SIAM J. SCI. COMPUT.
Vol. 35, No. 3, pp. B576–B605

POSITIVITY PRESERVATION AND ADVECTION ALGORITHMS WITH APPLICATIONS TO EDGE PLASMA TURBULENCE*

J. L. PETERSON† AND G. W. HAMMETT‡

Abstract. The unique conditions of edge tokamak plasmas motivate the employment of efficient numerical methods that can robustly handle steep temperature and density gradients. To that end, we compare an Arakawa finite difference technique with some recent high-resolution upwind methods that are designed to minimize nonmonotonic overshoots while preserving the accuracy of solutions at smooth extrema. Versions of these algorithms are able to rigorously preserve positivity when that is physically relevant (such as for the advection of particle density or temperature). We explain in detail the use of these methods for the nonlinear Poisson bracket, an operator applicable to neutral fluid, gyrofluid, gyrokinetic, and general Hamiltonian simulations. For one-dimensional passive advection, the high-resolution upwind techniques maintain monotonicity and approach minimal levels of phase error and dissipation, especially at long wavelengths. In a two-dimensional incompressible Navier–Stokes vortex merging problem we find that extrema-preserving methods can resolve details at lower resolution than the Arakawa technique, are less dissipative than traditional finite volume methods while still minimizing overshoots and ensuring positivity, and model nonlinear cascade behavior fairly well without additional subgrid damping.

Key words. numerical simulation, fluid, advection, positivity, high-order upwind

AMS subject classifications. 76D05, 65M06, 65M08

DOI. 10.1137/120888053

1. Introduction.

The unique environment of the edge region of fusion plasmas complicates the study of turbulence. Analytic approximations made to simplify tokamak core simulations do not necessarily hold true. Fluctuation levels may grow as large as the background distribution function, thereby limiting the applicability of perturbative formulations. Furthermore, steep temperature and density gradients and the general range of scales in edge turbulence strain computational resources. As the density and temperature can vary by two orders of magnitude or more over very short distances in the edge region (for instance, from the top of the pedestal to the scrape-off layer or in the vicinity of plasma blobs), the resolution requirements to accurately model the system can be challenging. These large gradients can lead to many of the same difficulties, such as unphysical overshoots and negative solutions, that motivated the development of modern shock-capturing algorithms.

In choosing from the virtual zoo of numerical algorithms the best possible method for edge turbulence simulation, one should consider many different properties of the computational scheme. First of all, it is important to consider the various conservation properties of the method. How well does the scheme preserve the density, momentum,

*Submitted to the journal's Computational Methods in Science and Engineering section August 14, 2012; accepted for publication (in revised form) February 25, 2013; published electronically May 2, 2013. This work was supported by the Center for the Study of Plasma Microturbulence, a U.S. Department of Energy Scientific Discovery Through Advanced Computing project. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 and by the Princeton Plasma Physics Laboratory under contract DE-AC02-09CH11466.

http://www.siam.org/journals/sisc/35-3/88805.html

†Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA 94551 (peterson76@llnl.gov).

‡Princeton University, Princeton Plasma Physics Laboratory, P.O. Box 451, Princeton, NJ 08543 (hammett@pppl.gov).

energy, enstrophy and/or entropy of the system if physical dissipation is turned off? But one should also consider other important properties beyond integral conservation laws. For example, a second consideration is that one would like the system to satisfy certain physical constraints. Positivity-preserving algorithms can ensure that the temperature, density, and distribution function solutions never drop below zero. This is particularly important in the presence of shocks, blobs, and other steep-gradient features.

Third, the accuracy of the employed method is important. High-order schemes can be advantageous, as they reduce the resolution required to accurately model the physics of the system. Any reduction in resolution gets amplified according to the dimensions of the problem. A factor of 2 reduction in resolution leads to a factor of 2^d reduction in memory and a factor of $\approx 2^{d+1}$ reduction in computing time, which is a factor of ≈ 64 speedup for five-dimensional gyrokinetics. (Though the reduction in the number of grid points is partially offset by the increased work per grid point needed for higher-order methods, this can still be advantageous because many modern computers are limited by the bandwidth between memory and processor, so it is relatively efficient to do the extra calculations per memory access needed for higher-order methods). Finally, when dealing with turbulence simulations, one must consider the ability of the algorithm to deal with cascades in k-space.

The challenges with edge plasma turbulence simulation are not, it should be noted, confined to plasmas. The similarities between the equations governing drift-wave turbulence, two-dimensional (2D) neutral fluid turbulence, and general Hamiltonian systems, expand this problem to a much wider realm. While each physical problem has its own ideal algorithm, the lessons learned from simulating systems relevant to edge plasma turbulence can be applied to other disciplines.

Naulin and Nielsen [22] compared a spectral method, a well-known Arakawa finite difference method [1], and a third-order finite volume WENO method [19, 29] due to Kurganov and Levy [15], for the 2D high Reynolds number Navier–Stokes problem of vortex merging [25]. Naulin and Nielsen's conclusion was that the inherent numerical dissipation from the upwinding in the third-order WENO method was so large that it was relatively inefficient for high Reynolds number flows. Their analysis focused on the conservation properties of the algorithms. Over time, the dissipation of the WENO method damped out fine features in the vorticity, ω, thereby reducing the total enstrophy $\Omega = \frac{1}{2}\int \omega^2 dA$ significantly faster than viscosity alone would at high resolution. Since the Arakawa finite difference method is designed mathematically to conserve Ω [1], Naulin and Nielsen concluded that the Arakawa method is preferable to the third-order WENO scheme.

Our aim is to reexamine the vortex merger problem to test other aspects of numerical algorithms that may also be important for gyrokinetic and gyrofluid edge plasma simulations. In particular, while previous work focused on the conservation properties of algorithms, other properties such as positivity preservation and the modeling of cascades may be more important for edge turbulence applications. We extend previous comparisons to include other modern finite volume methods with lower dissipation than a third-order upwind method, including some recent high-order methods designed to reduce clipping of extrema by distinguishing smooth extrema from discontinuities. These are the piecewise parabolic method (PPM) (in both its original form [7] and a recent smooth extrema extension [6]), the high-order finite volume method of Suresh and Huynh [30], and a hybrid PPM/Suresh–Huynh method. Section 2 lays out the vortex merger problem while section 3 documents each algorithm

and some of their unique properties. We present results for inviscid one-dimensional passive advection and for full 2D nonlinear hydrodynamics in section 4.

2. The model system and general problem parameters. Due to the universality of nonlinear multidimensional advection, we take as our model system the same employed in [22], namely, a 2D neutral fluid vortex merger. We begin with the incompressible, unforced, 2D Navier–Stokes equations

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\vec{v} = \nu \nabla^2 \vec{v} - \nabla p. \tag{2.1}$$

Here, $\vec{v} = v_x \hat{x} + v_y \hat{y}$ is the velocity vector of the fluid confined in the x-y plane, and ν is the kinematic viscosity. (The mass density has been normalized to unity.) The scalar pressure, p, is governed by the incompressibility condition, $\nabla \cdot \vec{v} = 0$. One can apply the curl operator to express (2.1) in terms of the scalar vorticity, $\omega = (\nabla \times \vec{v}) \cdot \hat{z}$, and a stream function, $\psi : (\nabla \psi \times \hat{z} = \vec{v})$. The new system of equations is

$$\frac{\partial \omega}{\partial t} = -\left[\omega, \psi\right] + \nu \nabla^2 \omega, \tag{2.2}$$

$$\nabla^2 \psi = -\left(\omega + \omega_{\text{offset}}\right). \tag{2.3}$$

A Poisson equation, (2.3), relates the vorticity and the stream function. The constant ω_{offset} will be discussed in greater detail below. The bracket term, $[\omega, \psi]$, is the standard Poisson bracket, given by

$$[\omega, \psi] = \vec{v} \cdot \nabla \omega = \frac{\partial \omega}{\partial x}\frac{\partial \psi}{\partial y} - \frac{\partial \omega}{\partial y}\frac{\partial \psi}{\partial x}. \tag{2.4}$$

The Poisson bracket term is our canonical nonlinear advection term. While this formulation is specifically for neutral fluids, this nonlinear term is ubiquitous, since many systems, including the gyrokinetic equations for magnetized plasma turbulence, are described by Hamiltonian dynamics that can be expressed in terms of Poisson brackets [28]. For instance, in plasma simulations one is interested in the advection of plasma density n by the $\vec{E} \times \vec{B}$ drift velocity. In such cases $\vec{v}_{E\times B} \cdot \nabla n \propto [\Phi, n]$, where Φ is the electrostatic potential. As such, discretizations of (2.4) can be applied to plasma physics turbulence problems. All of the algorithms in section 3 represent (2.4) differently.

Following the details outlined in [22], we solve (2.2) and (2.3) on a doubly-periodic domain of dimensions 10×10 length units. The uniform grid spacing is $h = \Delta x = \Delta y = 10/N$, with N being the number of grid points in either the x or y directions. Grid indices in the x and y directions are indicated by i and j, respectively, with $i, j \in [0, N-1]$, so that $\omega_{i,j} \doteq \omega(x_i, y_i) = \omega(ih, jh)$.

As our focus is on the nonlinear Poisson bracket, (2.4), we use an explicit "4-point stencil" for the viscous term on the right-hand side of (2.2). Specifically we set

$$\nabla^2 \omega_{i,j} = \frac{1}{h^2}\left(\omega_{i-1,j} + \omega_{i+1,j} + \omega_{i,j-1} + \omega_{i,j+1} - 4\omega_{i,j}\right). \tag{2.5}$$

While the authors of [22] employ an implicit form for (2.5), there is no significant difference in solutions for the small time steps used in our simulations. Additionally, to solve the Poisson equation, (2.3), we use a spectral technique based on the FFTW package [8] that takes advantage of the fact that in k-space the inversion of (2.3) is algebraic.

The test problem in [22] consists of two merging Gaussian vortexes. They are initially 3 length units apart, have unit maximum, and are given in standard Gaussian form as

$$\omega_l(\vec{r}_{i,j}) = e^{-\|\vec{r}_{i,j}-\vec{r}_l\|^2/(2\sigma^2)}. \tag{2.6}$$

Explicitly, $\|\vec{r}_{i,j}\| = \sqrt{x_i^2 + y_i^2} = h\sqrt{i^2 + j^2}$ is a distance on the grid, $\vec{r}_l$ is the location of the center of Gaussian number l, and σ is the standard deviation of the monopole. Following [22] we define our Gaussian monopoles such that $2\sigma^2 = (0.8)^2$ or $\sigma = 0.8/\sqrt{2} \approx 0.566$. The initial positions of the two peaks are $\vec{r}_1 = (3.5, 5.0)$ and $\vec{r}_2 = (6.5, 5.0)$. After summing these two Gaussians to give the initial vorticity $\omega_{i,j} = \omega_1(\vec{r}_{i,j}) + \omega_2(\vec{r}_{i,j})$, we calculate a small offset to the Poisson equation equivalent to ensuring that the total net vorticity on the grid $\sum_{i,j}(\omega_{i,j} + \omega_{\text{offset}})$ has zero total circulation, so that the Poisson equation is invertible on a periodic domain. Specifically, this offset is written as

$$\omega_{\text{offset}} = -\frac{h^2}{L^2}\sum_{i,j}\omega_{i,j}. \tag{2.7}$$

While in two-dimensional hydrodynamics the local vorticity $\omega_{i,j}$ can have either sign, the equivalent Poisson equation in plasmas involves contributions from the ion and electron charge densities, both of which must individually be of only one sign, so positivity preservation of the ion and electron particle density is important.

Simulations are further parametrized by the Reynolds number, defined here as $\text{Re} = \sigma v^0_{\max}/\left(\sqrt{2}\nu\right)$, where $v^0_{\max}$ is the maximum scalar velocity based on the initial conditions and for this simulation is approximately 0.25. The one-dimensional (1D) passive advection test case in section 4.1 is inviscid ($\nu = 0$), and the fully nonlinear 2D problem in section 4.2 uses a high Reynolds number ($\text{Re} = 100,000$).

The time steps are variable during the course of the simulation and are designed to keep the Courant number at $CFL = v_{\max}\Delta t/h = 0.1$, well below the stability limit. At time step n, $\Delta t^n = 0.1h/v^{n-1}_{\max}$. Since the CFL condition is defined in terms of the maximum grid velocity, it is locally below this value in most cells.

3. The Poisson bracket algorithms. We compare six different algorithms for the spatial discretization of the Poisson bracket, (2.4). Like the authors of [22] we examine Arakawa finite differencing [1] and a third-order WENO scheme [15, 19, 29]. Additionally, we compare the behavior of a variant of PPM with and without a limiter designed to improve the accuracy at smooth extrema [6, 7]. We also consider a limiter method meant to decrease the diffusion of solutions at smooth extrema as laid out by Suresh and Huynh in [30]. While many of these high-resolution upwind algorithms have been tested for 1D linear and shock problems, here we also compare them side by side with Arakawa on a 2D nonlinear incompressible flow problem.

3.1. Arakawa finite differencing.

3.1.1. A second-order solution (Arakawa/Arakawa2). A classic implementation of the Poisson bracket, (2.4), is the finite difference method[1] proposed by Arakawa [1, 2]. It has the advantages of a straightforward, closed-form equation and of preserving discrete analogs of the exact linear and quadratic conservation properties of the Poisson bracket. That is, it guarantees conservation of the mean vorticity, which results from $\int d^2r\,[\psi,\omega]=0$, and conservation of both quadratic invariants of 2D hydrodynamics, the energy and the enstrophy Ω (the total squared vorticity, $\Omega=(1/2)\int d^2r\,|\omega|^2\to(1/2)h^2\sum_{i,j}|\omega_{i,j}|^2$), which result from the discrete equivalents of the properties $\int d^2r\,[\psi,\omega]\psi=0$ and $\int d^2r\,[\psi,\omega]\omega=0$. In Arakawa's method, the Poisson bracket is represented by

$$
\begin{aligned}
(3.1)\quad [\omega,\psi]=-\frac{1}{12h^2}\times\{&(\psi_{i,j-1}+\psi_{i+1,j-1}-\psi_{i,j+1}-\psi_{i+1,j+1})(\omega_{i+1,j}+\omega_{i,j})\\
&-(\psi_{i-1,j-1}+\psi_{i,j-1}-\psi_{i-1,j+1}-\psi_{i,j+1})(\omega_{i-1,j}+\omega_{i,j})\\
&+(\psi_{i+1,j}+\psi_{i+1,j+1}-\psi_{i-1,j}-\psi_{i-1,j+1})(\omega_{i,j+1}+\omega_{i,j})\\
&-(\psi_{i+1,j-1}+\psi_{i+1,j}-\psi_{i-1,j-1}-\psi_{i-1,j})(\omega_{i,j-1}+\omega_{i,j})\\
&+(\psi_{i+1,j}-\psi_{i,j+1})(\omega_{i+1,j+1}+\omega_{i,j})\\
&-(\psi_{i,j-1}-\psi_{i-1,j})(\omega_{i-1,j-1}+\omega_{i,j})\\
&+(\psi_{i,j+1}-\psi_{i-1,j})(\omega_{i-1,j+1}+\omega_{i,j})\\
&-(\psi_{i+1,j}-\psi_{i,j-1})(\omega_{i+1,j-1}+\omega_{i,j})\}.
\end{aligned}
$$

Note that one can consider a 1D passive advection limit, where $\psi=vy$ and ω is independent of y, in which case the Arakawa formula applied to (2.2) reduces to

$$
(3.2)\qquad \frac{\partial\omega_i}{\partial t}=-v\frac{\omega_{i+1}-\omega_{i-1}}{2\Delta x}+\nu\frac{\partial^2\omega}{\partial x^2},
$$

which is just the standard form for centered second-order finite differencing of the advection term. It is well known that this expression can lead to unphysical oscillations in the solution, particularly in regions of sharp gradients compared to the grid resolution, unless the diffusion term is sufficiently strong [17].

3.1.2. A higher-order Arakawa solution (Arakawa4). It is possible to use Richardson extrapolation [26, 27] to construct a higher-order Arakawa discretization. The lower-order Arakawa discretization is a symmetric approximation to the Poisson bracket on a grid of size h:

$$
(3.3)\qquad [\omega,\psi]_h=[\omega,\psi]_{\text{True}}+C_2h^2+C_4h^4+\cdots.
$$

In this case $[\omega,\psi]_{\text{True}}$ is the true value of the Poisson bracket, and the second term represents the error in the discretization, which is proportional to h^2 and scales with

[1] Although written here as finite differences, the Arakawa algorithm can be considered as having a generalized finite volume form, where the first four lines in (3.1) correspond to the usual fluxes, between cell (i,j) and cells to its east, west, north, and south, but the next four lines represent some kind of corner fluxes, between the center cell and cells to the NE, SW, NW, and SE. In principle one could apply limiters to these fluxes to satisfy monotonicity constraints as one does with other finite volume methods, but that would break the quadratic conservation properties of the Arakawa method. Following [22], we use the finite difference form without limiters.

a constant factor C_2. By applying the same algorithm but on a larger grid spacing, we can equally write

$$[\omega,\psi]_{2h} = [\omega,\psi]_{\text{True}} + 4C_2h^2 + 16C_4h^4 + \cdots, \tag{3.4}$$

which can be obtained from (3.1) by the transformation $h \Rightarrow 2h$ and $\{i,j\} \pm 1 \Rightarrow \{i,j\} \pm 2$. These can be combined algebraically to eliminate the second-order error term, thus giving us a new Poisson bracket discretization that has an error proportional to the fourth power of the grid spacing, much like many of the higher-order finite volume methods presented in the following section. Explicitly this becomes

$$\frac{4}{3}[\omega,\psi]_h - \frac{1}{3}[\omega,\psi]_{2h} = [\omega,\psi]_{\text{True}} - 4C_4h^4 + \cdots. \tag{3.5}$$

The discretization given by (3.5) is a fourth-order Arakawa finite difference approximation to the Poisson bracket with the same conservation properties as the lower-order method detailed above. The grid stencil is now five points in any direction, the same as the higher-order finite volume methods, with similar implications for parallelization.

Since the original vortex merging problem in [22] used the second-order Arakawa method, we focus in subsequent sections primarily on the second-order method. Generic references to "Arakawa" refer to the second-order method, although in some places we mark it as "Arakawa2" when comparing with the fourth-order variant, which is marked as "Arakawa4."

3.2. Finite volume. While finite difference methods focus on the solution at points, finite volume techniques evolve the integral-averaged quantity within a grid cell. Specifically, we average (2.2) over a grid cell to obtain the following conservative equation for the volume-averaged vorticity, $\bar{\omega}_{i,j} \equiv h^{-2}\int\int_{i,j}\omega dxdy$:

$$\frac{\partial\bar{\omega}_{i,j}}{\partial t} = \nu\nabla^2\bar{\omega}_{i,j} - \frac{1}{h}\left(F^x_{i+1/2,j} - F^x_{i-1/2,j} + F^y_{i,,j+1/2} - F^y_{i,j-1/2}\right). \tag{3.6}$$

The form of (3.6) is the same as (2.2), with the Poisson bracket term represented as a divergence of fluxes. Fundamentally we are evolving the same equation, but philosophically it is important to remember that these are volume-averaged vorticities, not necessarily the point-value vorticity at the grid center. The fluxes through the grid cell are defined in terms of the velocities and vorticities along grid edges,

$$F^x_{i+1/2,j} = v^x_{i+1/2,j} \times \omega^*_{i+1/2,j}, \tag{3.7}$$

$$F^y_{i,j+1/2} = v^y_{i,j+1/2} \times \omega^*_{i,j+1/2}. \tag{3.8}$$

The choice of ω^*depends upon the direction of the velocity. Finite volume algorithms allow discontinuities at the grid boundaries, so along each grid edge there exist two values for the vorticity, depending upon which side of the interface one looks. Specifically, let us define the vorticity in grid cell (i,j) at the boundaries of that cell $(i \pm 1/2, j \pm 1/2)$ as

$$\omega^R_{i,j} = \omega_{i+1/2,j}, \tag{3.9}$$

$$\omega^L_{i,j} = \omega_{i-1/2,j}, \tag{3.10}$$

$$\omega^U_{i,j} = \omega_{i,j+1/2}, \tag{3.11}$$

$$\omega^D_{i,j} = \omega_{i,j-1/2}. \tag{3.12}$$

The heart of every finite volume method lies in the determination of these "right," "left," "up," and "down" edge vorticities $\{\omega^R_{i,j}, \omega^L_{i,j}, \omega^U_{i,j}, \omega^D_{i,j}\}$. Once calculated, we choose ω^* to be the edge value of the vorticity in the cell from which a fluid element would originate (the "upwind" direction). Mathematically, this is represented in the x-direction as

$$\omega^*_{i+1/2,j} = \begin{cases} \omega^R_{i,j}, & v^x_{i+1/2,j} > 0, \\ \omega^L_{i+1,j}, & v^x_{i+1/2,j} < 0, \end{cases} \tag{3.13}$$

$$\omega^*_{i-1/2,j} = \begin{cases} \omega^R_{i-1,j}, & v^x_{i-1/2,j} > 0, \\ \omega^L_{i,j}, & v^x_{i-1/2,j} < 0. \end{cases} \tag{3.14}$$

The y-direction is symmetric under the transformation $\{R, L, i, j\} \to \{U, D, j, i\}$:

$$\omega^*_{i,j+1/2} = \begin{cases} \omega^U_{i,j}, & v^y_{i,j+1/2} > 0, \\ \omega^D_{i,j+1}, & v^y_{i,j+1/2} < 0, \end{cases} \tag{3.15}$$

$$\omega^*_{i,j-1/2} = \begin{cases} \omega^U_{i,j-1}, & v^y_{i,j-1/2} > 0, \\ \omega^D_{i,j}, & v^y_{i,j-1/2} < 0. \end{cases} \tag{3.16}$$

Proceeding with any finite volume method involves the calculation of the vorticities $\{\omega^R_{i,j}, \omega^L_{i,j}, \omega^U_{i,j}, \omega^D_{i,j}\}$ and the edge velocities $\{v^x_{i\pm1/2,j}, v^y_{i,j\pm1/2}\}$. The velocity calculation is straightforwardly based upon the stream function. We use a centered discretization for the velocities $\vec{v} = \nabla\psi \times \hat{z}$ and define $v^x_{i,j} = (\psi_{i,j+1} - \psi_{i,j-1})/2h$ and $v^y_{i,j} = -(\psi_{i+1,j} - \psi_{i-1,j})/2h$. We then interpolate linearly to the cell edges

$$v^x_{i\pm1/2,j} = \frac{1}{2}\left(v^x_{i,j} + v^x_{i\pm1,j}\right), \tag{3.17}$$

$$v^y_{i,j\pm1/2} = \frac{1}{2}\left(v^y_{i,j} + v^y_{i,j\pm1}\right). \tag{3.18}$$

Although we will be using higher-order methods for the interpolations of ω to the cell faces, the final algorithm is formally only second-order accurate in multiple dimensions because we are using simple midpoint evaluations of the face-averaged fluxes such as $F^x_{i+1/2,j} = \langle v^x \omega^* \rangle_{i+1/2,j} = (1/h) \int_{y_{j-1/2}}^{y_{j+1/2}} dy\, v^x(x_{i+1/2}, y) \omega^*(x_{i+1/2}, y) \approx v^x_{i+1/2,j} \omega^*_{i+1/2,j}$. The higher-order aspects of the ω interpolations are still useful for reducing numerical dissipation. (Following up on a suggestion by one of the reviewers, on nonuniform grids, it would probably be better to interpolate the streamfunction to the corners of the grid cells first and then difference across a face to determine the velocity through that face. This would preserve the divergence-free property of the flow.) Extensions to full higher-order accuracy can be done; see, for example, [5].

The edge vorticity calculation is unique to each finite volume method. However, all techniques involve two basic steps:

1. Interpolate from cell-averaged vorticities to cell boundaries. This is usually accomplished by fitting surrounding vorticity values to a high-order polynomial function. Let us call these estimates $\{\omega^{\dagger R}_{i,j}, \omega^{\dagger L}_{i,j}, \omega^{\dagger U}_{i,j}, \omega^{\dagger D}_{i,j}\}$.
2. Limit these initial interpolations, $\{\omega^{\dagger R}_{i,j}, \omega^{\dagger L}_{i,j}, \omega^{\dagger U}_{i,j}, \omega^{\dagger D}_{i,j}\}$, according to a set of predefined conditions. These initial estimates may (or may not) be altered during the process, but the result is the set $\{\omega^R_{i,j}, \omega^L_{i,j}, \omega^U_{i,j}, \omega^D_{i,j}\}$ to be used in (3.13)–(3.16). Most often this is to prevent overshoots and undershoots in the solution, but the exact mechanism varies with the algorithm.

Once one has determined the edge velocities $\{v^x_{i\pm1/2,j}, v^y_{i,j\pm1/2}\}$ from (3.17) and (3.18) and $\{\omega^R_{i,j}, \omega^L_{i,j}, \omega^U_{i,j}, \omega^D_{i,j}\}$ from the method-specific interpolation/limiting algorithm, the procedure continues by using (3.13)–(3.16) to determine the values of ω^*. Hence, one can use (3.7) and (3.8) to determine the appropriate flux in (3.6). The coding to evolve the equation in time is identical for a finite volume method and a finite difference equation, with the Poisson bracket term in (2.2) replaced by the flux terms in (3.6).

We now describe the algorithms for the interpolation and limiting of edge-valued vorticities in the x-direction. The y-direction is symmetric under the transformation: $\{R, L, i, j\} \to \{U, D, j, i\}$.

3.2.1. Third-order WENO (WENO3). We employ the third-order WENO method as suggested by Kurganov and Levy [15]. The interpolations are parabolic, while the limiters rigorously preserve monotonicity and reduce to piecewise constant values at extrema. First, we determine if the current grid cell vorticity is either a maximum or a minimum. If so, we set the edge vorticity values to be equal to the cell-averaged value:

$$\begin{aligned} &\text{if sign}\left(\bar\omega_{i+1,j} - \bar\omega_{i,j}\right) \neq \text{sign}\left(\bar\omega_{i,j} - \bar\omega_{i-1,j}\right) \text{ then} \\ &\qquad \omega^R_{i,j} = \omega^L_{i,j} = \bar\omega_{i,j}. \end{aligned} \tag{3.19}$$

Otherwise, we are not at an extremum and we proceed with our parabolic interpolations for $\{\omega^{\dagger R}_{i,j}, \omega^{\dagger L}_{i,j}\}$. Defining $\Delta^1_i = (\bar\omega_{i+1,j} - \bar\omega_{i-1,j})$ and $\Delta^2_i = (\bar\omega_{i+1,j} - 2\bar\omega_{i,j} + \bar\omega_{i-1,j})$ we have

$$\omega^{\dagger R}_{i,j} = \bar\omega_{i,j} + \frac{1}{4}\Delta^1_i + \frac{1}{12}\Delta^2_i, \tag{3.20}$$

$$\omega^{\dagger L}_{i,j} = \bar\omega_{i,j} - \frac{1}{4}\Delta^1_i + \frac{1}{12}\Delta^2_i. \tag{3.21}$$

The limiters to ensure that rigorously no overshoots and undershoots appear in the solution proceed thusly:

$$\text{if sign}\left(\bar\omega_{i,j} - \omega^{\dagger L}_{i,j}\right) \neq \text{sign}\left(\omega^{\dagger L}_{i,j} - \bar\omega_{i-1,j}\right) \text{ then}$$

$$\omega^L_{i,j} = \bar\omega_{i-1,j}, \tag{3.22}$$

$$\omega^R_{i,j} = \bar\omega_{i-1,j} + \frac{5}{2}\left(\bar\omega_{i,j} - \bar\omega_{i-1,j}\right), \tag{3.23}$$

$$\text{else if sign}\left(\bar\omega_{i,j} - \omega^{\dagger R}_{i,j}\right) \neq \text{sign}\left(\omega^{\dagger R}_{i,j} - \bar\omega_{i+1,j}\right) \text{ then}$$

$$\omega^R_{i,j} = \bar\omega_{i+1,j}, \tag{3.24}$$

$$\omega^L_{i,j} = \bar\omega_{i+1,j} + \frac{5}{2}\left(\bar\omega_{i,j} - \bar\omega_{i+1,j}\right). \tag{3.25}$$

If neither of these conditions is met, the limiter is not triggered and the initial estimates are used: $\omega^{R,L}_{i,j} = \omega^{\dagger R,L}_{i,j}$.

The limiters in the WENO3 scheme serve to ensure that edge vorticity values lie between the values of adjacent cells and rigorously preserve this monotonicity.

3.2.2. The piecewise parabolic method (PPM4). Originally developed by Colella and Woodward [7], PPM is a finite volume method that seeks solutions comprised of local parabolas. Like the WENO3 method in section 3.2.1, it becomes piecewise constant at extrema. We use a fourth-order variant of PPM (PPM4) for the

spatial discretizations, as described in [6]. As with all of the other spatial-differencing algorithms, we will use the Runge–Kutta (RK) time-advancement method described in section 3.4 (instead of the original PPM time-advancement method), as was also done in [5]. In our notation the initial interpolation becomes

$$\omega_{i,j}^{\dagger R} = \frac{7}{12}\left(\bar{\omega}_{i,j} + \bar{\omega}_{i+1,j}\right) - \frac{1}{12}\left(\bar{\omega}_{i-1,j} + \bar{\omega}_{i+2,j}\right), \tag{3.26}$$

$$\omega_{i,j}^{\dagger L} = \frac{7}{12}\left(\bar{\omega}_{i,j} + \bar{\omega}_{i-1,j}\right) - \frac{1}{12}\left(\bar{\omega}_{i-2,j} + \bar{\omega}_{i+1,j}\right). \tag{3.27}$$

Next, PPM4 completes the interpolation by applying monotonicity constraints to eliminate solution oscillations. These limiters are triggered when monotonicity is violated. Explicitly these conditions are:

$$\begin{aligned} &\text{if } \left(\bar{\omega}_{i,j} - \omega_{i,j}^{\dagger R}\right)\left(\bar{\omega}_{i+1,j} - \omega_{i,j}^{\dagger R}\right) > 0 \text{ then} \\ &\qquad \omega_{i,j}^{\dagger R} = \left(\bar{\omega}_{i,j} + \bar{\omega}_{i+1,j}\right)/2 - \left(\delta\omega_{i+1}^{*} - \delta\omega_{i}^{*}\right)/6, \end{aligned} \tag{3.28}$$

$$\begin{aligned} &\text{if } \left(\bar{\omega}_{i-1,j} - \omega_{i,j}^{\dagger L}\right)\left(\bar{\omega}_{i,j} - \omega_{i,j}^{\dagger L}\right) > 0 \text{ then} \\ &\qquad \omega_{i,j}^{\dagger L} = \left(\bar{\omega}_{i-1,j} + \bar{\omega}_{i,j}\right)/2 - \left(\delta\omega_{i}^{*} - \delta\omega_{i-1}^{*}\right)/6. \end{aligned} \tag{3.29}$$

We define

$$\delta\omega_i^* = \begin{cases} \operatorname{sign}\left(\Delta_i^1\right)\min\left(\frac{1}{2}\left|\Delta_i^1\right|, 2\left|\Delta_i^L\right|, 2\left|\Delta_i^R\right|\right), & \Delta_i^R\Delta_i^L > 0 \\ 0, & \text{otherwise,} \end{cases} \tag{3.30}$$

where $\Delta_i^1 = \bar{\omega}_{i+1,j} - \bar{\omega}_{i-1,j}$, $\Delta_i^L = \bar{\omega}_{i,j} - \bar{\omega}_{i-1,j}$, and $\Delta_i^R = \bar{\omega}_{i+1,j} - \bar{\omega}_{i,j}$. As mentioned, limiters only apply when monotonicity is violated in the edge interpolations. In smooth regions away from extrema, there will be no change to $\omega_{i,j}^{\dagger\{R,L\}}$ from (3.28)–(3.29).

The next step in the PPM4 method is to apply additional limiters, which depend on whether or not the solution is a local extremum, defined not only by the averaged values, $\bar{\omega}$, but also by the initial interpolations, $\omega^{\dagger}$. Specifically, the solution is a local maximum or minimum if

$$\left(\omega_{i,j}^{\dagger R} - \bar{\omega}_{i,j}\right)\left(\bar{\omega}_{i,j} - \omega_{i,j}^{\dagger L}\right) \le 0 \text{ or } \left(\bar{\omega}_{i-1,j} - \bar{\omega}_{i,j}\right)\left(\bar{\omega}_{i,j} - \bar{\omega}_{i+1,j}\right) \le 0. \tag{3.31}$$

If the initial interpolation satisfies (3.31), we apply the piecewise constant limiters $\omega_{i,j}^R = \omega_{i,j}^L = \bar{\omega}_{i,j}$, like (3.19).

Away from extrema we apply further limiters that constrain the amount of variation in the local reconstructions. We test and limit each of the edge interpolations $\{\omega_{i,j}^{\dagger R}, \omega_{i,j}^{\dagger L}\}$ using the logic

$$\begin{aligned} &\text{if } \left|\omega_{i,j}^{\dagger R,L} - \bar{\omega}_{i,j}\right| \ge n\left|\bar{\omega}_{i,j} - \omega_{i,j}^{\dagger L,R}\right| \\ &\qquad \text{then } \omega_{i,j}^{R,L} = \bar{\omega}_{i,j} + n\left(\bar{\omega}_{i,j} - \omega_{i,j}^{\dagger L,R}\right) \\ &\qquad \text{else } \omega_{i,j}^{R,L} = \omega_{i,j}^{\dagger R,L}. \end{aligned} \tag{3.32}$$

The parameter n here controls how much variation between the edge interpolations and the cell average is allowed before limiting the solution. (If the solution varies

too much, that is used as a sign that we are near a shock or underresolved region and the limiter allows a discontinuity to form in the reconstructed solution across the cell edge.) Higher n allows higher-order accuracy in smooth regions, but requires a smaller time step and delays the switch to allowing discontinuities when needed. We have used $n = 3$ here to avoid limiting the accuracy of monotonic cubic interpolations. It has been pointed out [20] that $n = 2$ can be used (as is done in the original PPM) with only rare and slight loss of local accuracy (because locally quadratic behavior near flat regions is more generic than locally cubic behavior, in the sense that locally cubic behavior requires both the first and second derivative to vanish simultaneously at a point, which in many cases might be rare).

While more complex limiters for the parabolic interpolant exist [6], we follow [5] and [20] and choose the limiters described above.

3.2.3. Smooth extrema preserving piecewise parabolic method (XPPM). A major limitation of many finite volume methods is a reduction of accuracy at smooth extrema [6, 30]. As noted above, many algorithms resort to first-order (piecewise constant) solutions at local maxima and minima. One method of improving the performance of these methods is to apply a new limiter at extrema, to replace (3.31). A recently developed limiter [6] uses estimates of the second derivative to relax the constraints on solutions at extrema, thereby allowing for solutions with higher accuracy.

The method is identical to that described in section 3.2.2, with the exception of the limiters. Interpolation is done according to (3.26) and (3.27). But instead of (3.28) and (3.29), we use the following (which corrects a typo in (19) of [6]):

$$\text{if } \left(\bar{\omega}_{i,j} - \omega^{\dagger R}_{i,j}\right)\left(\bar{\omega}_{i+1,j} - \omega^{\dagger R}_{i,j}\right) > 0 \text{ then}$$

$$\omega^{\dagger R}_{i,j} = \frac{1}{2}\left(\bar{\omega}_{i,j} + \bar{\omega}_{i+1,j}\right) - \frac{h^2}{6}\left(D^2\omega\right)_{i+1/2,j,\lim}, \tag{3.33}$$

$$\text{if } \left(\bar{\omega}_{i-1,j} - \omega^{\dagger L}_{i,j}\right)\left(\bar{\omega}_{i,j} - \omega^{\dagger L}_{i,j}\right) > 0 \text{ then}$$

$$\omega^{\dagger L}_{i,j} = \frac{1}{2}\left(\bar{\omega}_{i-1,j} + \bar{\omega}_{i,j}\right) - \frac{h^2}{6}\left(D^2\omega\right)_{i-1/2,j,\lim}. \tag{3.34}$$

The value of $(D^2\omega)_{\{i,j\}\pm 1/2,\lim}$ is determined from approximations to the second derivative:

$$\begin{aligned}
\left(D^2\omega\right)_{i+1/2,j} &= \frac{3}{h^2}\left(\bar{\omega}_{i,j} - 2\omega^{\dagger R}_{i,j} + \bar{\omega}_{i+1,j}\right),\\
\left(D^2\omega\right)_{i+1/2,j,L} &= \frac{1}{h^2}\left(\bar{\omega}_{i-1,j} - 2\bar{\omega}_{i,j} + \bar{\omega}_{i+1,j}\right),\\
\left(D^2\omega\right)_{i+1/2,j,R} &= \frac{1}{h^2}\left(\bar{\omega}_{i,j} - 2\bar{\omega}_{i+1,j} + \bar{\omega}_{i+2,j}\right).
\end{aligned} \tag{3.35}$$

If $(D^2\omega)_{i+1/2,j}$, $(D^2\omega)_{i+1/2,j\{L,R\}}$ all have the same sign, then

$$\begin{aligned}
\left(D^2\omega\right)_{i+1/2,j,\lim} &= s\,\min\left\{C\left|\left(D^2\omega\right)_{i+1/2,j,L}\right|, C\left|\left(D^2\omega\right)_{i+1/2,j,R}\right|, \left|\left(D^2\omega\right)_{i+1/2,j}\right|\right\};\\
s &= \operatorname{sign}\left(\left(D^2\omega\right)_{i+1/2,j}\right).
\end{aligned} \tag{3.36}$$

Otherwise, $(D^2\omega)_{i+1/2,j,\lim} = 0$. A value for the constant C moderately higher than 1 allows the algorithm to use the most accurate estimate of the second derivative

in relatively smooth regions where all estimates of the second derivative are similar, while turning on limiters when there are significant oscillations near the grid scale. Following [6] we select $C = 1.25$. To determine $(D^2\omega)_{i-1/2,j,\mathrm{lim}}$, let $i \to i-1$ and $\omega^{\dagger R}_{i,j} \to \omega^{\dagger L}_{i,j}$.

While modifications to limiters away from extrema exist [6], we follow [5] and [20] and focus only on algorithmic modifications at extrema as defined by (3.31). Recent work on further improvements to the XPPM approach can be found in [21].

Away from these points our implementation of the XPPM method is identical to the PPM4 algorithm with (3.32). The differences occur near extrema. When (3.31) is satisfied, instead of proceeding with (3.19) (as does the PPM4 algoritihm), the XPPM algorithm uses its own smooth extrema limiter [6], which begins with several approximations to the second derivative:

$$\begin{aligned}
\left(D^2\omega\right)_{i,j} &= \frac{6}{h^2}\left(\omega^{\dagger L}_{i,j} - 2\bar{\omega}_{i,j} + \omega^{\dagger R}_{i,j}\right),\\
\left(D^2\omega\right)_{i,j,C} &= \frac{1}{h^2}\left(\bar{\omega}_{i-1,j} - 2\bar{\omega}_{i,j} + \bar{\omega}_{i+1,j}\right),\\
\left(D^2\omega\right)_{i,j,L} &= \frac{1}{h^2}\left(\bar{\omega}_{i-2,j} - 2\bar{\omega}_{i-1,j} + \bar{\omega}_{i,j}\right),\\
\left(D^2\omega\right)_{i,j,R} &= \frac{1}{h^2}\left(\bar{\omega}_{i,j} - 2\bar{\omega}_{i+1,j} + \bar{\omega}_{i+2,j}\right).
\end{aligned} \tag{3.37}$$

If $(D^2\omega)_{i,j}$, $(D^2\omega)_{i,j\{C,L,R\}}$ all have the same sign, then

$$\left(D^2\omega\right)_{i,j,\mathrm{lim}} = s\,\min\left\{C\left|\left(D^2\omega\right)_{i,j,L}\right|, C\left|\left(D^2\omega\right)_{i,j,R}\right|, C\left|\left(D^2\omega\right)_{i,j,C}\right|, \left|\left(D^2\omega\right)_{i,j}\right|\right\}. \tag{3.38}$$

Otherwise, $(D^2\omega)_{i,j,\mathrm{lim}} = 0$. The coefficient $s \equiv \mathrm{sign}[(D^2\omega)_{i,j}]$, and C is defined as above, after (3.36). The final limiter is

$$\omega^{R,L}_{i,j} = \begin{cases} \bar{\omega}_{i,j} + \left(\omega^{\dagger R,L}_{i,j} - \bar{\omega}_{i,j}\right)\frac{\left(D^2\omega\right)_{i,j,\mathrm{lim}}}{\left(D^2\omega\right)_{i,j}}, & \left(D^2\omega\right)_{i,j} \neq 0,\\ \bar{\omega}_{i,j}, & \left(D^2\omega\right)_{i,j} = 0. \end{cases} \tag{3.39}$$

The net effect of this is that at smooth extrema, the limiters turn off and the high-order interpolation for $\omega^{\dagger R,L}_{i,j}$ is used, while at poorly resolved nonsmooth extrema, it switches to approach a piecewise constant reconstruction.

3.2.4. Fifth-order Suresh–Huynh limiters (SuHu). Another method to ameliorate the loss of accuracy at extrema comes from Suresh and Huynh [30]. The technique uses geometrical arguments to maintain the order of the underlying interpolation method at smooth local extrema and guarantees global positivity for linear problems. It should be noted that no such guarantee is made on nonlinear problems such as the Navier–Stokes problem here. While it is possible to apply the SuHu limiter to any initial edge interpolation, we employ a quartic polynomial reconstruction that results in a fifth-order method,

$$\omega^{\dagger R}_{i,j} = \frac{1}{60}\left(2\bar{\omega}_{i-2,j} - 13\bar{\omega}_{i-1,j} + 47\bar{\omega}_{i,j} + 27\bar{\omega}_{i+1,j} - 3\bar{\omega}_{i+2,j}\right), \tag{3.40}$$

$$\omega^{\dagger L}_{i,j} = \frac{1}{60}\left(2\bar{\omega}_{i+2,j} - 13\bar{\omega}_{i+1,j} + 47\bar{\omega}_{i,j} + 27\bar{\omega}_{i-1,j} - 3\bar{\omega}_{i-2,j}\right). \tag{3.41}$$

As shown by (3.40) and (3.41), the polynomial interpolations are symmetric.

After interpolation, each edge value is limited by a function of the vorticities of the cell and its neighbors. Let us call that algorithm $l(\omega^\dagger, \omega_{-2}, \omega_{-1}, \omega_0, \omega_{+1}, \omega_{+2})$. Symmetry allows the same function to be used for each of the edge values $\{\omega^R_{i,j}, \omega^L_{i,j}, \omega^U_{i,j}, \omega^D_{i,j}\}$:

$$\omega^R_{i,j} = l\left(\omega^{\dagger R}_{i,j}, \bar{\omega}_{i-2,j}, \bar{\omega}_{i-1,j}, \bar{\omega}_{i,j}, \bar{\omega}_{i+1,j}, \bar{\omega}_{i+2,j}\right), \tag{3.42}$$

$$\omega^L_{i,j} = l\left(\omega^{\dagger L}_{i,j}, \bar{\omega}_{i+2,j}, \bar{\omega}_{i+1,j}, \bar{\omega}_{i,j}, \bar{\omega}_{i-1,j}, \bar{\omega}_{i-2,j}\right). \tag{3.43}$$

The function $\hat{\omega} = l(\omega^\dagger, \omega_{-2}, \omega_{-1}, \omega_0, \omega_{+1}, \omega_{+2})$ begins by calculating a "monotonicity-preserving" limit for ω, ω^{MP}:

$$\omega^{MP} = \omega_0 + \text{minmod}\left[(\omega_{+1} - \omega_0), \alpha\,(\omega_0 - \omega_{-1})\right]. \tag{3.44}$$

The minmod$[x, y]$ function is the median of x, y, and 0: minmod$[x, y] = (\text{sign}(x) + \text{sign}(y)) \times \min(|x|, |y|)/2$. The multiple-argument minmod$[z_1, \ldots, z_k]$ function returns the smallest argument if all are positive, the largest argument if all are negative, and zero otherwise. And α is a numerical constant. Higher values of α allow higher-order accuracy in the interpolations but also reduce the allowable time step that will ensure monotonicity in a strong stability preserving (SSP) RK algorithm. As suggested in [30], we set $\alpha = 4$. For a further discussion of α and time stepping, see section 3.4.

Part of the efficiency of the SuHu limiter is a check to see if any limiters must be applied to the edge value. If we are within a certain tolerance ($\epsilon = 10^{-10}$), we do not have to apply any limiters to $\omega^\dagger$:

$$\text{if } \left(\omega^\dagger - \omega_0\right)\left(\omega^\dagger - \omega^{MP}\right) < \epsilon \text{ then } \hat{\omega} = \omega^\dagger. \tag{3.45}$$

Otherwise, we proceed with the limiting procedure, beginning with some second differences:

$$d_{-1} = \omega_{-2} + \omega_0 - 2\omega_{-1}, \tag{3.46}$$

$$d_0 = \omega_{-1} + \omega_{+1} - 2\omega_0, \tag{3.47}$$

$$d_1 = \omega_0 + \omega_{+2} - 2\omega_{+1}. \tag{3.48}$$

We then modify these differences:

$$d^{M4}_+ = \text{minmod}\left[4d_0 - d_1, 4d_1 - d_0, d_0, d_1\right], \tag{3.49}$$

$$d^{M4}_- = \text{minmod}\left[4d_{-1} - d_0, 4d_0 - d_{-1}, d_0, d_{-1}\right]. \tag{3.50}$$

The limiting algorithm continues by calculating various geometric factors. The meaning and reasoning behind these are given in [30], but for simplicity, let us leave these details and proceed with the algorithm:

$$\omega^{UL} = \omega_0 + \alpha\,(\omega_0 - \omega_{-1}), \tag{3.51}$$

$$\omega^{AVG} = \frac{1}{2}(\omega_0 + \omega_{+1}), \tag{3.52}$$

$$\omega^{MD} = \omega^{AVG} - \frac{1}{2}d^{M4}_+, \tag{3.53}$$

$$\omega^{LC} = \omega_0 + \frac{1}{2}(\omega_0 - \omega_{-1}) + \frac{4}{3}d^{M4}_-. \tag{3.54}$$

These values are used to calculate the minimum and maximum allowable values for $\omega^\dagger$:

$$\omega_{\min} = \max\left[\min\left(\omega_0, \omega_{+1}, \omega^{MD}\right), \min\left(\omega_0, \omega^{UL}, \omega^{LC}\right)\right], \tag{3.55}$$

$$\omega_{\max} = \min\left[\max\left(\omega_0, \omega_{+1}, \omega^{MD}\right), \max\left(\omega_0, \omega^{UL}, \omega^{LC}\right)\right]. \tag{3.56}$$

Finally one modifies the edge value by taking the median of these two limits:

$$\hat{\omega} = \text{median}\left(\omega^\dagger, \omega_{\min}, \omega_{\max}\right). \tag{3.57}$$

The equations (3.44)–(3.57) make up the entire limiting function, $l(\omega^\dagger, \omega_{-2}, \omega_{-1}, \omega_0, \omega_{+1}, \omega_{+2})$, which can be applied to each edge interpolation, $\{\omega_{i,j}^{\dagger R}, \omega_{i,j}^{\dagger L}\}$, using (3.42) and (3.43). Even though the SuHu method seems outwardly complex, the symmetries in both the interpolation and limiting phases of the algorithm allow for easy coding. The SuHu limiter maintains the order of the underlying algorithm even at extrema and guarantees monotonicity preservation for 1D linear passive advection problems, but global constraints might not be enforced in the presence of nonlinearities. Positivity is also insured if a time-step restriction is satisfied.

3.2.5. Suresh–Huynh/piecewise parabolic method hybrid (SuHu-PPM). To form a simple hybrid technique, we use the PPM interpolations as laid forth in section 3.2.2 but the limiters proposed by Suresh and Huynh [30] and detailed in section 3.2.4. This is not out of the spirit of the SuHu method, as [30] states that the limiters therein can be applied to any type of interpolation, and here we apply them to the fourth-order PPM interpolations. (This hybrid choice was motivated by results found in section 4.) Explicitly, we use (3.26)–(3.27) for the initial interpolations, but apply the limiters in (3.42)–(3.43).

Although the value of α in (3.44) can be set to 3 when using the fourth-order initial interpolations of PPM, thereby allowing a slightly larger time step (see section 3.4), we leave $\alpha = 4$ for the results in this paper.

3.3. On parallelization. Considering the recent advancements in high-powered computation, it is prudent to mention a few words on an algorithm's ability to parallelize efficiently over multiple processors and overall performance. Because higher-order methods generally involve more FLOPS per memory access (and can reuse data in cache from adjacent cells in the higher-order interpolations), one will generally expect higher-order methods to perform better on modern computer architectures where memory bandwidth is usually more of a bottleneck than the CPU.

Even though the test problem contained here was run on a single serial processor, all of the methods tested lend themselves to domain decomposition. The number of requisite ghost cells each processor needs depends upon the algorithm. The Arakawa method needs one layer to achieve its second-order accuracy, as cell (i, j) needs information from its immediate neighbors $(i \pm 1, j \pm 1)$. A higher-order Arakawa method, such as the one discussed in section 3.1.2, would require more information. For instance a fourth-order method needs $(i \pm 2, j \pm 2)$ [23] and, thus, two layers of ghost cells. The WENO3 method similarly only needs one layer of extra information per processor to calculate edge values $\{\omega_{i,j}^R, \omega_{i,j}^L, \omega_{i,j}^U, \omega_{i,j}^D\}$. However, in order to properly calculate the fluxes through the faces of cell (i, j), we must know the edge vorticity values on both sides of the interfaces, or specifically $\{\omega_{i-1,j}^R, \omega_{i+1,j}^L, \omega_{i,j-1}^U, \omega_{i,j+1}^D\}$. This implies that each processor must first calculate all of the $\{\omega_{i,j}^R, \omega_{i,j}^L, \omega_{i,j}^U, \omega_{i,j}^D\}$ on its subgrid using one ghost layer, then exchange information with its neighbors to be

able to calculate the correct fluxes and advance the solution and then exchange the newly calculated vorticity values $\omega_{i,j}^{n+1}$ with its neighboring processors.

Alternatively, one could subdivide the grid into domains with two ghost layers. This would allow each processor to accurately calculate all of the $\{\omega_{i,j}^R, \omega_{i,j}^L, \omega_{i,j}^U, \omega_{i,j}^D\}$ and $\{\omega_{i-1,j}^R, \omega_{i+1,j}^L, \omega_{i,j-1}^U, \omega_{i,j+1}^D\}$ on its subdomain, thereby consolidating two message passing events into one. In most cases this will probably be better as it will reduce the total latency, but the choice of which method to use would depend on details of the specific computer architecture (such as the latency and bandwidth of the interprocessor communication relative to CPU speed), and thus the relative efficiencies of having a slightly larger domain on each processor and the speed at which processors can exchange information. Similarly, the PPM, XPPM, and SuHu methods could all run with two ghost cells, but it will probably be more efficient to have three.

3.4. Time-advancement algorithms and positivity constraints. To advance the system in time, we employ a third-order SSP-RK algorithm (SSP-RK3), as described in [11]. It is a multistage process that is slightly complicated by the coupled equations (2.2) and (2.3). Written in a generic form, (2.2) is

$$\frac{\partial \omega}{\partial t} = \xi(\omega, \psi). \tag{3.58}$$

The SSP-RK3 method evolves this equation in three stages for every total time step, Δt, with the total process a convex average of forward Euler steps. If each Euler step preserves certain properties (such as monotonicity, as the limiters of the various finite volume methods are designed to do), then the convex average of the final method will preserve these properties as well. If we define the vorticity and stream function at time step n as ω^n and ψ^n, then the SSP-RK3 method is

$$\omega^\star = \omega^n + \Delta t\ \xi(\omega^n, \psi^n), \tag{3.59}$$

$$\omega' = \frac{3}{4}\omega^n + \frac{1}{4}\omega^\star + \frac{1}{4}\Delta t\ \xi(\omega^\star, \psi^\star), \tag{3.60}$$

$$\omega^{n+1} = \frac{1}{3}\omega^n + \frac{2}{3}\omega' + \frac{2}{3}\Delta t\ \xi(\omega', \psi'). \tag{3.61}$$

It should be noted that each vorticity/stream function pair must satisfy the Poisson equation (2.3), so the SSP-RK3 algorithm requires three inversions of (2.3) per time step.

As mentioned, we run the simulations with a CFL number of 0.1 to ensure that our comparison of spatial algorithms is not muddled by temporal errors. While our focus is primarily on the different algorithms for evaluating the Poisson bracket, the choice of time-integration scheme in a plasma edge turbulence simulation is important as well.

The SSP-RK3 algorithm is one of a set of schemes designed to maintain the properties (such as monotonicity and positivity) of the simulation throughout the multistage integration process. There is a trade-off with these multistage algorithms. As the number of stages increases, so can the order accuracy of the method and the maximum stable time step. However, adding additional stages increases the amount of work, as the Poisson equation must be inverted and the Poisson bracket must be evaluated at each stage.

SSP-RK algorithms are comprised of many Euler-like substages, each of which makes a temporal advance of Δt_1. In order for an SSP-RK algorithm to preserve

TABLE 3.1

Stability-based efficiency of various time-advancement algorithms [10], *measured by work required to integrate forward in time using the maximum possible time step to preserve stability, monotonicity, and positivity. The parameter α depends on the spatial discretization algorithm, and the length of a substep is Δt_1.*

Algorithm	Number of stages N_{stages}	Time-step size $\Delta t/\Delta t_1$	Relative efficiency ϵ_t
SSP-RK3	3	1	$0.333/(1+\alpha)$
5-stage SSP-RK4	5	1.508	$0.302/(1+\alpha)$
Low-storage 10-stage SSP-RK4	10	6	$0.6/(1+\alpha)$

monotonicity (and/or positivity, if relevant), it must satisfy a constraint on the maximum allowable substage time advance [10]. Namely, if we define a CFL time scale by $\Delta t_{\text{CFL}} = h/v$, then to preserve monotonicity or positivity, each substage of an SSP-RK method must satisfy a constraint of the form

$$\Delta t_{1,\max} = \frac{\Delta t_{\text{CFL}}}{1+\alpha}, \tag{3.62}$$

where the value of α depends on the spatial discretization algorithm and generally increases with the order of the spatial discretization. (Possible choices of α and its relation to monotonicity preservation are discussed in [30]. We will use their suggested value of $\alpha = 4$ for the cases in this paper. Values of α for positivity preservation are discussed below.) The length of each of these substeps is dependent upon the time-advancement algorithm. For instance, in the SSP-RK3 algorithm described in (3.59)–(3.61), $\Delta t_1 = \Delta t$, the total time step. While the ratio of the total time step to a substep, $\Delta t/\Delta t_1$, is algorithmically dependent, it does define the maximum allowable total time step, $\Delta t_{\max} = (\Delta t/\Delta t_1)\Delta t_{1,\max}$. We can define an SSP-based relative efficiency for an algorithm as the ratio of this maximum allowable time step (relative to the Courant time Δt_{CFL}) to the number of stages in the RK scheme:

$$\begin{aligned} \epsilon_t &= \frac{\Delta t_{\max}}{\Delta t_{\text{CFL}}} \frac{1}{N_{\text{stages}}} \\ &= \frac{\Delta t}{\Delta t_1} \frac{1}{N_{\text{stages}}(1+\alpha)}. \end{aligned} \tag{3.63}$$

While a detailed discussion of various optimal SSP algorithms can be found in [10], Table 3.1 highlights the efficiencies of a few methods. The three-stage RK method we use, (3.59)–(3.61), is actually more efficient than an optimal five-stage, fourth-order method, because it has fewer stages. Even still, a recently discovered ten-stage SSP-RK4 algorithm [12] is nearly twice as efficient as the five-stage version, because it allows for a much larger time step. This analysis, however, only takes into consideration the work required for monotonicity and positivity. In situations where high accuracy is of greater concern, higher-order methods would be favorable.

In any case, we are focused primarily on spatial discretizations of the advection operator and so choose the simpler SSP-RK3 method with small enough time step so as not to obscure the spatial discretization analysis. An estimate of the time-stepping errors can be seen in Figure 4.3, where the dissipation from the Arakawa scheme is entirely from time integration errors.

Time-advancement schemes can also alter the monotonicity and positivity properties of the system. Two of the Poisson bracket algorithms, the XPPM and Suresh–

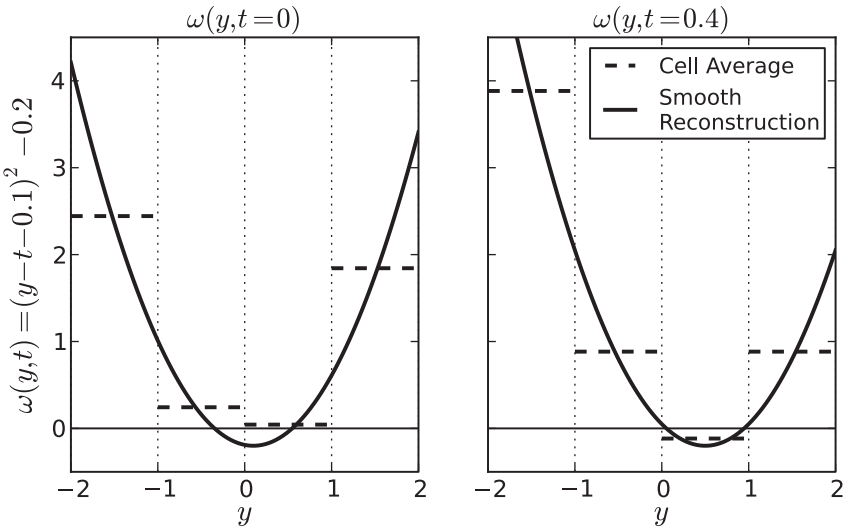

FIG. 3.1. *Illustration that there exist initial conditions where all of the cell averages are positive (left plot), but the extrema-preserving algorithms can lead to a negative cell average at a later time (right plot), as the minimum in the underlying reconstruction moves from a cell edge towards a cell center. Note that the underlying smooth reconstruction $\omega(y)$ can be negative in some regions even though the cell averages are everywhere positive initially.*

Huynh algorithms, are monotonicity preserving (in the sense described below), but they are not necessarily positivity preserving, a desirable property when simulating advection of physically positive quantities (examples include density, temperature, probabilities, or particle distribution functions). To understand the origin of this problem, note that these algorithms are monotonicity preserving (for constant advection in one dimension, if the time step is sufficiently small) in the sense that no new extrema are produced and an initially monotonic region may be displaced during a time step but remains monotonic within that displaced region. (This is the sense in which Suresh and Huynh refer to their algorithm as monotonicity preserving.[2]) In other words, the algorithms move existing extrema but do not create any new extrema. However, they do not necessarily prevent an existing extremum from being slightly accentuated, which in fact must be allowed in order to maintain accuracy near smooth extrema. Thus they do not necessarily ensure positivity of the solution.

An example illustrates how these kinds of monotonicity-preserving algorithms do not necessarily preserve positivity. Smooth extrema preserving algorithms like XPPM and SuHu are exact for smooth low-order solutions, which near a minimum can be approximated as a generic parabola, or $\omega(y,t) = C_0 + C_2(y - vt - y_0)^2$, for given coefficients C_0 and C_2, where the minimum is initially located at y_0. It is possible for the initial conditions for the cell-averaged values of $\bar{\omega}_i$ to all be positive, but to correspond to an underlying reconstruction $\omega(y,0)$ that goes negative at some locations. Then as the solution evolves in time, it is possible for a cell average to drop below zero, as illustrated in Figure 3.1, for the specific case of $C_0 = -0.2$, $C_2 = 1.0$, $v = 1.0$, and $y_0 = 0$. Of course, if $\omega(y)$ should be positive, then $C_0 \geq 0$, and the initial

[2]In some of the CFD literature, the term "monotonicity preserving" is defined to mean not only that there are no new extrema produced, but also that existing extrema are not accentuated. However, this is not the sense in which Suresh and Huynh use the term.

values of $\bar{\omega}_i$ should be such that $\bar{\omega}_i$ remains positive. But the converse is not true, as there are certain values for $\bar{\omega}_i$ that correspond to $C_0 < 0$, even though all of the $\bar{\omega}_i$ are positive at a given instant, which will then cause some $\bar{\omega}_i$ to go negative at a later time. (Such conditions might be produced not only by initial conditions but by other terms in the equations, such as loss terms by radiation or some other process.)

It is possible to guarantee positivity of the cell-averaged solution by supplementing these algorithms with a simple additional limiter. A single forward Euler time step of (3.6) (neglecting the diffusion term for simplicity) can be written in the form

$$\bar{\omega}_l^{n+1} = \bar{\omega}_l^n - \sum_s \frac{\Delta t\, \hat{v}_{s,l}}{h}\, \hat{\omega}_{s,l} \tag{3.64}$$

where $\hat{\omega}_{s,l}$ is the value of ω interpolated (with limiters) to the face s of the cell l at time level n, and $\hat{v}_{s,l}$ is the velocity directed *outward* through face s. (For two-dimensional problems, $s = 1 \ldots 4$, and $\hat{v}_{s,l} = -v_{i-1/2,j}$ for s corresponding to the left face of cell (i, j).) To ensure positivity of the solution at time t_{n+1}, we must satisfy $\sum_s (\Delta t\, \hat{v}_{s,l}/h)\, \hat{\omega}_{s,l} \le \bar{\omega}_l^n$. If we use a conservative upper bound on the left-hand side of this equation based on only the outward fluxes from this cell ($v_{s,l} \ge 0$), and if the edge interpolations are bounded by

$$0 \le \hat{\omega}_{s,l} \le (1 + \alpha)\, \bar{\omega}_l^n \tag{3.65}$$

for some constant α (to be discussed), then positivity is preserved if

$$\sum_s \Delta t\, \max(\hat{v}_{s,l}, 0)/h \le 1/(1 + \alpha) \tag{3.66}$$

is satisfied for all cells. The maximum value over all cells of the left-hand side of this equation defines a generalized CFL parameter, so positivity is satisfied if CFL $\le 1/(1 + \alpha)$.

To preserve the order of accuracy of the algorithm, one chooses α large enough to avoid limiting any allowable, positive solutions that are sufficiently smooth. For the fourth-order XPPM or SuHu-PPM, where solutions are of the form $\omega(y) \propto (y - y_0)^2$ near a zero, the maximum value for the ratio of the edge value to its cell-averaged value, $\hat{\omega}_{s,l}/\bar{\omega}_l^n$, is 4. Therefore, setting $\alpha = 3$ is the minimum value needed to guarantee positivity without reducing the order of accuracy. The value of $\alpha = 4$ used in the original SuHu algorithm will also ensure positivity without reducing the order of accuracy, if the edge interpolations are limited to satisfy (3.65) and the time step satisfies (3.66). In general, as the order of accuracy of the interpolations increases, one must use larger values of α in (3.65) to preserve that accuracy, but this increases the computational expense because the time-step size is reduced for larger α in (3.66).

These results for the choice of α apply in the asymptotic limit as the grid is refined. One might consider more general polynomials near a minimum. As an example for XPPM or SuHu-PPM, which use cubic interpolating polynomials, one could have $\omega(y) \propto (y - y_0)^2 (y - y_2)$ locally near a particular cell containing y_0. A value of $\alpha \approx 3.38$ would be necessary to avoid limiting any case where the point y_2 (where ω changes sign and this local polynomial expansion breaks down) lies at least two cells away. However, as the grid is refined so that the point y_2 is many cells away, then $\alpha = 3$ becomes an acceptable limit to preserve positivity and the order of accuracy.

For the fifth-order SuHu algorithm, where $\omega(y) \propto (y - y_0)^4$ is possible, one finds the maximum ratio of edge to cell-averaged values gives $\alpha \approx 6.43$, which would lead

to the time-step restriction, CFL < 0.13, somewhat stricter than the Suresh–Huynh original choice for just monotonicity, CFL < 0.2. However, if high-order minima are rare and it is more likely to have distinct second-order minima, $\omega(y) \propto (y - y_0)^2(y - y_2)^2$, then the value of $\alpha \sim 3$–4 may be sufficient in most cases, allowing a larger time step with little loss of accuracy. The limiter in (3.65) will preserve positivity if the SSP-RK time step is sufficiently small, with CFL < 0.2 for our parameters.

A flexible way to implement a positivity limiter, without imposing the limit on the time step in (3.66), is to bound the edge interpolations to satisfy

$$\hat{\omega}_{s,l} < \bar{\omega}_l^n / \max(1/(1+\alpha), \mathrm{CFL}_l), \tag{3.67}$$

where the local CFL parameter in cell l is defined by $\mathrm{CFL}_l = \sum_s \Delta t \max(\hat{v}_{s,l}, 0)/h$. The parameter α is as given above to avoid losing accuracy for smooth positive solutions in regions where the local CFL parameter is small enough, though there will be some loss of accuracy by this positivity limiter in regions where the local CFL parameter exceeds $1/(1+\alpha)$. For certain types of flows (including kinetic plasma problems of interest to us), the local CFL number on most of the grid will be much smaller than the maximum CFL number on the grid, so that positivity can thus be ensured everywhere, without loss of accuracy on most of the grid, while allowing a larger time step. We leave exploration of this possibility as future work.

As mentioned in [6], another way to preserve positivity could be to couple finite volume methods with the flux-corrected transport method (FCT) [31], and some approaches to this could allow a larger time step. The standard FCT method requires twice the work per time step, but if it is used only to enforce positivity, then a simpler version can be implemented within a single step [32]. As has been pointed out [20], it is not necessary to apply the FCT algorithm on every substage of an RK time-advancement algorithm. Instead, one could apply FCT only after a full RK time step, using the fluxes averaged over the RK substages as the starting point for FCT. (This comes at the expense of storing these average fluxes at each cell face, which requires d times as much memory as the cell averages in d dimensions.) If an individual RK substage generates negative values, the corrections from the other substages are likely to make it positive again, or at least reduce the magnitude of the overshoots. (If the solution is everywhere positive after the final RK stage on most time steps, then the FCT limiting could be skipped on those steps.) This could allow a larger time step for preserving positivity without reducing the order of accuracy. Note that negative values during intermediate RK stages can be set to a floor value if necessary when calculating fluxes, while still satisfying the conservation form as long as only the fluxes are modified. Storage of the fluxes could be avoided by using the time-step algorithm in [33], though this algorithm is only formally second-order accurate for nonlinear problems [34]. Tests of FCT-based approaches are beyond the scope of this paper.

4. Results.

4.1. One-dimensional advection. By skipping the Poisson equation inversion, (2.3), with the initial conditions $v_x = 0$, $v_y = 1$ and without viscosity, $\nu = 0$, the equations of motion become simple advection, and (2.1) reduces to a wave equation. Exact solutions are easily tractable, as the initial disturbance propagates with constant unit velocity. After a transit period ($t = 10m$, $m \in \mathbb{N}$), any distortions from the initial conditions are only properties of the numerical algorithm.

A standard test problem in 1D advection is the transit of a Gaussian and a step-function pulse (see, for instance, [17]). Since our problem is formulated in two

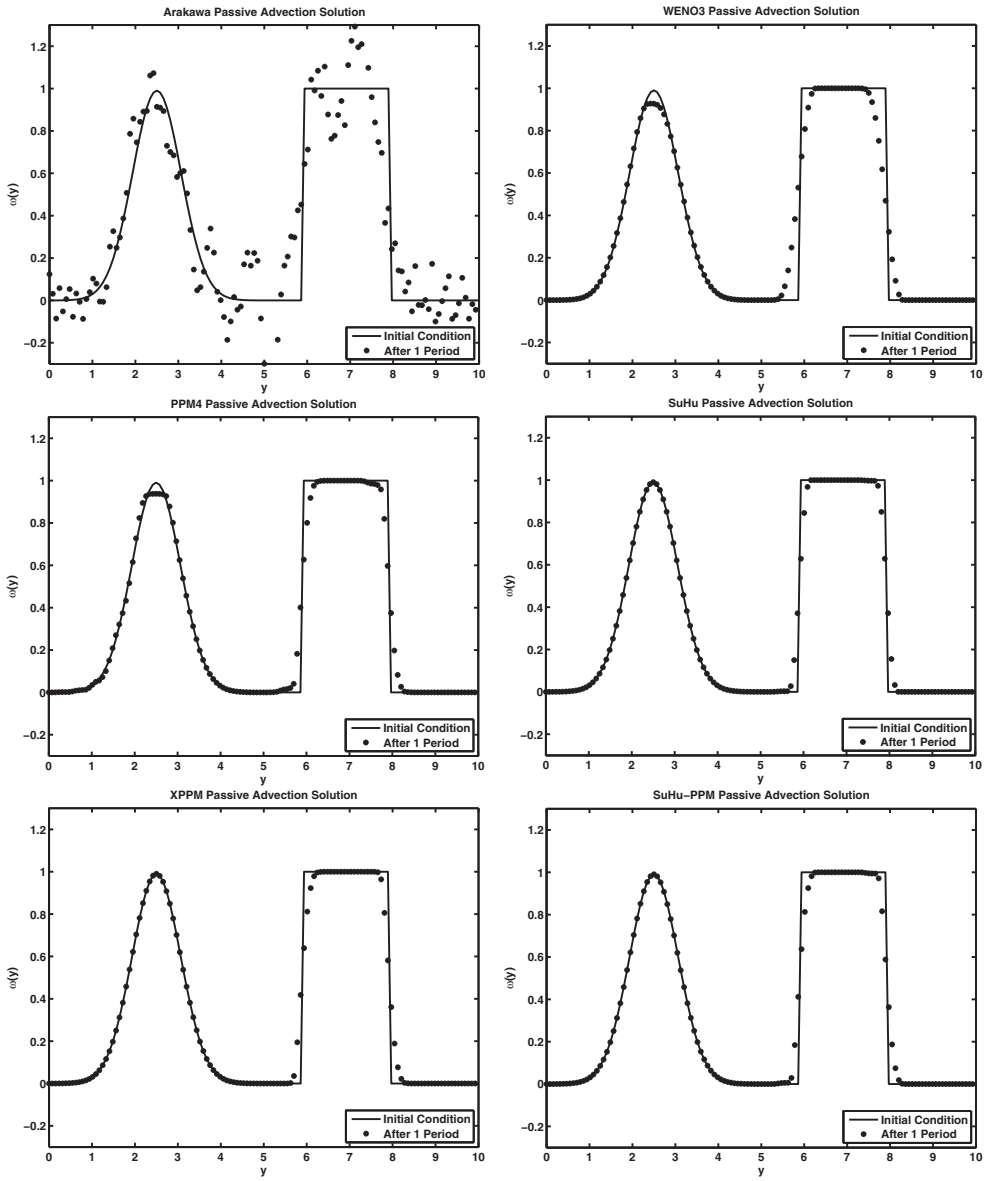

FIG. 4.1. *Advection of a Gaussian and step-function pulse for one transit (dots) through a* 1*D periodic domain, overlaid with the initial conditions (solid line). Left, top to bottom: second-order Arakawa, PPM*4*, and XPPM. Right, top to bottom: WENO*3*, SuHu, SuHu-PPM. In all cases,* $\nu = 0$, $v_y = 1$.

dimensions, we advect 2D structures (the Gaussian and a square block) in one dimension and compare solutions through the center of the structures along the direction of motion. That is, after initialization, we let these pulses transit through one period of motion in the y-direction and compare solutions at fixed values of x.

Figure 4.1 shows the overlaid initial (solid) and final (dotted) solutions for the six methods (all with CFL $= v\Delta t/\Delta x = 0.1$ so that time discretization errors are negligible and the results do not change with smaller Δt). The second-order Arakawa result displays a large phase error, as evidenced by the mismatch in peak locations.

Furthermore, one clearly sees spurious oscillations that build up behind the pulses and propagate through the mesh. These kinds of errors could be particularly significant in cases where nonlinear interactions of coherent structures are important, so one would like to minimize phase errors between various components of the solution.

Reducing the spurious oscillations seen with the Arakawa method may require very high resolution and/or including relatively strong viscosity. Much of the total error in the Arakawa solution originates from the step function, which propagates through the mesh and adversely affects the smooth pulse. (Indeed, even very high resolution will not eliminate spurious oscillations for Arakawa with a step-function initial profile.) However, even without the step function, the solution would still oscillate to unphysical negative values (for a Gaussian with a full width at half maximum (FWHM) of 1, and 12.8 points per FWHM, the oscillations in the solution will reach -6% of the peak value after propagating a distance of 10). In many respects this is not unexpected, as in the 1D limit the Arakawa discretization reduces to a second-order center-differenced method (see (3.2)). The WENO3 and PPM4 solutions display significant diffusion near the Gaussian peak (PPM4 is slightly less diffusive), but both show relatively small phase errors and asymmetries. However, the finite volume methods with smooth extrema preserving limiters reproduce the initial conditions quite well, with peak amplitude drops of $\sim 10^{-7}$. Additionally, no finite volume result has spurious oscillations; the solution stays above the initial minimum, preserving positivity.

As another test, we can initialize the vorticity to be a cosine wave in the direction of propagation with wave number k, $\omega^0(y, t = 0) = \cos(ky)$. After a time $T = 2\pi/(kv)$ the wave should have moved exactly one wavelength and reproduce the initial condition. For linear algorithms, the numerical solution will be a cosine wave, with possible phase shift and amplitude errors: $\omega^T(y) = \omega(y, t = T) = a\cos(ky + \phi)$. Because of the nonlinearities in the limiters of these algorithms, a standard linear von Neumann analysis cannot be done, so we measure amplitude and phase errors that build up during one advection period in the following way. If there were no damping, the numerical cosine wave would satisfy $\langle \omega^2 \rangle = 1/2$ for all $k = 2\pi m/L$ (except $m = 0, N/2$), where $\langle \cdots \rangle$ denotes the average operator along the y-direction. Let us characterize a numerical damping factor γ_n by

$$\langle \omega^2 \rangle (T) = \frac{1}{2} e^{-2\gamma_n T}. \tag{4.1}$$

The phase error can be determined by the law of cosines, through the dot product of the initial and final solutions:

$$\cos(\phi) = \frac{\sum_j \omega_j^0 \omega_j^T}{\sqrt{\sum_j \left(\omega_j^0\right)^2} \sqrt{\sum_j \left(\omega_j^T\right)^2}}. \tag{4.2}$$

If ω^T and ω^0 are exactly out of phase ($\omega^0 = -\omega^T$), then (4.2) calculates $\phi = \cos^{-1}(-1) = \pi$. If they are in phase, $\phi = \cos^{-1}(1) = 0$. Alternatively, one can calculate the phase error less robustly by a nonlinear least-squares fit for the coefficients a and ϕ of $\omega(y, T) = a\cos(ky + \phi)$. We define the phase error E_ϕ as $(1 - \cos(\phi))/2$, which maps $\phi \in [0, \pi] \to E_\phi \in [0, 1]$.

An ideal algorithm would have small values for both γ_n and ϕ for all wave numbers $k\Delta y/\pi \in (0, 1)$. If one is primarily interested in large-scale physical features, low damping and phase errors at small k would be particularly desirable, even though,

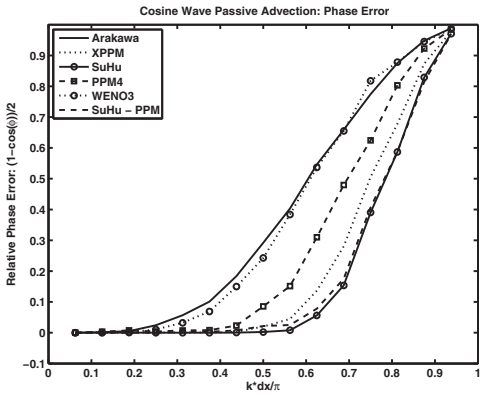

FIG. 4.2. *Phase errors after one period for advection of a cosine wave as a function of wave number.*

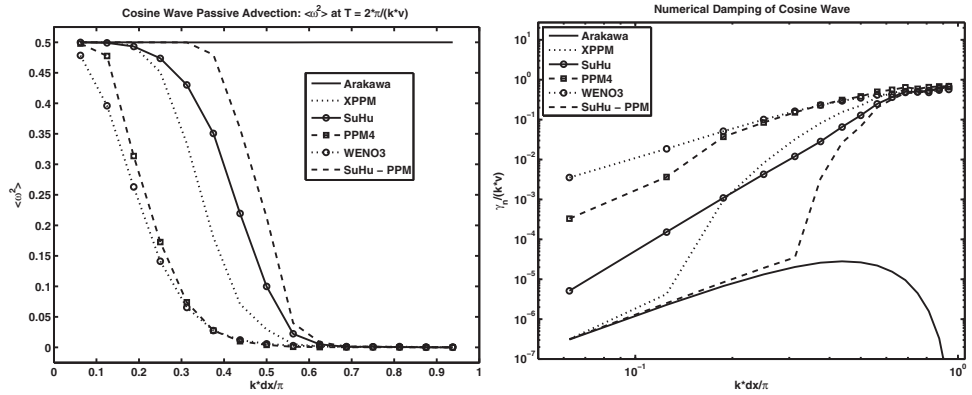

FIG. 4.3. $\langle\omega^2\rangle$ *as a function of wave number for the passive advection of a cosine wave after one period* T *for the different algorithms (left), along with associated numerical damping factors* γ_n *(right) as defined by* $\langle\omega^2\rangle(T) = \frac{1}{2}e^{-2\gamma_n T}$.

in general, the phase and amplitude errors are not simple functions of only the wave number for most of these algorithms because of the nonlinearities in the limiters they use. As shown in Figure 4.2, the phase errors of the higher-order algorithms (XPPM, SuHu, SuHu-PPM) are all much lower than those of the WENO3, PPM4, and Arakawa methods across all wavelengths.

We display $\langle\omega^2\rangle$ with calculated numerical damping rates, γ_n, of each method in Figure 4.3. As the Arakawa spatial discretization method does not introduce any dissipation itself, the curve labelled Arakawa represents the small residual numerical dissipation from the third-order time-stepping method. While the WENO3 and PPM4 methods have relatively high rates of numerical dissipation, the extremum-preserving methods have significantly lower dissipation. XPPM drops to the minimum possible dissipation at long wavelengths, because it reverts to an undamped centered method in well-resolved regions. The observation that SuHu is better than XPPM at higher wave number while XPPM is better at lower wave number motivated the choice of the SuHU-PPM hybrid, which uses the centered fourth-order interpolation of the initial step of PPM in combination with the SuHu limiters. The SuHu-PPM method nearly

overlaps the true solution as it approaches the minimum level of dissipation for long wavelengths of $k\Delta y \lesssim \pi/3$.

It is possible to understand some of the asymptotic scalings for the numerical dissipation rate in Figure 4.3. If (3.20) and (3.21) from the WENO3 method were used to interpolate the boundary values with no further limiters, then this would correspond to an upwind-biased third-order method, which would have a numerical damping rate $\gamma_n \sim |v|(\Delta x)^3 k^4$. As seen in the damping rates of Figure 4.3, the additional limiters in the WENO3 method give larger damping, $\gamma_n \propto k^3$. This behavior is because WENO3 reverts to a piecewise constant interpolation near extrema, which is equivalent to simple first-order upwind that locally introduces a numerical damping $\sim |v|\Delta x k^2$. Since the fraction of cells at extrema is proportional to $k\Delta x$, the resulting average dissipation is a numerical damping rate $\sim k^3$. Similar arguments apply to the regular PPM method, which is also dominated by clipping errors at extrema.

4.2. Two-dimensional vortex merger. We now turn to the fully 2D nonlinear problem as described in section 2. The two Gaussian monopoles begin to swirl around each other and merge together. During the process, sideband rings propagate outward across the mesh. Details on the vortex merging process can be found in [25], but it should be noted that the vorticity should never drop below the floor as defined by the background. Any such solutions are solely artifacts of the numerical method.

In Figure 4.4 we show $\mathrm{Re} = 10^5$ vorticity solutions at $t = 100$ for WENO3, second-order Arakawa (Arakawa2), SuHu, XPPM, and SuHu-PPM, at two different resolutions, $N = 128$ (left) and $N = 1024$ (right). All plots are on the same common color scale, which spans the initial minimum and maximum vorticity. Values outside of this range are marked in white (if they are negative, indicating a positivity violation) or black (if they are too large, indicating an overshoot).

Beginning with the low-resolution simulations, we see that the WENO3 solution is extremely affected by numerical dissipation. The solution has washed out all fine features and has joined the two peaks into a single ellipsoid shape. Furthermore, the orientation of that ellipse does not agree with any of the other solutions, but WENO3 does not violate positivity, as expected. The low-resolution Arakawa solution is also plagued with numerical errors, as indicated by the numerous positivity violations and the unphysical overshoots near the center of the domain. However, the SuHu, XPPM, and hybrid SuHu-PPM methods more faithfully reproduce the solution at low resolutions. Both of the merging peaks are resolved alongside the expanding banded arms. The SuHu low-resolution solution violates positivity slightly in a few places, which is interesting, although not surprising, because the SuHu limiters are designed to only ensure positivity in linear problems, like passive advection, and make no such guarantee for nonlinear problems, like vortex merging. Although the two methods with the PPM interpolations (XPPM and SuHu-PPM) do well at low resolutions, the SuHu-PPM method shows the most detail.

The $N = 1024$ extremum-preserving and Arakawa solutions show crisp details, such as the wake features seen trailing the vortexes, which are washed out by the WENO3 method. This is in agreement with the observations made by Naulin and Nielsen [22]. However, the high-resolution Arakawa solution still has excessive oscillations and positivity violations. This is seen not only in the white and black cells of Figure 4.4, but also in a 1D slice through the domain center, as shown in Figure 4.5.

With respect to overall levels of dissipation, we present in Figure 4.6 the total enstrophy Ω at $t = 100$ (left plot) and at $t = 500$ (right plot), for different resolutions: $N = 1024, 512, 256, 128$. Plotted alongside is a curve marked "optimal subgrid,"

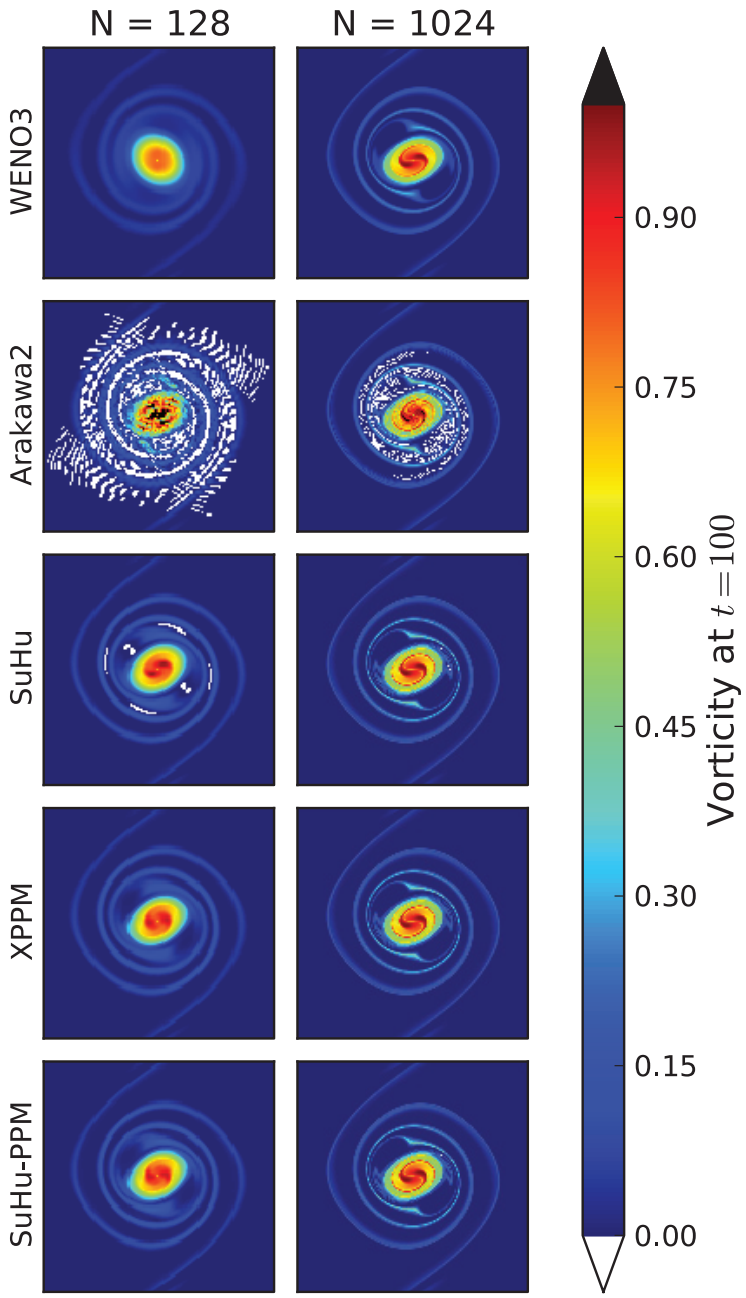

FIG. 4.4. *Low-* ($N = 128$) *and high-resolution* ($N = 1024$) *vorticity solutions for the vortex merger problem with high Reynolds number* $Re = 10^5$ *at* $t = 100$, *with the color scale spanning the minimum and maximum initial vorticity. Positivity violations appear white. Unphysical overshoots are black.*

which corresponds to a hypothetical subgrid model that leaves the spectrum unperturbed except for truncation at $k_{\max} = \pi/(\Delta x) = k_{\min} N/2$. The total enstrophy is calculated by summing the enstrophy spectrum Ω_k over all k up to $k_{\max}$ from a well-resolved $N = 1024$ simulation using the fourth-order Arakawa method described

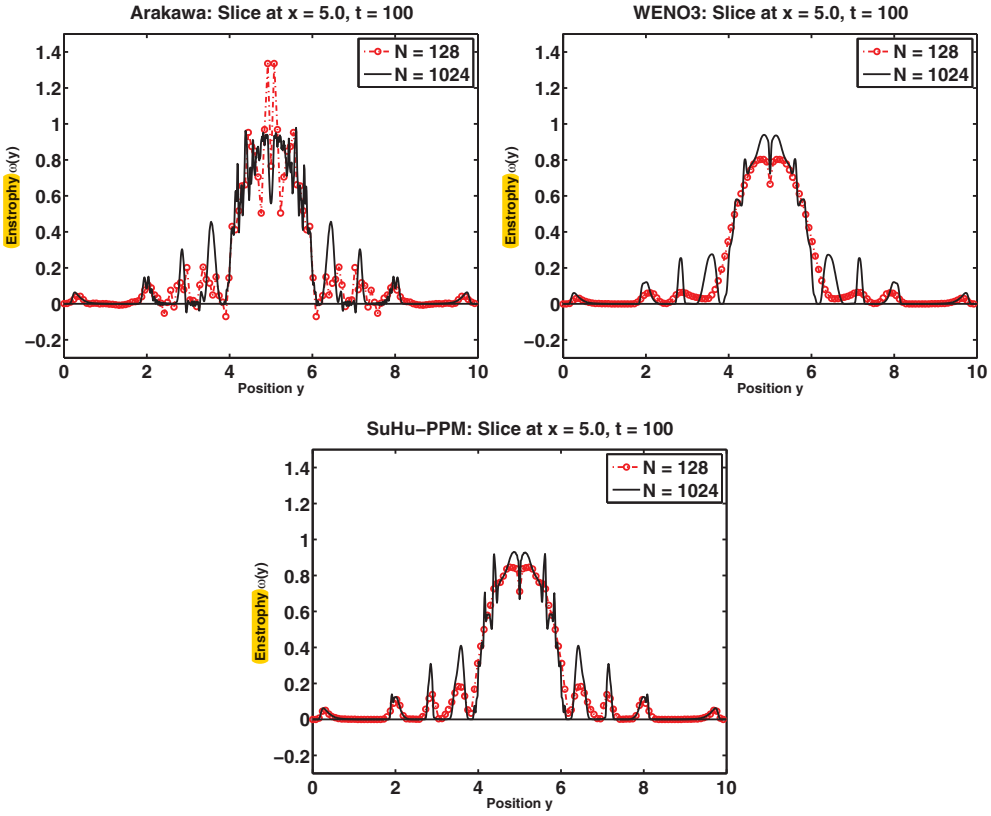

FIG. 4.5. *1D slices of vorticity, $\omega(x = 5.0, y, t = 100)$ for the second-order Arakawa (top left), WENO3 (top right), and SuHu-PPM (bottom) techniques.*

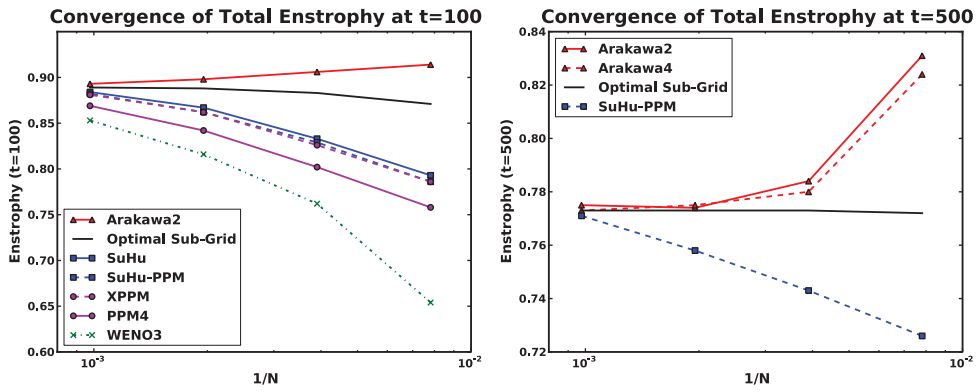

FIG. 4.6. *Convergence of the total enstrophy at time $t = 100$ (left) and time $t = 500$ (right) versus $1/N$, for grid sizes $N = 128$, 256, 512, 1024, for the high Reynolds number case. Also plotted are the solutions from an optimal subgrid model.*

in section 3.1.2. The optimal subgrid curves are fairly flat, indicating that little of the enstrophy is above these cutoffs.

The solutions for both the WENO3 and Arakawa methods at $t = 100$ agree with those found in [22]. The numerical dissipation in WENO3 makes the enstrophy signif-

Errata 4/8/2017: "Enstrophy" in these y-axis labels should be "vorticity".

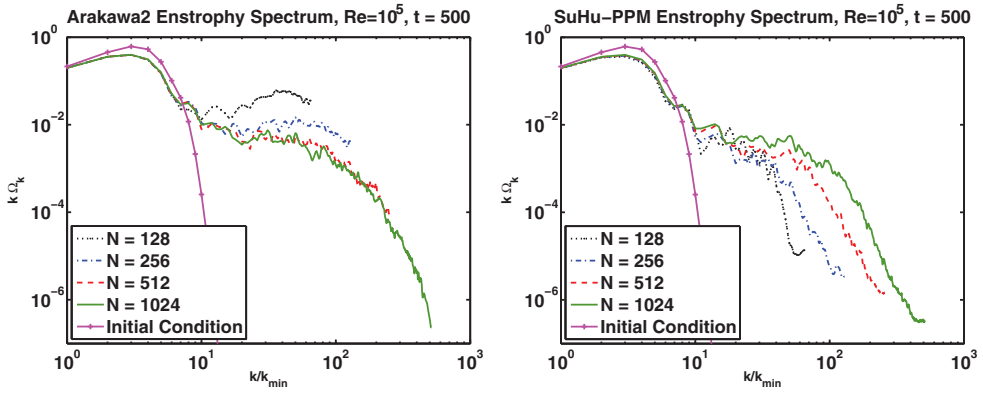

FIG. 4.7. *Enstrophy spectra for high Reynolds number at different resolutions, for the second-order Arakawa method (left) and the SuHu-PPM method (right). At lower resolutions, the Arakawa method experiences an artificial pileup of enstrophy at high k.*

icantly lower than that calculated via the Arakawa method, which exactly preserves the discretized enstrophy. As such, the Arakawa solution only contains dissipation from the explicit viscous term and from the time-stepping algorithm, which is small. The extremum-preserving algorithms are the least dissipative of the finite volume schemes; however, even the PPM4 method presents a significant improvement over WENO3. (A correction to the finite volume methods that accounts for the fact that finite volume methods actually deal with cell-averaged vortices $\overline{\omega}_{i,j}$ and that $[\overline{\omega_{i,j}}]^2$ is slightly larger than $\overline{\omega_{i,j}^2}$ is very small and has no visual impact on this plot.) While Arakawa is closest to the correct total enstrophy when dissipation is not yet important (at early times) and converges most quickly at high resolution, it can have problems at low resolution at later times (as seen in the $t = 500$ plot) when dissipation should occur but the dissipation scale is not adequately resolved. Even at early times when Arakawa is closer to the correct total enstrophy because it has more enstrophy than the finite volume methods, the phases between the high k modes containing that extra entrophy are not accurate. This leads to the large spurious oscillations seen in the low resolution results in Figure 4.5.

In general, some level of dissipation is desirable for simulations of turbulent or other nonlinear cascades, as illustrated in another way by the enstrophy spectra Ω_k shown in Figure 4.7. We plot $k\Omega_k$ versus normalized wave number $k/k_{\min}$, with $k_{\min} = 2\pi/L$, for different resolution simulations at late time, $t = 500$, for the same $\mathrm{Re} = 10^5$ vortex merger problem as before. The left plot is for the Arakawa technique and the right plot is for the SuHu-PPM hybrid technique (XPPM and SuHu give similar results). The 1D enstrophy spectrum Ω_k plotted here is the sum over all modes $\vec{k} = k_x\hat{x} + k_y\hat{y}$ with $\|\vec{k}\|$ lying in the bin $k \pm -\Delta k/2$, so that the total enstrophy is $\Omega = \sum_k \Omega_k$, where $k = k_{\min}, 2k_{\min}, \ldots, \sqrt{2}(N/2)k_{\min}$.

As can be seen in Figure 4.7, both methods show a falloff in enstrophy at high resolution. Additionally, the energy-containing larger scales ($k/k_{\min} < 10$ in this case) look similar. But the Arakawa method at lower resolution has an artificial pileup of enstrophy at high k. Because of the conservation properties of Arakawa, the enstrophy is trapped in the system if there is no dissipation (or the dissipation scale is not sufficiently resolved), and instead of cascading down to unresolved smaller scales, the enstrophy is reflected back from the grid scale, leading to a pileup. This pileup, or bottleneck, problem is well known in turbulence simulations and is the

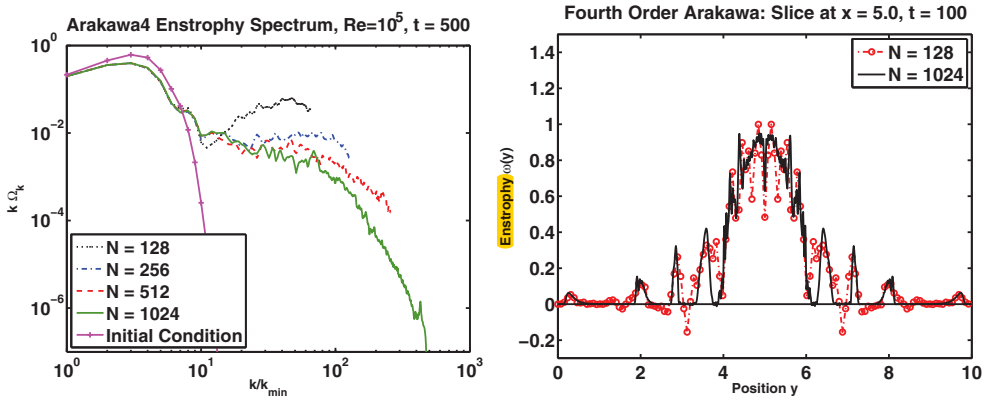

FIG. 4.8. *Utilizing a higher-order Arakawa method prevents neither spectrum pileup in low resolution cases (left) nor spurious oscillations (right).*

motivation for adding some kind of artificial dissipation or hyperviscosity to such systems [3, 28]. The time it takes for nonlinearities to cascade part of the solution to dissipation scales is often very fast (for example, in fully developed three-dimensional Kolmogorov turbulence, energy reaches the dissipation scale in a time of order of the large-scale eddy turnover time, independent of Reynolds number), so dissipation quickly becomes significant in such cases.

In the lowest resolution $N = 128$ Arakawa case, the pileup spectrum for $k/k_{\min} > 20$ is roughly similar to what one might expect for thermalization of enstrophy, where there is no net cascade of enstrophy. For 2D thermal equilibrium, the energy spectrum is $E_k \propto k/(\alpha + \beta k^2)$, so the asymptotic result at high k would give $k\Omega_k \propto k^2$ for positive β coefficient [13, 14]. This is similar to recent observations of partial thermalization at high k for hyperviscosities $\propto \nabla^{2p}$ with excessively high p exponents [9]. Note that the standard expectations for 2D turbulence [13, 24] with a net cascade would give $E_k \sim k^{-3}$ in the enstrophy intertial range, and thus $k\Omega_k \sim k^0$, though these results for fully developed turbulence are not necessarily expected to hold for the coherent vortex merger problem under study here.

Increasing the order of accuracy of a method does not necessarily fix the pileup and positivity problems seen at lower order. The fourth-order Arakawa method still traps enstrophy in the system and allows nonphysical oscillations. Figure 4.8, a recreation of the tests in Figures 4.5 and 4.7 with the fourth-order Arakawa method, shows similar results to the lower-order Arakawa method. Since enstrophy is exactly conserved by the Arakawa methods, it cannot leave the system at the grid scale without explicit damping, and the power spectrum is altered if the dissipation scale is not adequately resolved. The higher-order Arakawa solution still displays spurious oscillations, visibly violating positivity even in the highest resolution simulation.

Higher-order versions of the Arakawa Poisson bracket representation share the conservation properties of their lower-order counterparts. Since positivity violation, spurious oscillation, and turbulent spectrum alteration are all related to these conservation properties, it is not possible to eliminate these problems by merely employing a higher-order solution.

5. Discussion. The SuHu-PPM method (and other similar methods) is more robust at lower resolution, giving the qualitatively correct enstrophy inertial range,

4/8/2017: "Enstrophy" in this y-axis label should be "vorticity".

because it essentially introduces a kind of subgrid model that provides some needed dissipation near the grid scale. While the numerical dissipation in these methods is the minimum necessary to ensure monotonicity (in 1D linear advection), and acts qualitatively as a subgrid model that improves robustness on coarse grids, it is not necessarily the optimal amount of damping for a subgrid model purely from a turbulence modeling perspective (if small amounts of nonmonotonic overshoots can be tolerated). For fluctuations near the grid scale, these methods revert to first-order upwind differencing, which corresponds to a damping rate of order $|v|/\Delta x$, while eddy-viscosity concepts [16] indicate that a subgrid model of the transfer from barely resolved scales to unresolved scales should introduce a damping rate near the grid scale of order of the shearing rate, $|\nabla v|$. For a Kolmogorov spectrum $E(k) \sim k^{-5/3}$, where most of the energy is at large scales but most of the shearing is done by small scales, this means that upwind damping at the grid scale is too strong by a factor of approximately $\sqrt{k_{\max}^2 \int dk\, E(k)}/\sqrt{\int dk\, k^2 E(k)} \sim (k_{\max}/k_0)^{1/3}$, where k_0 is the wave number of the energy-containing eddies and $k_{\max} \sim 1/(\Delta x)$. While this damping is too strong at the grid scale at high resolution, where $k_{\max}/k_0$ is large, the numerical dissipation drops off relatively rapidly for $k/k_{\max} < 1/2$ to $1/3$ (see Figures 4.3 and 4.7), like a hyperviscosity, so longer wavelength fluctuations are relatively unaffected.

For other types of flow, such as those with shear, the numerical dissipation in the finite volume methods might not be enough because it only damps modes with short scales in the direction of the flow, irrespective of the scales perpendicular to the flow direction. An example of such is passive advection $\partial\omega/\partial t + \nabla \cdot (\vec{v}\omega) = 0$ in a sheared flow, defined for some numerical constant s as $\vec{v} = \hat{x}sy$, with the initial condition $\omega(t = 0) = \cos(k_0 x)$. The exact solution should be $\omega = \cos(k_0 x - k_0 s t y)$. If $k_0 \ll k_{\max}$ then the numerical dissipation will be negligible, even though the actual solution should cascade to unresolved scales when the y-component of the wave number $k_0 s t = k_{\max}$. This absence of damping will cause a recurrence error on time scales larger than $k_{\max}/(k_0 s)$. For many turbulent systems, the gradients of the advected quantity will be randomly oriented with respect to the flow direction, so there will be sufficient damping near the grid scale. To further improve robustness one could consider adding a subgrid model to introduce damping related to sheared flows.

It is also possible that a subgrid model [16] (even a relatively simple Smagorinsky model) added to the Arakawa method could significantly improve its robustness with respect to the enstrophy spectral shape and the pileup problem, though such a subgrid model by itself would not be enough to eliminate the phase errors that exist even in uniform advection or preserve exact positivity or monotonicity. One could further add limiters only to try to preserve positivity, which would introduce less dissipation than limiters that also try to preserve monotonicity. (These subgrid models and limiters would of course break the conservation properties that the Arakawa method has by itself.) We leave exploration of these possibilities to future work.

Although the extrema-preserving methods were the best performers in our test problems, one should exercise caution when implementing them. In our implementation, the XPPM, SuHu, and SuHu-PPM were comparable in speed to each other but were approximately 2x slower in CPU time per time step per grid point than the Arakawa methods, because we did not focus on optimization. In particular, the code looped over the whole grid twice each time step for the finite volume methods for convenience, because the finite volume advection operators had been implemented separately in the x and y directions. Advection problems are typically memory-bandwidth limited, so this explains the factor of two slowdown compared to the Arakawa methods, which only looped through the grid once to implement the full advection operator.

In essence, this is an example of poor cache optimization. The implementation of the finite volume algorithms could be rewritten to do everything in a single loop, which should speed them up by about 2x. (That the second- and fourth-order Arakawa methods had similar execution times at all grid resolutions further supports the idea of memory-bandwidth limitation.) In modern architectures, where FLOPS are cheap relative to memory access, the differences in speed per grid point for these various algorithms should be small if they are implemented optimally. In problems where an expensive Reimann solver takes most of the CPU time, this would also reduce the sensitivity of the total CPU time to the order of the reconstruction algorithm. Even if the higher-order extrema-preserving methods are somewhat more expensive per grid point, they should often be a better choice, particularly in higher dimensions, because of their better performance at lower grid resolution. Since the compute time for advection problems in d dimensions scales as CN_1^{d+1} (including the shorter time step at higher resolution), where N_1 is the number of grid points in each direction, a modest decrease in the required resolution N_1 can easily offset a factor of two in the coefficient C.

6. Conclusions. We have compared a number of discretizations for the Poisson bracket nonlinearity for 1D passive advection and for the vortex merger problem in 2D incompressible Navier–Stokes. The ubiquitous nature of this nonlinearity allows for the application of these techniques to other equations, notably those used to describe edge gyrokinetic and gyrofluid plasmas.

In agreement with previous work [22], we find that the numerical dissipation in the third-order WENO method [15, 19, 29] can be significant for some applications. The Arakawa method [1] is designed to be dissipation free, and so of course does better in terms of conservation properties in cases where there should not be much dissipation. However, depending on the application at hand, there are other properties of interest beside dissipation rates, such as spectral shape, phase errors, spurious oscillations, and positivity. We have shown that using a higher-order variant of the Arakawa method alone cannot fix positivity violation and spectral pileups.

We have also extended the analysis to include a number of recent finite volume techniques that are both accurate to a higher order and seek to increase the accuracy of solutions at smooth extrema, leading to lower levels of numerical dissipation. These include the high-order limiter implementation of Suresh and Huynh [30], Colella and Sekora's smooth extrema limiter [6] for the piecewise parabolic method [7], and a hybrid technique that uses PPM4 interpolations and the Suresh and Huynh limiter.

The extrema-preserving techniques do extremely well in the passive advection problem, eliminating extraneous oscillations and reducing phase errors while keeping dissipation minimal. In the fully nonlinear problem they are less dissipative than the conventional finite volume methods, reduce unphysical oscillations, resolve details fairly well at low resolution, and model cascade behavior. All methods of course converge to the right answer when the dissipation scale is well resolved, but these recent finite volume methods are fairly robust even at lower resolutions, because their numerical dissipation acts as an approximate subgrid model (though not necessarily an optimal one).

Positivity is an important property to preserve for physical quantities such as density, temperature, and distribution function in simulations of edge plasmas. While the finite volume methods studied here do fairly well at preventing artificial overshoots and preserving positivity, because of the modifications to the limiters to preserve the accuracy of smooth extrema they may still produce small negative overshoots.

However, it is possible to strictly preserve positivity with a small supplement to the limiters used in these algorithms, similar to the constraints they already use to enforce monotonicity.

There are various ways the time-step constraint might be further relaxed while maintaining positivity. One possibility, mentioned in [6], is to couple these algorithms with an FCT algorithm [31].

Another class of techniques that may be worth exploring for these types of problems are discontinuous Galerkin algorithms [4, 18, 35], as they combine some of the favorable features of finite volume and finite elements, and have relatively low phase errors while keeping a local stencil for efficient implementation.

Acknowledgments. We wish to thank J. Candy, P. Colella, D. Martin, and J. Stone for their insight.

REFERENCES

[1] A. Arakawa, *Computational design for long-term numerical integration of the equations of fluid motion: Two-dimensional incompressible flow. Part* I, J. Comput. Phys., 1 (1966), pp. 119–143.
[2] A. Arakawa, *Introduction to "Computational design for long-term numerical integration of the equations of fluid motion: Two-dimensional incompressible flow. Part* I", J. Comput. Phys., 135 (1997), pp. 101–102.
[3] V. Borue and S. A. Orszag, *Forced three-dimensional homogeneous turbulence with hyperviscosity*, Europhys. Lett., 29 (1995), pp. 687–692.
[4] B. Cockburn and C.-W. Shu, *The Runge-Kutta discontinuous Galerkin method for conservation laws* V: *Multidimensional systems*, J. Comput. Phys., 141 (1998), pp. 199–224.
[5] P. Colella, M. R. Dorr, J. A. F. Hittinger, and D. F. Martin, *High order finite-volume methods in mapped coordinates*, J. Comput. Phys., 230 (2011), pp. 2952–2976.
[6] P. Colella and M. D. Sekora, *A limiter for PPM that preserves accuracy at smooth extrema*, J. Comput. Phys., 227 (2008), pp. 7069–7076.
[7] P. Colella and P. R. Woodward, *The piecewise parabolic method (PPM) for gas-dynamical simulations*, J. Comput. Phys., 54 (1984), pp. 174–201.
[8] M. Frigo and S. G. Johnson, *The design and implementation of FFTW*3, Proc. IEEE, 93 (2005), pp. 216–231.
[9] U. Frisch, S. Kurien, R. Pandit, W. Pauls, S. S. Ray, A. Wirth, and J.-Z. Zhu, *Hyperviscosity, Galerkin truncation, and bottlenecks in turbulence*, Phys. Rev. Lett., 101 (2008), 144501.
[10] S. Gottlieb, D. I. Ketcheson, and C.-W. Shu, *High order strong stability preserving time discretizations*, J. Sci. Comput., 38 (2009), pp. 251–289.
[11] S. Gottlieb, C.-W. Shu, and E. Tadmor, *Strong stability preserving high-order time discretization methods*, SIAM Rev., 43 (2001), pp. 89–112.
[12] D. I. Ketcheson, *Highly efficient strong stability-preserving Runge-Kutta methods with low-storage implementations*, SIAM J. Sci. Comput., 30 (2008), pp. 2113–2136.
[13] R. H. Kraichnan, *Inertial ranges in two-dimensional turbulence*, Phys. Fluids, 10 (1967), pp. 1417–1423.
[14] R. H. Kraichnan and D. Montgomery, *Two-dimensional turbulence*, Rep. Prog. Phys., 43 (1980), pp. 547-619.
[15] A. Kurganov and D. Levy, *A third-order semidiscrete central scheme for conservation laws and convection-diffusion equations*, SIAM J. Sci. Comput., 22 (2000), pp. 1461–1488.
[16] M. Lesieur, *Turbulence in Fluids*, Springer, Dordrecht, The Netherlands, 2008.
[17] R. J. Leveque, *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press, Cambridge, UK, 2002.
[18] J.-G. Liu and C.-W. Shu, *A high-order discontinuous Galerkin method for* 2*D incompressible flows*, J. Comput. Phys., 160 (2000), pp. 577–596.
[19] X.-D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.
[20] D. Martin and P. Colella, *Private communication*, 2008.
[21] P. McCorquodale and P. Colella, *A high-order finite-volume method for conservation laws on locally refined grids*, Comm. Appl. Math. Comput. Sci., 6 (2011), pp. 1–25.

[22] V. Naulin and A. H. Nielsen, *Accuracy of spectral and finite difference schemes in 2D advection problems*, SIAM J. Sci. Comput., 25 (2003), pp. 104–126.
[23] I. Navon and Z. Alperson, *Application of fourth-order finite differences to a baroclinic model of the atmosphere*, Meteor. Atmos. Phys., 27 (1978), pp. 1–19.
[24] K. Ohkitani, *Wave number space dynamics of enstrophy cascade in a forced two-dimensional turbulence*, Phys. Fluids A, 3 (1991), pp. 1598–1611.
[25] J. J. Rasmussen, A. H. Nielsen, and V. Naulin, *Dynamics of vortex interactions in two-dimensional flows*, Phys. Scripta, T98 (2002), pp. 29–33.
[26] L. F. Richardson, *The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam*, R. Soc. Lond. Philos. Trans. Ser. A Math. Phys. Eng. Sci., 210 (1911), pp. 307–357.
[27] L. F. Richardson and J. A. Gaunt, *The deferred approach to the limit*, R. Soc. Lond. Philos. Trans. Ser. A Math. Phys. Eng. Sci., 226 (1927), pp. 299-349.
[28] B. D. Scott, *An implicit simulation of drift-wave turbulence in a sheared magnetic field*, J. Comput. Phys., 78 (1988), pp. 114–137.
[29] C.-W. Shu, *Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory Schemes for Hyperbolic Conservation Laws*, Technical report TR-97-65, Institute for Computer Applications in Science and Engineering, (ICASE), Hampton, VA, 1997.
[30] A. Suresh and H. T. Huynh, *Accurate monotonicity-preserving schemes with Runge-Kutta time stepping*, J. Comput. Phys., 136 (1997), pp. 83–99.
[31] S. T. Zalesak, *Fully multidimensional flux-corrected transport algorithms for fluids*, J. Comput. Phys., 31 (1979), pp. 335–362.
[32] D. R. Durran, *Numerical Methods for Fluid Dynamics With Applications to Geophysics*, 2nd ed., Springer, New York, 2010.
[33] L. J. Wicker and W. C. Skamarock, *Time-splitting methods for elastic models using forward time schemes*, Monthly Weather Rev., 130 (2002), pp. 2088–2097.
[34] R. J. Purser, *Accuracy considerations of time-splitting methods for models using two-time-level schemes*, Monthly Weather Rev., 135 (2007), pp. 1158–1164.
[35] T. Zhou, Y. Li, and C.-W. Shu, *Numerical comparison of WENO finite volume and Runge-Kutta discontinuous Galerkin methods*, J. Sci. Comput., 16 (2001), pp. 145–171.