Communications in Applied Mathematics and Computational Science

AN UNSPLIT, HIGHER-ORDER GODUNOV METHOD USING QUADRATIC RECONSTRUCTION FOR ADVECTION IN TWO DIMENSIONS

SANDRA MAY, ANDREW NONAKA, ANN ALMGREN AND JOHN BELL

vol. 6 no. 1 2011

mathematical sciences publishers

COMM. APP. MATH. AND COMP. SCI.
Vol. 6, No. 1, 2011

msp

AN UNSPLIT, HIGHER-ORDER GODUNOV METHOD USING QUADRATIC RECONSTRUCTION FOR ADVECTION IN TWO DIMENSIONS

SANDRA MAY, ANDREW NONAKA, ANN ALMGREN AND JOHN BELL

Linear advection of a scalar quantity by a specified velocity field arises in a number of different applications. Important examples include the transport of species and energy in low Mach number models for combustion, atmospheric flows and astrophysics, and contaminant transport in Darcy models of saturated subsurface flow. In this paper, we present a customized finite volume advection scheme for this class of problems that provides accurate resolution for smooth problems while avoiding undershoot and overshoot for nonsmooth profiles. The method is an extension of an algorithm by Bell, Dawson and Shubin (BDS), which was developed for a class of scalar conservation laws arising in porous media flows in two dimensions. The original BDS algorithm is a variant of unsplit, higher-order Godunov methods based on construction of a limited bilinear profile within each computational cell. The new method incorporates quadratic terms in the polynomial reconstruction, thereby reducing the L^1 error and better preserving the shape of advected profiles while continuing to satisfy a maximum principle for constant coefficient linear advection. We compare this new method to several other approaches, including the bilinear BDS method and unsplit piecewise parabolic (PPM) methods.

1. Introduction

The focus of much of the literature on numerical methods for hyperbolic partial differential equations is on general systems of conservation laws, particularly the compressible Euler equations (see [14] for an overview of the literature). However, there are a number of important problems in science and engineering where we need to solve linear advection problems of the form

$$s_t + (us)_x + (vs)_y = 0, \tag{1}$$

where $s = s(x, y, t)$ is a scalar field and (u, v) represents a known velocity field. One important example of this type of problem arises in projection algorithms for incompressible and other low Mach number flows, where the velocity field used

MSC2000: 35-04, 35L65.
Keywords: Godunov method, scalar conservation law, two-dimensional quadratic reconstruction.

for advection is constructed during the time step using a projection that enforces the divergence constraint. Applications of these low Mach number projection algorithms include low Mach number terrestrial combustion [11], nuclear flame simulation [4], low Mach number stratified atmospheric [23] and astrophysical flows [19], as well as general variable-density incompressible flow [2]. In these cases the density, species, and other scalar quantities are advected by a velocity field that is calculated before the advection step is performed. Advection problems also arise in contaminant transport in saturated groundwater flow [20]. We note that in several of the above problems, the full evolution equation for s often includes a right hand side representing reactions, diffusion or other processes. However, discretization approaches typically separate the computation of the advective flux from the treatment of the other terms. In particular, diffusion is typically treated in a form in which explicit hyperbolic fluxes appear as source terms in an implicit discretization of diffusion; reactions are typically included via operator splitting. Consequently, here we will focus on the homogeneous system; the reader is referred to the literature cited above for discussion of how to incorporate other processes.

There are several aspects of the class of problems we are considering that are worth noting. First, in most of these applications the velocity field is determined by solving a constraint equation that explicitly encapsulates a specific discrete form of the divergence of the velocity field with which we want the hyperbolic discretization to be consistent. Furthermore, we only have a limited characterization of the velocity field, typically integral averages of the normal component on edges of grid cells. Another aspect of the class of problems being considered is that they can be highly sensitive to overshoot and undershoot. For example, many chemical reaction systems are ill-defined when a species has a negative concentration. Similarly, although harder to detect, errors associated with overshoot in a species concentration can be significantly enhanced by the kinetics mechanism. Thus, we would like a method that provides an accurate discretization and preserves the shape of advected profiles while avoiding overshoot and undershoot. One final consequence of the type of problems we consider is that the computational cost is dominated by elliptic solvers, reaction networks, and/or calls to the equation of state, so the overall cost of advection is minor in comparison; thus accuracy is of more importance than cost in choosing the advection algorithm.

There is a vast literature on numerical methods for first-order hyperbolic partial differential equations, all of which can potentially be adapted to advection by a known velocity field. It is beyond the scope of this paper to survey all of that work; however, we will briefly describe some of the main themes underlying some of these approaches. We first note that dimensional operator splitting does not work well for advection by a nonconstant divergence-free velocity field. In a dimensionally split

approach, the fluid can experience an artificial compression in one sweep combined with an artificial expansion in another sweep, which can lead to significant artifacts. An example showing these types of artifacts is presented in [1]. Thus, we restrict ourselves here to unsplit discretizations.

The first unsplit second-order Godunov method, based on linear reconstruction, was presented by Colella [7], and was later extended to three dimensions by Saltzman [21]. Colella [7] motivated the development of the unsplit Godunov algorithm by introduction of the corner transport upwind (CTU) method. The CTU method is a first-order upwind advection scheme that incorporates diagonal coupling based on a piecewise-constant approximation and the geometry of characteristics for constant coefficient advection. However, the geometric interpretation was abandoned in the extension to general systems of conservation laws. Miller and Colella [17] developed a version of the unsplit scheme based on the piecewise parabolic method (PPM) of Colella and Woodward [9]. This approach uses the same formalism as the piecewise linear algorithms but constructs a parabolic rather than a linear profile in each coordinate direction. There has been some recent work aimed at improving the limiters for PPM. Colella and Sekora [8] developed a new PPM limiter that preserves accuracy at smooth extrema but suffers from sensitivity to roundoff error; more recently McCorquodale and Colella [16] introduced an improvement to that limiter which is less sensitive to roundoff error (P. Colella, private communication, 2010).

LeVeque [13] introduced higher-order advection schemes based on geometric ideas derived from a wave propagation perspective. The WAF approach of Billett and Toro [5] and the Mot-ICE-P1 scheme of Noelle [18] use similar geometric ideas and, in fact, share a number of features of the scheme that will be our starting point. Smolarkiewicz and collaborators developed multidimensional advection schemes for geophysical flows based on flux-corrected transport ideas; see [23] and the references cited therein. Another class of schemes is the ADER-type schemes developed by Toro and collaborators; see, for example, [24]. These schemes are somewhat more algebraic in their construction, using a Cauchy–Kowalewski procedure and Taylor series expansion to evaluate approximations at quadrature nodes on space-time edges of cells. Another class of schemes that has become popular for a wide range of problems is WENO-type schemes. The reader is referred to [22] for a general discussion of these types of methods. Of particular interest for multidimensional advection are unsplit, multidimensional versions of WENO such as those by Levy, Puppo, and Russo [15] and Kurganov and Petrova [12]. A final category of schemes is discontinuous Galerkin finite element methods. There have been recent special issues of journals focused on discontinuous Galerkin; see [10; 6]. Unlike the finite volume schemes discussed above, discontinuous Galerkin methods advance an entire polynomial representation in time.

Although all of the above literature is applicable to linear advection, most of these methods are designed for more general conservation laws. One approach to solving (1) is simply to adapt a method for general systems to the special case considered here. However, we wish to exploit the special structure of the linear advection problem to design a finite volume method that best meets the targets of accuracy and shape preservation without overshoot or undershoot and fits the existing constraints in terms of specification of the velocity field.

The method presented here is an extension of the two-dimensional, higher order scheme for linear advection and scalar conservation laws developed by Bell, Dawson and Shubin [3]. It exploits the observation that the equation is (trivially) diagonalizable and bases the construction of the fluxes on the detailed geometry of the characteristics. For constant coefficient advection, the Bell, Dawson and Shubin (BDS) scheme is numerically equivalent to fitting a profile within each cell, analytically advecting the reconstructed solution and averaging the solution onto the grid. We note also that both the method presented in this paper and the original BDS algorithm are fully explicit in time, and do not require a Runge–Kutta procedure.

This original BDS algorithm constructs a limited bilinear profile within each cell. The method presented here extends the BDS approach by constructing a limited, two-dimensional biquadratic representation of the solution within each cell. The two key elements of the algorithm are the construction of the limited quadratic profile and the modification of the quadrature rules used to compute the fluxes. In the next section, we review the original BDS algorithm. We then discuss the modifications needed to include the quadratic terms in the reconstruction and how to modify the flux computation to account for those terms. Next we present computational results comparing the quadratic BDS algorithm with the original BDS algorithm and two variations of the unsplit PPM algorithm. The initial tests are for advection by a prescribed velocity field. We then illustrate the performance of the algorithm for advection of density and a tracer in a variable density projection algorithm. Finally we present comparisons with some alternative schemes that have been discussed in the literature along with some discussions of their characteristics.

2. Bilinear BDS method

2.1. ***Overview.*** The BDS method was originally developed for scalar conservation laws that arise in porous media flow, of the form

$$s_t + [uf(s)]_x + [vg(s)]_y = qh(s), \tag{2}$$

where (u, v) represents a spatially dependent velocity field and the right side represents point sources and sinks of fluid of strength $q(x, y)$ and composition h.

The fluid was assumed to be incompressible; that is,

$$u_x + v_y = 0 \tag{3}$$

away from the support of q. A detailed description of the method can be found in [3]. Since our focus here is on linear advection in low Mach number models, we will restrict our consideration to the case where $f(s) = g(s) = s$ and not consider sources or sinks of fluid, so that $q = 0$. Although the test cases will all consider a divergence-free velocity field so that the equation satisfies a maximum principle, we will not make any assumptions about the divergence of the velocity in the specification of the algorithm. We summarize the original BDS method in three steps:

Step I. Construct an appropriately limited bilinear polynomial representation of s at time t^n in each cell (i, j).

Step II. Define edge values $s_{i+1/2,j}$, $s_{i,j+1/2}$, etc., by integrating over time and space assuming the bilinear profile.

Step III. Update the solution at time t^{n+1} using a conservative update,

$$\begin{aligned} s_{ij}^{n+1} = s_{ij}^n &- \frac{\Delta t}{\Delta x}(u_{i+1/2,j}s_{i+1/2,j} - u_{i-1/2,j}s_{i-1/2,j}) \\ &- \frac{\Delta t}{\Delta y}(v_{i,j+1/2}s_{i,j+1/2} - v_{i,j-1/2}s_{i,j-1/2}). \end{aligned} \tag{4}$$

Here Δt is the time step and Δx and Δy are the mesh spacings in the x- and y-directions, respectively.

In the next section we describe the construction of the bilinear polynomial; following that we discuss how to construct the face values by integrating over time and space.

2.2. ***Construction of the bilinear polynomial.*** Here, we describe an algorithm for Step I, the construction of a bilinear polynomial representation of s at time t^n, written in the form

$$p_{ij}^{\mathrm{bl}}(x, y) = s_{xy,ij}\,(x - x_i)(y - y_j) + s_{x,ij}\,(x - x_i) + s_{y,ij}\,(y - y_j) + \hat{s}, \tag{5}$$

where (x_i, y_j) denotes the cell center of cell (i, j).

To obtain estimates for the corner values on each cell, a multidimensional analog of the procedure used by Colella and Woodward [9] was chosen. For equally spaced grids this leads to

$$\begin{aligned} s_{i+1/2,j+1/2} = \big[&s_{i-1,j-1} - 7(s_{i,j-1} + s_{i+1,j-1}) + s_{i+2,j-1} \\ &-7s_{i-1,j} + 49(s_{ij} + s_{i+1,j}) - 7s_{i+2,j} \\ &-7s_{i-1,j+1} + 49(s_{i,j+1} + s_{i+1,j+1}) - 7s_{i+2,j+1} \\ &+s_{i-1,j+2} - 7(s_{i,j+2} + s_{i+1,j+2}) + s_{i+2,j+2}\big]/144. \end{aligned} \tag{6}$$

The estimates for the four corner values LL, LH, RL, and RH (left low, left high, right low, and right high) are then used to calculate slopes on the cell (i, j):

$$s_{x,ij} = \frac{(\mathrm{RH} + \mathrm{RL}) - (\mathrm{LH} + \mathrm{LL})}{2\Delta x}, \tag{7a}$$

$$s_{y,ij} = \frac{(\mathrm{LH} + \mathrm{RH}) - (\mathrm{LL} + \mathrm{RL})}{2\Delta y}, \tag{7b}$$

$$s_{xy,ij} = \frac{(\mathrm{RH} - \mathrm{RL}) - (\mathrm{LH} - \mathrm{LL})}{\Delta x \Delta y}. \tag{7c}$$

The constant term $\hat{s}$ is given by s_{ij}. Note that the integral over the linear and bilinear terms vanishes because the polynomial is centered at the cell center. So by construction, the average value of the polynomial over the cell (i, j) equals the cell value s_{ij}.

Remark. The presence of the bilinear term $s_{xy,ij}$ leads to an improved preservation of shapes for off-axis movement compared to methods which only include linear terms (and possibly pure quadratic terms) in their profile reconstruction.

2.3. ***Limiting the bilinear polynomial.*** As noted in [3], the limiting of (5) can be cast as an optimization problem: minimize, in each cell, the L^2 norm of the difference between the limited polynomial and the original interpolation function given by (7a)–(7c) subject to two constraints:

(1) The average of the polynomial evaluated at the four corners of cell (i, j) must equal the cell average s_{ij}.

(2) The polynomial evaluated at a corner must lie between the minimum and the maximum of the cell averages of the four cells surrounding the corner.

However, to reduce the computational cost a simple heuristic algorithm was developed that produced results within 10% of the results obtained from the minimization procedure in terms of overall L^1 error in a variety of test cases. This heuristic algorithm goes as follows:

Step i. Compute the values of the bilinear polynomial at the cell corners:

$$\mathrm{LL}_{\mathrm{temp}} = s_{ij} - \frac{\Delta x}{2} s_{x,ij} - \frac{\Delta y}{2} s_{y,ij} + \frac{\Delta x}{2}\frac{\Delta y}{2} s_{xy,ij}, \tag{8a}$$

$$\mathrm{LH}_{\mathrm{temp}} = s_{ij} - \frac{\Delta x}{2} s_{x,ij} + \frac{\Delta y}{2} s_{y,ij} - \frac{\Delta x}{2}\frac{\Delta y}{2} s_{xy,ij}, \tag{8b}$$

$$\mathrm{RL}_{\mathrm{temp}} = s_{ij} + \frac{\Delta x}{2} s_{x,ij} - \frac{\Delta y}{2} s_{y,ij} - \frac{\Delta x}{2}\frac{\Delta y}{2} s_{xy,ij}, \tag{8c}$$

$$\mathrm{RH}_{\mathrm{temp}} = s_{ij} + \frac{\Delta x}{2} s_{x,ij} + \frac{\Delta y}{2} s_{y,ij} + \frac{\Delta x}{2}\frac{\Delta y}{2} s_{xy,ij}. \tag{8d}$$

Step ii. Check to see if each temporary value is in the range defined by the four neighboring cell values, for example, check whether $\mathrm{LL}_{\mathrm{temp}}$ lies between $\min_{\mathrm{LL}}$ and $\max_{\mathrm{LL}}$, where these limits are defined by

$$\min_{\mathrm{LL}} = \min(s_{i-1,j-1}, s_{i,j-1}, s_{i-1,j}, s_{ij}), \tag{9a}$$

$$\max_{\mathrm{LL}} = \max(s_{i-1,j-1}, s_{i,j-1}, s_{i-1,j}, s_{ij}). \tag{9b}$$

If all of the temporary values $\mathrm{LL}_{\mathrm{temp}}, \ldots, \mathrm{RH}_{\mathrm{temp}}$ happen to lie between their respective bounds, the polynomial does not need to be limited. In that case, skip Steps iii and iv, and keep the original values for s_x, s_y, and s_{xy} as computed in equations (7a)–(7c). Otherwise, constrain these temporary values so they do not introduce any new extrema, for example, set

$$\mathrm{LL}_{\mathrm{temp}} = \max[\min(\mathrm{LL}_{\mathrm{temp}}, \max_{\mathrm{LL}}), \min_{\mathrm{LL}}]. \tag{10}$$

Step iii. Iterative Loop:

(a) Compute the difference between the sum of the temporary values and the cell average, s_{ij}, multiplied by four:

$$\mathrm{sumdif} = (\mathrm{LL}_{\mathrm{temp}} + \mathrm{LH}_{\mathrm{temp}} + \mathrm{RL}_{\mathrm{temp}} + \mathrm{RH}_{\mathrm{temp}}) - 4s_{ij}. \tag{11}$$

Assume for now that sumdif ≥ 0; the case where sumdif ≤ 0 is analogous.

(b) Find out which temporary corner values are larger (smaller) than s_{ij} by more than $\epsilon = 10^{-10}$. Let kdp be the number of corners with this property.

(c) *Loop over corners:* If the temporary corner value is larger than s_{ij} by more than $\epsilon = 10^{-10}$, make the following assignments (using corner $\mathrm{LL}_{\mathrm{temp}}$ as an example assuming that $\mathrm{LL}_{\mathrm{temp}}$ fulfills the criterion in (b)):

- redfac $\leftarrow \min[\mathrm{sumdif}/\mathrm{kdp}, \mathrm{LL}_{\mathrm{temp}} - \min(s_{i-1,j-1}, s_{i,j-1}, s_{i-1,j}, s_{ij})]$
- kdp $\leftarrow$ kdp $- 1$
- sumdif $\leftarrow$ sumdif $-$ redfac
- $\mathrm{LL}_{\mathrm{temp}} \leftarrow \mathrm{LL}_{\mathrm{temp}} -$redfac

Step iv. Compute the final slopes following equations (7a)–(7c), using $\mathrm{LL}_{\mathrm{temp}}$, etc., rather than LL, etc.

Remark. Numerical tests have shown that three iterations are sufficient to complete the limiting process in Step iii.

2.4. ***Construction of edge states.*** The other key part of the BDS algorithm is the calculation of the edge states $s_{i+1/2,j}$, $s_{i,j+1/2}$, etc., in Step II that are used to construct the update terms in (4) in Step III. For clarity of exposition, we focus on the problem

$$s_t + us_x + vs_y = 0, \quad u, v > 0 \text{ constant}; \tag{12}$$

the extension to spatially varying (u, v) will be described later.

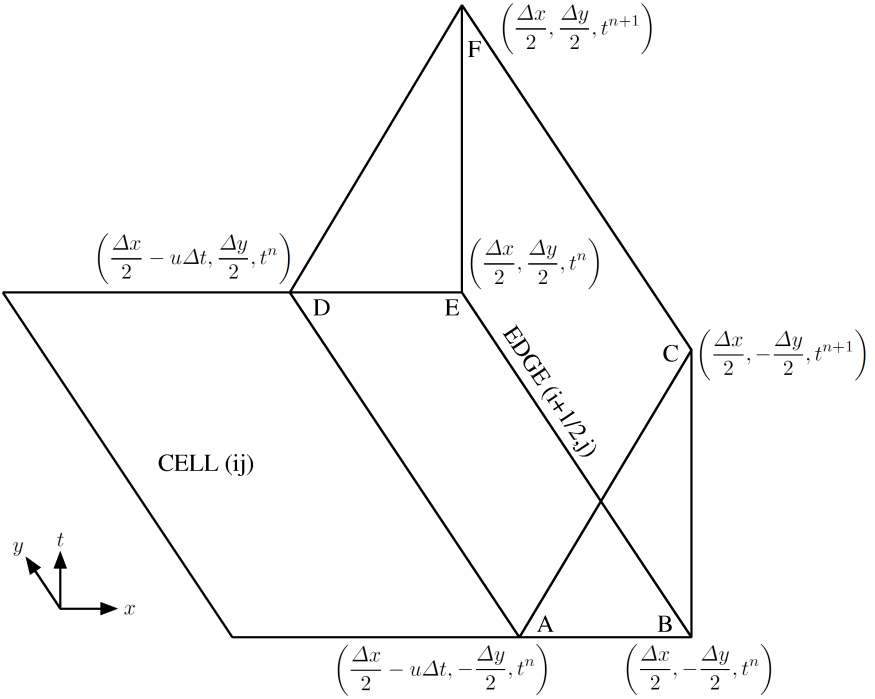


Figure 1. Characteristic domain of dependence of edge $(i+1/2,\, j)$. x- and y-coordinates specified relative to block center. Figure taken from [3].

2.4.1. *Constant velocity.* Since $u > 0$, the characteristic domain of dependence of edge $(i+1/2,\, j)$ is the space-time region $ABCDEF$ as depicted in Figure 1. Thus, to compute $s_{i+1/2,\,j}$ we compute the average of s over the face $BCEF$, which we denote $s^L_{i+1/2,\,j}$. To obtain $s^L_{i+1/2,\,j}$, we integrate (12) over $ABCDEF$ and use the divergence theorem, taking advantage of the fact that the resulting integral over face $ACDF$ vanishes to obtain

$$u\, s^L_{i+1/2,\,j}\, \Delta t \Delta y = u \iint_{BCEF} s\, dy\, dt$$
$$= \iint_{ABDE} s\, dx\, dy + v \iint_{ABC} s\, dx\, dt - v \iint_{DEF} s\, dx\, dt. \quad (13)$$

By construction, s is piecewise bilinear and the edges of $ABDE$ are aligned with the coordinate axes. Therefore, one can use the midpoint formula to evaluate the integral over $ABDE$ exactly:

$$\iint_{ABDE} s\, dx\, dy = u\Delta t \Delta y\, s_{M,F}, \quad \text{where } s_{M,F} = \frac{\Delta x - u\Delta t}{2} s_{x,ij} + s_{ij}. \quad (14)$$

Consequently,

$$u\, s^{L}_{i+1/2,j} = \frac{u}{\Delta t \Delta y} \iint_{BCEF} s\, dy\, dt$$
$$= u\, s_{M,F} - \frac{v}{\Delta t \Delta y}\Big(\iint_{DEF} s\, dx\, dt - \iint_{ABC} s\, dx\, dt \Big). \qquad (15)$$

To evaluate the integrals over the surface triangles we use the same idea. Integrating $s_t + u s_x + v s_y = 0$ over the volume $DEFG$ shown in Figure 2, one can express the integral over DEF in terms of the integral over DEG, using the observation that contributions over the faces GEF and GFD vanish, to obtain:

$$v \iint_{DEF} s\, dx\, dt = \iint_{DEG} s\, dx\, dy. \qquad (16)$$

For the evaluation of the integral on the right side, the midpoint quadrature rule can be applied for the constant and linear parts of the bilinear polynomial on cell (i, j). For the bilinear term this rule is not exact. Therefore, the bilinear term is evaluated at the midpoints of the three edges and their sum is divided by three. The evaluation of the integral over the face ABC in (15) is analogous, noting that the characteristic domain of dependence extends into cell $(i, j-1)$, and therefore we evaluate the bilinear polynomial on cell $(i, j-1)$.

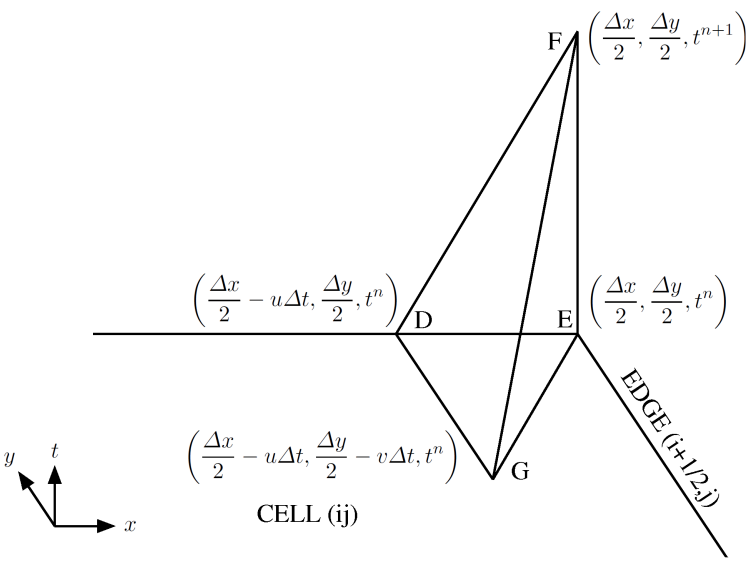


Figure 2. Characteristic domain of dependence of triangle DEF. The x- and y-coordinates are specified relative to block center. Figure taken from [3].

2.4.2. *Nonconstant velocity.* Here we generalize the construction above to the more general problem where (u, v) is spatially varying. At each edge we need to calculate the upwind edge state. If we assume $u_{i+1/2,j} > 0$, then we need to calculate $s^L_{i+1/2,j}$. For nonconstant velocity, we write the equation in the form

$$s_t + us_x + (vs)_y + su_x = 0,$$

and integrate over region $ABCDEF$ to obtain

$$\begin{aligned} &s^L_{i+1/2,j} \\ &\quad = s_{M,F} - \frac{\Delta t}{2}\frac{(u_{i+1/2,j} - u_{i-1/2,j})}{\Delta x} s_{M,F} - \frac{\Delta t}{2\Delta y}(v_{i,j+1/2}\Gamma^+ - v_{i,j-1/2}\Gamma^-), \end{aligned} \tag{17}$$

where Γ^+ and Γ^- represent the average values of the flux vs over the triangles DEF and ABC, respectively, and

$$s_{M,F} = \frac{\Delta x - u_{i+1/2,j}\Delta t}{2} s_{x,ij} + s_{ij}. \tag{18}$$

Here the term

$$-\frac{\Delta t}{2}\frac{(u_{i+1/2,j} - u_{i-1/2,j})}{\Delta x} s_{M,F}$$

approximates the volume integral of su_x over $ABCDEF$ using explicit Euler quadrature in time and treating u_x as a constant given by the difference of the edge velocities.

To compute the transverse correction term Γ^+, we write the equation in the form

$$s_t + us_x + vs_y + s(u_x + v_y) = 0.$$

If $v_{i,j+1/2} > 0$, we define

$$\begin{aligned} s^+_m &= \frac{1}{m(DEG)} \iint_{DEG} s\, dx\, dy \\ &= \frac{s_{xy,ij}}{12}\left[3\Delta x\Delta y - 4u_{i+1/2,j}\Delta t\Delta y - 2v_{i,j+1/2}\Delta x\Delta t + 3u_{i+1/2,j}v_{i,j+1/2}(\Delta t)^2\right] \\ &\qquad + \frac{s_{x,ij}}{6}(3\Delta x - 4\Delta t u_{i+1/2,j}) + \frac{s_{y,ij}}{6}(3\Delta y - 2\Delta t v_{i,j+1/2}) + s_{ij}. \end{aligned} \tag{19}$$

Then

$$\begin{aligned} \Gamma^+ &= \frac{1}{m(DEG)}\left(\iint_{DEG} s\, dx\, dy - \iiint_{DEFG} s(u_x + v_y)\, dx\, dy\, dt \right) \\ &= s^+_m - \frac{1}{m(DEG)} \iiint_{DEFG} s(u_x + v_y)\, dx\, dy\, dt \\ &= s^+_m \cdot \left[1 - \frac{\Delta t}{3}\left(\frac{u_{i+1/2,j} - u_{i-1/2,j}}{\Delta x} + \frac{v_{i,j+1/2} - v_{i,j-1/2}}{\Delta y}\right)\right], \end{aligned} \tag{20}$$

where we have again used explicit Euler quadrature in time for the volume integral.

Otherwise if $v_{i,j+1/2} < 0$, we define

$$u_{\text{proj}} = \begin{cases} u_{i+1/2,j+1} & \text{if } u_{i+1/2,j} \cdot u_{i+1/2,j+1} > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{21}$$

that is, in the second case we project corner G in Figure 2 from cell $(i+1,\ j+1)$ onto the edge $(i+1/2,\ j+1)$. This ensures that the characteristic domain of dependence is contained within one cell. Then s_m^+ is given by

$$\begin{aligned} s_m^+ = \frac{s_{xy,i,j+1}}{12}\big[&-3\Delta x\Delta y + 2(u_{i+1/2,j} + u_{\text{proj}})\Delta t\Delta y - 2v_{i,j+1/2}\Delta x\Delta t \\ &+ (u_{i+1/2,j} + 2u_{\text{proj}})v_{i,j+1/2}(\Delta t)^2\big] \\ &+ \frac{s_{x,i,j+1}}{6}\left[3\Delta x - 2\Delta t(u_{i+1/2,j} + u_{\text{proj}})\right] \\ &+ \frac{s_{y,i,j+1}}{6}\left(-3\Delta y - 2\Delta t v_{i,j+1/2}\right) + s_{i,j+1} \end{aligned} \tag{22}$$

and

$$\Gamma^+ = s_m^+ \cdot \left[1 - \frac{\Delta t}{3}\left(\frac{u_{i+1/2,j+1} - u_{i-1/2,j+1}}{\Delta x} + \frac{v_{i,j+3/2} - v_{i,j+1/2}}{\Delta y}\right)\right]. \tag{23}$$

The formula for Γ^- is analogous. We note that Γ^+ computed for edge $(i+1/2,\ j)$ is, in general, not the same as Γ^- computed for edge $(i+1/2,\ j+1)$.

If $u_{i,j+1/2} < 0$, we need to calculate the edge value from cell $(i+1,\ j)$, which we denote by $s^R_{i+1/2,j}$, using formulae analogous to the above. The calculation of $s_{i,j+1/2}$ is done similarly.

3. New quadratic BDS method

3.1. ***Overview.*** The main drawback of the bilinear BDS method compared to PPM-style methods is that it uses only a bilinear polynomial for the profile reconstruction. As a result, the method is only second-order accurate. To address that limitation we now include quadratic terms in the polynomial reconstruction. Each step of the quadratic BDS method is similar to that of the bilinear method; Step III is unchanged, while Step I and Step II now differ because we work with a quadratic rather than bilinear polynomial.

3.2. ***Construction of the quadratic polynomial.*** Here, we describe an algorithm for the construction of a quadratic polynomial representation of s at time t^n, written in the form

$$\begin{aligned} p^{\text{q}}_{ij}(x,\ y) = s_{xx,ij}(x - x_i)^2 + s_{yy,ij}(y - y_j)^2 + s_{xy,ij}(x - x_i)(y - y_j) \\ + s_{x,ij}(x - x_i) + s_{y,ij}(y - y_j) + \bar{s}. \end{aligned} \tag{24}$$

We denote the constant term by $\bar{s}$ instead of $\hat{s}$ (as we did for the bilinear polynomial p^{bl}) since the constant term will no longer be equal to s_{ij}. Note that this is not a full biquadratic polynomial; no mixed quadratic terms are included. The construction and limiting of a full biquadratic polynomial would be considerably more complicated and would not lead to a higher order of convergence.

We begin the construction of the quadratic polynomial as in the bilinear method, using (6) to define corner values for each cell. We will determine the quadratic terms independently.

To obtain an estimate for s_{xx} at the center of cell (i, j) we construct the quartic polynomial whose cell average matches the cell averages $s_{i-2,j}$, $s_{i-1,j}$, s_{ij}, $s_{i+1,j}$, and $s_{i+2,j}$. We then approximate the second derivative of the function s by the second derivative of the quartic polynomial at the center of cell (i, j). This leads to the following formulae:

second derivative at cell center of cell (i, j) in x-direction =

$$\frac{1}{8(\Delta x)^2}\left(-s_{i-2,j} + 12s_{i-1,j} - 22s_{ij} + 12s_{i+1,j} - s_{i+2,j}\right), \quad (25a)$$

second derivative at cell center of cell (i, j) in y-direction =

$$\frac{1}{8(\Delta y)^2}\left(-s_{i,j-2} + 12s_{i,j-1} - 22s_{ij} + 12s_{i,j+1} - s_{i,j+2}\right). \quad (25b)$$

By construction, these formulae are exact for one-dimensional polynomials up to order four. (In fact, they are even exact for a quintic polynomial due to their symmetry.) The coefficients for the quadratic terms $s_{xx,ij}$ and $s_{yy,ij}$ are then given by dividing the estimates for the second derivatives by two.

We then want to construct a polynomial out of the above information. We calculate $s_{xy,ij}$, $s_{x,ij}$, and $s_{y,ij}$ from the estimates for the corner values RH, RL, LH, and LL using equations (7a)–(7c), and $s_{xx,ij}$ and $s_{yy,ij}$ out of the estimates for the second derivatives:

$$s_{xx,ij} = \tfrac{1}{2}(\text{second derivative at cell center of cell } (i, j) \text{ in } x\text{-direction}), \quad (26a)$$

$$s_{yy,ij} = \tfrac{1}{2}(\text{second derivative at cell center of cell } (i, j) \text{ in } y\text{-direction}). \quad (26b)$$

To make sure that the average of the polynomial over cell (i, j) equals the cell average s_{ij}, we redefine the constant term

$$\bar{s} = s_{ij} - \frac{1}{\Delta x \Delta y}\iint_{\mathrm{cell}(i,j)} s_{xx,ij}(x - x_i)^2 + s_{yy,ij}(y - y_j)^2\, dx\, dy \quad (27a)$$

$$= s_{ij} - \tfrac{1}{12}\left[s_{xx,ij}(\Delta x)^2 + s_{yy,ij}(\Delta y)^2\right]. \quad (27b)$$

It is straightforward to show that this algorithm reconstructs a polynomial of the form (24) exactly.

3.3. ***Limiting the quadratic profile.*** The limiting of the quadratic polynomial (24) is split up into three parts:

Step 1. Test whether all estimated corner values are smaller (or larger) than the cell average s_{ij}. This can happen, for example, if the peak of a Gaussian lies in the middle of a cell. If that is the case, we set the polynomial to be constant on that cell with the value s_{ij} and the limiting process is complete.

Step 2. Otherwise, we attempt to accept the polynomial with unlimited slopes $s_{xy,ij}$, $s_{x,ij}$, and $s_{y,ij}$ and only limit the quadratic coefficients $s_{xx,ij}$ and $s_{yy,ij}$ appropriately. The resulting polynomial is tested to see if it is suitable. If so, the limiting is complete.

Step 3. In the third step, we first construct the *limited* bilinear profile using the original BDS approach. We then adjust $s_{xx,ij}$ and $s_{yy,ij}$ to construct a suitable quadratic polynomial.

The tests in Step 2 usually fail close to a discontinuity, that is, we need to use the more restrictive limiting in Step 3 for that case. Furthermore, these tests do ensure that the minimum and maximum values of the polynomial (if accepted in Step 2) are bounded by the cell values of neighboring cells (in order to satisfy a maximum principle for constant coefficient linear advection). We first discuss our general strategy for limiting the quadratic terms before giving detailed algorithms.

For the limiting of $s_{xx,ij}$ and $s_{yy,ij}$ we mainly consider the partial derivatives of the quadratic polynomial p^{q} in the x- and in the y-direction, or to be more precise: we determine whether p^{q}_x and/or p^{q}_y vanish in the interior of the cell. In Step 2 of our limiting procedure we limit the quadratic coefficients if both partial derivatives happen to vanish in the same cell, that is, if we happen to have an interior extremum or saddle point. In the more restrictive Step 3 of our limiting we limit $s_{xx,ij}$ if p^{q}_x vanishes in the interior of the cell independently of p^{q}_y. The limiting is set up in such a way that the position of the root of p^{q}_x is projected onto the edge closer to that position (w.r.t. the x-coordinate). The same holds true for $s_{yy,ij}$.

In describing the limiting procedure for $s_{xx,ij}$ in more detail, we will suppress the ij index in (24). The partial derivative of $p^{\mathrm{q}}(x, y)$ with respect to x is given by

$$p^{\mathrm{q}}_x(x, y) = 2s_{xx}(x - x_i) + s_{xy}(y - y_j) + s_x. \tag{28}$$

Setting $p^{\mathrm{q}}_x(x, y) = 0$ then corresponds to

$$-s_x - s_{xy}(y - y_j) = 2s_{xx}(x - x_i). \tag{29}$$

The right side of (29) varies within $\left[-2|s_{xx}|\frac{\Delta x}{2}, 2|s_{xx}|\frac{\Delta x}{2}\right]$ in cell (i, j). Therefore, in order for (29) to never hold within cell (i, j) (which is equivalent to saying that

$p_x^{\mathrm{q}}(x, y) \neq 0$ within cell (i, j)), we need

$$|s_x + s_{xy}(y - y_j)| \geq 2|s_{xx}|\frac{\Delta x}{2} \qquad \text{for all } y \in \left[y_j - \frac{\Delta y}{2}, y_j + \frac{\Delta y}{2}\right]. \tag{30}$$

Since $s_x + s_{xy}(y - y_j)$ is a linear function of y, this can't be true if the function values for $y_j - \Delta y/2$ and $y_j + \Delta y/2$ have opposite signs. So if

$$\operatorname{sign}\left(s_x + s_{xy}\frac{\Delta y}{2}\right) \cdot \operatorname{sign}\left(s_x - s_{xy}\frac{\Delta y}{2}\right) < 0, \tag{31}$$

then there exists a $\hat{y} \in [y_j - \Delta y/2, y_j + \Delta y/2]$ such that $p_x^{\mathrm{q}}(x_i, \hat{y}) = 0$. Otherwise the two terms have the same sign and we can take the one with the smaller absolute value as a limiting value. We define

$$\mathrm{cmp} = \min\left(\left|s_x + s_{xy}\frac{\Delta y}{2}\right|, \left|s_x - s_{xy}\frac{\Delta y}{2}\right|\right) \tag{32}$$

and ask for $\mathrm{cmp} \geq \Delta x|s_{xx}|$ to be true. In Step 2, if (31) is satisfied *and* if additionally one of the two conditions analogous to (31) and (32) for s_{yy} is true then we set $s_{xx} = 0$. Otherwise (i.e., (31) not true) if $\mathrm{cmp} < \Delta x|s_{xx}|$ *and* if additionally one of the two analogous conditions for s_{yy} is true we redefine s_{xx} as

$$s_{xx} = \operatorname{sign}(s_{xx})\frac{\mathrm{cmp}}{\Delta x}, \tag{33}$$

that is, we project the position of the root of p_x^{q} from the interior of the cell onto the edge. The analogous limiting applies for s_{yy}. In our second, more restrictive algorithm used in Step 3 we limit s_{xx} and s_{yy} independently of each other. That means if (31) is true, we set $s_{xx} = 0$. Otherwise if $\mathrm{cmp} < \Delta x|s_{xx}|$, we define

$$s_{xx} = \operatorname{sign}(s_{xx})\frac{\mathrm{cmp}}{\Delta x}. \tag{34}$$

The limiting for s_{yy} is analogous.

With this basic approach to limiting the quadratic term, we now provide the details of Step 2 and Step 3 of the limiting procedure. We start with the algorithm used in Step 2 which is designed for smooth areas of the solution.

3.3.1. *Limiting in smooth parts of the solution.* The idea for this algorithm is to try to keep the unlimited polynomial if possible, but still satisfy a maximum principle. To achieve that goal we take the unlimited slopes $s_{x,ij}$, $s_{y,ij}$, and $s_{xy,ij}$ as given by (7a)–(7c). The estimates for the quadratic terms $s_{xx,ij}$ and $s_{yy,ij}$ are only limited if both p_x^{q} and p_y^{q} vanish in the same cell as described above. Since the overall algorithm is designed to satisfy a maximum principle for constant coefficient advection, we need to check whether the minimum and maximum values of the polynomial over the entire cell lie inside the corresponding bounds. The limiting of $s_{xx,ij}$ and $s_{yy,ij}$ ensures that the minimum and maximum lie on the boundary of the cell. Hence, we check whether all corner values and the minimum/maximum

on all four edges lie between the cell values of neighboring cells. Since the edges are aligned with the coordinate axes, the two-dimensional biquadratic polynomial simplifies to a one-dimensional quadratic polynomial there. Consider the upper y-edge, that is, fix $y = \Delta y/2$. If there is an interior extremum at all, that is, if

$$|s_{x,ij} + s_{xy,ij}\Delta y/2| < |s_{xx,ij}\Delta x|, \tag{35}$$

then the extremum has to have the x-coordinate

$$x_{extr,+} = -\frac{s_{x,ij} + s_{xy,ij}\Delta y/2}{2s_{xx,ij}} \tag{36}$$

relative to x_i. The formulae for the other three edges are deduced analogously. This leads to the following algorithm:

Step 2: Limiting of quadratic profile in smooth parts of solution.

Assume given the estimated corner values $\mathrm{LL}_{ij}, \ldots, \mathrm{RH}_{ij}$ and estimated coefficients $s_{xx,ij}$ and $s_{yy,ij}$ using equations (6) as well as (25a), (26a) and (25b), (26b).

(1) Calculate the unlimited slopes $s_{x,ij}$, $s_{y,ij}$, $s_{xy,ij}$ out of $\mathrm{LL}_{ij}, \ldots, \mathrm{RH}_{ij}$ using (7a)–(7c).

(2) Check whether both p_x^{q} and p_y^{q} vanish in the interior of the same cell:
set $\mathrm{test1}_{xx} \leftarrow$ false, $\mathrm{test2}_{xx} \leftarrow$ false, $\mathrm{test1}_{yy} \leftarrow$ false, $\mathrm{test2}_{yy} \leftarrow$ false.

- *if* $\mathrm{sign}(s_{x,ij} + s_{xy,ij}\Delta y/2) \cdot \mathrm{sign}(s_{x,ij} - s_{xy,ij}\Delta y/2) < 0$, set $\mathrm{test1}_{xx} \leftarrow$ true.
- *else if* $\mathrm{cmp} < \Delta x|s_{xx,ij}|$, where cmp is defined by (32), set $\mathrm{test2}_{xx} \leftarrow$ true.

Proceed analogously with $\mathrm{test1}_{yy}$ and $\mathrm{test2}_{yy}$. If $\mathrm{test1}_{xx}$ evaluates to true *and* either one of the tests for yy is true, set $s_{xx,ij} \leftarrow 0$. If $\mathrm{test2}_{xx}$ evaluates to true *and* either one of the tests for yy is true, set $s_{xx,ij} \leftarrow \mathrm{sign}(s_{xx,ij})\,\mathrm{cmp}/\Delta x$. Proceed analogous for $s_{yy,ij}$.

(3) Adjust the constant term $\bar{s}$ using (27b).

(4) Check whether the reconstructed polynomial lies in bounds:

- Evaluate the quadratic polynomial p^{q} given in (24) at the four corners. For each corner, check whether the value of the polynomial lies between the cell averages of the four cells surrounding that corner.
- Calculate the extremal position on each of the four edges (if one exists) using equations (35) and (36) and evaluate p^{q} there. Check whether the value of that point lies between the cell averages of the four cells closest to the position of the extremum.

If all tests are satisfied, keep that polynomial and skip (5) immediately below.

(5) Limit $s_{xx,ij}$ independently of $s_{yy,ij}$ if one of the following conditions holds true:

- *if* $\operatorname{sign}(s_{x,ij} + s_{xy,ij}\Delta y/2) \cdot \operatorname{sign}(s_{x,ij} - s_{xy,ij}\Delta y/2) < 0$, set $s_{xx,ij} \leftarrow 0$.
- *else if* $\text{cmp} < \Delta x|s_{xx,ij}|$ (see (32)), set $s_{xx,ij} \leftarrow \operatorname{sign}(s_{xx,ij})\,\text{cmp}/\Delta x$.

Calculate $s_{yy,ij}$ following the same rules. Check again whether the corners are in bounds. If yes, keep that polynomial. If not, this algorithm was *not* successful (and we'll continue with *Step 3* described below).

Remark. In (5), we only need to check the corners, because by limiting $s_{xx,ij}$ and $s_{yy,ij}$ the way we do there, the minimum and maximum values of p^q now occur at the corners.

Numerical tests suggest that this limiting leads to very good performance in the sense of the overall L^1 error for smooth initial data.

3.3.2. *Limiting ensuring monotonicity.* The algorithm in this subsection is designed for discontinuities. Whereas we took the unlimited slopes $s_{x,ij}$, $s_{y,ij}$, and $s_{xy,ij}$ in the algorithm above, we now apply the limiting procedure from the original bilinear BDS method to the slopes $s_{x,ij}$, $s_{y,ij}$, and $s_{xy,ij}$. In this way we make sure that we preserve the behavior of the bilinear BDS method close to discontinuities. Additionally, we limit the coefficients $s_{xx,ij}$ and $s_{yy,ij}$ using the more restrictive way described above such that the polynomial on cell (i, j) is monotone in x- and y-direction over the whole cell. This leads to the following algorithm:

Step 3: Limiting of quadratic profile close to discontinuities.

Assume the estimated corner values $\text{LL}_{ij}, \ldots, \text{RH}_{ij}$ and estimated coefficients $s_{xx,ij}$ and $s_{yy,ij}$ using equations (6) as well as (25a), (26a) and (25b), (26b) are given.

(1) Use the limiting procedure in the original bilinear BDS to limit the (bi-)linear coefficients, i.e., limit $s_{x,ij}$, $s_{y,ij}$, and $s_{xy,ij}$ following the algorithm given in Section 2.3.

(2) Limit $s_{xx,ij}$ if one of the following conditions holds true:
- *if* $\operatorname{sign}(s_{x,ij} + s_{xy,ij}\Delta y/2) \cdot \operatorname{sign}(s_{x,ij} - s_{xy,ij}\Delta y/2) < 0$, set $s_{xx,ij} \leftarrow 0$.
- *else if* $\text{cmp} < \Delta x|s_{xx,ij}|$ (see (32)), set $s_{xx,ij} \leftarrow \operatorname{sign}(s_{xx,ij})\,\text{cmp}/\Delta x$.

Calculate $s_{yy,ij}$ analogously.

(3) Adjust the constant term $\bar{s}$ using (27b) .

(4) Test whether the corner values of the fully reconstructed quadratic polynomial exceed the cell averages of the neighboring cells. If this is the case, set $s_{xx,ij} \leftarrow 0$ and $s_{yy,ij} \leftarrow 0$, and set the constant term equal to s_{ij}.

The last step ensures that the corner values of the quadratic polynomial lie between the minimum/maximum values of the neighboring cells, that is, it ensures that the algorithm satisfies a maximum principle for constant coefficient linear

advection. Without the quadratic terms the four corners of the cell (i, j) have the values

$$\pm\frac{\Delta x\Delta y}{4}s_{xy,ij} \pm \frac{\Delta x}{2}s_{x,ij} \pm \frac{\Delta y}{2}s_{y,ij} + s_{ij}, \tag{37}$$

which are guaranteed to lie between the minimum/maximum bounds due to the bilinear limiting strategy. Then we add the quadratic terms and change the constant term from s_{ij} to $\bar{s} = s_{ij} - \frac{1}{12}s_{xx,ij}(\Delta x)^2 - \frac{1}{12}s_{yy,ij}(\Delta y)^2$. Consequently, the corners have the values

$$\begin{aligned} &s_{xx,ij}\frac{(\Delta x)^2}{4} + s_{yy,ij}\frac{(\Delta y)^2}{4} \pm \frac{\Delta x\Delta y}{4}s_{xy,ij} \pm \frac{\Delta x}{2}s_{x,ij} \pm \frac{\Delta y}{2}s_{y,ij} + \bar{s} \\ &\quad = s_{xx,ij}\frac{(\Delta x)^2}{6} + s_{yy,ij}\frac{(\Delta y)^2}{6} \pm \frac{\Delta x\Delta y}{4}s_{xy,ij} \pm \frac{\Delta x}{2}s_{x,ij} \pm \frac{\Delta y}{2}s_{y,ij} + s_{ij}. \end{aligned} \tag{38}$$

So compared to the bilinear polynomial p^{bl} the values of the quadratic polynomial p^{q} differ at every corner by $\frac{1}{6}s_{xx,ij}(\Delta x)^2 + \frac{1}{6}s_{yy,ij}(\Delta y)^2$ (which is a constant for every cell).

For smooth initial data, this constant often even helps to keep the corner values in bounds. In our numerical tests, violations were usually seen only for discontinuous initial data. Therefore, we chose the straightforward way to fix this problem just described: at the end of the limiting routine, we check whether the values of the quadratic polynomial at the corners lie inside bounds. If that's the case, we are done. Otherwise, we set $s_{xx,ij} = 0$ and $s_{yy,ij} = 0$, that is, we go back to the bilinear polynomial.

This leads to a method that obeys the maximum principle for constant coefficient linear advection (up to numerical roundoff error):

- The minimum and maximum value of the quadratic polynomial p^{q}_{ij} on cell (i, j) are limited by the values on the boundary in Step 2 of the limiting process and by the values at the corners in Step 3.
- The values on the boundary and the corner values of the quadratic polynomial don't exceed the minimum/maximum of the neighboring cell averages.

3.4. ***Construction of edge states.*** The formalism based on integrating over the characteristic domain of dependence remains the same as in the bilinear scheme. The only changes that we need to make are to substitute higher order quadrature formulae to evaluate the integrals over the quadratic terms exactly. Let us first consider the integral

$$\iint_{ABDE} s(x, y)\,dx\,dy, \tag{39}$$

for the case $u_{i+1/2,j} > 0$ appearing in the calculation of the flux $s^L_{i+1/2,j}$ in (13). For the bilinear, linear, and constant terms of the polynomial, we can use the midpoint

formula as before. The integrals over the quadratic terms can be calculated explicitly:

$$\begin{aligned}
&\iint_{ABDE} s_{xx,ij}(x-x_i)^2\,dx\,dy \\
&\quad = s_{xx,ij}\Delta y \int_{x_{i+1/2}-u_{i+1/2,j}\Delta t}^{x_{i+1/2}} (x-x_i)^2\,dx \\
&\quad = s_{xx,ij}\Delta y\left[\tfrac{1}{4}(\Delta x)^2 u_{i+1/2,j}\Delta t - \tfrac{1}{2}\Delta x(u_{i+1/2,j}\Delta t)^2 + \tfrac{1}{3}(u_{i+1/2,j}\Delta t)^3\right], \qquad (40)
\end{aligned}$$

and

$$\iint_{ABDE} s_{yy,ij}(y-y_j)^2\,dx\,dy = \tfrac{1}{12}s_{yy,ij}u_{i+1/2,j}\Delta t(\Delta y)^3. \qquad (41)$$

That means that $s_{M,F}$ given by (18) is replaced by the following $s^Q_{M,F}$ in formula (17):

$$\begin{aligned}
s^Q_{M,F} &= \bar{s} + \frac{\Delta x - u_{i+1/2,j}\Delta t}{2} s_{x,ij} \\
&\quad + s_{xx,ij}\left[\tfrac{1}{4}(\Delta x)^2 - \tfrac{1}{2}\Delta x u_{i+1/2,j}\Delta t + \tfrac{1}{3}(u_{i+1/2,j}\Delta t)^2\right] + \tfrac{1}{12}s_{yy,ij}(\Delta y)^2. \qquad (42)
\end{aligned}$$

Additionally, we need to adjust the calculations of Γ^+ and Γ^- appropriately. This corresponds to changing the evaluations of the contributions coming from the triangles DEF and ABC. We evaluate the linear part of the polynomial with the midpoint rule. For the quadratic and bilinear terms we use the same rule as used for the bilinear term in the original BDS: we evaluate the terms at the midpoints of all three edges and divide the corresponding sum by three. In this way, we are evaluating all two-dimensional integrals exactly. The same changes hold true for all the other cases considered in Section 2.4.2 (i.e., the calculation of $s^R_{i+1/2,j}$, $s^L_{i,j+1/2}$, and $s^R_{i,j+1/2}$). We note that for constant coefficient advection, analogous to the original BDS algorithm, the quadratic BDS algorithm is equivalent to fitting a limited quadratic profile to the solution at time t^n, advecting that profile exactly and averaging it back onto the grid, which guarantees that the solution satisfies a maximum principle.

4. Numerical results

In this section we present a series of numerical tests using our new quadratic method (BDS_Q). In Section 4.1, we advect smooth and discontinuous initial data using a constant velocity field. We explore the effects of angle dependence of the velocity field on the accuracy and overshoot of our method. In Section 4.2, we explore the effects of advecting smooth and discontinuous data in a velocity field that varies in space. In Section 4.3, we integrate the algorithm into a variable density projection method [2], and thus the velocity varies in both space and time. In this example, we examine the behavior of the density in addition to a passively advected scalar.

In each of these sections, we compare our results to the original bilinear method (BDS_BL, [3]) and two PPM methods. The first, which we call PPM1, is the PPM algorithm [9] that has been in use for over 25 years. The second, which we call PPM2, is based on a recent effort [8] to preserve accuracy at smooth extrema. The algorithm as described in [8] suffers from sensitivity to roundoff error; we incorporate the correction to that as described in [16]. Finally, in Section 4.4 we discuss the performance of BDS_Q on some additional test problems discussed in the literature and compare results to other schemes.

In Sections 4.1 and 4.2 we solve the equation

$$s_t + (us)_x + (vs)_y = 0 \tag{43}$$

for both smooth and discontinuous initial conditions, where (u, v) is a specified velocity field. For the smooth case we define

$$s(x, y, t = 0) = e^{-60r^2}, \tag{44}$$

where $r^2 = (x-1)^2 + (y-1)^2$ on the domain $(0, 2)^2$. The discontinuous initial data is given by a round tophat of the form

$$s(x, y, t = 0) = \begin{cases} 1 & \text{if } r < 0.2, \\ 0 & \text{otherwise,} \end{cases} \tag{45}$$

where $r = \sqrt{(x-0.5)^2 + (y-0.5)^2}$ on the domain $(0, 1)^2$. Given the analytic profile of s at $t = 0$, we discretize the smooth initial data using Gaussian quadrature rules, which give us a fourth-order estimate of the cell average. As a result, while the maximum of the analytical function in (44) is 1, the numerical maximum will be slightly lower for each resolution. To discretize the discontinuous round tophat we approximate the integral of the function over the cell by dividing the cell into 16 subcells, evaluating the analytic function at the center of each subcell, then averaging the 16 values to define the average over the original cell.

In Section 4.3, we integrate the algorithm into a variable density projection method [2]. First we consider the advection of a passive tracer in a constant density incompressible flow. Here although the velocity varies in space and time the tracer does not couple back to the fluid. In the second example we consider the advection of density in a variable density flow. For this case, the density couples back into the evolution of the velocity field.

For all of the tests we set the time step based on the CFL condition

$$\Delta t = \sigma_{\mathrm{CFL}} \min_{ij} \left(\frac{\Delta x}{|u_{ij}|}, \frac{\Delta y}{|v_{ij}|} \right),$$

with a CFL number, $\sigma_{\mathrm{CFL}} = 0.9$, and impose periodic boundary conditions on all faces.

Before presenting our results we would like to comment on the computational efficiency of the BDS_Q algorithm as compared to BDS_BL and PPM2. In our testing, BDS_BL is a factor of 1.79 times more expensive than PPM2, and BDS_Q is a factor of 2.47 times more expensive than PPM2. However, in our intended applications the computational cost is dominated by elliptic solvers, reaction networks, and/or calls to the equation of state, so the overall cost of advection is minor, even with the more expensive BDS_Q algorithm.

4.1. *Constant velocity advection.*

4.1.1. *Smooth initial data.* In this section we consider the evolution of s with initial data given by (44). The motivation for extending BDS_BL to BDS_Q by adding the quadratic terms was to increase the accuracy for problems with smooth initial data, while maintaining the lack of under- and overshoot that we see with BDS_BL for discontinuous problems. We report the L^1 norm of error relative to the exact solution for each method at three different resolutions for both the limited and unlimited forms of each algorithm in Table 1, where $(u, v) = (1, 0)$, and Table 2, where $(u, v) = (1, 0.2)$. As expected we see third-order convergence for unlimited BDS_Q as opposed to second-order convergence for BDS_BL, i.e., the error decreases by a factor of 8 rather than 4 for each factor 2 decrease in mesh spacing. With limiting, the ratio of errors with BDS_Q decreases slightly in the off-axis test, but we observe that BDS_Q is the only method to maintain such high convergence rates in this test. All other methods demonstrate second-order convergence for the off-axis test, even though PPM2 shows a high rate of convergence for the axis-aligned case.

Looking at the magnitudes of errors as opposed to the ratios, we see that for flow aligned with the x-axis, the errors are lowest using PPM2, which for this particular problem is equivalent to PPM without limiters. However, this relative advantage disappears when the flow does not align with a coordinate axis. At the highest

Method	100^2 Error	Ratio	200^2 Error	Ratio	400^2 Error
BDS_Q	1.89e-04	8.0	2.36e-05	8.3	2.83e-06
BDS_BL	6.18e-04	4.1	1.49e-04	4.1	3.62e-05
PPM1	5.86e-04	5.0	1.18e-04	5.1	2.30e-05
PPM2	4.48e-05	14.	3.16e-06	12.	2.62e-07
BDS_Q, no limiting	5.80e-05	8.7	6.69e-06	8.2	8.18e-07
BDS_BL, no limiting	5.45e-04	4.0	1.37e-04	4.0	3.45e-05
PPM, no limiting	4.48e-05	14.	3.16e-06	12.	2.62e-07

Table 1. Error in the L^1 norm at $t = 2$ for smooth initial data with $(u, v) = (1, 0)$.

Method	100^2 Error	Ratio	200^2 Error	Ratio	400^2 Error
BDS_Q	1.33e-03	7.2	1.85e-04	7.4	2.51e-05
BDS_BL	4.71e-03	4.1	1.15e-03	4.0	2.89e-04
PPM1	7.88e-03	3.8	2.07e-03	3.9	5.29e-04
PPM2	7.86e-03	4.0	1.98e-03	4.0	4.97e-04
BDS_Q, no limiting	7.49e-04	8.4	8.95e-05	8.1	1.10e-05
BDS_BL, no limiting	4.53e-03	4.0	1.13e-03	4.0	2.82e-04
PPM, no limiting	7.88e-03	4.0	1.98e-03	4.0	4.97e-04

Table 2. Error in the L^1 norm at $t = 10$ for smooth initial data with $(u, v) = (1, 0.2)$.

resolution of the off-axis test, the error in the solution is more than an order of magnitude smaller with BDS_Q than with any of the other methods.

Another metric for the performance of a method for scalar advection is the degree to which it preserves the maximum of a smooth peak, and the degree to which the final solution under- or overshoots the minimum and maximum, respectively, of the original solution. Analytically, scalar advection with a divergence-free velocity field should preserve the maximum and minimum of the original solution.

In Table 3 we show the peak value of the solution for the axis-aligned flow at final time; in this case, none of the methods exhibit undershoot. In Table 4 we show the peak value at the final times and the largest value of undershoot for the off-axis case. Disappointingly, the peak for BDS_Q is slightly lower than the peaks for BDS_BL and for eitherPPM method; this is an issue we hope to address in future work. It is clear that the reduction results from the limiting in BDS_Q; without limiting the maximum for BDS_Q is comparable to that of unlimited PPM. We also verify

Method	100^2 max	200^2 max	400^2 max
Analytic	0.98417	0.99601	0.99900
BDS_Q	0.95344	0.98454	0.99493
BDS_BL	0.95432	0.98493	0.99506
PPM1	0.96476	0.98873	0.99642
PPM2	0.98405	0.99598	0.99900
BDS_Q, no limiting	0.98281	0.99585	0.99897
BDS_BL, no limiting	0.98279	0.99588	0.99899
PPM, no limiting	0.98405	0.99598	0.99900

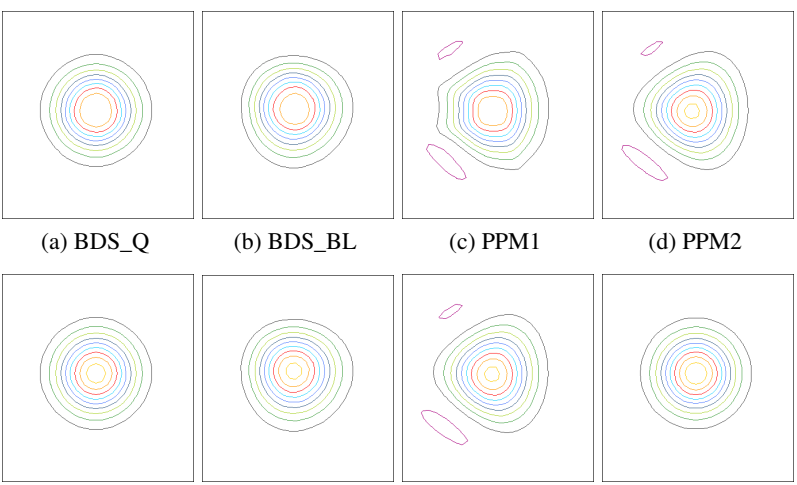
Table 3. Peak at $t = 2$ for smooth initial data with $(u, v) = (1, 0)$. None of the methods exhibit undershoot for this problem.

Method	100^2 max	100^2 min	200^2 max	200^2 min	400^2 max	400^2 min
Analytic	0.98417	0.00000	0.99601	0.00000	0.99900	0.00000
BDS_Q	0.87065	0.00000	0.95442	0.00000	0.98383	0.00000
BDS_BL	0.86967	0.00000	0.95790	0.00000	0.98598	0.00000
PPM1	0.88597	–0.02133	0.96678	–0.00003	0.98907	0.00000
PPM2	0.94632	–0.01701	0.99261	0.00000	0.99877	0.00000
BDS_Q, no limiting	0.96600	0.00000	0.99377	0.00000	0.99873	0.00000
BDS_BL, no limiting	0.99512	–0.00316	0.99321	0.00000	0.99875	0.00000
PPM, no limiting	0.94632	–0.01783	0.99261	0.00000	0.99877	0.00000

Table 4. Maxima and minima at $t = 10$ for smooth initial data with $(u, v) = (1, 0.2)$.

in this test that both BDS_BL and BDS_Q show no undershoot for the off-axis problem, unlike both PPM approaches, which undershoot on the coarser grids.

A final metric we consider is the distortion of the original shape at the final time. Analytically, the initial data should stay undistorted throughout the entire evolution. Figure 3 shows contours of the solution at $t = 10$ for evolution with $(u, v) = (1, 0.2)$. BDS_Q clearly preserves the round shape most accurately, with some distortion evident for BDS_BL. The solutions for PPM1 and PPM2 show



(a) BDS_Q (b) BDS_BL (c) PPM1 (d) PPM2

(e) unlim. BDS_Q (f) unlim. BDS_BL (g) unlim. PPM (h) analytic

Figure 3. Final solution for smooth initial data, $(u, v) = (1, 0.2)$, and 100^2 resolution. There are nine contour lines evenly spaced from 0.1 to 0.9; in addition, the violet contour encloses the region where $s < -0.01$.

significant distortion. We can also see in this figure the flattening of the peak for the limited versions of BDS_BL and BDS_Q, as well as the undershoot for the PPM methods.

4.1.2. *Discontinuous initial data.* In this section we consider the evolution of discontinuous initial data given by (45), again considering two different velocity fields, $(u, v) = (1, 0)$ and $(u, v) = (1, 0.2)$. In Tables 5 and 6 we show the L^1 norm of error for each method at three different resolutions for both the limited and unlimited forms of each algorithm. In this table we no longer show the ratios of error because we do not expect to approach the asymptotic convergence rates with discontinuous initial data. We make three observations from these tables: first, the errors for the different methods are much closer to each other for discontinuous initial data than for smooth initial data; second, unlike for smooth initial data, for both velocity fields and all the methods, limiting reduces the L^1 error of the solution; third, for the axis-aligned flow field the errors for the PPM methods are slightly

Method	100^2 Error	200^2 Error	400^2 Error
BDS_Q	5.40e-03	3.30e-03	1.99e-03
BDS_BL	5.69e-03	3.56e-03	2.23e-03
PPM1	3.67e-03	2.18e-03	1.29e-03
PPM2	3.83e-03	2.32e-03	1.40e-03
BDS_Q, no limiting	6.97e-03	4.22e-03	2.52e-03
BDS_BL, no limiting	7.65e-03	5.01e-03	3.33e-03
PPM, no limiting	7.71e-03	4.68e-03	2.85e-03

Table 5. Error in the L^1 norm at $t = 1$ for discontinuous initial data with $(u, v) = (1, 0)$.

Method	100^2 Error	200^2 Error	400^2 Error
BDS_Q	1.23e-02	7.30e-03	4.34e-03
BDS_BL	1.45e-02	9.13e-03	5.82e-03
PPM1	2.31e-02	1.53e-02	1.02e-02
PPM2	2.33e-02	1.57e-02	1.06e-02
BDS_Q, no limiting	1.49e-02	8.79e-03	5.17e-03
BDS_BL, no limiting	2.22e-02	1.49e-02	9.96e-03
PPM, no limiting	2.86e-02	1.91e-02	1.27e-02

Table 6. Error in the L^1 norm at $t = 5$ for discontinuous initial data with $(u, v) = (1, 0.2)$.

Method	100^2 max	100^2 min	200^2 max	200^2 min	400^2 max	400^2 min
Analytic	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
BDS_Q	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
BDS_BL	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
PPM1	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
PPM2	1.00000	0.00000	1.00443	0.00000	1.00080	0.00000
BDS_Q, no limiting	1.10593	–0.10032	1.09505	–0.09264	1.09159	–0.08632
BDS_BL, no limiting	1.10527	–0.10060	1.13222	–0.13264	1.15713	–0.15713
PPM, no limiting	1.17344	–0.17344	1.14547	–0.14547	1.16618	–0.16618

Table 7. Maxima and minima at $t = 1$ for discontinuous initial data with $(u, v) = (1, 0)$.

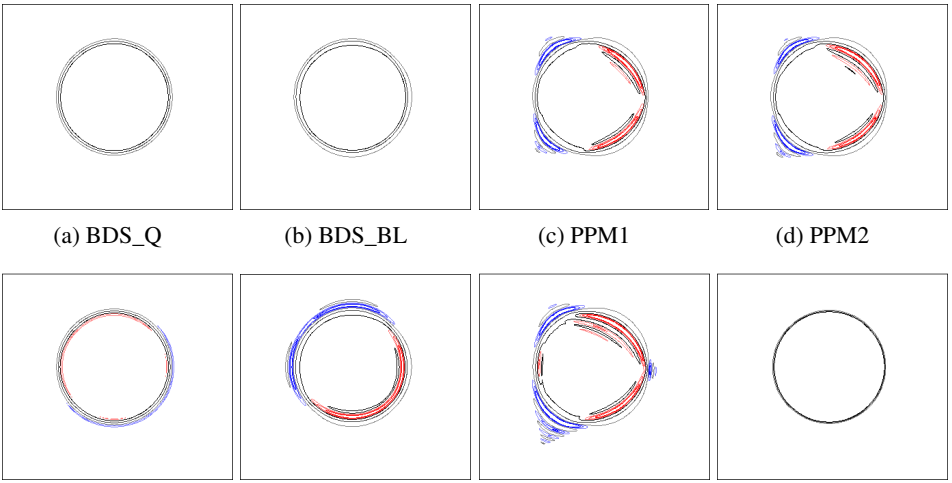
Method	100^2 max	100^2 min	200^2 max	200^2 min	400^2 max	400^2 min
Analytic	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
BDS_Q	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
BDS_BL	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
PPM1	1.18611	–0.28803	1.21003	–0.32074	1.22745	–0.30010
PPM2	1.19335	–0.29089	1.21697	–0.29623	1.23178	–0.28074
BDS_Q, no limiting	1.10668	-0.06558	1.09510	-0.07061	1.09311	-0.07344
BDS_BL, no limiting	1.17508	-0.16271	1.19262	-0.18948	1.21179	-0.20986
PPM, no limiting	1.21288	-0.33820	1.23017	-0.32474	1.24244	-0.32613

Table 8. Maxima and minima at $t = 5$ for discontinuous initial data with $(u, v) = (1, 0.2)$.

lower; for the diagonal flow the BDS methods have slightly lower error.

In Tables 7 and 8 we show the maxima and minima of the solution at the final time. For the axis-aligned flow we see that, of the methods with limiters, only PPM2 introduces new maxima to the solution. However, for the off-axis flow the solutions for both PPM1 and PPM2 overshoot by more than 20% at the two higher resolutions, while BDS_BL and BDS_Q retain the initial maximum value of 1. We see similar results with the minima. We also note that while the L^1 error decreases with mesh spacing, the magnitude of the over- and undershoot does not.

Again we look at the distortion of the final solution for the case with the off-axis velocity field. In Figure 4 we see contours of the solution with contours in the regions of undershoot and overshoot given in color. We notice a slight spreading



(a) BDS_Q (b) BDS_BL (c) PPM1 (d) PPM2

(e) unlim. BDS_Q (f) unlim. BDS_BL (g) unlim. PPM (h) analytic

Figure 4. Final solution for discontinuous initial data, $(u, v) = (1, 0.2)$, and 400^2 resolution. There are three black contours from 0.05 to 0.95, three red contours from 1.05 to 1.15 marking the overshoot and three blue contours going from -0.15 to -0.05 marking the undershoot.

of the contours for all of the methods but no over- or undershoot for the limited BDS_Q and BDS_BL methods. All the PPM solutions show severe distortion from the over- and undershoot.

Finally, we examine the question of how the over- and undershoot vary with the angle of the velocity relative to the x-axis. Table 9 shows the maximum and minimum of each solution after 500 time steps for each of the limited methods using the discontinuous initial data at 100^2 resolution. We see that the $(u, v) = (1, 0.2)$ case is representative of off-axis velocities, and in fact the over- and undershoot seem to peak for both PPM1 and PPM2 when the velocity field is approximately 30° off the x-axis.

4.2. ***Variable velocity advection.*** Here we consider the velocity field given by $(u, v) = [1.0, \sin(\pi x)]$ in the domain $(0, 2)^2$. We advect s until $t = 10$ for both smooth and discontinuous initial data, with the discontinuous data now centered at $(1, 1)$ instead of $(0.5, 0.5)$ as was done earlier. The goal of this test is to examine if the conclusions of the previous section hold when the velocity field is no longer spatially constant.

4.2.1. *Smooth initial data.* Results for smooth initial data are shown in Tables 10 and 11. In Table 10 we report the L^1 error at $t = 10$, in Table 11 we show the

Method	(u, v) $(1,0)$	$(1,0.2)$	$(1,0.4)$	$(1,0.5)$	$(1,0.6)$	$(1,0.8)$	$(1,1)$
				Maximum			
Analytic	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
BDS_Q	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
BDS_BL	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
PPM1	1.00000	1.18762	1.19635	1.20484	1.20307	1.17445	1.16009
PPM2	1.00324	1.18856	1.21372	1.22929	1.22770	1.17534	1.15975
				Minimum			
Analytic	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
BDS_Q	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
BDS_BL	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
PPM1	0.00000	–0.28937	–0.32127	–0.32719	–0.32405	–0.32244	–0.28363
PPM2	0.00000	–0.27999	–0.32064	–0.32297	–0.32934	–0.31379	–0.26969

Table 9. Maxima and minima after 500 time steps for each limited method using discontinuous initial data at 100^2 resolution.

minimum and maximum value at $t = 10$ for all methods considered. Analogously to the conclusions in Section 4.1.1 for off-axis flow we observe, for smooth initial data:

- third-order accuracy for BDS_Q and second-order accuracy for other methods,
- flattening of the peak for BDS_Q and BDS_BL relative to PPM,
- BDS_Q outperforms BDS_BL by every metric except for a slightly lower peak at 400^2,
- no under- or overshoot for the BDS algorithms, some undershoot for the PPM algorithms.

Method	100^2 Error	Ratio	200^2 Error	Ratio	400^2 Error
BDS_Q	2.61e-03	8.0	3.25e-04	7.3	4.45e-05
BDS_BL	4.96e-03	4.4	1.13e-03	4.2	2.70e-04
PPM1	5.29e-03	4.1	1.28e-03	4.2	3.07e-04
PPM2	4.46e-03	4.0	1.11e-03	4.0	2.79e-04
BDS_Q, no limiting	1.88e-03	8.1	2.32e-04	8.2	2.84e-05
BDS_BL, no limiting	4.68e-03	4.4	1.07e-03	4.1	2.58e-04
PPM, no limiting	4.43e-03	4.0	1.12e-03	4.0	2.79e-04

Table 10. Error in the L^1 norm at $t = 10$ for smooth initial data with $(u, v) = [1.0, \sin(\pi x)]$.

Method	100^2 max	100^2 min	200^2 max	200^2 min	400^2 max	400^2 min
Analytic	0.98417	0.00000	0.99601	0.00000	0.99900	0.00000
BDS_Q	0.83400	0.00000	0.94420	0.00000	0.98117	0.00000
BDS_BL	0.82682	0.00000	0.94381	0.00000	0.98189	0.00000
PPM1	0.88298	–0.01126	0.97013	–0.00014	0.99172	0.00000
PPM2	0.93607	–0.01069	0.98961	–0.00013	0.99824	0.00000
BDS_Q, no limiting	0.94487	–0.00163	0.99076	0.00000	0.99835	0.00000
BDS_BL, no limiting	0.92712	–0.01581	0.98848	–0.00027	0.99810	0.00000
PPM, no limiting	0.93715	–0.01224	0.98961	-0.00012	0.99824	0.00000

Table 11. Maxima and minima at $t = 10$ for smooth initial data with $(u, v) = [1.0, \sin(\pi x)]$.

Note that both BDS implementations, with limiting, do not undershoot at any time with respect to the scale used in our tables. In fact, neither BDS_Q nor BDS_BL show any under/overshoot larger than 10^{-9} at any of our reported resolutions. (This error depends directly on the choice of ϵ in the limiting algorithm described in Section 2.3.) By contrast, the PPM1/PPM2 algorithms first exhibit undershoot with magnitude greater than 10^{-5} at $t = 0.882$ (PPM1, 100^2), $t = 0.918$ (PPM2, 100^2), $t = 5.193$ (PPM1, 200^2), and $t = 5.148$ (PPM2, 200^2). We observe no undershoot on our finest grid for the PPM methods with respect to the scale in Table 11.

4.2.2. *Discontinuous initial data.* Repeating the same tests with discontinuous initial data leads to the results shown in Tables 12, 13, and 14. In these tables, we report the L^1 error at $t = 10$ as well as the minimum and maximum value both after only one time step and at $t = 10$. Analogous to the conclusions in Section 4.1.2 for off-axis flow, we observe for discontinuous initial data:

Method	100^2 Error	200^2 Error	400^2 Error
BDS_Q	2.98e-02	1.77e-02	1.05e-02
BDS_BL	3.34e-02	2.04e-02	1.25e-02
PPM1	3.59e-02	2.30e-02	1.49e-02
PPM2	3.58e-02	2.27e-02	1.47e-02
BDS_Q, no limiting	3.54e-02	2.13e-02	1.26e-02
BDS_BL, no limiting	4.41e-02	2.87e-02	1.86e-02
PPM, no limiting	4.00e-02	2.50e-02	1.58e-02

Table 12. Error in the L^1 norm at $t = 10$ for discontinuous initial data with $(u, v) = [1.0, \sin(\pi x)]$.

Method	100^2 max	100^2 min	200^2 max	200^2 min	400^2 max	400^2 min
Analytic	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
BDS_Q	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
BDS_BL	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
PPM1	1.04271	–0.04141	1.04393	–0.04301	1.04444	–0.04062
PPM2	1.03456	–0.03403	1.03931	–0.03666	1.04231	–0.04037
BDS_Q, no limiting	1.05006	–0.05824	1.05962	–0.05489	1.04745	–0.05812
BDS_BL, no limiting	1.04201	–0.04645	1.04771	–0.04659	1.04002	–0.04863
PPM, no limiting	1.05559	–0.05606	1.06203	–0.06010	1.05846	–0.06410

Table 13. Maxima and minima after one time step for discontinuous initial data with $(u, v) = [1.0, \sin(\pi x)]$.

Method	100^2 max	100^2 min	200^2 max	200^2 min	400^2 max	400^2 min
Analytic	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
BDS_Q	0.99544	0.00000	0.99998	0.00000	1.00000	0.00000
BDS_BL	0.99665	0.00000	0.99999	0.00000	1.00000	0.00000
PPM1	1.10007	–0.09099	1.10253	–0.11853	1.12603	–0.14435
PPM2	1.09626	–0.09067	1.09608	–0.12249	1.12464	–0.15188
BDS_Q, no limiting	1.13956	–0.06600	1.12081	–0.06652	1.11089	–0.06679
BDS_BL, no limiting	1.22656	–0.11437	1.22889	–0.14710	1.22822	–0.17816
PPM, no limiting	1.15451	–0.12074	1.12697	–0.14695	1.13860	–0.17400

Table 14. Maxima and minima at $t = 10$ for discontinuous initial data with $(u, v) = [1.0, \sin(\pi x)]$.

- a lower error for BDS_Q than for any of the other algorithms,
- no undershoot or overshoot for the BDS algorithms at any time,
- an overshoot for the PPM algorithms of $> 3\%$ after one time step, and $> 9\%$ at $t = 10$.

We conclude that the behavior of the BDS_Q algorithm with a spatially varying velocity field is consistent with that observed with a constant velocity field.

4.3. ***Shear layer example.*** Here we consider a temporally and spatially evolving velocity field that is determined by solving the incompressible Euler equations using the approximate projection algorithm described in [2]. We note that although the algorithm uses an approximate projection to define the cell-centered velocity

at the end of each time step, the edge-based velocities that are used for advection are discretely divergence-free (to the tolerance of the iterative solver that is used to enforce the constraint). We initialize the problem with parallel shear layers perturbed slightly; the initial velocity field is given by

$$(u, v)|_{t=0} = \left(\tanh\left[60(0.07 - |y - 0.5|)\right], 0.5\sin(2\pi x)\right). \tag{46}$$

In the first test we initialize the density to be constant everywhere; because the flow is divergence-free the density will remain constant. We solve (43) for a passively advected tracer, s, that is initialized with discontinuous data described in (45). In Tables 15 and 16 we see the over- and undershoot associated with each method after one time step and at $t = 0.23$, respectively. Both PPM1 and PPM2 over/undershoot by 3–4% after one time step, and by 6–25% at the final time. The BDS methods do not over/undershoot at any time during the simulation. Figure 5 shows the solution for both BDS_Q and PPM2; the locations of the overshoot and undershoot are evident in the blue and red coloring of the figure.

In the second test we consider a variable density case in which the initial density, ρ, is given by

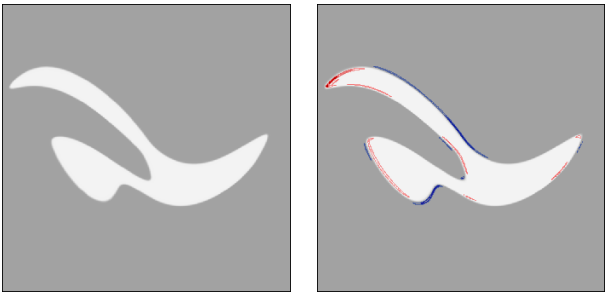
$$\rho(x, y, t = 0) = 1.5 + 0.5\tanh\left[600(0.02 - |y - 0.5|)\right] \tag{47}$$

	128^2		256^2		512^2	
Method	max	min	max	min	max	min
Analytic	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
BDS_Q	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
BDS_BL	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
PPM1	1.04177	–0.03766	1.04200	–0.03845	1.04166	–0.04293
PPM2	1.03531	–0.03570	1.03942	–0.03798	1.03976	–0.04026

Table 15. Maxima and minima of s after one time step for the constant density shear layer problem with discontinuous initial data.

	128^2		256^2		512^2	
Method	max	min	max	min	max	min
Analytic	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
BDS_Q	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
BDS_BL	1.00000	0.00000	1.00000	0.00000	1.00000	0.00000
PPM1	1.13413	–0.05983	1.19031	–0.07449	1.20885	–0.09356
PPM2	1.17185	–0.06706	1.23665	–0.08806	1.25714	–0.10572

Table 16. Maxima and minima of s at $t = 0.23$ for the constant density shear layer problem with discontinuous initial data.



(a) BDS_Q (b) PPM2

Figure 5. Solution at time $t = 0.23$ for the constant density shear layer problem with discontinuous initial data and resolution 512^2. Undershoot between -0.1 and -0.01 is marked in blue. Overshoot between 1.01 and 1.15 is marked in red. More intense colors are used for areas of higher over- and undershoot.

and initial velocity field given by

$$(u, v)|_{t=0} = \{\tanh\left[600(0.07 - |y - 0.5|)\right], 3.5\sin(2\pi x)\}. \tag{48}$$

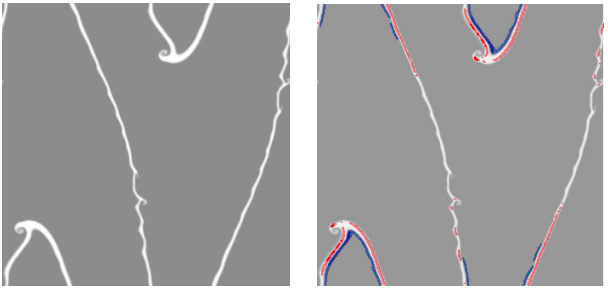
For this case, the mass conservation equation,

$$\rho_t + (u\rho)_x + (v\rho)_y = 0, \tag{49}$$

is part of the evolution. We note that this test differs from all the others in that the density couples back into the calculation of the velocities at each time step. In Table 17 we see the over- and undershoot associated with each method at $t = 0.19$. As in the previous tests both PPM1 and PPM2 create noticeable under- and overshoot while neither of the BDS methods do. In fact, the BDS methods do not over- or undershoot at any time during the simulation, whereas the PPM methods suffer at early times. For example, for the 512^2 simulations, we first observe an overshoot $> 10^{-5}$ at $t = 0.017$ (PPM1) and $t = 0.003$ (PPM2). Figure 6 shows the solution for

	128^2		256^2		512^2	
Method	max	min	max	min	max	min
Analytic	2.00000	1.00000	2.00000	1.00000	2.00000	1.00000
BDS_Q	1.92762	1.00000	1.99013	1.00000	1.99967	1.00000
BDS_BL	1.92277	1.00000	1.99376	1.00000	1.99981	1.00000
PPM1	2.02653	0.89027	2.10101	0.86719	2.13961	0.87522
PPM2	2.09583	0.80600	2.09924	0.87192	2.16143	0.84435

Table 17. Maxima and minima of ρ at $t = 0.19$ for the variable density shear layer problem.



(a) BDS_Q (b) PPM2

Figure 6. Solution at time $t = 0.23$ for the variable density shear layer problem at resolution 512^2. Undershoot between 0.85 and 0.99 is marked in blue. Overshoot between 2.01 and 2.15 is marked in red. More intense colors are used for areas of higher over- and undershoot.

both BDS_Q and PPM2; the locations of the overshoot and undershoot are again evident in the blue and red coloring of the figure.

4.4. ***Additional comparisons.*** In this section, we examine the results of BDS_Q on some test problems for linear advection that have been discussed in the literature. It is impossible to provide a comprehensive comparison of BDS to the full gamut of possible advection schemes; here we will consider the third-order central WENO scheme of Levy, Puppo, and Russo (LPR) [15] and the ADER schemes discussed by Toro and Titarov [24]. LPR is a multidimensional staggered grid scheme that uses a limited quadratic reconstruction algorithm. The ADER schemes use a similar reconstuction but use a Cauchy–Kowaleski procedure and Taylor series expansion to approximate the flux at Gaussian quadrature nodes on the space-time edges of the control volume.

First we consider the linear advection test problem, $s_t + s_x + s_y = 0$, examined in [15]. The initial conditions are given by

$$s(x, y, t = 0) = \sin^2(\pi x) \sin^2(\pi y)$$

on the domain $(0, 1)^2$ and errors for LPR are computed at $t = 1$ based on a CFL of 0.425. The BDS_Q scheme has a less restrictive stability limit so we use a CFL of 0.9, which represents approximately the same fraction of the maximum stable time step. We first compare the unlimited version of BDS_Q with the unlimited version of LPR. Tests over a range of resolutions from 10^2 to 160^2 show that both methods converge at third order accuracy. Furthermore, at each resolution, the accuracy of BDS_Q is a factor of at least 20 better than LPR in both the L^1 and L^∞ norms. When we repeat the experiment with the limited schemes, BDS_Q is approximately

a factor of 10 more accurate on the 10^2 grid but the accuracy of LPR improves relative to BDS_Q to the point that on the 160^2 grid, BDS_Q is less than a factor of two more accurate in the L^1 norm and a factor of 2.03 worse in the L^∞ norm.

We conjecture that this relative loss of accuracy reflects the more restrictive limiting of BDS_Q needed to enforce a maximum principle. To test this hypothesis, we have implemented the WENO reconstruction in [15] and tested it in conjunction with the BDS_Q flux computation. The resulting hybrid scheme for the smooth problems has errors that are an order of magnitude less than LPR in the L^1 and L^∞ norms on a 160^2 grid. However, on a discontinuous advection example, this less restrictive reconstruction leads to approximately 2% overshoot and undershoot. These tests confirm the conjecture and indicate that if we do not need to enforce a maximum principle, a less restrictive reconstruction can be used that will result in reduced errors for smooth problems.

We next provide a comparison to the higher-order ADER schemes of Toro and Titarev [24]. They consider a variable coefficient linear advection example of the form (1), referred to as the frontogenesis problem. The initial conditions are given by

$$s(x, y, t = 0) = \tanh \frac{y}{\delta}, \quad \delta = 1$$

on the domain $(-5, 5)^2$, and the velocity field is constant in time, given by

$$(u, v) = \omega(r)(-y, x),$$

with $\omega(r) = (1/r)U_T(r)$, $U_T(r) = 2.5980762\ \mathrm{sech}^2 r \tanh r$, and $r^2 = x^2 + y^2$. The errors are computed at $t = 4$ as compared to the exact solution,

$$s(x, y, t) = \tanh[y \cos(\omega t) - x \sin(\omega t)].$$

As with the LPR scheme, the stability limit of the ADER schemes appears to be a CFL of 0.5. Here we compare results of BDS_Q at a CFL of 0.9 for comparison to the ADER schemes at CFL of 0.45, which again represents the same fraction of the maximum allowable time step. At a resolution of 100^2, errors in the ADER3 scheme are intermediate between the BDS_Q scheme with and without limiting in both L^1 and L^∞. (The errors are quite close and lie within a narrow band.) We conjecture that this again reflects the more restrictive limiting in BDS_Q needed to enforce a maximum principle. As the grid is refined, the errors in ADER3 improve more rapidly than BDS_Q. In particular ADER3 shows convergence rates higher than third order whereas BDS_Q is between second and third order accurate. The issue here is related to the representation of the velocity field that is used by the different schemes. For the intended applications of BDS_Q, we typically obtain the velocity field from the solution of an elliptic PDE, thus the only characterization of the advection velocity is the integral average of the normal component over an edge.

This restriction leads to a reduction of convergence to less than third order for the frontogenesis problem with BDS_Q. The ADER schemes use derivative information about the velocity field that we assume is not available, allowing them to construct a more accurate solution on finer grids. Unfortunately, this also implies that the ADER schemes cannot be directly applied in the context considered here; it is not clear whether it would be possible to modify the algorithm to work with the restricted velocity information without a loss of accuracy. We have also considered the case when $\delta \to 0$ so that the discrete initial scalar field changes from -1 to 1 across the $y = 0$ interface. We note that BDS_Q was able to treat the discontinuous initial data without undershoot or overshoot over a range of mesh spacings and CFL conditions.

5. Summary and conclusions

We have presented a new finite volume scheme for linear advection in two dimensions that is based on reconstructing an appropriately limited multidimensional biquadratic profile and deriving a flux based on the multidimensional geometry of the characteristics. This scheme, which we refer to as BDS_Q, is third-order accurate for smooth problems and satisfies a maximum principle when the advective velocity field is spatially constant. Numerical evidence shows that the method continues to satisfy the maximum principle in more general circumstances. The new method is compared to two variations of unsplit PPM, which represent state-of-the-art algorithms for general systems of conservation laws. For advection that is not aligned with one of the coordinate axes, the new algorithm has better accuracy than either PPM scheme. Furthermore, the PPM algorithms do not satisfy a maximum principle and are subject to significant overshoot and undershoot. We have also shown that our method is competitive with modern WENO and ADER schemes.

Two aspects of the BDS_Q algorithm differ from the PPM algorithms and are important for guaranteeing a maximum principle. First, the BDS_Q algorithm uses a fully multidimensional limiting, whereas the PPM schemes limit in one direction only. Thus, the reconstructed profile over the entire cell in PPM can introduce new maxima and minima. Also, from a geometric point of view, the transverse flux corrections in the PPM schemes corresponding to $\Gamma^{+,-}$ are not evaluated at the correct location to mimic exact advection of the reconstructed profile. Note, however, that the transverse corrections to the edge states represent a higher-order effect in time so that the differences in the schemes are reduced as the CFL becomes smaller.

The extension of the method presented here to a more general scalar conservation law as done in [3] is straightforward. There are also a number of potential improvements that could be considered for the reconstruction phase of the algorithm, such as establishing a more formal approach to limiting, not only in the present context but also for the construction of higher-order approximations and to avoid limiting

smooth extrema. Additionally of interest would be the extension of this approach to three dimensions; we plan to pursue this in future work.

Acknowledgments

The work at LBNL was supported by the Applied Mathematics Program of the DOE Office of Advanced Scientific Computing Research under the U.S. Department of Energy under contract No. DE-AC02-05CH11231. S. May also received support from the Courant Institute of Mathematical Sciences under the U.S. Department of Energy under contract No. DE-FG02-88ER25053. Visualization was performed using the VisIt visualization software [25].

References

[1] A. S. Almgren, V. E. Beckner, J. B. Bell, M. S. Day, L. H. Howell, C. C. Joggerst, M. J. Lijewski, A. Nonaka, M. Singer, and M. Zingale, *CASTRO: A new compressible astrophysical solver, I: Hydrodynamics and self-gravity*, Astrophys. J. **715** (2010), no. 2, 1221–1238.

[2] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, *A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations*, J. Comput. Phys. **142** (1998), no. 1, 1–46. MR 99k:76096 Zbl 0933.76055

[3] J. B. Bell, C. N. Dawson, and G. R. Shubin, *An unsplit, higher order Godunov method for scalar conservation laws in multiple dimensions*, J. Comput. Phys. **74** (1988), 1–24.

[4] J. B. Bell, M. S. Day, C. A. Rendleman, S. E. Woosley, and M. A. Zingale, *Adaptive low Mach number simulations of nuclear flame microphysics*, J. Comput. Phys. **195** (2004), no. 2, 677–694.

[5] S. J. Billett and E. F. Toro, *On WAF-type schemes for multidimensional hyperbolic conservation laws*, J. Comput. Phys. **130** (1997), no. 1, 1–24. MR 97i:65148 Zbl 0873.65088

[6] B. Cockburn and C.-W. Shu, *Foreword [to the Proceedings of the First International Symposium on DG Methods]*, J. Sci. Comput. **22/23** (2005), 1–3. MR 2142187

[7] P. Colella, *Multidimensional upwind methods for hyperbolic conservation laws*, J. Comput. Phys. **87** (1990), no. 1, 171–200. MR 91c:76087 Zbl 0694.65041

[8] P. Colella and M. D. Sekora, *A limiter for PPM that preserves accuracy at smooth extrema*, J. Comput. Phys. **227** (2008), no. 15, 7069–7076. MR 2009d:76079 Zbl 1152.65090

[9] P. Colella and P. R. Woodward, *The piecewise parabolic method (ppm) for gas-dynamical simulations*, J. Comput. Phys. **54** (1984), 174–201.

[10] C. Dawson, *Foreword [special issue on discontinuous galerkin methods for incompressible elastic materials]*, Comput. Methods Appl. Mech. Engrg. **195** (2006), 3183.

[11] M. S. Day and J. B. Bell, *Numerical simulation of laminar reacting flows with complex chemistry*, Combust. Theory Modelling **4** (2000), no. 4, 535–556.

[12] A. Kurganov and G. Petrova, *A third-order semi-discrete genuinely multidimensional central scheme for hyperbolic conservation laws and related problems*, Numer. Math. **88** (2001), no. 4, 683–729. MR 2002e:65118 Zbl 0987.65090

[13] R. J. Leveque, *High-resolution conservative algorithms for advection in incompressible flow*, SIAM J. Numer. Anal. **33** (1996), no. 2, 627–665. MR 98b:76049 Zbl 0852.76057

[14] _____, *Finite volume methods for hyperbolic problems*, Cambridge University Press, 2002. Zbl 1010.65040

[15] D. Levy, G. Puppo, and G. Russo, *Compact central WENO schemes for multidimensional conservation laws*, SIAM J. Sci. Comput. **22** (2000), no. 2, 656–672. MR 2001d:65110 Zbl 0967.65089

[16] P. McCorquodale and P. Colella, *A high-order finite-volume method for hyperbolic conservation laws on locally-refined grids*, Commun. Appl. Math. Comput. Sci. **6** (2011), no. 1, 1–25.

[17] G. H. Miller and P. Colella, *A conservative three-dimensional Eulerian method for coupled solid-fluid shock capturing*, J. Comput. Phys. **183** (2002), no. 1, 26–82. MR 2003j:76080 Zbl 1057.76558

[18] S. Noelle, *The MoT-ICE: a new high-resolution wave-propagation algorithm for multidimensional systems of conservation laws based on Fey's method of transport*, J. Comput. Phys. **164** (2000), no. 2, 283–334. MR 2001f:76061

[19] A. Nonaka, A. S. Almgren, J. B. Bell, M. J. Lijewski, C. M. Malone, and M. Zingale, *MAESTRO: An adaptive low Mach number hydrodynamics algorithm for stellar flows*, Astrophys. J. Suppl. **188** (2010), no. 2, 358–383.

[20] G. S. H. Pau, A. S. Almgren, J. B. Bell, and M. J. Lijewski, *A parallel second-order adaptive mesh algorithm for incompressible flow in porous media*, Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. **367** (2009), no. 1907, 4633–4654. MR 2010j:76127 Zbl 1192.76056

[21] J. Saltzman, *An unsplit 3D upwind method for hyperbolic conservation laws*, J. Comput. Phys. **115** (1994), no. 1, 153–168. MR 1300337 Zbl 0813.65111

[22] C.-W. Shu, *High order weighted essentially nonoscillatory schemes for convection dominated problems*, SIAM Rev. **51** (2009), no. 1, 82–126. MR 2010a:65162 Zbl 1160.65330

[23] P. K. Smolarkiewicz and L. G. Margolin, *MPDATA: a finite-difference solver for geophysical flows*, J. Comput. Phys. **140** (1998), no. 2, 459–480. MR 98m:86002 Zbl 0935.76064

[24] E. F. Toro and V. A. Titarev, *ADER schemes for scalar non-linear hyperbolic conservation laws with source terms in three-space dimensions*, J. Comput. Phys. **202** (2005), no. 1, 196–215. MR 2005h:65140 Zbl 1061.65103

[25] *VisIt User's Manual*, version 1.5, Lawrence Livermore National Laboratory, Livermore, CA, 2005.

Received August 23, 2010. Revised February 7, 2011.

SANDRA MAY: may@cims.nyu.edu
Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, Mail Code: 0711, New York, NY 10012, United States

ANDREW NONAKA: AJNonaka@lbl.gov
Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, MS 50A-1148, Berkeley, CA 94720, United States

ANN ALMGREN: ASAlmgren@lbl.gov
Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, MS 50A-1148, Berkeley, CA 94720, United States

JOHN BELL: jbbell@lbl.gov
Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, MS 50A-1148, 1 Cyclotron Road, Berkeley, CA 94720, United States

Communications in Applied Mathematics and Computational Science

pjm.math.berkeley.edu/camcos

See inside back cover or pjm.math.berkeley.edu/camcos for submission instructions.

The subscription price for 2011 is US $70/year for the electronic version, and $100/year for print and electronic. Subscriptions, requests for back issues from the last three years and changes of subscribers address should be sent to Mathematical Sciences Publishers, Department of Mathematics, University of California, Berkeley, CA 94720-3840, USA.

Communications in Applied Mathematics and Computational Science, at Mathematical Sciences Publishers, Department of Mathematics, University of California, Berkeley, CA 94720-3840 is published continuously online. Periodical rate postage paid at Berkeley, CA 94704, and additional mailing offices.

CAMCoS peer review and production are managed by EditFLOW™ from Mathematical Sciences Publishers.

PUBLISHED BY
mathematical sciences publishers
http://msp.org/
A NON-PROFIT CORPORATION
Typeset in LaTeX

Communications in Applied Mathematics and Computational Science

vol. 6 no. 1 2011