

# <sup>1</sup> GCM-Filters: A Python package for Diffusion-based Spatial Filtering of Gridded Data from General Circulation Models

<sup>4</sup> TBD<sup>1</sup>

<sup>5</sup> 1 TBD

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

## Software

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

## <sup>6</sup> Summary

<sup>7</sup> GCM-Filters is a python package that allows scientists to perform spatial filtering analysis  
<sup>8</sup> in an easy, flexible and efficient way. The package implements the filtering method that was  
<sup>9</sup> introduced by Grooms et al. (2021). The filtering algorithm is analogous to smoothing via  
<sup>10</sup> diffusion; hence the name *diffusion-based filters*. GCM-Filters is designed to work with  
<sup>11</sup> gridded data that is produced by General Circulation Models (GCMs) of ocean, weather, and  
<sup>12</sup> climate. Spatial filtering of GCM data is a common analysis method in the Earth Sciences,  
<sup>13</sup> for example to study oceanic and atmospheric motions at different spatial scales or to develop  
<sup>14</sup> subgrid-scale parameterizations for ocean models.

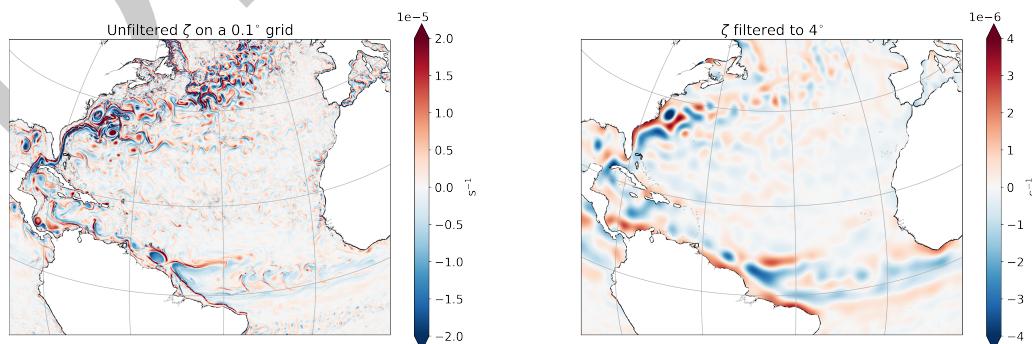
**Editor:** [Editor Name](#)

**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](#)).

<sup>15</sup> GCM-Filters provides filters that are highly configurable, with the goal to be useful for a wide  
<sup>16</sup> range of scientific applications. The user has different options for selecting the filter scale and  
<sup>17</sup> filter shape. The filter scale can be defined in several ways: a fixed length scale (e.g., 100  
<sup>18</sup> km), a scale tied to a model grid scale (e.g., 1°), or a scale tied to a varying dynamical scale  
<sup>19</sup> (e.g., the Rossby radius of deformation). As an example, Figure 1 shows unfiltered and filtered  
<sup>20</sup> relative vorticity, where the filter scale is set to a model grid scale of 4°. GCM-Filters also  
<sup>21</sup> allows for anisotropic, i.e., direction-dependent, filtering. Finally, the filter shape – currently:  
<sup>22</sup> either Gaussian or Taper – determines how sharply the filter separates scales above and below  
<sup>23</sup> 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  $\zeta$  on the left. The plots are made with `matplotlib` (Hunter, 2007) and `cartopy` (Met Office, 2010 - 2015).

## 24 Statement of Need

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

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

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

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

- 55 ▪ Grid types with **scalar Laplacians** that can be used for filtering scalar fields, for example  
56 temperature or vorticity (see Figure 1). The currently implemented grid types are  
57 compatible with different ocean GCM grids including MOM5 (MOM 5 Development  
58 Team, n.d.), MOM6 (Adcroft et al., 2019) and the POP2 (POP2-CESM Development  
59 Team, n.d.) tripole grid.
- 60 ▪ Grid types with **vector Laplacians** that can be used for filtering vector fields, for example  
61 horizontal velocity ( $u, v$ ). The currently implemented grid type is compatible with ocean  
62 GCM grids that use an Arakawa C-grid convention; examples include MOM6 (Adcroft  
63 et al., 2019) and the MITgcm (Campin et al., 2021).

64 Users are encouraged to contribute more grid types and Laplacians via pull requests.

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

<sup>70</sup> **Acknowledgements**

<sup>71</sup> **References**

- <sup>72</sup> 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](https://doi.org/10.1029/2019MS001726)
- <sup>73</sup> 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). *MIT-gcm/MITgcm: checkpoint67z* (Version checkpoint67z) [Computer software]. Zenodo. <https://doi.org/10.5281/zenodo.4968496>
- <sup>74</sup> Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. <https://dask.org>
- <sup>75</sup> Grooms, I., Loose, N., Abernathey, R., Steinberg, J., Bachman, S. D., Marques, G., Guillaumin, A. P., & Yankovsky, E. (2021). Diffusion-based smoothers for spatial filtering of gridded geophysical data. *Earth and Space Science Open Archive*, 27. <https://doi.org/10.1002/essoar.10506591.1>
- <sup>76</sup> Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, 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>
- <sup>77</sup> Hoyer, S., & Hamman, J. (2017). Xarray: ND labeled arrays and datasets in python. *Journal of Open Research Software*, 5(1). <https://doi.org/10.5334/jors.148>
- <sup>78</sup> Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- <sup>79</sup> Met Office. (2010 - 2015). *Cartopy: A cartographic python library with a matplotlib interface*. <http://scitools.org.uk/cartopy>
- <sup>80</sup> MOM 5 Development Team. (n.d.). *MOM 5: The modular ocean model*. <https://github.com/mom-ocean/MOM5>
- <sup>81</sup> Okuta, R., Unno, Y., Nishino, D., Hido, S., & Loomis, C. (2017). CuPy: A NumPy-compatible library for NVIDIA GPU calculations. *Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Thirty-First Annual Conference on Neural Information Processing Systems (NIPS)*. [http://learningsys.org/nips17/assets/papers/paper\\_16.pdf](http://learningsys.org/nips17/assets/papers/paper_16.pdf)
- <sup>82</sup> POP2-CESM Development Team. (n.d.). *POP2-CESM*. <https://github.com/ESCOMP/POP2-CESM>
- <sup>83</sup> 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, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>