filter property for many physical quantities, for example energy or ocean salinity. More details on the filter properties can be found in Grooms et al. (2021).

An important goal of GCM-Filters is to enable computationally efficient filtering. The user can employ GCM-Filters on either CPUs or GPUs, with NumPy (Harris et al., 2020) or CuPy (Okuta et al., 2017) input data. GCM-Filters leverages Dask (Dask Development Team, 2016) and Xarray (Hoyer & Hamman, 2017) to support filtering of larger-than-memory datasets and computational flexibility.

Usage

The main GCM-Filters class that the user will interface with is the `gcm_filters.Filter` object. When creating a filter object, the user specifies how they want to smooth their data, including the desired filter shape and filter scale. At this stage, the user also picks the grid type that matches their GCM data, given a predefined list of grid types. Each grid type has an associated discrete Laplacian, and requires different *grid variables* that the user must provide (the latter are usually available to the user as part of the GCM output). Currently, GCM-Filters provides a number of different grid types and associated discrete Laplacians:

- Grid types with **scalar Laplacians** that can be used for filtering scalar fields, for example temperature or vorticity (see [Figure 1](#)). The currently implemented grid types are compatible with different ocean GCM grids including MOM5 (MOM 5 Development Team, 2012), MOM6 (Adcroft et al., 2019) and the POP2 (Smith et al., 2010) tripole grid.
- Grid types with **vector Laplacians** that can be used for filtering vector fields, for example horizontal velocity (u, v). The currently implemented grid type is compatible with ocean GCM grids that use an Arakawa C-grid convention; examples include MOM6 (Adcroft et al., 2019) and the MITgcm (Campin et al., 2021).

Atmospheric model grids are not yet supported, but could be implemented within the GCM-Filters package. Users are encouraged to contribute more grid types and Laplacians via pull requests. While we are excited to share GCM-Filters in its current beta state at version 0.1.3, we plan to continue improving and maintaining the package for the long run and welcome new contributors from the broader community.

Acknowledgements

This work was supported by the National Science Foundation grants OCE 1912302, OCE 1912325, OCE 1912332, OCE 1912420, GEO 1912357, and the NOAA grant CVP NA19OAR4310364. Busecke received support from the Gordon and Betty Moore Foundation.

References

- Adcroft, A., Anderson, W., Balaji, V., Blanton, C., Bushuk, M., Dufour, C. O., Dunne, J. P., Griffies, S. M., Hallberg, R., Harrison, M. J., Held, I. M., Jansen, M. F., John, J. G., Krasting, J. P., Langenhorst, A. R., Legg, S., Liang, Z., McHugh, C., Radhakrishnan, A., ... Zhang, R. (2019). The GFDL global ocean and sea ice model OM4.0: Model description and simulation features. *Journal of Advances in Modeling Earth Systems*, 11(10), 3167–3211. <https://doi.org/https://doi.org/10.1029/2019MS001726>
- Campin, J.-M., Heimbach, P., Losch, M., Forget, G., edhill3, Adcroft, A., amolod, Menemenlis, D., dfer22, Hill, C., Jahn, O., Scott, J., stephdut, Mazloff, M., Fox-Kemper, B., antnguyen13, Doddridge, E., Fenty, I., Bates, M., ... dussin, raphael. (2021). *MITgcm/MITgcm: checkpoint67z* (Version checkpoint67z) [Computer software]. Zenodo. <https://doi.org/10.5281/zenodo.4968496>
- Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. <https://dask.org>
- Grooms, I., Loose, N., Abernathey, R., Steinberg, J. M., Bachman, S. D., Marques, G., Guillaumin, A. P., & Yankovsky, E. (2021). Diffusion-Based Smoothers for Spatial Filtering

- 103 of Gridded Geophysical Data. *Journal of Advances in Modeling Earth Systems*, 13(9),
104 e2021MS002552. <https://doi.org/10.1029/2021MS002552>
- 105 Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau,
106 D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk,
107 M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., ... Oliphant,
108 T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
109
- 110 Hoyer, S., & Hamman, J. (2017). Xarray: ND labeled arrays and datasets in python. *Journal*
111 *of Open Research Software*, 5(1). <https://doi.org/10.5334/jors.148>
- 112 Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science &*
113 *Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 114 Met Office. (2010 - 2015). *Cartopy: A cartographic python library with a matplotlib interface*.
115 <http://scitools.org.uk/cartopy>
- 116 MOM 5 Development Team. (2012). *MOM 5: The modular ocean model*. <https://github.com/mom-ocean/MOM5>
117
- 118 Okuta, R., Unno, Y., Nishino, D., Hido, S., & Loomis, C. (2017). CuPy: A NumPy-compatible
119 library for NVIDIA GPU calculations. *Proceedings of Workshop on Machine Learning Sys-*
120 *tems (LearningSys) in the Thirty-First Annual Conference on Neural Information Process-*
121 *ing Systems (NIPS)*. http://learningsys.org/nips17/assets/papers/paper_16.pdf
- 122 Smith, R., Jones, P., Briegleb, B., Bryan, F., Danabasoglu, G., Dennis, J., Dukowicz, J., Eden,
123 C., Fox-Kemper, B., Gent, P., Hecht, M., Jayne, S., Jochum, M., Large, W., Lindsay,
124 M., K., Norton, N., Peacock, S., Vertenstein, M., & Yeager, S. (2010). *The parallel*
125 *ocean program (POP) reference manual*. [http://www.cesm.ucar.edu/models/cesm1.0/](http://www.cesm.ucar.edu/models/cesm1.0/pop2/doc/sci/POPRefManual.pdf)
126 [pop2/doc/sci/POPRefManual.pdf](http://www.cesm.ucar.edu/models/cesm1.0/pop2/doc/sci/POPRefManual.pdf)
- 127 Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D.,
128 Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M.,
129 Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson,
130 E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scien-
131 tific Computing in Python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>
132