

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

Nora Loose¹, Ryan Abernathey², Ian Grooms¹, Julius Busecke², Arthur Barthé³, Elizabeth Yankovsky³, Gustavo Marques⁴, Jacob Steinberg⁵, Andrew 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 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.

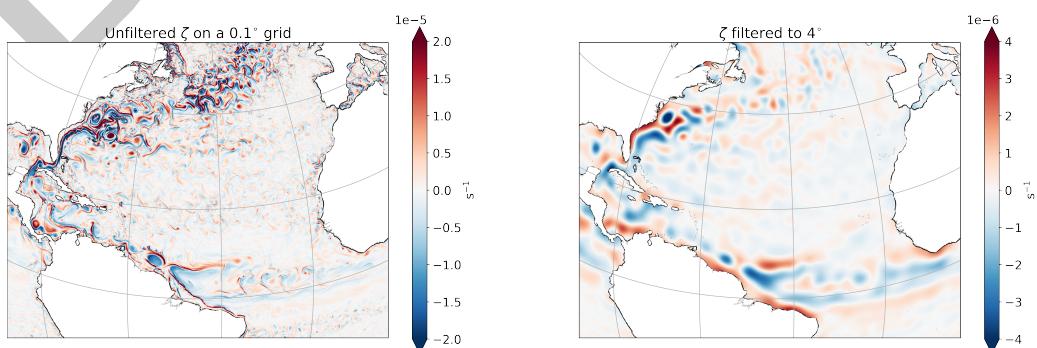


Figure 1: (Left) Snapshot of unfiltered surface relative vorticity $\zeta = \partial_x v - \partial_y u$ from a global 0.1° simulation with MOM6 (Adcroft et al., 2019). (Right) Relative vorticity filtered to 4°, obtained by applying GCM-Filters to the field ζ on the left. The plots are made with matplotlib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

30 Statement of Need

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

- 38
- 39 1. GCM data comes on irregular curvilinear grids with spatially varying grid-cell geometry.
 - 40 2. Continental boundaries require careful and special treatment when filtering ocean GCM
41 output.
 - 42 3. Earth Science applications benefit from configurable filters, where the definition of filter
43 scale and shape is flexible.
 - 44 4. GCM output is often too large to process in memory, requiring distributed and / or
45 delayed execution.

46 The `GCM-Filters` algorithm (Grooms et al., 2021) applies a discrete Laplacian to smooth a field
47 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
48 diffusion is internally implemented in GCMs. The no-flux boundary conditions ensures that
49 the filter preserves the integral: $\int_{\Omega} \bar{f}(x, y) dA = \int_{\Omega} f(x, y) dA$, where f is the original field,
50 \bar{f} the filtered field, and Ω the ocean domain. Conservation of the integral is a desirable filter
51 property for many physical quantities, for example energy or ocean salinity. More details on
52 the filter properties can be found in Grooms et al. (2021).

53

54 An important goal of `GCM-Filters` is to enable computationally efficient filtering. The user
55 can employ `GCM-Filters` on either CPUs or GPUs, with NumPy (Harris et al., 2020) or CuPy
56 (Okuta et al., 2017) input data. `GCM-Filters` leverages Dask (Dask Development Team,
57 2016) and Xarray (Hoyer & Hamman, 2017) to support filtering of larger-than-memory
58 datasets and computational flexibility.

59 Usage

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

- 67
- 68 ▪ Grid types with **scalar Laplacians** that can be used for filtering scalar fields, for example
69 temperature or vorticity (see [Figure 1](#)). The currently implemented grid types are
70 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
71 grid.
 - 72 ▪ Grid types with **vector Laplacians** that can be used for filtering vector fields, for example
73 horizontal velocity (u, v). The currently implemented grid type is compatible with ocean
74 GCM grids that use an Arakawa C-grid convention; examples include MOM6 (Adcroft
75 et al., 2019) and the MITgcm (Campin et al., 2021).

76 Atmospheric model grids are not yet supported, but could be implemented within the GCM-F
77 ilters package. Users are encouraged to contribute more grid types and Laplacians via pull
78 requests.

79 Acknowledgements

80 This work was supported by the National Science Foundation grants OCE 1912302,
81 OCE 1912325, OCE 1912332, OCE 1912420, GEO 1912357, and the NOAA grant CVP
82 NA19OAR4310364.

83 References

- 84 Adcroft, A., Anderson, W., Balaji, V., Blanton, C., Bushuk, M., Dufour, C. O., Dunne, J.
85 P., Griffies, S. M., Hallberg, R., Harrison, M. J., Held, I. M., Jansen, M. F., John, J.
86 G., Krasting, J. P., Langenhorst, A. R., Legg, S., Liang, Z., McHugh, C., Radhakrishnan,
87 A., ... Zhang, R. (2019). The GFDL global ocean and sea ice model OM4.0: Model
88 description and simulation features. *Journal of Advances in Modeling Earth Systems*,
89 11(10), 3167–3211. <https://doi.org/https://doi.org/10.1029/2019MS001726>
- 90 Campin, J.-M., Heimbach, P., Losch, M., Forget, G., edhill3, Adcroft, A., amolod, Men-
91 emenlis, D., dfer22, Hill, C., Jahn, O., Scott, J., stephdut, Mazloff, M., Fox-Kemper,
92 B., antnguyen13, Doddridge, E., Fenty, I., Bates, M., ... dussin, raphael. (2021). *MIT-*
93 *gcm/MITgcm: checkpoint67z* (Version checkpoint67z) [Computer software]. Zenodo.
94 <https://doi.org/10.5281/zenodo.4968496>
- 95 Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. <https://dask.org>
- 96 Grooms, I., Loose, N., Abernathey, R., Steinberg, J. M., Bachman, S. D., Marques, G.,
97 Guillaumin, A. P., & Yankovsky, E. (2021). Diffusion-Based Smoothers for Spatial Filtering
98 of Gridded Geophysical Data. *Journal of Advances in Modeling Earth Systems*, 13(9),
99 e2021MS002552. <https://doi.org/10.1029/2021MS002552>
- 100 Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau,
101 D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk,
102 M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., ... Oliphant,
103 T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- 104 Hoyer, S., & Hamman, J. (2017). Xarray: ND labeled arrays and datasets in python. *Journal*
105 *of Open Research Software*, 5(1). <https://doi.org/10.5334/jors.148>
- 106 Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science &*
107 *Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 108 Met Office. (2010 - 2015). *Cartopy: A cartographic python library with a matplotlib interface*.
109 <http://scitools.org.uk/cartopy>
- 110 MOM 5 Development Team. (2012). *MOM 5: The modular ocean model*. <https://github.com/mom-ocean/MOM5>
- 111 Okuta, R., Unno, Y., Nishino, D., Hido, S., & Loomis, C. (2017). CuPy: A NumPy-compatible
112 library for NVIDIA GPU calculations. *Proceedings of Workshop on Machine Learning Sys-*
113 *tems (LearningSys) in the Thirty-First Annual Conference on Neural Information Process-*
114 *ing Systems (NIPS)*. http://learningsys.org/nips17/assets/papers/paper_16.pdf

- 118 Smith, R., Jones, P., Briegleb, B., Bryan, F., Danabasoglu, G., Dennis, J., Dukowicz, J., Eden,
119 C., Fox-Kemper, B., Gent, P., Hecht, M., Jayne, S., Jochum, M., Large, W., Lindsay,
120 M., K., Norton, N., Peacock, S., Vertenstein, M., & Yeager, S. (2010). *The parallel*
121 *ocean program (POP) reference manual.* [http://www.cesm.ucar.edu/models/cesm1.0/
122 pop2/doc/sci/POPRefManual.pdf](http://www.cesm.ucar.edu/models/cesm1.0/pop2/doc/sci/POPRefManual.pdf)
- 123 Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D.,
124 Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M.,
125 Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson,
126 E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scien-
127 tific Computing in Python. *Nature Methods*, 17, 261–272. [https://doi.org/10.1038/
128 s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2)

DRAFT