Diffusion-based smoothers for spatial filtering of gridded geophysical data

I. Grooms[1], N. Loose[1], R. Abernathey[2], J. M. Steinberg[3], S. D. Bachman[4], G. Marques[4], A. P. Guillaumin[5], E. Yankovsky[5], and L. Zanna[5]

[1]Department of Applied Mathematics, University of Colorado, Boulder, Colorado, USA
[2]Lamont Doherty Earth Observatory of Columbia University, Palisades, New York, USA
[3]Woods Hole Oceanographic Institution, Woods Hole, Massachusetts, USA
[4]Climate and Global Dynamics Division, National Center for Atmospheric Research, Boulder, Colorado, USA
[5]Courant Institute of Mathematical Sciences, New York University, New York, New York, USA

Key Points:

- A new way to apply a spatial low-pass filter to gridded data is developed
- The new method can be applied in any geometry since it only requires a discrete Laplacian operator
- ~~A Python package implementing the new method is developed~~ We develop an open-source python package that implements the method with support for CPU and GPU

Corresponding author: Ian Grooms, `ian.grooms@colorado.edu`

Abstract

We describe a new way to apply a spatial filter to gridded data from models or observations, focusing on low-pass filters. The new method is analogous to smoothing via diffusion, and its implementation requires only a discrete Laplacian operator appropriate to the data. The new method can approximate arbitrary filter shapes, including Gaussian and boxcar filters, and can be extended to spatially-varying and anisotropic filters. The new diffusion-based smoother's properties are illustrated with examples from ocean model data and ocean observational products, and its speed is compared with similar methods from the literature. The new method performs efficiently, especially when implemented on graphics processing units (GPUs). ~~An open source Python package is developed that implements the method~~A open-source python package implementing this algorithm, called gcm-filters, is currently under development.

Plain Language Summary

"The large scale part" and "the small scale part" of ~~things~~ quantities like velocity, temperature, and pressure fluctuations are important for a range of questions in Earth system science. This paper describes a precise way of defining these quantities, as well as an efficient method for diagnosing them from gridded data, especially the data produced by Earth system models.

1 Introduction

Spatial scale is an organizing concept in Earth system science: ~~Atmospheric~~ atmospheric synoptic scales and convective scales, and oceanic mesoscales and submesoscales, for example, are ubiquitous touchstones in atmospheric and oceanic dynamics. The ~~nearly-inescapable~~ pervasive idea of an energy spectrum is fundamentally based on the idea of partitioning energy (or variance) across a range of spatial scales. ~~Nevertheless, diagnosing the~~ Despite this central importance, diagnosing dynamics at different spatial scales ~~often relies~~ remains challenging. When analysing remote-sensing or simulation data, scientists instead often rely on time averaging as ~~a proxy because it is far easier~~ proxy for separating scales, which is more computationally convenient than spatial filtering~~;~~. Temporal filtering is often of interest in its own right, but in situations where spatial filtering is called for this trade of spatial for temporal filtering ~~is enabled~~ can be justified by the fact that dynamics at different spatial scales are frequently also associated with different time scales.

Spatial filtering has recently begun to replace time averages and zonal averages in a priori studies of subgrid-scale parameterization for ocean models. ~~For example, Nadiga (2008) and Grooms et al. (2013) used an elliptic inversion to define spatial filters in rectangular Cartesian domains.Grooms and Kleiber (2019) used Fourier-based filtering methods for primitive equation model output in a Cartesian rectangular domain. The foregoing examples are somewhat atypical in that they don't explicitly use a spatial convolution filter; several recent investigations have used convolutions to define the filter. For example,~~ A canonical model for spatial filtering is given by kernel convolution

$$\bar{f}(\mathbf{x}) = \int_{\mathbb{R}^d} G(\mathbf{x}-\mathbf{x}')f(\mathbf{x}')\mathrm{d}\mathbf{x}', \tag{1}$$

where G is the convolution kernel, $\mathbf{x}'$ is a dummy integration variable, and $\mathbb{R}^d$ denotes the set of all real vectors of length d. Berloff (2018), Bolton and Zanna (2019), Ryzhov et al. (2019), and Haigh et al. (2020) all used convolution filters to study subgrid-scale parameterization in the context of quasigeostrophic dynamics in a rectangular Cartesian domain. Lu et al. (2016), Aluie et al. (2018), Khani et al. (2019), ~~and~~

~~Stanley, Bachman, and Grooms (2020)~~Stanley, Bachman, and Grooms (2020), and Guillaumin and Zanna (2021) used approximate spatial convolutions on the sphere to filter ocean general circulation model output, and Aluie (2019) showed how to correctly define convolution on the sphere in such a way that the filter commutes with spatial derivatives. A 'top hat' or 'boxcar' kernel (i.e. an indicator function over a circle or a square, respectively) is used in all these studies, except for Bolton and Zanna (2019)~~and Stanley, Bachman, and Grooms (2020),~~ , Stanley, Bachman, and Grooms (2020), and Guillaumin and Zanna (2021) who used Gaussian kernels. Spatial convolution is not the only way to define or implement spatial filters. For example, Nadiga (2008) and Grooms et al. (2013) used an elliptic inversion to define spatial filters for quasigeostrophic model output, and Grooms and Kleiber (2019) used Fourier-based filtering methods for primitive equation model output, all in rectangular Cartesian domains.

We make a semantic distinction between spatial filtering and coarse graining. In our use of the terms, coarse graining is an operation that produces output at a lower resolution (i.e. smaller number of grid points) than the input, whereas spatial filtering produces output at the same resolution as the input. (Note that this terminology is not uniformly adopted in the literature; cf. ~~(Aluie et al., 2018)~~Aluie et al. (2018).) Berloff (2005), Porta Mana and Zanna (2014), Williams et al. (2016), Stanley, Grooms, et al. (2020), and Zanna and Bolton (2020) are all examples where coarse graining was used in the context of ocean model subgrid-scale parameterization. The term 'averaging' is sometimes used instead of filtering. They are essentially synonymous when the filter kernel ~~has~~ G is non-negative~~weights~~, but a filter whose kernel has negative ~~weights~~ values cannot be described as an average, so we opt to use the more general term. A low-pass filter can be described as a smoother, which is the focus here, but the methods described here can be straightforwardly adapted to band-pass or high-pass filters.

This paper introduces a new way of designing and implementing spatial filters that relies only on a discrete Laplacian operator for the data. Because ~~they rely~~ it relies on the discrete Laplacian to smooth a field through an iterative process reminiscent of diffusion, we refer to the new ~~methods~~ method as diffusion-based filters. The ~~new filters provide an efficient way of implementing something close to a Gaussian kernel convolution; they also allow the scale selectiveness (i.e. the shape) of the filter to be tuned as desired. Because they require only the ability to apply a discrete Laplacian operator, these filters can be used with a wide range of data types, including output from models on unstructured grids. The new filters are computationally efficient, with comparable speed to similar methods implemented in Fortran or in widely-used Python packages, and are extremely fast on graphics processing units (GPUs). In the presence of boundaries, and when the filter scale varies over the domain, the new filters do not commute with spatial derivatives. In domains without boundaries, the new filters will commute with any spatial derivative that commutes with the Laplacian, at least provided that the filter length scale does not change over the domain. If desired, ocean boundaries can be eliminated by treating values on land as zero, following (Aluie et al., 2018).~~

~~The~~ paper is structured as follows. Section 2 describes the new filters along with their properties. Examples using model data and observations are provided in section 3 to illustrate the various filter properties described in section 2. ~~Some speed tests are performed in section ??, and conclusions~~ Conclusions are offered in section 4.

2 Spatial filtering of gridded data

2.1 Review

Spatial filtering of gridded data is a well developed field, both for general applications and in the context of geophysical data. The focus here is on filtering in the context of fluid models, especially atmosphere and ocean models. ~~Shapiro filters(Shapiro, 1970) were~~ To place our new method into context, we review existing filtering techniques, and distinguish between implicit and explicit filters.

Shapiro (1970) introduced a class of filters, widely used to improve the performance of early finite-difference weather models. Shapiro filters are essentially discrete spatial convolution filters optimized to remove the smallest scales that can be represented on a grid, while leaving the other scales as close to unchanged as possible. Sagaut and Grohens (1999) reviewed some of the more recent approaches to convolution-based filtering for large-eddy simulation (LES). Sadek and Aluie (2018) developed two discrete convolution kernels for the purpose of accurately extracting the energy spectrum using convolution filters rather than Fourier methods.

Germano (1986) introduced an implicit differential filter of the form

$$(1 - L^2\Delta)\bar{f} = f, \tag{2}$$

where $\bar{f}$ is the filtered field, L is the filter length scale, and Δ is the Laplacian. It is 'implicit' because applying the filter to data involves solving a system of equations. This is the kind of spatial filter used by Nadiga (2008) and Grooms et al. (2013) in the context of subgrid-scale parameterization in quasigeostrophic ocean models, and ~~this same~~ a similar fractional elliptic equation underlies the approach to spatial filtering of scattered data recently developed by Robinson and Grooms (2020). ~~Sagaut and Grohens (1999) review some explicit filters for large-eddy simulation (LES).~~ Raymond (1988) and Raymond and Garder (1991) ~~develop~~ developed implicit filters for meteorological applications using higher order differential operators. Guedot et al. (2015) developed higher order implicit differential filters on unstructured meshes for engineering applications. ~~Sadek and Aluie (2018) develop two discrete convolution type filters for the purpose of accurately extracting the energy spectrum using filters rather than Fourier methods. The new approach developed here results in high order explicit differential filters.~~ Note that the term 'high order' here refers to the differential operator, though it has been used elsewhere with different meanings (Sagaut & Grohens, 1999; Sadek & Aluie, 2018).

The new approach developed here results in high order explicit differential filters, meaning that they use a discrete Laplacian, but that they do not require solving a system of equations.

2.2 Spatial filtering basics

Most intuition about spatial filtering and spatial scales is built on the foundation of kernel convolution and Fourier analysis~~. To wit, one can define a spatial filter as follows~~

$$\bar{f}(\mathbf{x}) = \int_{\mathbb{R}^d} G(\mathbf{x} - \mathbf{x}')f(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

~~where G is the convolution kernel and $\mathbf{x}'$ is a dummy integration variable.~~ , in the context of equation (1). The well-known convolution theorem (e.g. Hunter & Nachtergaele, 2001, Theorem 11.35) states that the Fourier transform of $\bar{f}$ is proportional to $\hat{G}\hat{f}$, where $\hat{\cdot}$ denotes the Fourier transform and the proportionality constant depends on the dimension d and on the normalization convention chosen in the definition of the Fourier transform.

Fourier analysis enables us to understand the effect of spatial convolution filtering in terms of length scales. We consider the function f to be a sum of many Fourier modes, each of which has a distinct spatial scale. The Fourier transform of the kernel, $\hat{G}$, then describes how each Fourier mode is modified by the spatial filtering operation. Filter kernels are usually symmetric about the origin, which makes $\hat{G}$ real-valued, so that spatial filtering only changes the amplitude of the Fourier modes and not their phase. If $\hat{G}(k) = 1$ for a particular Fourier mode then the corresponding length scale is left unchanged in $\bar{f}$, whereas if $\hat{G}(k) = 0$ for a particular Fourier mode then the corresponding length scale is removed from $\bar{f}$. By modifying the amplitudes of the Fourier modes, spatial filtering controls the scale content of $\bar{f}$.

~~In~~ One of the simplest kernels is the so-called boxcar function, defined in one spatial dimension ~~the~~ as

$$G_L(x) = \begin{cases} 1/L & |x| < L/2 \\ 0 & |x| > L/2 \end{cases} \tag{3}$$

This represents averaging all the points in the neighborhood with the same weight, and the parameter L defines the size of the neighborhood. (In higher dimensions the boxcar filter is nonzero over a square region, while a 'top-hat' filter is nonzero over a circular or spherical region.) The Fourier transform of ~~a~~ the boxcar filter of width L is

$$\hat{G}_L(k) = \mathrm{sinc}\left(\frac{kL}{2\pi}\right) \tag{4}$$

where $\mathrm{sinc}(x) = \sin(\pi x)/(\pi x)$ and k is the wavenumber. This function decays only as $1/k$ at large k, so it does not correspond to a sharp separation between length scales. Conversely, a 'spectral truncation' filter has a kernel whose Fourier transform is a boxcar, and ~~whose kernel~~ the kernel itself is a sinc function. The boxcar and spectral truncation filters illustrate the concept that a short-range kernel does not separate scales well, and a filter that makes a sharp separation between scales requires a very long-range kernel. Figure 1 shows the boxcar and sinc convolution kernels, to illustrate that the more scale-selective sinc kernel has a much longer range. In practice there is a tradeoff between choosing a kernel that makes as clean a scale separation as possible and choosing a kernel whose range is short enough to apply efficiently.

It is usually desirable for the filter to preserve the integral, and to commute with derivatives, i.e.

$$\int_{\mathbb{R}^d} f(\mathbf{x})\mathrm{d}\mathbf{x} = \int_{\mathbb{R}^d} \bar{f}(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{5}$$

$$\frac{\partial \bar{f}}{\partial x_i} = \overline{\frac{\partial f}{\partial x_i}}. \tag{6}$$

Any convolution filter commutes with derivatives, and preservation of the integral is easily ensured by the condition

$$\int_{\mathbb{R}^d} G(\mathbf{x})\mathrm{d}\mathbf{x} = 1. \tag{7}$$

In the presence of boundaries the convolution formula (1) no longer works, since $f(\mathbf{x})$ is not defined on $\mathbb{R}^d$. One option, used by Aluie et al. (2018), is to simply set $f(\mathbf{x}) = 0$ outside the domain boundaries. The more common option is to vary the kernel near the boundaries so that the filter formula changes to

$$\bar{f}(\mathbf{x}) = \int_\Omega G(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')\mathrm{d}\mathbf{x}', \tag{8}$$

where $\Omega \subset \mathbb{R}^d$ is the spatial domain. This kind of spatial filter no longer commutes with spatial derivatives, though it still preserves the integral as long as the kernel is appropriately normalized.

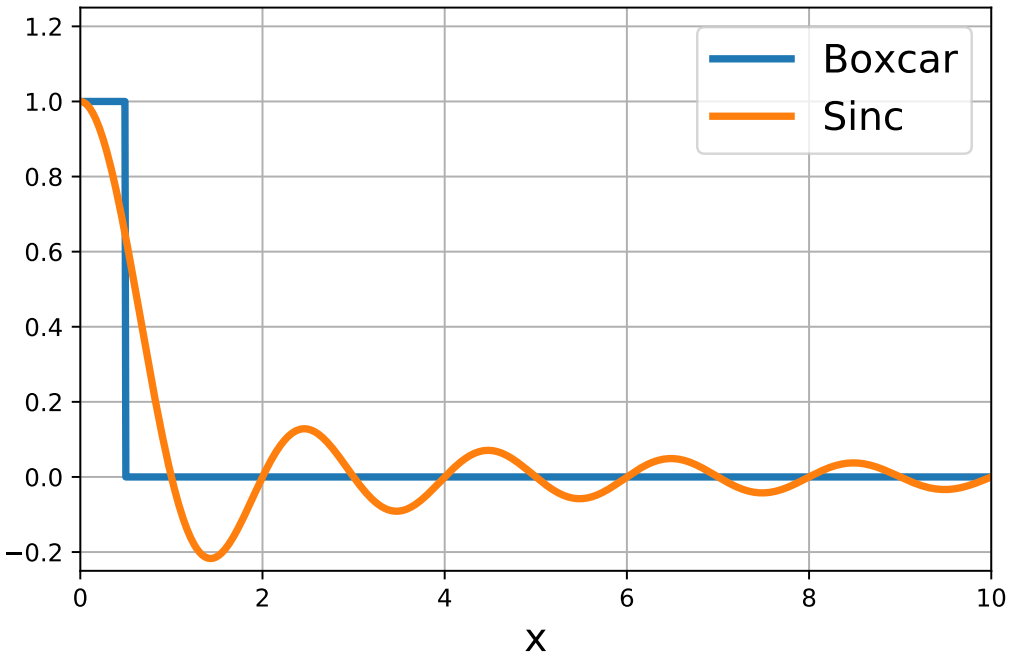

Figure 1. The boxcar function of width 1 and sinc(x).

The background intuition for kernel-based spatial filters in this subsection is developed entirely for functions on Euclidean spaces. The ~~situation~~ definition of convolution-based spatial filters is considerably more complicated on a sphere; see (Aluie, 2019) for details.

2.3 Diffusion-based smoothers

2.3.1 Discrete integral & Laplacian

To generalize the foregoing ideas to ~~complex domain~~ more complicated domains and grid geometries we begin with a transition to the discrete representation. The field to be filtered is no longer a continuous function, but a vector $\mathbf{f}$; for example, if we wish to filter temperature on a grid of n points, then we think of the values of temperature on the grid as a vector in $\mathbb{R}^n$. To lay a foundation for the analysis we need two ingredients; the first is a discrete integral

$$\int_\Omega f(\mathbf{x})\mathrm{d}\mathbf{x} \approx \sum_i w_i f_i, \tag{9}$$

where Ω denotes the spatial domain and w_i are positive weights. Cartesian geometry is assumed for ease of presentation, but the discrete integral could easily approximate an integral over the sphere or some other smooth manifold without changing the analysis. For a typical finite-volume model the weight w_i will simply be the area (or volume, if the integral is over ~~volume~~three spatial dimensions) of the i^{th} grid cell. If the weights w_i are all positive then we can define a discrete inner product

$$\langle \mathbf{f}, \mathbf{g} \rangle = \sum_i w_i f_i g_i. \tag{10}$$

The area integral can be expressed in terms of the inner product as $\langle \mathbf{1}, \mathbf{f} \rangle$, where $\mathbf{1}$ is a vector whose entries are all 1.

The second ingredient is a discrete Laplacian, i.e. some operation on $\mathbf{f}$ that produces an approximation of Δf on the grid. We write this operation in matrix form as $\mathbf{Lf}$, though it is certainly not necessary to actually construct the matrix $\mathbf{L}$. We assume that the discrete Laplacian is negative semi-definite, and self-adjoint with respect to the discrete inner product, i.e for any $\mathbf{f}$ and $\mathbf{g}$

$$\langle \mathbf{f}, \mathbf{Lf} \rangle \leq 0, \text{ and } \langle \mathbf{f}, \mathbf{Lg} \rangle = \langle \mathbf{Lf}, \mathbf{g} \rangle. \tag{11}$$

This is automatically guaranteed for finite-volume discretizations of the Laplacian with no-flux boundary conditions.

2.3.2 Connecting the discrete Laplacian to spatial scales

Since the discrete Laplacian is self-adjoint and negative semi-definite, the eigenvalues of $\mathbf{L}$ are all real and non-positive, and there is an eigenvector basis $\mathbf{q}_1, \ldots, \mathbf{q}_n$ of $\mathbb{R}^n$ that is ~~orthogonal~~ orthonormal with respect to the discrete inner product. This is directly analogous to the Fourier analysis of the foregoing section: Fourier modes on $\mathbb{R}^d$ are eigenfunctions of the Laplacian on $\mathbb{R}^d$. In fact, with an equispaced grid and periodic boundaries the eigenvectors $\mathbf{q}_i$ are exactly the discrete Fourier modes. In both the Fourier version and the discrete version the eigenvalues can be interpreted as describing the spatial scale of the corresponding eigenfunction:

$$\Delta e^{\mathbf{i}\mathbf{k}\cdot\mathbf{x}} = -k^2 e^{\mathbf{i}\mathbf{k}\cdot\mathbf{x}}, \quad \mathbf{L}\mathbf{q}_i = -k_i^2 \mathbf{q}_i. \tag{12}$$

On the left in the above expression $k = \|\mathbf{k}\|$ represents the familiar Fourier wavenumber corresponding to a wavelength of $2\pi/k$, while on the right the eigenvalue $-k_i^2$ has been written with similar notation to emphasize the similarity. Precisely because $\mathbf{L}$ is a discretization of the Laplacian, the length $2\pi/k_i$ should roughly correspond to the length scale of the eigenvector $\mathbf{q}_i$.

Continuing the analogy with the previous section, it is possible to write the vector to be filtered as a sum over eigenfunctions of the discrete Laplacian:

$$\mathbf{f} = \sum_{i=1}^{n} \hat{f}_i \mathbf{q}_i. \tag{13}$$

~~Applying~~ We next show that we can filter $\mathbf{f}$ by applying a function $p(-\mathbf{L})$ to ~~both sides~~ it. From equation (13), we see that this results in

$$p(-\mathbf{L})\mathbf{f} = \sum_{i=1}^{n} \hat{f}_i p(k_i^2) \mathbf{q}_i = \sum_{i=1}^{n} \hat{f}_i \hat{G}(k_i) \mathbf{q}_i, \tag{14}$$

where the notation $\hat{G}(k) = p(k^2)$ has been deliberately used to emphasize the connection to the Fourier convolution theorem recalled in the previous section: ~~If~~ if the expansion coefficients of $\mathbf{f}$ are $\hat{f}_i$, then the expansion coefficients of $p(-\mathbf{L})\mathbf{f}$ are $\hat{G}(k_i)\hat{f}_i$. (The notation p is used for both the matrix and ~~a~~ scalar versions of the function; a familiar example might be $p(-\mathbf{L}t) = e^{-\mathbf{L}t}$ and $p(0) = e^0 = 1$.) If one defined the function p in such a way that

$$\hat{G}(k) = \begin{cases} 1 & k < k_* \\ 0 & k \geq k_* \end{cases}, \tag{15}$$

then multiplying $\mathbf{f}$ by $p(-\mathbf{L})$ would correspond to projecting $\mathbf{f}$ onto large-scale modes defined by $k_i < k_*$. This would be analogous to a spectral truncation filter. Since the discrete filter is a function of a discrete Laplacian, it is natural to suspect that the filter should commute with derivatives; this question is addressed in Appendix B.

2.3.3 Polynomial approximation of the target filter

For the large data sets produced by Earth system models computing the eigenvalues and eigenvectors of $\mathbf{L}$ is prohibitively expensive, and even solving linear systems involving $\mathbf{L}$ can be expensive. By contrast, simply applying $\mathbf{L}$ is usually inexpensive. In practice this means that it is inexpensive to compute ~~$p(-\mathbf{L})\boldsymbol{f}$~~ $p(-\mathbf{L})\mathbf{f}$ when p is a polynomial. (The implicit differential filters of Germano (1986) and Guedot et al. (2015) correspond to letting $1/p$ be a polynomial.)

We propose to define our new filters as $\bar{\mathbf{f}} = p(-\mathbf{L})\mathbf{f}$, where p is a polynomial

$$p(-\mathbf{L}) = a_0\mathbf{I} + a_1(-\mathbf{L}) + \ldots + a_N(-\mathbf{L})^N. \tag{16}$$

To show that such a filter preserves the integral, note that $p(-\mathbf{L})$ is ~~self-adoint~~ self-adjoint with respect to the discrete inner product, and

$$\langle \mathbf{1}, \bar{\mathbf{f}} \rangle = \langle \mathbf{1}, p(-\mathbf{L})\mathbf{f} \rangle = \langle p(-\mathbf{L})\mathbf{1}, \mathbf{f} \rangle = \langle a_0\mathbf{1}, \mathbf{f} \rangle, \tag{17}$$

~~Where~~ where we have used the fact that $\mathbf{L1} = \mathbf{0}$ for any consistent discretization of the Laplacian with no-flux boundary conditions. The condition $a_0 = p(0) = 1$ thus guarantees that the spatial filter will preserve the integral. It also ensures that the filter will leave large scales approximately unchanged; in order to remove small scales p should decay towards zero as k increases.

We can choose a specific shape for p by means of standard polynomial approximation of a 'target' filter $\hat{G}_t$. For example, ~~note that the Fourier transform of a Gaussian convolution kernel with standard deviation L is~~

$$\hat{G}(k) = \exp\left\{-\frac{L^2k^2}{2}\right\}.$$

~~In~~ in order to construct a filter that acts like a convolution-based spatial filter with a Gaussian kernel of standard deviation L, one might choose a target filter of the form

$$\hat{G}_t(k) = \exp\left\{-\frac{L^2k^2}{2}\right\}. \tag{18}$$

The goal would then be to find a polynomial p such that $p(k^2) \approx \hat{G}_t(k)$. In general this is not possible with an explicit filter because polynomials grow without bound as $k \to \pm\infty$; thankfully it is only necessary for the approximation to hold over the range of scales represented on the grid, specifically for $0 \le k \le k_n$ where $-k_n^2$ is the most-negative eigenvalue of $\mathbf{L}$. If k_n is not known, some reasonable proxy can be used to define the range of scales over which p should act like a spatial filter~~, e.g.~~. For example, on a quadrilateral grid one might use $0 \le k \le \sqrt{d}\pi/\mathrm{d}x_{\min}$ where $\mathrm{d}x_{\min}$ is the ~~smallest grid spacing~~ length of the smallest grid cell edge and d is the spatial dimension of the grid.

In Appendix A we present a least-squares approach for finding a polynomial p such that $p(k^2)$ approximates $\hat{G}_t(k)$. The left column of ~~Fig. ??~~ Figure 2 shows three examples of target filters, along with their approximations $p(k^2)$ using polynomials of degree $N = 3, 5$, and 11. The top row shows the boxcar target (4) with length scale ~~$L = 4$~~ $L = 8$ (nondimensional), and the middle row shows the Gaussian target that corresponds to a Gaussian kernel with standard deviation ~~$2/\sqrt{3}$~~ $4/\sqrt{3}$ (nondimensional). The bottom row shows a target that we here ~~christen~~ label 'taper.'

The taper target is a piecewise polynomial with a continuous first derivative. It is $\hat{G}_t(k) = 0$ for k above some cutoff $k_c = 2\pi/L$, with ~~$L = 4$~~ $L = 8$ (nondimensional) in ~~Fig. ??~~Figure 2. For $0 \le k \le k_c/X$ it takes the value $\hat{G}_t(k) = 1$ where X controls

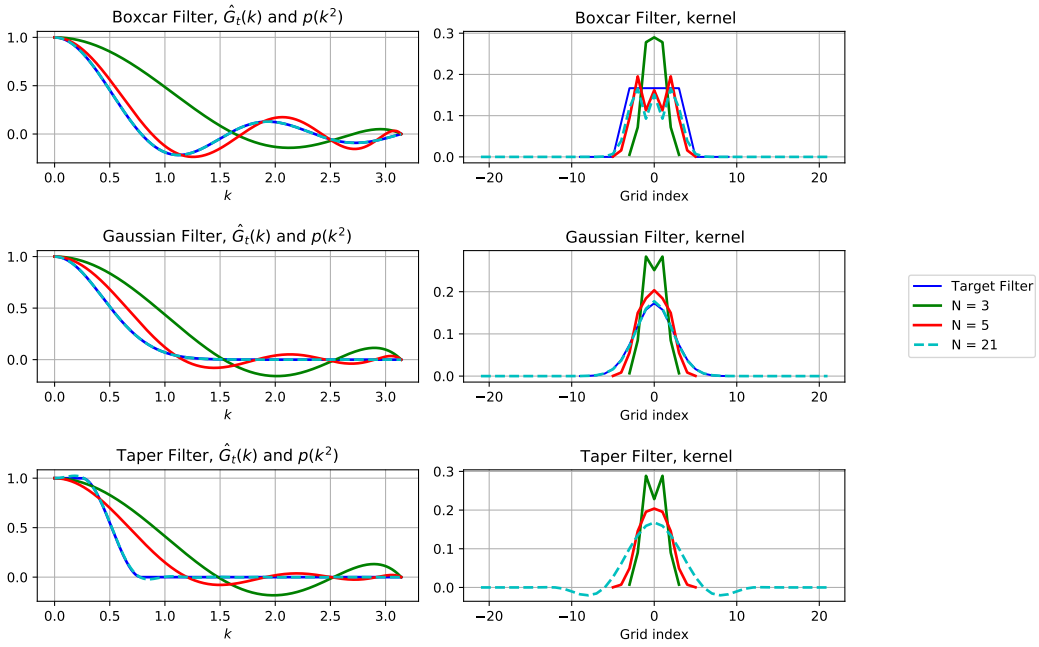

Figure 2. Left: Target filters $\hat{G}_t(k)$ and their approximations $p(k^2)$. Right: The equivalent kernel weights in one dimension on an equispaced grid of size 1. Top Row: A boxcar filter of width ~~4~~8; Middle Row: A Gaussian filter with standard deviation ~~$2/\sqrt{3}$~~$4/\sqrt{3}$; Bottom Row: The taper filter. All length scales in this figure are nondimensional. There is no blue line in the lower right panel because the taper filter is defined directly in terms of its target $\hat{G}_t(k)$, rather than via its convolution kernel, as for the boxcar and Gaussian filters.

the width of the transition region; $X = \pi$ in ~~Fig. ??~~Figure 2. For wavenumbers in the transition region $k_c/X \le k \le k_c$ the taper target is a cubic polynomial. As the width of the transition region goes to zero ($X \to 1$) the taper target approaches the spectral truncation filter, which is a step function at wavenumber k_c. The left column of ~~Fig. ??~~ Figure 2 shows that the number of steps N required to achieve an accurate approximation of the target filter depends on the shape of the target filter, with more scale-selective targets like the taper requiring more steps N.

2.3.4 Definition of filter scale

We provide a single definition of the 'filter scale' for the boxcar, Gaussian, and taper targets as follows. The filter scale for a boxcar kernel is simply the width of the kernel L (not the half-width). Per equation (4), the boxcar filter exactly zeros out the wavenumber $k = 2\pi/L$. Since the taper filter also zeros out wavenumber $2\pi/L$, it is natural to let L define the 'filter scale' for both the boxcar and taper filters. The filter scale for a Gaussian is chosen so that the standard deviation of the Gaussian and boxcar kernels match for a given filter scale (cf. Sagaut & Grohens, 1999). This is achieved by defining the 'filter scale' L for a Gaussian to be $\sqrt{12}$ times the standard deviation of the Gaussian kernel, i.e. to extract the standard deviation σ from the filter scale L use $\sigma = L/(2\sqrt{3})$.

2.3.5 Filter algorithm

Once the approximating polynomial has been found, the filtered field ~~$p(-\mathbf{L})\mathbf{f}$~~ $p(-\mathbf{L})\mathbf{f}$ can be efficiently computed using an iterative algorithm based on the polynomial roots. In general, any polynomial with real coefficients has roots that are either real, or come in complex-conjugate pairs. We can thus write

$$p(s) = a_N(s - s_1)\cdots(s - s_M)(s^2 - 2sR\{s_{M+2}\} + |s_{M+2}|^2)\cdots(s^2 - 2sR\{s_N\} + |s_N|^2), \quad (19)$$

where M is the number of real roots, the roots are $s_1, \ldots, s_N$, and $R\{\cdot\}$ and $I\{\cdot\}$ denote the real and imaginary parts of a complex number, respectively. The quadratic terms can also be written $|s - s_k|^2 = (s - R\{s_{M+2}\})^2 + (I\{s_{M+2}\})^2$. The condition $p(0) = 1$ implies

$$p(s) = \left(1 - \frac{s}{s_1}\right)\cdots\left(1 - \frac{s}{s_M}\right)\left(1 + \frac{-2sR\{s_{M+2}\} + s^2}{|s_{M+2}|^2}\right)\cdots\left(1 + \frac{-2sR\{s_N\} + s^2}{|s_N|^2}\right). \quad (20)$$

Based on this representation, the filtered field $\bar{\mathbf{f}} = p(-\mathbf{L})\mathbf{f}$ can be computed in $M + (N - M)/2$ ~~steps~~ stages as follows. First the real roots are dealt with via

$$\bar{\mathbf{f}} = \mathbf{f} \quad (21a)$$

$$\bar{\mathbf{f}} \leftarrow \bar{\mathbf{f}} + \frac{1}{s_k}\mathbf{L}\bar{\mathbf{f}}, \quad k = 1, \ldots, M. \quad (21b)$$

These stages are called Laplacian stages. Next the complex roots are dealt with via

$$\bar{\mathbf{f}} \leftarrow \bar{\mathbf{f}} + \frac{2R\{s_k\}}{|s_k|^2}\mathbf{L}\bar{\mathbf{f}} + \frac{1}{|s_k|^2}\mathbf{L}^2\bar{\mathbf{f}}, \quad k = M + 2, M + 4, \ldots, N. \quad (22)$$

These stages are called biharmonic stages because of the need to apply the discrete biharmonic operator $\mathbf{L}^2$.

In ~~exact arithmetic the iterations corresponding to the real and complex roots~~ the absence of roundoff errors the Laplacian and biharmonic stages can be applied in any order, and once they are both complete $\bar{\mathbf{f}}$ contains the filtered field (though at any point in the middle of the iterations $\bar{\mathbf{f}}$ has no particular meaning). However, in practice the order can have an impact on numerical stability. This issue is discussed

in the following subsection. The form of equation (21b) is directly reminiscent of time integration of the diffusion equation via an explicit Euler discretization, and in some sense the method can be thought of as smoothing through diffusion.

2.4 Numerical Stability

Recall that per equation (13) we can formally expand the field to be filtered as a sum of eigenvectors of the discrete Laplacian, and that per equation (14) the effect of the filter is simply to modify the coefficients in this expansion. The same idea applies to a single stage in the iterative application of the filter. A single Laplacian stage multiplies the expansion coefficients by

$$1 - \frac{k_i^2}{s_k}. \tag{23}$$

Any modes i such that $k_i^2 > 2s_k$ will have their coefficients $\hat{f}_i$ amplified at this stage, and smaller scales will experience greater amplification. (The sign of the coefficients will also be changed; the real roots s_k are generally positive.) In contrast, when $|1 - k_n^2/s_k| < 1$ none of the modes will experience amplification and the smallest scales will be damped.

A single biharmonic stage multiplies the expansion coefficients by

$$\left|1 - \frac{k_i^2}{s_k}\right|^2. \tag{24}$$

As a function of k_i this is a positive parabola that equals 1 at $k_i = 0$. When the real part of s_k is negative all modes are amplified with increasing amplification at small scales. When the real part of s_k is positive, modes with $k_i^2 > 2\mathcal{R}\{s_k\}$ will be amplified, with increasing amplification at small scales.

A filter that attempts to remove a wide range of scales, i.e. one where the filter scale is much larger than the grid scale, will have many small values of s_k that correspond to stages where the small scales are amplified, and will also have many stages overall since it requires more stages to accomplish more smoothing. If there are several stages in succession that cause amplification at the small scales, it can lead to extreme amplification at small scales, including extreme amplification of any roundoff errors present in the small scales. This combination of many amplifying stages, together with a large number of stages for roundoff errors to accumulate, can lead to inaccurate results or even blowup of the filtered field. To avoid this we recommend choosing a specific order for the roots s_k, such that stages that amplify small scales are always followed by stages that damp small scales.

To illustrate these ideas we set up a simple toy problem with a one-dimensional, periodic, equispaced grid of 256 points in a nondimensional domain of size 2π, and a spectral discrete Laplacian. The eigenvectors of the discrete Laplacian are the discrete Fourier modes with wavenumbers $k = -127, \ldots, 128$, and the eigenvalues are exactly $-k^2$. The filter polynomial p is constructed by directly specifying the roots s_k, rather than by approximating some target filter $\hat{G}_t$. The roots s_k are the integers from 43 to 170, squared (i.e. there are $N = 128$ stages). This filter should thus exactly zero out all discrete wavenumbers with $|k| \geq 43$, while smoothly damping wavenumbers with $|k| < 43$. The field to be filtered is constructed to have discrete Fourier transform $\hat{f}_k = e^{\mathrm{i}\theta_k}$ where θ_k are independent and uniformly distributed on $[0, 2\pi)$. This initial condition is chosen so that the discrete Fourier transform of the final filtered field should, in the absence of roundoff errors, have absolute value equal to $|p(k^2)|$.

Figure 3 shows the amplitude of the Fourier modes of the field as it progresses through the stages of the filter. The left panel shows the result for a filter where s_k are

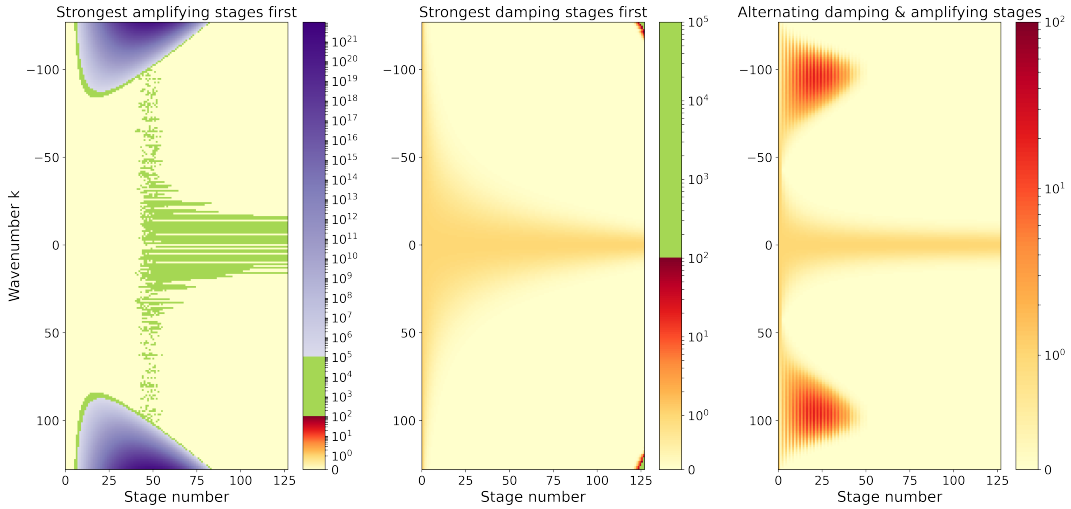

Figure 3. Amplitude of the Fourier coefficients of $\bar{\mathbf{f}}$ as it proceeds through the filter stages. In each panel the abscissa is filter stage while the ordinate is the wavenumber. In the left panel s_k are arranged in increasing order. In the center panel the s_k are decreasing. In the right panel the damping and amplifying stages alternate.

ordered from least to greatest, such that the first stages amplify the small scales while the last stages damp them. The small scales grow to amplitudes on the order of 10^{21} within the first 50 stages. The subsequent stages manage to damp these small scales back out, but the solution is so corrupted by the effect of roundoff errors that the final solution is completely inaccurate: the large scales have amplitudes on the order of 10^4.

The center panel of ~~Fig.~~ Figure 3 shows the effect of arranging s_k in decreasing order, such that the last stages amplify the small scales while the first stages damp them. The filter behaves quite well until the final few stages, where the small scales are amplified to the order of 10^4. Evidently the initial damping stages introduce small amplitude roundoff errors into the small scales which are then amplified in the final stages.

The right panel of ~~Fig.~~ Figure 3 shows the effect of arranging the s_k so that the small scales are alternately amplified and then damped. In the early stages of the filter there is a range of intermediate scales that begins to amplify, though they maintain modest amplitudes less than 100. These intermediate scales are eventually damped back out in the later stages, leading to a well-behaved and accurate solution.

The stages in the right panel of Figure 3 are arranged in the following simple way. We first compute the impact of each stage on the smallest scale, given by setting $k_i = k_{\max}$ in the absolute value of expression (23) and in expression (24). These values are then ordered, and the stage order is set by selecting the smallest value (strongest damping) first, followed by the largest value (strongest amplification), followed by the next-smallest value, etc.

2.5 Impact of the order of accuracy of the discrete Laplacian

~~The foregoing section made no reference to the accuracy of the discrete Laplacian as an approximation of the continuous Laplacian.~~ This section gives a simple example to show that higher-order discretizations of the Laplacian should be better able to sharply distinguish between scales near the grid scale. The fundamental idea of

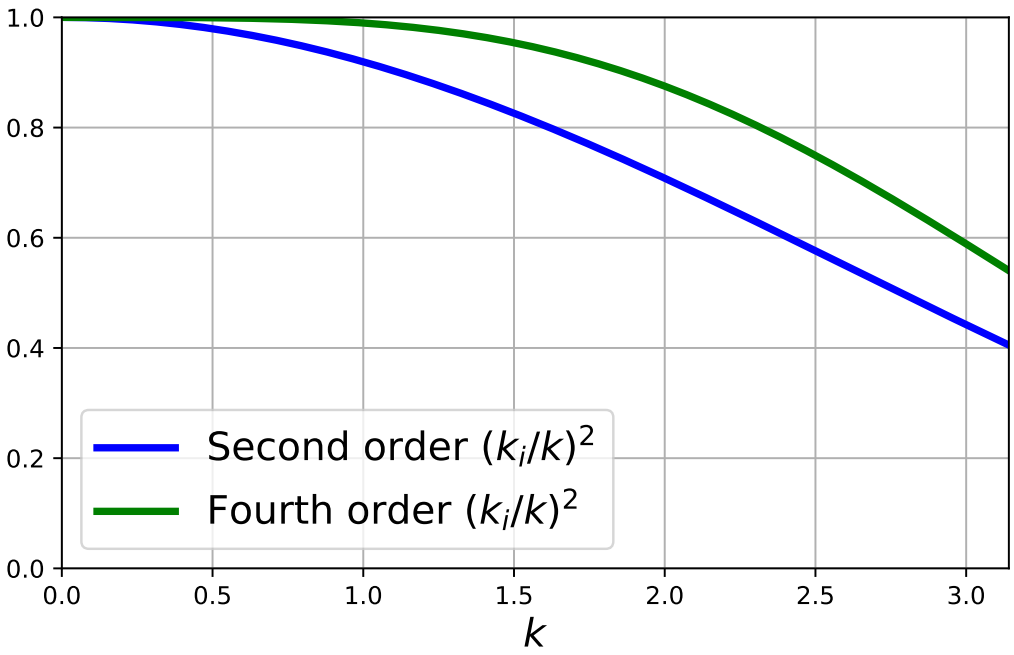

Figure 4. The ratio of the eigenvalues $-k_i^2$ of the discrete Laplacians to the true value $-k^2$. The second-order Laplacian is shown in blue and the fourth-order Laplacian is shown in green.

~~the preceding section~~ section 2.3 is that the eigenvalues of the discrete Laplacian correspond to the spatial length scale of the eigenvector in the same way that this correspondence works for the continuous Fourier problem, i.e. if ~~the~~ $-k_i^2$ is an eigenvalue of the discrete Laplacian ~~is $-k_i^2$~~ then the length scale of the corresponding eigenvector $\mathbf{q}_i$ is assumed to be $2\pi/k_i$. This connection between eigenvalues and length scales can be inaccurate at small length scales.

For example, consider the following two discrete Laplacians on an infinite or periodic one-dimensional equispaced grid with grid spacing 1 (nondimensional)

$$(\mathbf{L}_2\mathbf{f})_j = f_{j-1} - 2f_j + f_{j+1} \tag{25}$$

$$(\mathbf{L}_4\mathbf{f})_j = -\frac{1}{12}f_{j-2} + \frac{4}{3}f_{j-1} - \frac{5}{2}f_j + \frac{4}{3}f_{j+1} - \frac{1}{12}f_{j+2}. \tag{26}$$

For both of these Laplacians the discrete Fourier modes

$$(\mathbf{q}_k)_j = e^{\mathrm{i}kj} \tag{27}$$

are eigenvectors, where $0 \le k \le \pi$ is the discrete wavenumber, $\mathbf{L}_2$ is second order, and $\mathbf{L}_4$ is fourth order. (Note that notation has been changed from $\mathbf{q}_i$ in ~~the previous section~~ section 2 to $\mathbf{q}_k$ here, so that k is the discrete wavenumber rather than i.) ~~If the discretization were perfect then~~ For a spectral discretization the eigenvalues would be $-k^2$, but the eigenvalues for the second and fourth order Laplacians are

$$\mathbf{L}_2\mathbf{q}_k = -4\sin^2\left(\frac{k}{2}\right)\mathbf{q}_k \tag{28}$$

$$\mathbf{L}_4\mathbf{q}_k = -\frac{2}{3}\left(7 - \cos(k)\right)\sin^2\left(\frac{k}{2}\right)\mathbf{q}_k. \tag{29}$$

The fact that these are not equal to $-k^2$ is tantamount to saying that the filter will incorrectly identify the length scales of the eigenfunctions. ~~More precisely, in~~ Figure 4 shows the ratio of the discrete eigenvalues (28) and (28) to the correct value $-k^2$. In both cases the wavenumber implied by the eigenvalue is smaller than the true wavenumber k, meaning that these Laplacians treat small scales as if they were larger-scale than they really are. Both Laplacians have accurate eigenvalues at large scales, but the fourth order Laplacian's eigenvalues are more accurate at small scales. A filter that uses the fourth order Laplacian will thus be more accurate when the filter is attempting to separate scales near the limit of resolution. If one is attempting, for example, to get an accurate estimate of the energy spectrum at scales near the grid scale using the ~~filtering~~ diffusion-based filter of section 2.3 in combination with the method of Sadek and Aluie (2018) for estimating the spectrum, then it would be important to use a high-order discretization of the Laplacian. On the other hand, if the filter is attempting to remove the entire range of small scales where the second-order Laplacian is inaccurate, then the second order Laplacian will work as well as higher-order Laplacians.

2.6 Spatially varying filter properties

The filters developed in ~~the preceeding section~~ section 2.3 are based on the isotropic Laplacian, and are therefore isotropic in the sense that they provide an equal amount of smoothing in every direction. The filter coefficients are the same over the whole domain, so the degree of smoothing is also constant over the domain. This can be generalized to anisotropic and spatially-varying filters by letting $\mathbf{L}$ be a discretization of $\nabla \cdot \mathbf{K}(\mathbf{x})\nabla$ where $\mathbf{K}(\mathbf{x})$ is a symmetric and positive definite tensor that varies in space (cf. Báez Vidal et al., 2016). (In this context $\mathbf{K}$ is nondimensional, since the dimensions are carried by the polynomial roots s_i.)

Consider first the isotropic case $\mathbf{K} = \kappa\mathbf{I}$ with constant κ, and assume that the filter polynomial $p(k^2)$ has been designed as described in ~~the foregoing section~~ section 2.3 under the assumption $\kappa = 1$. If the filter polynomial is used with constant $\kappa \neq 1$ then the filter polynomial $p(k^2)$ is replaced by $p(\kappa k^2)$. This is tantamount to rescaling the filter length scale by $\sqrt{\kappa}$. For example, if the original filter with $\kappa = 1$ had a characteristic length scale of L then the filter using $\kappa \neq 1$ has a characteristic length scale of $\sqrt{\kappa}L$.

Next consider the case of an isotropic Laplacian with spatially-varying κ, and assume that κ varies slowly over the domain. The filter polynomial p is designed to have length scale L if $\kappa = 1$. In regions where $\kappa > 1$ the filter will have a longer length scale $\sqrt{\kappa}L$, while in regions where $\kappa < 1$ the filter will have a smaller length scale. (If κ varies on length scales smaller than the filter scale then the behavior of the filter is hard to predict, so this situation should be avoided.)

Finally, consider the case of an anisotropic Laplacian with symmetric and positive definite $\mathbf{K}$ that varies over the domain. At each point in the domain $\mathbf{K}$ has two orthogonal eigenvectors corresponding to different directions, and the eigenvalues indicate the strength of smoothing in each direction. One natural application of the anistropic Laplacian is to apply a filter whose length scale is tied to the local grid scale, which is especially relevant for Earth system models whose grid cell sizes vary in space. This can be achieved by aligning the eigenvectors of $\mathbf{K}$ with the local orthogonal grid directions, and letting the respective eigenvalues determine the amount of filtering in each direction.

A major caveat to the above discussion is that values of $\kappa > 1$ can lead to unexpected behavior. Consider, for example, the filter polynomial

$$p(\kappa k^2) = (1 - 0.7\kappa k^2)(1 - 0.8\kappa k^2)\cdots(1 - 1.2\kappa k^2)(1 - 1.3k^2), \tag{30}$$

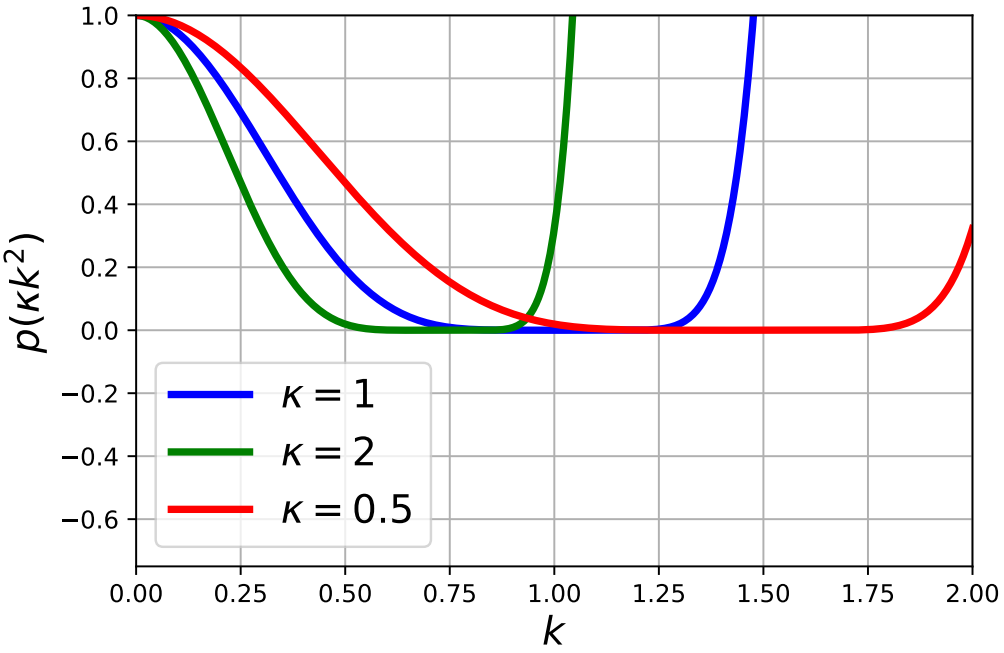

Figure 5. The effect of changing κ on the filter polynomial $p(\kappa k^2)$ for the polynomial p from equation (30).

where the scales that can be represented on the grid are associated with wavenumbers $0 \leq k \leq 1$ ~~. Figure ??~~ and the standard case uses $\kappa = 1$. The blue line in Figure 5 shows that $p(k^2)$ only acts as a smoother over the range of scales associated with $0 \leq k \leq 1$; at larger k that are not represented on the grid the filter will significantly amplify these scales~~, while switching their sign~~. Using $\kappa > 1$ has the effect of bringing this undesirable filter behavior into the range of scales represented on the grid, as can be seen in the orange line corresponding to $\kappa = 2$ in ~~Fig. ??~~Figure 5. In contrast, using $\kappa \leq 1$ has no such problems (the green line in ~~Fig. ??~~Figure 5). It is thus desirable to specify $\kappa \leq 1$ whenever possible.

Consider, for example, a one-dimensional non-uniform grid with maximum grid spacing $h_{\max}$, minimum grid spacing $h_{\min}$, and local grid spacing h. To apply a filter that smooths locally to a scale ~~n~~ m times larger than the local grid, one could choose the filter scale to be ~~$L = nh_{\min}$~~ $L = mh_{\min}$ and then set $\kappa = (h/h_{\min})^2$. Locally the filter scale is rescaled to ~~$\sqrt{\kappa}L = (h/h_{\min})(nh_{\min}) = nh$~~ $\sqrt{\kappa}L = (h/h_{\min})(mh_{\min}) = mh$ as desired, but at the same time $\kappa \geq 1$ which will lead to undesirable behavior at the small scales. Instead, one can achieve the same effect by setting the filter scale to ~~$L = nh_{\max}$~~$L = mh_{\max}$, and then setting $\kappa = (h/h_{\max})^2$. The local filter scale is again ~~$L = nh$~~$L = mh$, but with $\kappa \leq 1$ over the whole domain.

We next describe a more *ad hoc* method of tying the local filter scale to the local grid scale. This method is not without drawbacks, but it is simpler and faster than the method based on an anisotropic and spatially-varying Laplacian. We call this filter the simple fixed factor filter.

Let $\mathbf{L}_0$ be the discretization of the Laplacian if all the cells had the same size. Since the cell sizes are assumed equal, the matrix $\mathbf{L}_0$ should be symmetric. If we simply replaced $p(-\mathbf{L})$ by $p(-\mathbf{L}_0)$ in the definition of the filter it would imply that we

were filtering *as if* all the grid cells were the same size, which is equivalent to making the scale of the filter relative to the scale of the local grid. Unfortunately this would no longer preserve the integral. To rectify this problem we propose a cell-size weighted filter, which amounts to the following recipe:

- Weight the input data by cell sizes
- Apply the filter assuming the cell sizes are equal
- Divide the result by the cell sizes.

We next show that this filter preserves the integral at the discrete level. First note that weighting by the cell size is equivalent to multiplication by a diagonal matrix $\mathbf{W}$ whose diagonal entries are the cell sizes, so the above filter corresponds to

$$\bar{\mathbf{f}} = \mathbf{W}^{-1} p(-\mathbf{L}_0)\mathbf{W}\mathbf{f}. \tag{31}$$

The inner product (10) can be written in the form $\langle \mathbf{f}, \mathbf{g} \rangle = \mathbf{f}^T \mathbf{W} \mathbf{g}$, and recall that the discrete integral is $\langle \mathbf{1}, \mathbf{f} \rangle$. To prove that the new filter conserves the integral we follow (17), and find that

$$\langle \mathbf{1}, \bar{\mathbf{f}} \rangle = \mathbf{1}^T \mathbf{W} \mathbf{W}^{-1} p(-\mathbf{L}_0)\mathbf{W}\mathbf{f} = p(0)\mathbf{1}^T \mathbf{W}\mathbf{f} = \langle \mathbf{1}, \mathbf{f} \rangle. \tag{32}$$

The above sequence uses the facts that $\mathbf{L}_0$ is symmetric, which implies $\mathbf{1}^T \mathbf{L}_0 = (\mathbf{L}_0 \mathbf{1})^T$, that any consistent discretization of the Laplacian with no-flux boundary conditions will have $\mathbf{L}_0 \mathbf{1} = \mathbf{0}$, and that $p(0) = 1$.

Applying the discrete Laplacian under the assumption that all cell sizes are equal is much simpler than using an anisotropic Laplacian, and the algorithm can thus be much faster. On the other hand, this ad hoc method no longer has the property that the constant vector is left unchanged by the filter. Note that the simple fixed factor filter is anisotropic whenever the grid spacing is anisotropic, and it is spatially-varying whenever the grid spacing is non-uniform.

2.7 Variance reduction

In some situations it is desirable to enforce that the filtered field has less total variance than the unfiltered field, i.e. for functions

$$\int_\Omega f(\mathbf{x})^2 \mathrm{d}\mathbf{x} \geq \int_\Omega \bar{f}(\mathbf{x})^2 \mathrm{d}\mathbf{x} \tag{33}$$

and for ~~vectors~~ the discrete case

$$\langle \mathbf{f}, \mathbf{f} \rangle \geq \langle \bar{\mathbf{f}}, \bar{\mathbf{f}} \rangle. \tag{34}$$

To translate this into a condition on the diffusion-based ~~smothers~~ smoothers developed here, expand $\mathbf{f}$ in the orthonormal basis of eigenvectors of $\mathbf{L}$

$$\mathbf{f} = \sum\nolimits_{i=1}^{n} \hat{f}_i \mathbf{q}_i. \tag{35}$$

The condition of variance reduction becomes

$$\sum\nolimits_{i=1}^{n} \hat{f}_i^2 \geq \sum\nolimits_{i=1}^{n} \hat{f}_i^2 \left(p(k_i^2)\right)^2. \tag{36}$$

In order for this to be satisfied for any possible vector $\mathbf{f}$ this requires $|p(k_i)^2| \leq 1$ for every k_i up to the largest one represented on the model grid, i.e. k_n. The eigenvalues $-k_i^2$ of the discrete Laplacian are usually not known exactly, so a sufficient condition for variance reduction would be that $|p(k^2)| \leq 1$ for every $0 \leq k \leq k_{\max}$ where $k_{\max} \geq k_n$. It is worth noting that this condition applies to p and not to the target filter. Even if the target filter satisfies this condition, the polynomial p might not satisfy it. (In all examples in the left column of Figure 2 both the target filter and the approximating polynomials do satisfy this condition.) It is also worth noting that failure to satisfy this condition does not guarantee that the filtered field has more total variance than the unfiltered field, ~~it simply means~~ but only that it might happen in some cases.

2.8 The effective kernel implied by the diffusion-based filter

If the spatial filter were defined by a discrete approximation of a kernel-based spatial filter (8) then the value of $\bar{f}$ at the i^{th} grid cell would be

$$\bar{f}_i = \langle \mathbf{g}_i, \mathbf{f} \rangle = \sum_j w_j g_{ij} f_j, \tag{37}$$

where $\mathbf{g}_i$ is the effective filter kernel corresponding to the i^{th} cell. Note that $\bar{f}_i = \langle \mathbf{e}_i, \bar{\mathbf{f}} \rangle / w_i$, where $\mathbf{e}_i$ is a vector of zeros with 1 at the i^{th} grid cell. Next note that

$$\bar{f}_i = \frac{1}{w_i} \langle \boldsymbol{e}_i, p(-\mathbf{L})\mathbf{f} \rangle = \frac{1}{w_i} \langle p(-\mathbf{L})\mathbf{e}_i, \mathbf{f} \rangle, \tag{38}$$

which implies that $\mathbf{g}_i = p(-\mathbf{L})\mathbf{e}_i / w_i$. We can thus compute the effective filter kernel that corresponds to $p(-\mathbf{L})$ at the i^{th} grid cell by applying the filter to $\boldsymbol{e}_i$ and then dividing the result by w_i. The same arguments can be used to find the effective filter kernel associated with the spatially-varying filters of ~~the preceding section~~ section 2.6.

Note that if the filter kernel ever takes a negative value, then it is no longer guaranteed to preserve positivity in the sense that $\bar{\mathbf{f}}$ may have negative values even when all the values in $\mathbf{f}$ are positive. The ~~Fourier~~ spectral truncation filter is such an example having negative weights.

The right column of ~~Fig. ??~~ Figure 2 computes the filter kernels associated with the polynomial approximations of the boxcar, Gaussian, and taper filters in the left column of ~~Fig. ??~~Figure 2. The standard equispaced, second-order Laplacian (25) was used, with a nondimensional grid size of 1. The upper right panel illustrates that the kernel associated with the polynomial approximation of the boxcar filter does not converge to the actual boxcar kernel, though it is close. One reason for this discrepancy is the fact that the boxcar target (4) was formulated by reference to a continuous Fourier transform, which is not a one-to-one match to the discrete version. Another reason is that the effective kernel depends on the discretization of the Laplacian; a higher-order discretization would result in a slightly different effective kernel. Despite these discrepancies, the effective kernel of the polynomial approximation to a Gaussian target still converges to a close approximation of the expected Gaussian kernel, as can be seen in the middle right panel of ~~Fig. ??~~Figure 2.

3 Illustrative Examples

In this section we present examples using model output and ~~data products~~ observational data to illustrate the various filter properties and capabilities. A open-source python package implementing the diffusion-based filters described in section 2, called gcm-filters, is currently under development. This Python code includes implementations of the discrete Laplacian on a variety of grids for different ocean general circulation models. All examples that show the filtering of two-dimensional data use a second-order discrete Laplacian (on a 5-point stencil) with no-flux boundary condition.

3.1 Effective Kernels

We begin with an example showing effective filter kernels for various configurations of the filters, noting especially how the filter kernel adapts near boundaries. Figure 6 shows the effective kernels for six different filter configurations, each at three locations in the equatorial Pacific Ocean. The grid is the 0.1 degree nominal resolution tripole grid of the Parallel Ocean Program (POP; Smith et al., 2010). The top row shows filters with a Gaussian target, while the bottom row shows filters with the taper target. It is clear that the taper target produces kernels with negative weights, while the Gaussian target does not. The left column corresponds to isotropic filters with

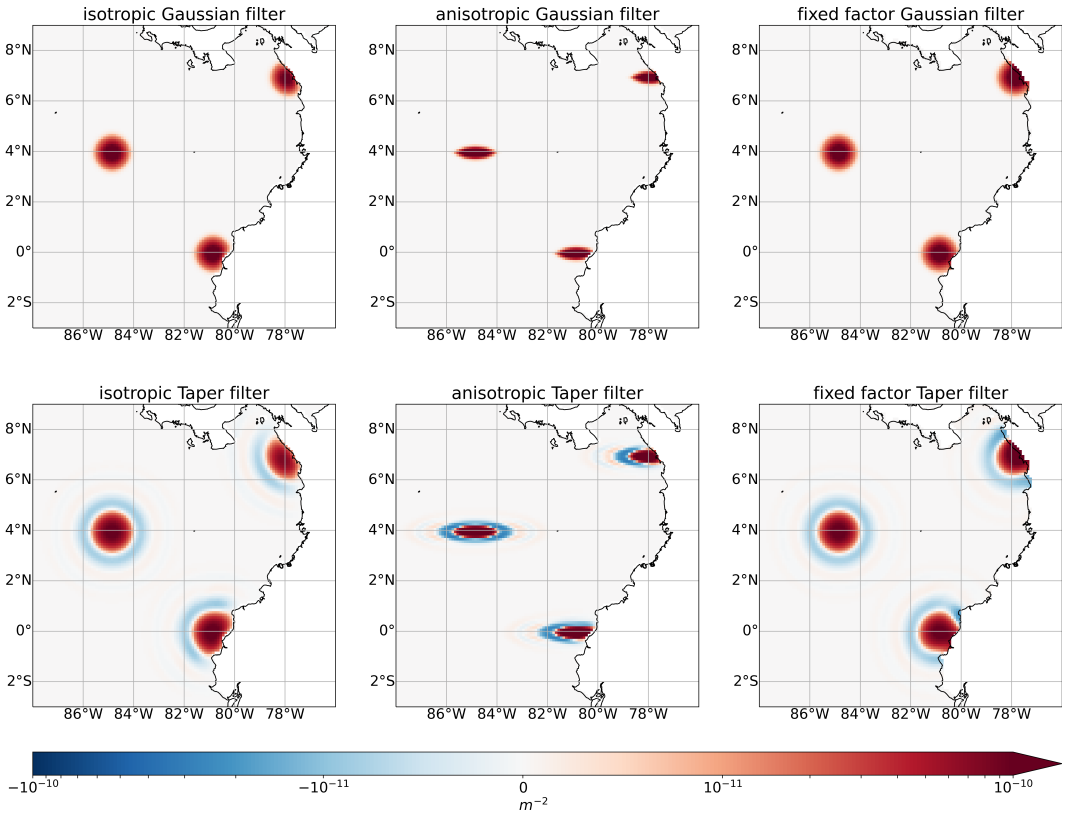

Figure 6. Effective filter kernels for six different filters on the 0.1 degree POP tripole grid, centered at 3 points in the equatorial Pacific. Top row: Gaussian; Bottom row: Taper. Left column: The isotropic version of the filter with a fixed filter scale of ~~112~~ 111.7 km. Center column: The anisotropic version of the ~~filter~~ filters in the left column, but with a length scale 10 times smaller in the meridional direction. Right column: The simple fixed factor filter from ~~section~~ Section 2.6~~. Kernel values are dimensionless~~, with filter scale equal to 10 times the local grid scale.

a fixed length scale of ~~112 km~~111.7 km, corresponding to 1 degree of latitude. The center column corresponds to anisotropic filters that are the same as the left column except that the filter scale has been decreased by a factor of 10 in the meridional direction. The right column corresponds to the simple fixed factor filter from ~~section 2.6. It is clear that the taper target produces kernels with negative weights, while the Gaussian target does not.~~ Section 2.6, where we chose a filter scale of 10 times the local grid scale. Since the grid is nominally 0.1 degree, the width of the fixed factor filter kernels in the right column is approximately equal to the width of the fixed scale filter kernels in the first column. Despite the similarity of the effective filter kernels in the first versus third columns, the computational effort of their underlying filters varies significantly: while the fixed scale Gaussian (Taper) filter in the first column requires $N = 90$ ($N = 319$) stages to achieve an approximation of the target filter with error less than 1%, the fixed factor versions in the third column need only $N = 18$ ($N = 64$) stages. This is explained by the fact that, by construction, the fixed scale filter has to remove a wider range of scales than the simple fixed factor filter, because the smallest grid spacing, $dx_{\min}$, of the POP grid is 2.2km. As a result, the filter with fixed scale $L = 111.7$ km has to remove scales up to 50 times the local grid scale, while the simple fixed factor filter removes only scales up to 10 times the local grid scale. The discrepancy in the number of necessary filter stages is reflected in the speed of the fixed scale versus fixed factor filters, which we will further discuss in Section 4.

~~The next example~~

3.2 Spatially varying filter scale

Figure 7 illustrates the ability of our filters to vary their length scales over the domain by using variable κ as described in ~~section~~ Section 2.6. We filter the vertical component of relative vorticity at the surface from the submesoscale-resolving MITgcm simulation of the Scotia Sea described in Bachman et al. (2017). In the map of the unfiltered vorticity (top panel) large scales are evident in the Antarctic Circumpolar Current to the east of Drake Passage, where the first baroclinic deformation radius tends to be $O(10)$ km and is generally smaller than the eddies themselves. Small scales are ubiquitous over the continental shelf off the eastern coast of Argentina, where the deformation radius is $O(1)$ km and is much closer to the eddy scale. We demonstrate the spatially-varying filter by choosing the filter length scale so that κ ~~so~~ is proportional to the local first baroclinic deformation radius. In making this choice we expect that more features will be filtered out in the areas where the dynamics tend to be larger than the deformation scale, as shown in the map of the filtered vorticity (middle panel) and the difference, i.e. the eddy vorticity field (lower panel). ~~Note that because the filter largely did not affect dynamics at or smaller than the deformation scale, the small-scale eddies over the continental shelf are part of the 'mean' field, rather than part of the eddies.~~

~~The next figure~~

3.3 Non-commutation of the filter and spatial derivatives

Figure 8 illustrates the lack of commutation of the filters with spatial derivatives in the presence of boundaries. We compute a large-scale part of the vertical component of relative vorticity in two ways, first by filtering the velocity and then computing vorticity from the result $\nabla \times \overline{\mathbf{u}}$, and second by computing the vorticity directly from the velocity and then applying the filter to the result $\overline{\nabla \times \mathbf{u}}$. The filter is isotropic, and uses a Gaussian target with a length scale of 100 km. The data is from a state-of-the-art climate model, CM2.6 (Delworth et al., 2012; Griffies, 2015), obtained through the Pangeo ~~platform~~ cloud data library (Abernathey et al., 2021). The ocean component of CM2.6 utilizes the GFDL-MOM5 numerical ocean code with a resolution of 0.1°. The

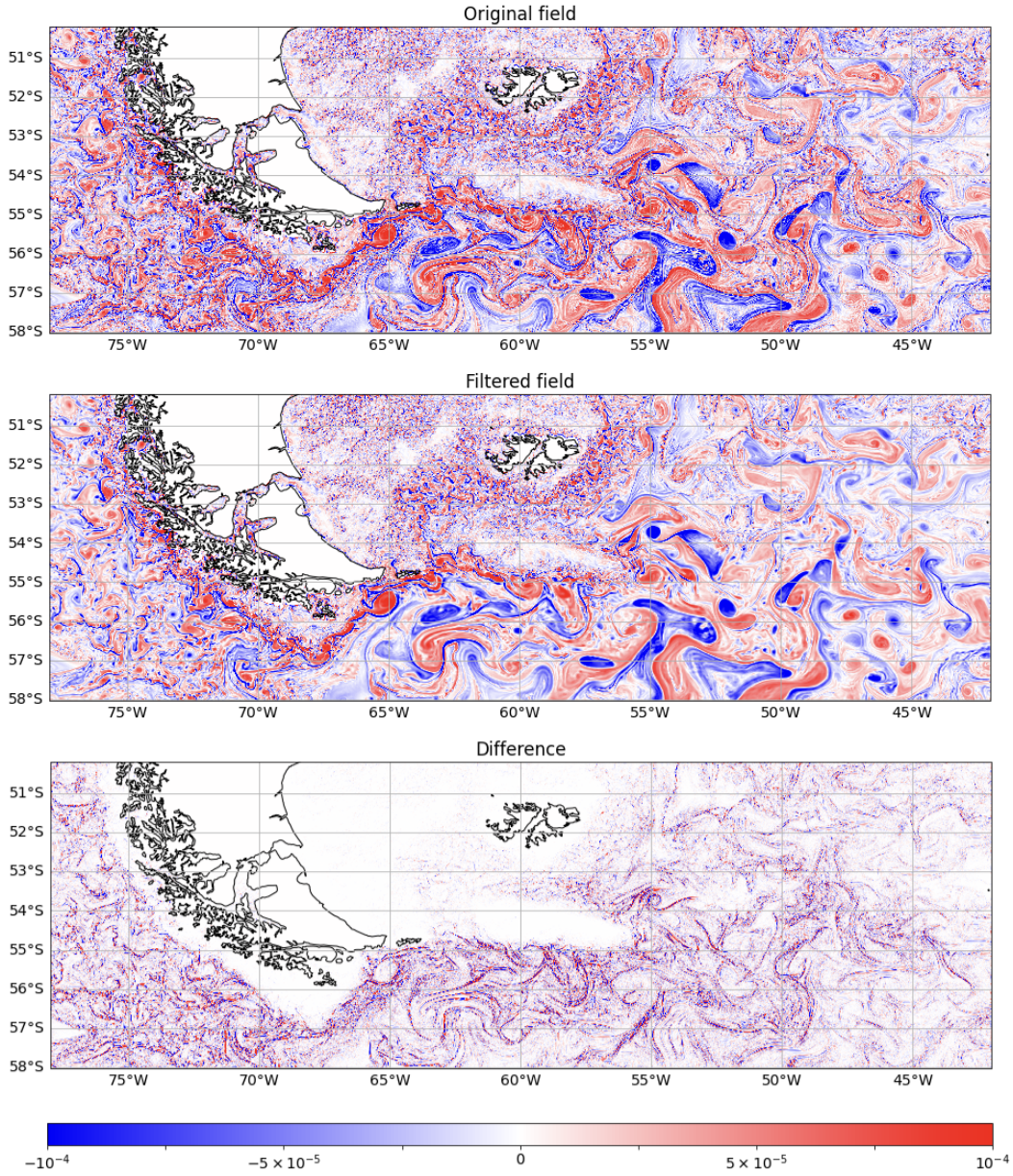

Figure 7. Surface relative vorticity from the MITgcm simulation in Bachman et al. (2017) demonstrating a spatially variable filter scale. The filter applied to the raw field (top panel) results in smoothing where the first baroclinic deformation radius is ~~large~~ small compared to the scale of the motion (middle panel), which is reflected in the difference between the raw and filtered fields (bottom panel). Units are s^{-1}.

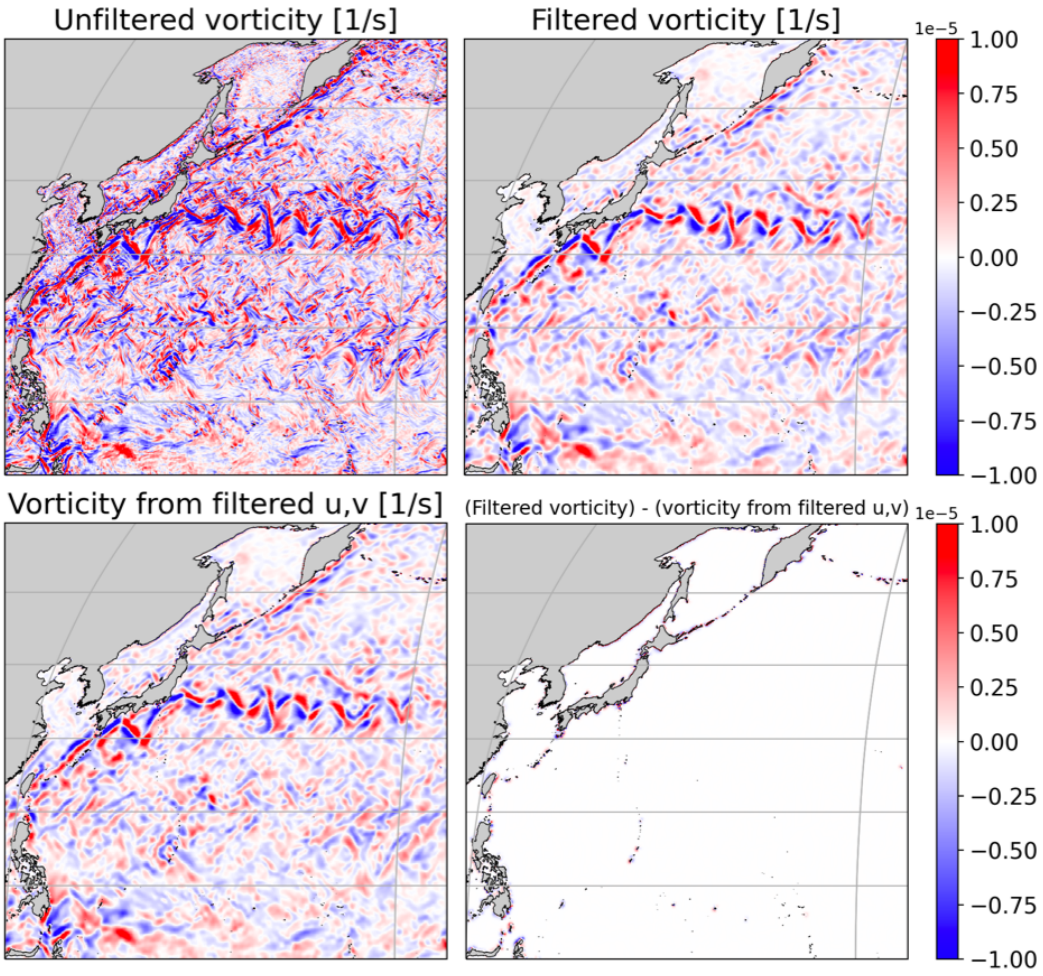

Figure 8. Surface relative vorticity fields taken from CM2.6 data. ~~Upper~~ The upper left panel ~~is~~ shows the unfiltered vorticity, the upper right ~~is~~ shows the filtered vorticity, the bottom left ~~is~~ panel shows the vorticity computed from filtered velocities, and the bottom right ~~is~~ panel shows the difference between the latter two fields.

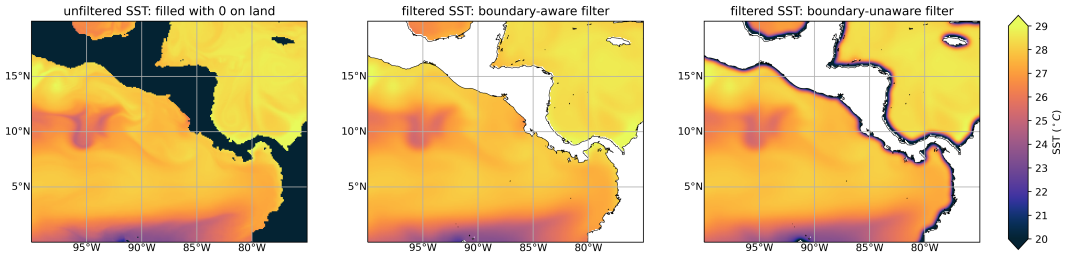

Figure 9. ~~Left:~~ The left panel shows SST from a single 5-day average of the 0.1° POP simulation in ~~(Johnson et al., 2016)~~Johnson et al. (2016), with values on land set to 0. ~~Center:~~ The ~~isotropic~~ center panel shows the simple Gaussian fixed factor filter with a ~~length~~ filter scale 10 times the local grid scale ~~of 112 km~~ applied to the left panel, ~~but~~ with a no-flux condition at land boundaries. ~~Right:~~ The ~~isotropic Gaussian filter with a length scale of 112 km applied to~~ right panel is filtered SST as in the ~~left~~ middle panel, but ignoring land boundaries instead of using a no-flux boundary condition. ~~Notice~~ Ignoring the ~~"~~boundaries introduces artificial cold ~~" band hugging the coasts in the right panel, which arises from filling land values with 0. This figure shows how fill values on land artificially affect the filtered fields in oceanic coastal regions. Apart from the differences~~ bands of waters near the ~~coast, the filtered fields are identical~~coasts. Units are degrees Celsius.

upper left panel ~~of Fig. 8~~ shows the raw vorticity in the northwest Pacific, while the upper right and lower left panels show the filtered vorticity and the vorticity obtained from the filtered velocity, respectively. The lower ~~left~~ right panel shows the difference between the two smoothed vorticities, and it is clear that the differences are extremely small over most of the domain. Significant differences arise only near the boundaries, as can be seen especially in the vicinity of the ~~Phillipines~~Philippines, which serves to illustrate the fact that the filter does not commute with derivatives near boundaries.

The ability to commute the filter with spatial derivatives can be restored by treating values on land as zero, following Aluie et al. (2018). To illustrate a potential downside of this approach we compare in ~~Fig.~~ Figure 9 the filtered sea surface temperature (SST) that results from the two approaches. The left panel shows the unfiltered SST, while the middle panel shows the SST filtered using the no-flux condition on the discrete Laplacian. The right panel shows the result of filtering SST over the entire domain, including land, while filling values on land with zero. The effect is to introduce ~~layers~~ bands of extremely cold water within one filter scale of the coast; in the right panel of Figure 9, SST reaches values as low as 6°C near Panama's coasts. It is thus clear that both methods have pros and cons near boundaries. The data used in ~~Fig.~~ Figure 9 are from the CORE-forced POP simulation described in ~~(Johnson et al., 2016)~~Johnson et al. (2016); the filter is the simple fixed factor filter of ~~section 2.6~~Section 2.6, with a filter scale 10 times the local grid scale ~~.~~and a Gaussian target.

3.4 Negative weights and eddy kinetic energy

The Gaussian filter's effective kernel has positive weights, while the more scale-selective taper filter's effective kernel typically has negative weights reminiscent of the sinc kernel that corresponds to the spectral truncation filter. These negative weights can produce negative values for non-negative quantities like eddy kinetic energy. We define eddy

kinetic energy (EKE) as

$$\text{EKE} = \frac{1}{2}\overline{|\mathbf{u}|^2} - \frac{1}{2}|\bar{\mathbf{u}}|^2. \tag{39}$$

This definition of EKE has the virtue that the total kinetic energy is exactly the sum of the mean and eddy kinetic energies. When the weights are positive, it can be shown usng the Cauchy-Schwartz inequality that the definition (39) will never produce a negative EKE.

Figure 10 illustrates the application of our filters to a single five-day average of AVISO estimates of absolute geostrophic velocity on a 0.25 degree grid obtained from Copernicus European Earth Observation program [`https://marine.copernicus.eu`] via Pangeo ~~(?, ?)~~(Abernathey et al., 2021). The upper left panel shows the unfiltered surface kinetic energy defined as $|\mathbf{u}|^2/2$. To compute mean surface kinetic energy we use the simple fixed factor Laplacian with a filter scale four times the local grid scale, i.e. a filter scale of 1 degree. The center panel in the upper row shows the mean kinetic energy defined as $|\bar{\mathbf{u}}|^2/2$ using a Gaussian target, while the upper right panel shows the mean kinetic energy obtained using the taper target. The lower panels show the surface eddy kinetic energy defined ~~as~~

$$\text{EKE} = \frac{1}{2}\overline{|\mathbf{u}|^2} - \frac{1}{2}|\bar{\mathbf{u}}|^2.$$

according to (39). It is clear that the negative weights in the taper filter lead to locally negative values of surface EKE.

The alternative definition $\overline{|\mathbf{u}'|^2}/2$ where $\mathbf{u}' = \mathbf{u} - \bar{\mathbf{u}}$ can also produce negative values of EKE when the filter has negative weights~~(not shown). The definition of EKE in equation (39) has the virtue that the total kinetic energy is exactly the sum of the mean and kinetic energies~~. As a simple example consider the case where $\mathbf{u}'$ is nonzero at only one grid point. Then $\overline{|\mathbf{u}'|^2}$ us proportional to the effective kernel centered at that point, and Figure 6 shows that the taper filter's effective kernel has negative weights.

3.5 Application to one-dimensional observational data

Our final example in Figure 11 illustrates the application of our filters to one-dimensional data, specifically along-track altimeter observations of absolute dynamic topography used to estimate cross-track geostrophic velocity. This example is included not only to highlight ~~only~~ additional capabilities of this filtering framework, but also to encourage its use on in-situ velocity or tracer measurements to permit scale-aware observational-model comparisons. We apply three filters (boxcar, Gaussian, and taper) to cross-track geostrophic velocity estimates along a single track of the Jason-2 altimeter located in the Western North Atlantic~~(figure 11)~~. Velocities are interpolated to 20 km spacing and then filtered to a 100 km filter scale. The upper panel ~~of Fig. 11~~ shows a single cycle of cross-track geostrophic velocity as a function of along-track distance moving north to south (grey lines show all cycles completed at 10 day intervals over a two year period). The single cycle (black) is then filtered using each ~~filter type three with EKE~~ of the three filter types with EKE shown in the lower panel. The three filters produce nearly indistinguishable large-scale fields, but the EKE defined according to equation (39), shown in the lower panel, displays notable differences. Specifically, the taper filter's negative weights lead to occasional negative values for EKE.

4 ~~Computational Cost~~

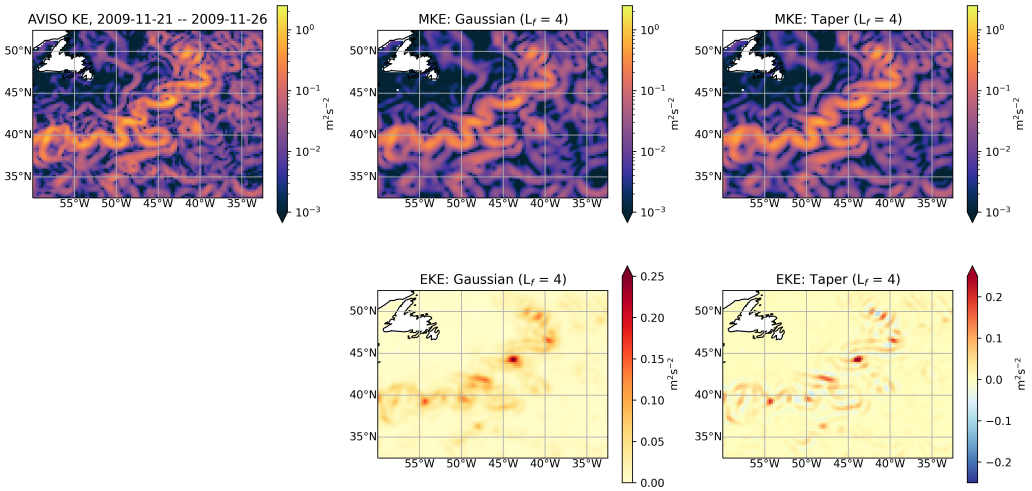

Figure 10. ~~Surface~~ The left panel shows surface kinetic energy calculated from absolute geostrophic velocities estimated using AVISO measurements of sea surface height. Velocities are provided on a 1/4° degree grid and filtered using a Gaussian (middle column) and taper (right column) simple fixed filter with filter scale 4 times the local grid scale. Definitions of mean kinetic energy (MKE) and eddy kinetic energy (EKE) are provided in the text.

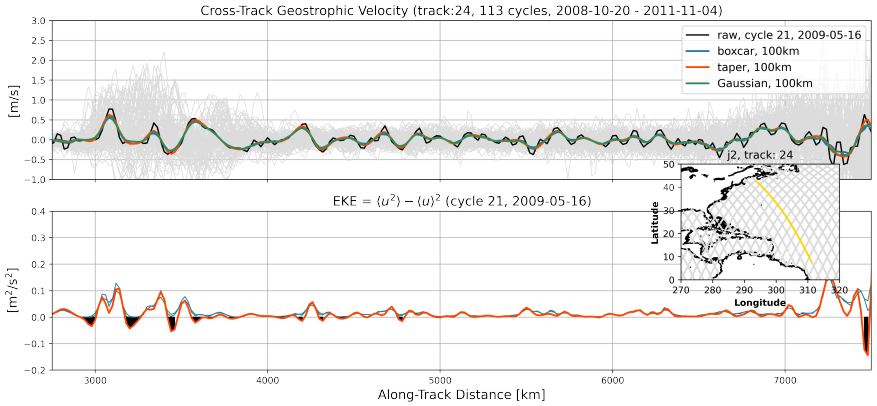

Figure 11. ~~Upper) Cross-track~~ The upper panel shows cross-track geostrophic velocities along the Jason-2 altimeter track number 24 spanning a two-year period (grey). A single cycle is selected (black) and filtered using the boxcar (blue), taper (orange), and ~~Gaussiam~~ Gaussian (green) filters to 100 km. The inset figure locates track 24 in the Western North Atlantic with along-track distance increasing north to south. ~~Lower) Eddy~~ The lower panel shows eddy kinetic energy defined using the cross-track geostrophic velocities above and filtered using boxcar, taper, and Gaussian filters. Shaded black regions identify locations of negative EKE associated with the taper filter.

The purpose of this section is to illustrate the speed of the filters proposed here, and to compare them to other methods from the literature. It bears noting that we compare to filters that are computing different but similar quantities, so the timings reported here should not be interpreted in the same vein as, say, timings for different implementations of a fast Fourier transform, where different implementations are all computing the exact same thing.

Fixed Factor SST Fixed Scale SST Fixed Factor T Fixed Scale TPython-CPU 6.14 s 64 s 84 s Fortran 0.39 s 278 s 28 s Python-GPU 1.59 s 1.19 s 1.64 s 18.1 s Mean wall times for fixed factor and fixed scale filters applied to temperature and SST data from a simulation of the POP model (Johnson et al., 2016).

We begin by comparing filters applied to temperature data from a single 5-day averaged output of the 0.1-degree POP simulation in (Johnson et al., 2016). We use our filters on both SST and on all 62 depth levels. We record times for our simple fixed factor filter with a Gaussian target and a filter scale 10 times larger than the local grid scale (using $N = 9$ stages), as well as for an isotropic Gaussian filter with a length scale of 1 degree of latitude, using $N = 90$ stages. For comparison we also record times for a parallel Fortran implementation of a convolution-based filter from Stanley, Bachman, and Grooms (2020). The Fortran code uses an isotropic Gaussian kernel with a fixed length scale of 1 degree of latitude, as well as an anisotropic Gaussian kernel with varying length scale equal to 10 times the local grid scale. These were both run on a clean node (no other processes)with two Intel Xeon E5-2680 v3 processors for a total of 24 cores. Tests were run 10 times each, and the mean times are reported in Table **??**. The Fortran code is significantly faster than the Python-based implementation of the simple fixed factor filter, though the gap narrows when filtering all 62 layers because of the Python code's improved ability to handle the embarrassingly-parallel nature of filtering 62 independent fields. In contrast, the Python code is significantly faster than the Fortran code when filtering to a fixed length scale.

The discrepancy can be attributed entirely to the way that the codes handle the POP tripole grid. The Python code only requires the ability to compute the Laplacian, which only requires the ability to pass information one cell deep across the tripolar seam in the Arctic ocean. In contrast, the Fortran implementation of the fixed-scale filter computes distance based on great circles, and handles the tripole grid by computing the filter kernel at *all* grid cells. This is an inefficient implementation, and with careful algorithm design the Fortran code's performance could be significantly improved. The point of this example though is not to show that the new methods are faster than any possible competitor, but rather to show that they are broadly comparable.

The Fortran and Python codes for the fixed-scale filter applied over all 62 layers were quite slow, as can be expected given the speed of applying the filters to a single layer, and times were not recorded.However, the Python code was ported to GPUs using the cupyPython library. The resulting code was run on a single node with an Nvidia V100 GPU, and the timings are reported in the third row of table **??**. The fixed factor filter applied to SST is approximately 4 times faster than on the CPU, but still fails to match the Fortran code. The fixed scale filter applied to SST is approximately 54 times faster than on the CPU, and approximately 230 times faster than the (admittedly inefficient) Fortran code. When the fixed factor code is applied over all 62 layers, the GPU code is approximately 50 times faster than the CPU code and 17 times faster than the Fortran code. The GPU Python code is the only one that completes the fixed scale filtering of all 62 layers in a reasonable time. This significant improvement on the GPU can be attributed to the internal parallelism that can be accessed for simple operations like a discrete Laplacian on a GPU, and indeed the

~~speedup increases as the problem size increases and more parallelism can be exploited.~~

~~A second point of reference was provided by comparing the new code to the Gaussian filter provided by scipy.ndimage.gaussian_filter. This filter assumes a regular and rectangular Cartesian grid, and is unable to handle continental boundaries except by filling them with synthetic data . This simplicity enables speed, and a somewhat comparable approach using our filters is to use the simple fixed factor Gaussian filter from section 2.6, and to fill values on land with zeros. The simple fixed factor filter with a no-flux treatment of boundariesis also included for comparison, and the filters are applied to the zonal velocity in the CM2.6 data from Fig. 8.~~

~~To compare these filters , we used a Gaussian with a filter scale 16 times larger than the local grid scale. For the Laplacian-based filter we varied the number of filter stages N, and for scipy.ndimage.gaussian_filterwe varied the number of standard deviations included in the kernel. (The Gaussian kernel is truncated to zero after a fixed number of standard deviations in scipy.ndimage.gaussian_filter.) The tests were run on a node with an Intel Xeon Platinum 8268 with 24 cores and process-level timing; results are shown in Fig. ??. The scipy.ndimage.gaussian_filterfilter is a little less than twice as fast as the Laplacianversion, and the no-flux treatment of boundaries adds approximately a factor of 3 in cost for the Laplacian version. Ultimately the new filters are of broadly comparable cost, but are far more flexible.~~

~~Timing of three filters applied to zonal velocity in the example from figure 8: Simple fixed factor Gaussian with no-flux at land boundaries (blue); Simple fixed factor Gaussian with zero-filled land values (orange); scipy.ndimage.gaussian_filter(green). The shaded bands indicate 1.96 standard deviations above and below the mean over 10 trials. The upper set of labels on the abscissa indicate the number of steps N used in the diffusion-based filters, while the lower labels indicate the number of standard deviations used in the Gaussian kernel by scipy. The stars indicate the default number of steps N in the diffusion-based filters, which guarantee an error of at most 1%. Units are seconds.~~

4 Conclusions

~~We have presented a new method for spatially filtering gridded data that only relies on the availability of a discrete Laplacian operator. The method involves repeated steps of the form (21b), and is therefore analogous to smoothing via diffusion~~ We have presented a new method for spatially filtering gridded data that only relies on the availability of a discrete Laplacian operator. The method involves repeated steps of the form (21b), and is therefore analogous to smoothing via diffusion. The new filters provide an efficient way of implementing something close to a Gaussian kernel convolution; they also allow the scale selectiveness (i.e. the shape) of the filter to be tuned as desired. As they require only the ability to apply a discrete Laplacian operator, these filters can be used with a wide range of data types, including output from models on unstructured grids, and gridded observational data sets. In the presence of boundaries, and when the filter scale varies over the domain, the new filters do not commute with spatial derivative. In domains without boundaries, the new filters will commute with any spatial derivative that commutes with the Laplacian, at least provided that the filter length scale does not change over the domain. If desired, ocean boundaries can be eliminated by treating values on land as zero, following Aluie et al. (2018). The basic method can be generalized to allow for anisotropic, i.e direction-dependent, as well as spatially-varying filter scales~~, and we have shown examples applying it to ocean model data and observational data, and demonstrated its computational efficiency~~. An open-source Python package implementing the method has been developed (?, ?). We note that related diffusion-based techniques have been

developed in the data assimilation community (e.g., ?, ?), though for the purpose of modeling correlations in the background error within a data assimilation system, rather than for spatial filtering analyses of model output. It is our hope that the new method and associated software will enable an increase in scale-dependent analysis of Earth system data, particularly for the purposes of subgrid-scale parameterization, though by no means limited to such.

Acknowledgments

We are grateful to J. Busecke for help with setting up the `gcm-filters` Python package, and to H. Khatri for discovering the numerical instability described in section 2.4. We thank A. Adcroft for helpful discussionson the design of spatially-varying filters. Data used in this article are available at ...; scripts used to generate the figures are available at .... A open-source python package implementing this algorithm, called `gcm-filters`, is currently under development (see `https://github.com/ocean-eddy-cpt/gcm-filters`). An early version of the package was used to generate the results in this paper. A paper describing the software itself is in preparation for Journal of Open Source Software, to coincide with the first release. In the present manuscript, our focus is the algorithm itself, not the implementation. We are grateful to F. Bryan for providing us with the output of the ~~simuations~~ simulations from (Johnson et al., 2016). I.G. and N.L. are supported by NSF OCE 1912332. [**other people put their grant numbers**]

Appendix A Solving the optimization problem to find the filter polynomial

We may find a polynomial that approximates the target filter by solving an optimization problem of the ~~following~~ form

$$p(s) = \arg\ \min \|\hat{G}_t(\sqrt{s}) - p(s)\|, \tag{A1}$$

where $s = k^2$ and p is a polynomial that must satisfy $p(0) = 1$. In order to enable rapid solution of this optimization problem it is convenient to use a weighted L^2 norm on $s \in [0, s_{\max}]$, where (as noted above) we may set $s_{\max} = k^2_{\max} = (\sqrt{d}\pi/\mathrm{d}x_{\min})^2$ where d is the dimension of the spatial domain. Using the Chebyshev norm is known to produce solutions that are close to the solution obtained from the max norm (Trefethen, 2019, theorem 16.1), so we adopt the Chebyshev norm

$$\|\hat{G}_t(\sqrt{s}) - p(s)\|^2_C = \int_0^{s_{\max}} \frac{(\hat{G}_t(\sqrt{s}) - p(s))^2}{\sqrt{s(s - s_{\max})}} \mathrm{d}s. \tag{A2}$$

The polynomial must satisfy $p(0) = 1$ in order to conserve the integral, and for convenience we also apply the condition $p(s_{\max}) = 0$. This allows us to solve the optimization problem using the Galerkin basis described by (Shen, 1995). To be precise, we let

$$p(s) = 1 - \frac{s}{s_{\max}} + \sum_{i=0}^{N-2} \hat{p}_i \phi_i(s), \tag{A3}$$

where $\phi_i(s)$ are the polynomial basis of Shen (1995), satisfying $\phi_i(0) = \phi_i(s_{\max}) = 0$, and $\phi_i(s)$ is a polynomial of degree $i + 2$. Collecting the Galerkin coefficients $\hat{p}_i$ into a vector $\hat{\mathbf{p}}$, standard arguments show that the optimal polynomial coefficients are given by the solution of

$$\mathbf{M}\hat{\mathbf{p}} = \mathbf{b}, \tag{A4}$$

where

$$M_{ij} = \langle \phi_i(s), \phi_j(s) \rangle_C \tag{A5}$$

$$b_i = \langle \phi_i(s), \hat{G}_t(\sqrt{s}) - 1 - \frac{s}{s_{\max}} \rangle_C, \tag{A6}$$

where and $\langle \cdot, \cdot \rangle_C$ denotes the Chebyshev inner product. The entries of $\mathbf{M}$ are known analytically (Shen, 1995), and the entries of $\mathbf{b}$ are computed using Gauss-Chebyshev quadrature with $N+1$ points.

Once a target filter $\hat{G}_t(k)$ has been specified, one must also choose the degree N of the polynomial p. As N increases the filter approaches the target filter, but at the same time the computational cost of the filter grows because applying the filter requires applying the discrete Laplacian N times. It is therefore desirable to choose some tradeoff between cost and accuracy. The Python package `gcm-filters` has a default setting for N that guarantees not more than 1% error in the difference between $\hat{G}_t$ and p; the user can also override this choice with any desired value of N.

Appendix B Commuting the filter and derivatives

This section explores conditions under which our filters commute with spatial derivatives, which was one of the main goals in the design of convolution-based spatial filters on the sphere in (Aluie, 2019)Aluie (2019). Filters with spatially-varying properties (cf. section Section 2.6) do not commute with derivatives, since they are analogous to integration against a spatially-varying kernel (i.e. equation (8)). We thus consider in this section only the versions of our filters with a fixed length scale. We first consider domains with boundaries, showing that our filters do not commute in this case, and then turn to the surface of a full sphere, without topographic boundaries.

Although our filters are defined entirely in discrete terms, it is natural to think in terms of the continuous limit, and this limit causes confusion. Consider for simplicity the case of the following filter for a scalar function $f(x)$ on $x \in [0, 1]$:

$$\bar{f} = \left(1 - \frac{1}{s_1}\Delta\right) f. \tag{B1}$$

This filter obviously commutes with derivatives, but it is in some sense not the correct continuous version of our discrete filter. The reason is that the discrete version always assumes no-flux boundary conditions on the data, because no other boundary condition is guaranteed to conserve the integral. Indeed the filter (B1) is not guaranteed to conserve the integral unless f satisfies no-flux (or periodic) boundary conditions. This is no limitation in the discrete case, since the no-flux Laplacian can be computed for any data. On the other hand, if one applies the discrete Laplacian with a no-flux assumption and then takes the limit of infinite resolution the result does not converge to Δf unless f actually satisfies no-flux boundary conditions. Instead, it converges to Δf plus Dirac delta distributions on the boundary. (This is analogous to the delta sheets of potential vorticity discussed by Bretherton (1966).)

In the correct continuous limit, equation (B1) is only defined for functions f that satisfy $f'(0) = f'(1) = 0$. With this more careful definition of the continuous limit of the filter, one can ask again whether it commutes with the spatial derivative. If one attempts to define $g(x) = f'(x)$ and then apply the filter to g, the result is not defined unless g also satisfies no-flux conditions, i.e. $f''(0) = f''(1) = 0$. So in the continuous limit, the filter will not commute with differentiation for functions with $f'' \neq 0$ on the boundaries. For higher-order filters the conditions for commutation are even more stringent, requiring derivatives up to high order to all be zero on the boundary.

An alternative perspective is afforded by the fact that our discrete filter is equivalent to a discrete kernel smoothing, per the arguments of section Section 2.8. In the presence of boundaries, the shape of the kernel varies in space, as can be seen in Fig. Figure 6. The continuous analog is integration against a spatially-varying kernel (equation (8)), which does not commute with spatial derivatives.

In the case without boundaries, e.g. on a sphere, there is no such difficulty. As long as the continuous differential operators commute (e.g. a Laplacian and a gradient), the discrete operators should also commute, at least up to discretization errors. The convolution-based spatial filters of Aluie (2019) only commute with derivatives in the absence of boundaries; this difficulty can be avoided by treating values outside the domain (e.g. on land) as zero (Aluie et al., 2018). A similar method can be used with our filters if desired: values outside the domain can be treated as zero (see right panel of Figure 9). The development in ~~section~~ Section 2.3 is based on a discrete approximation of a scalar Laplacian, or of the Laplace-Beltrami operator on a curved surface like the sphere. This can in principle be extended to vector Laplacians, for example if one wants a filter that commutes the filtering operation on a vector field with divergence or curl of the field, by simply replacing $\mathbf{L}$ with a discretization of the vector Laplacian.

References

Abernathey, R., Augspurger, T., Banihirwe, A., Blackmon-Luca, C., Crone, T., Gentemann, C., ... Signell, R. (2021). Cloud-native repositories for big scientific data. *Computing in Science and Engineering*(01), 1–1.

Aluie, H. (2019). Convolutions on the sphere: commutation with differential operators. *GEM-International Journal on Geomathematics*, *10*(1), 9.

Aluie, H., Hecht, M., & Vallis, G. K. (2018). Mapping the energy cascade in the North Atlantic Ocean: The coarse-graining approach. *J. Phys. Ocean.*, *48*(2), 225–244.

Bachman, S. D., Taylor, J., Adams, K., & Hosegood, P. (2017). Mesoscale and submesoscale effects on mixed layer depth in the Southern Ocean. *J. Phys. Ocean.*, *47*(9), 2173–2188.

Báez Vidal, A., Lehmkuhl, O., Trias, F. X., & Pérez-Segarra, C. D. (2016). On the properties of discrete spatial filters for CFD. *J. Comput. Phys.*, *326*, 474–498.

Berloff, P. S. (2005). On dynamically consistent eddy fluxes. *Dyn. Atmos. Oceans*, *38*(3-4), 123–146.

Berloff, P. S. (2018). Dynamically consistent parameterization of mesoscale eddies. Part III: Deterministic approach. *Ocean Model.*, *127*, 1–15.

Bolton, T., & Zanna, L. (2019). Applications of deep learning to ocean data inference and subgrid parameterization. *J. Adv. Model. Earth Syst.*, *11*(1), 376–399.

Bretherton, F. (1966). Critical layer instability in baroclinic flows. *Q. J. Roy. Meteor. Soc.*, *92*(393), 325–334.

Delworth, T., Rosati, A., Anderson, W., Adcroft, A., Balaji, V., Benson, R., ... R, Z. (2012). Simulated climate and climate change in the GFDL CM2.5 high-resolution coupled climate model. *J. Climate*, *25*(8), 2755-2781.

Germano, M. (1986). Differential filters for the large eddy numerical simulation of turbulent flows. *Phys. Fluids*, *29*(6), 1755–1757.

Griffies, S. M. (2015). A handbook for the GFDL CM2-0 model suite. *GFDL Climate Processes and Sensitivity Group, Technical Report 1*.

Grooms, I., & Kleiber, W. (2019). Diagnosing, modeling, and testing a multiplicative stochastic Gent-McWilliams parameterization. *Ocean Model.*, *133*, 1–10.

Grooms, I., Nadeau, L.-P., & Smith, K. S. (2013). Mesoscale eddy energy locality in an idealized ocean model. *J. Phys. Ocean.*, *43*.

Guedot, L., Lartigue, G., & Moureau, V. (2015). Design of implicit high-order filters on unstructured grids for the identification of large-scale features in large-eddy simulation and application to a swirl burner. *Phys. Fluids*, *27*(4), 045107.

Guillaumin, A., & Zanna, L. (2021). Stochastic deep learning parameterization of ocean momentum forcing. *Earth and Space Science Open Archive*, 31. doi: 10.1002/essoar.10506419.1

Haigh, M., Sun, L., Shevchenko, I., & Berloff, P. (2020). Tracer-based estimates of eddy-induced diffusivities. *Deep-sea Res. Pt. I*, 103264. doi: 10.1016/j.dsr.2020.103264

Hunter, J. K., & Nachtergaele, B. (2001). *Applied analysis*. World Scientific.

Johnson, B. K., Bryan, F. O., Grodsky, S. A., & Carton, J. A. (2016). Climatological annual cycle of the salinity budgets of the subtropical maxima. *J. Phys. Ocean.*, *46*(10), 2981–2994.

Khani, S., Jansen, M. F., & Adcroft, A. (2019). Diagnosing subgrid mesoscale eddy fluxes with and without topography. *J. Adv. Model. Earth Syst.*.

Lu, J., Wang, F., Liu, H., & Lin, P. (2016). Stationary mesoscale eddies, upgradient eddy fluxes, and the anisotropy of eddy diffusivity. *Geo. Res. Lett.*, *43*(2), 743–751.

Nadiga, B. (2008). Orientation of eddy fluxes in geostrophic turbulence. *Phil. Trans. R. Soc. A*, *366*(1875), 2489–2508.

Porta Mana, P., & Zanna, L. (2014). Toward a stochastic parameterization of ocean mesoscale eddies. *Ocean Model.*, *79*, 1–20.

Raymond, W. H. (1988). High-order low-pass implicit tangent filters for use in finite area calculations. *Mon. Weather Rev.*, *116*(11), 2132–2141.

Raymond, W. H., & Garder, A. (1991). A review of recursive and implicit filters. *Mon. Weather Rev.*, *119*(2), 477–495.

Robinson, G., & Grooms, I. (2020). A fast tunable blurring algorithm for scattered data. *SIAM J. Sci. Comput.*, *42*(4), A2281–A2299.

Ryzhov, E., Kondrashov, D., Agarwal, N., & Berloff, P. (2019). On data-driven augmentation of low-resolution ocean model dynamics. *Ocean Model.*, *142*, 101464.

Sadek, M., & Aluie, H. (2018). Extracting the spectrum of a flow by spatial filtering. *Phys. Rev. Fluids*, *3*(12), 124610.

Sagaut, P., & Grohens, R. (1999). Discrete filters for large eddy simulation. *Int. J. Numer. Meth. Fl.*, *31*(8), 1195–1220.

Shapiro, R. (1970). Smoothing, filtering, and boundary effects. *Rev. Geophy.*, *8*(2), 359–387.

Shen, J. (1995). Efficient spectral-Galerkin method II. Direct solvers of second-and fourth-order equations using Chebyshev polynomials. *SIAM J. Sci. Comput.*, *16*(1), 74–87.

Smith, R., Jones, P., Briegleb, B., Bryan, F., Danabasoglu, G., Dennis, J., ... others (2010). The parallel ocean program (POP) reference manual. *Los Alamos National Lab Technical Report*, *141*.

Stanley, Z., Bachman, S., & Grooms, I. (2020). Vertical structure of ocean mesoscale eddies with implications for parameterizations of tracer transport. *J. Adv. Model. Earth Syst.*, *12*(10), e2020MS002151.

Stanley, Z., Grooms, I., Kleiber, W., Bachman, S., Castruccio, F., & Adcroft, A. (2020). Parameterizing the impact of unresolved temperature variability on the large-scale density field: Part 1. Theory. *J. Adv. Model. Earth Syst.*, *12*(12), e2020MS002185.

Trefethen, L. (2019). *Approximation theory and approximation practice, extended edition*. SIAM, Philadelphia, USA.

Williams, P., Howe, N., Gregory, J., Smith, R., & Joshi, M. (2016). Improved climate simulations through a stochastic parameterization of ocean eddies. *J. Climate*, *29*, 8763–8781.

Zanna, L., & Bolton, T. (2020). Data-driven equation discovery of ocean mesoscale closures. *Geo. Res. Lett.*, *47*(17), e2020GL088376.