SIAM J. SCI. COMPUT.
Vol. 33, No. 4, pp. 2039–2062

A THREE-DIMENSIONAL, UNSPLIT GODUNOV METHOD FOR SCALAR CONSERVATION LAWS*

A. NONAKA†, S. MAY‡, A. S. ALMGREN†, AND J. B. BELL†

Abstract. Linear advection of a scalar quantity by a specified velocity field arises in a number of different applications. Of particular interest here is the transport of species and energy in low Mach number models for combustion, atmospheric flows, and astrophysics, as well as contaminant transport in Darcy models of saturated subsurface flow. An important characteristic of these problems is that the velocity field is not known analytically. Instead, an auxiliary equation is solved to compute averages of the velocities over faces in a finite volume discretization. In this paper, we present a customized three-dimensional 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. Here we present a three-dimensional extension of the original BDS algorithm that is based on a limited trilinear profile within each cell. We compare this new method to several other unsplit approaches, including piecewise linear methods, piecewise parabolic methods, and wave propagation schemes.

Key words. Godunov method, scalar conservation law, linear advection

AMS subject classifications. 35-04, 35L65

DOI. 10.1137/100809520

1. Introduction.

The literature on numerical methods for hyperbolic partial differential equations focuses on general systems of conservation laws, particularly the compressible Euler equations (see [20], e.g., 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

$$s_t + (us)_x + (vs)_y + (ws)_z = 0, \tag{1.1}$$

where $s = s(x, y, z, t)$ is a scalar field and $\mathbf{u} = (u, v, w)$ represents a known velocity field. Important examples of this type of problem arise in projection-based algorithms for incompressible and more general low Mach number flows in which there is some constraint on the divergence of $\mathbf{u}$. Applications of these low Mach number projection algorithms include low Mach number terrestrial combustion [14], nuclear flame simulation [6], low Mach number stratified atmospheric [32] and astrophysical flows [28], 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 [29]. In several of the above problems, the

*Received by the editors September 23, 2010; accepted for publication (in revised form) June 1, 2011; published electronically August 18, 2011. This work 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 DE-AC02-05CH11231.

http://www.siam.org/journals/sisc/33-4/80952.html

†Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (ajnonaka@lbl.gov, asalmgren@lbl.gov, jbbell@lbl.gov).

‡Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 (may@cims.nyu.edu). This author's work was supported by the Courant Institute of Mathematical Sciences under the U.S. Department of Energy under contract DE-FG02-88ER25053.

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.

The target application of the algorithm places several constraints on the numerical method. 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. For low Mach number flows $\mathbf{u}$ is constructed using a projection that enforces the divergence constraint; for contaminant transport, $\mathbf{u}$ is determined from Darcy's law and incompressibility. Furthermore, we have only a limited characterization of the velocity field, typically integral averages of the normal component on faces 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, errors associated with overshoot in a species concentration can be significantly enhanced by reaction. 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 amount of literature on numerical methods for conservation laws, 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 various 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 three-dimensional second-order Godunov method, based on linear reconstruction, was presented by Saltzman [30], which was a generalization of the two-dimensional scheme developed by Colella [12]. Colella [12] 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 [26] developed a three-dimensional unsplit scheme based on the piecewise parabolic method (PPM) of Colella and Woodward [11]. This approach uses the same formalism as the piecewise linear algorithms but constructs a parabolic rather than a linear profile in each coordinate direction. Recent work by Colella and collaborators has investigated the use of less-restrictive limiters for PPM. Colella and Sekora [10] developed a new PPM limiter that preserves accuracy at smooth extrema

but suffers from sensitivity to roundoff error; more recently McCorquodale and Colella [25] introduced an improvement to that limiter which is less sensitive to roundoff error.

LeVeque [18, 19] and Langseth and LeVeque [17] introduced higher-order advection schemes based on geometric ideas derived from a wave propagation perspective; these are now publicly available in the CLAWPACK package. Billett and Toro [8] present a three-dimensional weighted average flux (WAF) method based on similar geometric ideas. These approaches, as well as the two-dimensional Mot-ICE-P1 scheme of Noelle [27], share a number of features with the scheme of Bell, Dawson, and Shubin [5] (BDS) that will be our starting point. Lukáčová-Medvid'ová et al. [21, 22, 23] develop finite volume evolution Galerkin methods (FVEG) that use geometric ideas based on bicharacteristics to develop numerical algorithms for nondiagonalizable systems. Smolarkiewicz and collaborators developed multidimensional advection schemes for geophysical flows based on flux-corrected transport ideas; see [32, 33] and the references cited therein. Another class of schemes is the ADER-type schemes developed by Toro and collaborators; see, for example, Toro and Titarev [34]. 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 faces of cells. We note that the higher-order ADER schemes are not directly applicable in our context because they require additional information about the velocity field that is not available in our context. Another class of schemes that has become popular for a wide range of problems is WENO-type schemes. The reader is referred to a recent survey article by Shu [31] for a general discussion of these types of methods. Of particular interest for multidimensional advection are unsplit, multidimensional versions of WENO such as the two-dimensional semidiscrete algorithm of Kurganov and Petrova [16] and the generalization to three dimensions by Balbás and Qian [3]. A final category of schemes is discontinuous Galerkin finite element methods. There have been recent special issues of journals focused on discontinuous Galerkin; see Dawson [13] and Cockburn and Shu [9]. 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 (1.1) 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. In particular, the method presented here is an extension of the two-dimensional Bell, Dawson, and Shubin (BDS) scheme to three dimensions. 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 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 that the method presented in this paper and the original BDS algorithm are fully explicit in time, and do not require a Runge–Kutta procedure. As noted in [5], the method is easily extendable to scalar conservation laws of the form

$$s_t + [uf(s)]_x + [vg(s)]_y + [wh(s)]_z = q(s). \tag{1.2}$$

In this paper, for ease of exposition, we restrict our developments to linear advection, i.e., $f(s) = g(s) = h(s) = s, q(s) = 0$. However, the method does not generalize to general systems of conservation laws.

This original BDS algorithm constructs a limited bilinear profile within each cell. There are two issues in the extension of this approach to three dimensions. First, we need to construct a limited trilinear profile within each cell. Second, we need to specify the detailed characteristic domain of dependence of each face to assemble the different contributions to the flux. These contributions are expressed as transverse corrections to the flux normal to the surface, similar to the construction in the unsplit Godunov schemes discussed above.

In section 2, we review the developments from the original two-dimensional BDS algorithm. In section 3, we present the three-dimensional algorithm. In section 4, we present numerical results comparing the three-dimensional BDS method to unsplit PLM and PPM schemes as well as the CLAWPACK wave propagation algorithm. These comparisons examine convergence rates, shape preservation, and overshoot/undershoot of the methods for both spatially constant and spatially variable divergence-free velocities. We also illustrate the behavior of the method on advection of a tracer in an incompressible flow being evolved using a projection method.

2. Review of two-dimensional scheme. Here we give a short summary of the original two-dimensional BDS method [5] to provide the reader with a basic understanding of the idea of the BDS scheme and to facilitate the understanding of the extension to three dimensions. An alternative summary of the original BDS method is given in [24]. We use a rectilinear grid with constant grid spacings Δx and Δy and a finite volume representation in which s_{ij}^n denotes the average value of s over the cell with index (ij) at time t^n. The face between cells with indices (ij) and $(i+1,j)$ is denoted $(i+1/2,j)$. Corners are denoted in a similar way, e.g., $(i+1/2,j+1/2)$. At each face, the normal velocity (e.g., $u_{i+1/2,j}$) is known and assumed to be constant over the time step. We advance the solution in time using a three-step procedure:

- *Step* I: Construct a limited piecewise bilinear representation of the solution in each grid cell of the form

$$s_{ij}(x,y) = s_{ij} + s_{x,ij}\cdot(x-x_i) + s_{y,ij}\cdot(y-y_j) + s_{xy,ij}\cdot(x-x_i)(y-y_j), \tag{2.1}$$

  where (x_i, y_j) are the physical coordinates of cell center (ij) and $s_{x,ij}, s_{y,ij}$, and $s_{xy,ij}$ are approximations for partial derivatives of s in cell (ij).
- *Step* II: Construct edge states, $s_{i+1/2,j}$, etc., by integrating the piecewise bilinear profiles over the space-time region determined by the characteristic domain of dependence of the face.
- *Step* III: Advance the solution in time using the conservative update equation

$$\begin{aligned} s_{ij}^{n+1} = s_{ij}^n &- \frac{\Delta t}{\Delta x}(u_{i+1/2,j}s_{i+1/2,j} - u_{i-1/2,j}s_{i-1/2,j}) \\ &- \frac{\Delta t}{\Delta y}(v_{i,j+1/2}s_{i,j+1/2} - v_{i,j-1/2}s_{i,j-1/2}). \end{aligned} \tag{2.2}$$

To construct the piecewise bilinear representation in *Step* I, we first compute values of s at each cell corner using high-order (bicubic) polynomial interpolation. These corner values are used to compute the slopes $s_{x,ij}$, $s_{y,ij}$, and $s_{xy,ij}$. In order to guarantee a maximum principle (for constant coefficient linear advection) the corner values may need to be adjusted prior to the calculation of the slopes to prohibit the creation of new extrema. The extension of *Step* I to three dimensions is straightforward and is explained in full detail in section 3.1.

Step II is the heart of the BDS scheme and, even in two dimensions, is considerably more complicated than *Steps* I and III. We use characteristic analysis to determine

the characteristic domain of dependence of each face. Since the velocity field on edges is known, we can use the piecewise bilinear representation at time t^n to determine the "average" value of s that passes through a given face over a time step. Hereby, we represent the edge states $s_{i+1/2,j}$, etc., in terms of integrals over space of time t^n data which are evaluated exactly (for our bilinear profile) using quadrature formulae. Reviewing *Step* II of the two-dimensional method in detail is very helpful for the understanding of the three-dimensional description of *Step* II given in section 3.2. We note that in the derivation of the two-dimensional method, time is used as a third coordinate direction in the included figures to graphically determine the characteristic domain of dependence of a face. This is not possible for the three-dimensional method.

3. Three-dimensional scheme. We use the same notation as in the two-dimensional case, with k representing the index for the z-direction. As before, we advance the solution in time using an analogous three-step procedure:

- *Step* I: Construct a limited piecewise trilinear representation of the solution in each grid cell of the form

$$\begin{aligned} s_{ijk}(x,y,z) = {} & s_{ijk} + s_{x,ijk}\cdot(x-x_i) + s_{y,ijk}\cdot(y-y_j) + s_{z,ijk}\cdot(z-z_k) \\ & + s_{xy,ijk}\cdot(x-x_i)(y-y_j) + s_{xz,ijk}\cdot(x-x_i)(z-z_k) \\ & + s_{yz,ijk}\cdot(y-y_j)(z-z_k) + s_{xyz,ijk}\cdot(x-x_i)(y-y_j)(z-z_k). \end{aligned} \tag{3.1}$$

  This procedure is described in section 3.1.
- *Step* II: Construct edge states, $s_{i+1/2,j,k}$, etc., by integrating the limited piecewise trilinear profiles over the space-time region determined by the characteristic domain of dependence of the face. This procedure is described in section 3.2.
- *Step* III: Advance the solution in time using the conservative update equation

$$\begin{aligned} s_{ijk}^{n+1} = s_{ijk}^{n} & - \frac{\Delta t}{\Delta x}\left(u_{i+1/2,j,k}s_{i+1/2,j,k} - u_{i-1/2,j,k}s_{i-1/2,j,k}\right) \\ & - \frac{\Delta t}{\Delta y}\left(v_{i,j+1/2,k}s_{i,j+1/2,k} - v_{i,j-1/2,k}s_{i,j-1/2,k}\right) \\ & - \frac{\Delta t}{\Delta z}\left(w_{i,j,k+1/2}s_{i,j,k+1/2} - w_{i,j,k-1/2}s_{i,j,k-1/2}\right). \end{aligned} \tag{3.2}$$

3.1. Profile reconstruction. We now describe the construction of a limited piecewise trilinear representation of the solution in each grid cell of the form given in (3.1). Following [5], for each cell we begin by computing point values at each corner using nearby cell-average values. We limit these values to prevent new extrema and then use difference formulae to compute the required derivatives.

To begin, for each corner, we find the tricubic polynomial whose cell averages coincide with cell averages for the 64 cells (i.e., the 4^3 block of cells) surrounding the corner. The value of the tricubic polynomial at the corner point gives an initial estimate for the value at that corner. The tricubic interpolation is a straightforward extension of the bicubic stencil given in section 3 of [5], which is equivalent to a multidimensional tensor product of Woodward and Colella's formula [11]. The full stencil has 64 points, so for brevity we list only the coefficients from one octant, noting

that the contribution from the remaining octants is symmetric:

$$\begin{aligned} s_{i+1/2,j+1/2,k+1/2} = & \frac{343}{1728} s_{i+1,j+1,k+1} \\ & - \frac{49}{1728} \left(s_{i+2,j+1,k+1} + s_{i+1,j+2,k+1} + s_{i+1,j+1,k+2} \right) \\ & + \frac{7}{1728} \left(s_{i+2,j+2,k+1} + s_{i+2,j+1,k+2} + s_{i+1,j+2,k+2} \right) \\ & - \frac{1}{1728} s_{i+2,j+2,k+2} + \cdots . \end{aligned} \tag{3.3}$$

For each cell, we consider the eight associated corner values. We will use a limited version of these corner values to construct our trilinear approximation. We begin by defining eight temporary corner values, e.g.,

$$s_{ijk}^{+++} = s_{i+1/2,j+1/2,k+1/2}, \quad s_{ijk}^{-++} = s_{i-1/2,j+1/2,k+1/2}, \quad \text{etc.} \tag{3.4}$$

Then, we offset each of these temporary values by the same constant, so that the average of the temporary values is equal to the cell average, s_{ijk}. Next, we modify the temporary corner values using a heuristic procedure that is a direct three-dimensional extension of the procedure in [5]. This procedure attempts to satisfy the following two constraints:

- *Constraint* 1: The average of the eight temporary corner values equals the cell-average value.
- *Constraint* 2: Each temporary corner value is not an extremum relative to its neighboring eight cell-average values, e.g.,

$$\alpha_{i+1/2,j+1/2,k+1/2} \le s_{ijk}^{+++} \le \beta_{i+1/2,j+1/2,k+1/2}, \tag{3.5}$$

$$\begin{aligned} \alpha_{i+1/2,j+1/2,k+1/2} &= \min(s_{i+i',j+j',k+k'}), \\ \beta_{i+1/2,j+1/2,k+1/2} &= \max(s_{i+i',j+j',k+k'}), \quad i',j',k' = 0,1. \end{aligned} \tag{3.6}$$

We begin our heuristic procedure by enforcing *Constraint* 2 for each of the eight temporary corner values by setting, e.g.,

$$s_{ijk}^{+++} = \max\left[\min\left(s_{ijk}^{+++}, \beta_{i+1/2,j+1/2,k+1/2} \right), \alpha_{i+1/2,j+1/2,k+1/2} \right]. \tag{3.7}$$

Then, we iterate over the following steps in order to modify the temporary values so that we enforce *Constraint* 1 without violating *Constraint* 2:

- *Step* 1: Compute the sum of the differences between the temporary values and the cell-average value

$$\delta = \left(\sum_{\pm} \sum_{\pm} \sum_{\pm} s_{ijk}^{\pm\pm\pm} \right) - 8 s_{ijk}. \tag{3.8}$$

  Assume $\delta > 0$ (if $\delta < 0$ the algorithm is analogous, and if $\delta = 0$, we have "converged" and steps 2–4 are unnecessary).
- *Step* 2: Note which temporary corner values are larger than s_{ijk} by more than $\epsilon = 10^{-10}$. Define an integer n as the number of corner values that meet this criterion.

- *Step* 3: Looping over each of the temporary corner values that is larger than s_{ijk} by more than $\epsilon = 10^{-10}$, do the following (using corner s_{ijk}^{+++} as an example):
  - Define $\gamma = \min\left[(\delta/n), s_{ijk}^{+++} - \alpha_{i+1/2,j+1/2,k+1/2}\right]$.
  - Set $s_{ijk}^{+++} = s_{ijk}^{+++} - \gamma$.
  - Set $n = n - 1$.
  - Set $\delta = \delta - \gamma$.
- *Step* 4: Return to *Step* 1.

After at most six iterations, we compute the final slopes used for our trilinear polynomial using

$$s_{x,ijk} = \frac{(s_{ijk}^{+++} + s_{ijk}^{++-} + s_{ijk}^{+-+} + s_{ijk}^{+--}) - (s_{ijk}^{-++} + s_{ijk}^{-+-} + s_{ijk}^{--+} + s_{ijk}^{---})}{4\Delta x}, \tag{3.9}$$

$$s_{xy,ijk} = \frac{(s_{ijk}^{+++} + s_{ijk}^{++-} + s_{ijk}^{--+} + s_{ijk}^{---}) - (s_{ijk}^{-++} + s_{ijk}^{-+-} + s_{ijk}^{+-+} + s_{ijk}^{+--})}{2\Delta x\Delta y}, \tag{3.10}$$

$$s_{xyz,ijk} = \frac{(s_{ijk}^{+++} + s_{ijk}^{+--} + s_{ijk}^{-+-} + s_{ijk}^{--+}) - (s_{ijk}^{-++} + s_{ijk}^{+-+} + s_{ijk}^{++-} + s_{ijk}^{---})}{\Delta x\Delta y\Delta z}, \tag{3.11}$$

and analogous formulae for s_y, s_z, s_{xz}, and s_{yz}. We have now computed the limited trilinear polynomial representation for each cell. It was noted in [5] that the two-dimensional analogue of this heuristic procedure led to results comparable to solving the following minimization problem for each cell in every time step: Find the bilinear function that is closest to the bilinear function defined by the original interpolation in L^2 subject to *Constraints* 1 and 2. We have not observed any break down of this heuristic procedure in our simulations.

3.2. Construction of edge states. We now describe the construction of the edge states that will be used in the conservative update equation (3.2). We restrict our development to the computation of $s_{i+1/2,j,k}$; other faces are treated analogously. Here we assume that $u_{i+1/2,j,k} > 0$, noting that the case where $u_{i+1/2,j,k} < 0$ is treated analogously. Figure 3.1 shows cell R with index (ijk) and its eight surrounding neighbors with an offset in the y and/or z-directions. The red cells (T, V, X, and Z) are offset from cell R by ± 1 in either the y or z-direction and the green cells (S, U, W, and Y) are offset from cell R by ± 1 in the y-direction and ± 1 in the z-direction. Since $u_{i+1/2,j,k} > 0$, we solve for $s_{i+1/2,j,k}$ using piecewise trilinear representations of s from cells on the upwind side of face $(i + 1/2, j, k)$, i.e., we use some subset of cells R, S, T, U, V, W, X, Y, and Z.

It is important to note that Figures 1 and 2 in [5] are drawn in three dimensions, with two dimensions representing the x and y spatial coordinates and the third dimension representing time. For the figures in this paper, we must draw all three spatial dimensions and are unable to graphically depict time. Many of the surfaces and volumes described in this paper shrink over the time step, and thus, we choose to only draw surfaces and volumes depicted at t^n. The analogous figures from the two-dimensional method would exclude the third axis and only show surfaces in the x-y plane.

We begin by linearizing (1.1) about cell R and expanding the derivative with respect to x to obtain

$$s_t + u_{i+1/2,j,k} s_x + (vs)_y + (ws)_z + s u_{x,ijk} = 0. \tag{3.12}$$

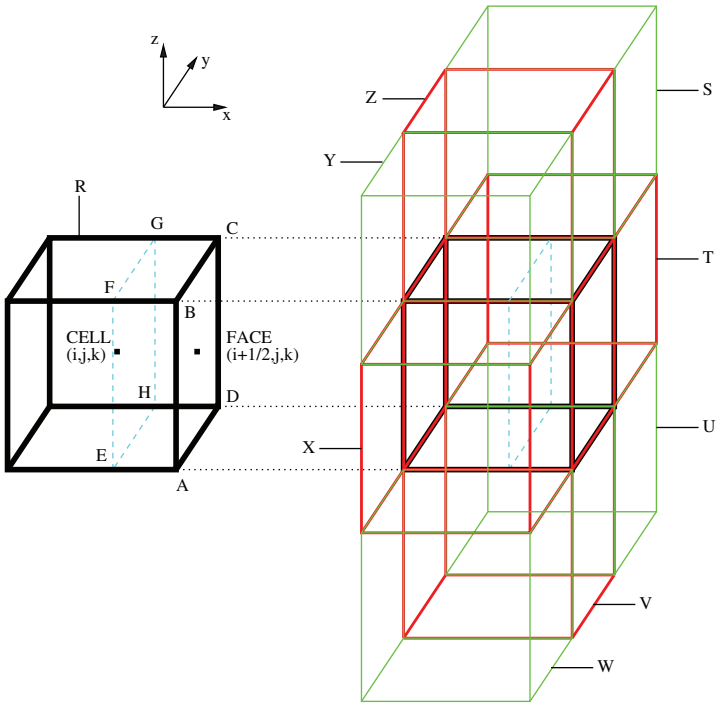


FIG. 3.1. *(Left) Cell R has index (ijk). Box $ABCDEFGH$ is the characteristic domain of dependence in x of face $ABCD$ at t^n since $u_{i+1/2,j,k} > 0$. (Right) Piecewise trilinear representations of s within cell R, red cells (T, V, X, and Z), and green cells (S, U, W, and Y) will be used to construct $s_{i+1/2,j,k}$. (A color version of this figure is available online.)*

As noted in [5], we have not expanded derivatives with respect to y or z. These terms need to be upwinded to correctly capture the three-dimensional characteristic domain of dependence. Also, each y and z face will be treated independently since in general the velocity field varies in space.

Referring to the left of Figure 3.1, the characteristic domain of dependence in x of face $ABCD$ over the time interval $t \in [t^n, t^{n+1}]$ is given by a box with configuration $ABCDEFGH$ at t^n that shrinks over the time step. In particular, points E, F, G, and H move in the positive x-direction with speed $u_{i+1/2,j,k}$ such that segments AE, BF, CG, and DH have length $|u_{i+1/2,j,k}|\Delta t$ at t^n and zero length at t^{n+1}. The intermediate-time lengths can be found by linearly interpolating between the lengths at t^n and t^{n+1}. We denote this space-time region as $\mathcal{D}^x$. For the rest of this paper, we will use a special notation in which an overbar for a given point is used as a reminder that the point changes location over time. If the overbar superscript is omitted, we are specifically referring to the location of the point at t^n.

We form an expression for $s_{i+1/2,j,k}$ by integrating (3.12) over $\mathcal{D}^x$:

$$\int_{t^n}^{t^{n+1}} \int\int\int_{ABCD\overline{EFGH}} \left[s_t + u_{i+1/2,j,k} s_x + (vs)_y + (ws)_z + s u_{x,ijk} \right] dx\, dy\, dz\, dt = 0. \tag{3.13}$$

Integrating by parts, noting that $u_{i+1/2,j,k}$ is constant over the time step and that the integrals on the time varying surfaces vanish because they are characteristic, this

simplifies to

$$
\begin{aligned}
0 = &- \int\int\int_{ABCDEFGH} s\ dx\ dy\ dz\Big|_{t^n} + u_{i+1/2,j,k} s_{i+1/2,j,k} \Delta y \Delta z \Delta t \\
&+ \int_{t^n}^{t^{n+1}} \int\int_{CD\overline{GH}} vs\ dx\ dz\ dt - \int_{t^n}^{t^{n+1}} \int\int_{AB\overline{EF}} vs\ dx\ dz\ dt \\
&+ \int_{t^n}^{t^{n+1}} \int\int_{BC\overline{FG}} ws\ dx\ dy\ dt - \int_{t^n}^{t^{n+1}} \int\int_{AD\overline{EH}} ws\ dx\ dy\ dt \\
&+ \int_{t^n}^{t^{n+1}} \int\int\int_{ABCD\overline{EFGH}} su_{x,ijk}\ dx\ dy\ dz\ dt. \qquad (3.14)
\end{aligned}
$$

The integrals containing vs (and ws) represent transverse fluxes, each of which corresponds to one of two possible cases, depending on the sign of $v_{i,j\pm 1/2,k}$ (and $w_{i,j,k\pm 1/2}$). As an example, consider the integral over $CD\overline{GH}$. In the first case (if $v_{i,j+1/2,k} > 0$), this integral eliminates the effects of characteristics that originate from box $ABCDEFGH$ at t^n but pass through face $CD\overline{GH}$, rather than face $ABCD$, during the time step. In the second case (if $v_{i,j+1/2,k} < 0$), this integral accounts for the effects of s that do not originate from box $ABCDEFGH$ at t^n but do pass through face $ABCD$ during the time step (such characteristics originate from either box S, T, or U). In the latter case, these external characteristics originate from red cell T from Figure 3.1. Note that all characteristics traced backward in time from face $ABCD$ at some $t \in [t^n, t^{n+1}]$ intersect either box $ABCDEFGH$ at t^n or one of the four faces $AB\overline{EF}$, $BC\overline{FG}$, $AD\overline{EH}$, or $CD\overline{GH}$, at some $t \in [t^n, t^{n+1}]$. Furthermore, any characteristic traced forward in time from box $ABCDEFGH$ at t^n intersects one of the faces $ABCD$, $AB\overline{EF}$, $BC\overline{FG}$, $AD\overline{EH}$, or $CD\overline{GH}$, at some $t \in [t^n, t^{n+1}]$.

The first term on the right-hand side of (3.14) is evaluated using the midpoint quadrature rule,

$$
\int\int\int_{ABCDEFGH} s\ dx\ dy\ dz\Big|_{t^n} \equiv s_M \cdot (u_{i+1/2,j,k} \Delta t) \Delta y \Delta z, \qquad (3.15)
$$

where s_M is the average value of s over box $ABCDEFGH$ at t^n. To compute s_M, we evaluate the trilinear function within cell R at the centroid of box $ABCDEFGH$. Since all bounding faces are aligned with the coordinate axes, this formula is exact for a trilinear polynomial. As in [5], the spatial part of the integral in the last term on the right-hand side of (3.14) is also evaluated using s_M with an explicit Euler treatment of the temporal integration. We now rewrite (3.14) to obtain the normal predictor equation:

$$
s_{i+1/2,j,k} = s_M - \frac{\Delta t}{2\Delta y}\left(\Gamma^{y+} - \Gamma^{y-}\right) - \frac{\Delta t}{2\Delta z}\left(\Gamma^{z+} - \Gamma^{z-}\right) - \frac{\Delta t}{2} s_M u_{x,ijk}. \qquad (3.16)
$$

Here, Γ^{y+} and Γ^{y-} (Γ^{z+} and Γ^{z-}) are the transverse flux corrections, which represent the average values of the flux vs (ws) over faces $CD\overline{GH}$ and $AB\overline{EF}$ ($BC\overline{GF}$ and $AD\overline{EH}$) over $t \in [t^n, t^{n+1}]$. The factors of $\Delta t/2$ in (3.16) come from the fact that we are evaluating the last five integrals on the right-hand side of (3.14) over space-time regions that are shrinking with respect to x in time. Thus, the space-time integral of these characteristic regions is equal to the corresponding area (or volume) at t^n multiplied by $\Delta t/2$. We compute spatial derivatives of velocity using the known face velocities, e.g., $u_{x,ijk} = (u_{i+1/2,j,k} - u_{i-1/2,j,k})/\Delta x$, which follows from the standard

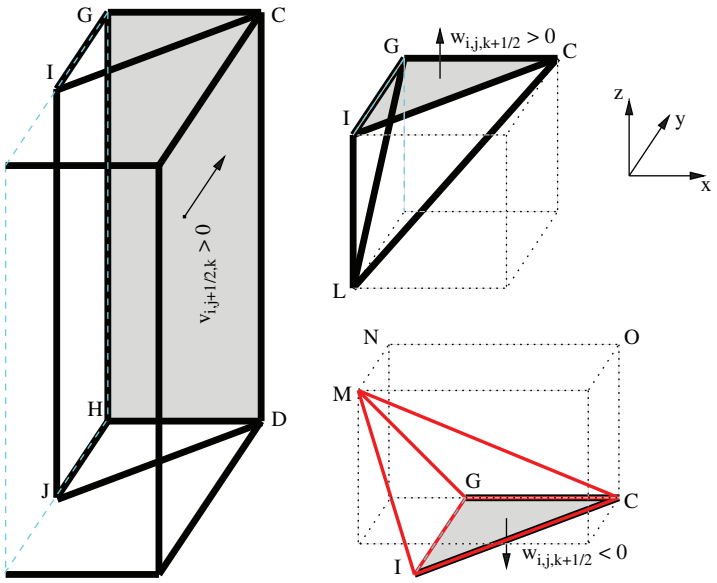


FIG. 3.2. *(Left) Volume $CDGHIJ$ is the characteristic domain of dependence in x and y of face $CD\overline{GH}$ at t^n for the case when $v_{i,j+1/2,k} > 0$. (Top right) Volume $CGIL$ is the characteristic domain of dependence of face $C\overline{GI}$ in all spatial dimensions at t^n for the case when $w_{i,j+1,k+1/2} > 0$. (Bottom right) Volume $CGIM$ is the characteristic domain of dependence of face $C\overline{GI}$ in all spatial dimensions at t^n for the case when $w_{i,j+1,k+1/2} < 0$. The red volume outline indicates that this region lies in red cell Z from Figure* 3.1. *(A color version of this figure is available online.)*

six-point divergence stencil for face-centered velocities (for many applications of this type of discretization methodology, this definition of divergence is implicit in the discretization of the elliptic partial differential equation used to compute the velocity field).

To complete the procedure we must compute Γ^{y+}/Γ^{y-} (and Γ^{z+}/Γ^{z-}) by evaluating the integrals of vs (and ws) over the faces $CD\overline{GH}/AB\overline{EF}$ (and $BC\overline{GF}/AD\overline{EH}$) over $t \in [t^n, t^{n+1}]$. We restrict the development to the integral over $CD\overline{GH}$ to compute Γ^{y+}; the others are computed in an analogous way. The integrand for this term represents the flux in the y-direction. For this reason we estimate an average value of s on face $CD\overline{GH}$ using linearization from the upwind side of face $(i, j + 1/2, k)$, and then compute the flux.

To compute Γ^{y+}, we first consider the case where $v_{i,j+1/2,k} > 0$. We linearize (1.1) about cell R and expand the derivatives with respect to x and y to obtain

$$s_t + u_{i+1/2,j,k} s_x + v_{i,j+1/2,k} s_y + (ws)_z + s u_{x,ijk} + s v_{y,ijk} = 0. \tag{3.17}$$

We have not expanded the derivative with respect to z; as before, these terms need to be upwinded independently at each edge to capture the characteristic domain of dependence accurately. Referring to the left of Figure 3.2, the characteristic domain of dependence in x and y of face $CD\overline{GH}$ over $t \in [t^n, t^{n+1}]$ is given by the space-time region bounded by volume $CD\overline{GHIJ}$ over $t \in [t^n, t^{n+1}]$. In particular,

- $C\overline{G}$ and $D\overline{H}$ have length $|u_{i+1/2,j,k}|\Delta t$ at t^n, length zero at t^{n+1}, with the length changing as a linear function of time.
- $\overline{GI}$ and $\overline{HJ}$ have length $|v_{i,j+1/2,k}|\Delta t$ at t^n, length zero at t^{n+1}, with the length changing as a linear function of time.

Consequently, triangular faces $C\overline{GI}$ and $D\overline{HJ}$ remain geometrically similar to triangular faces CGI and DHJ. We denote this space-time region as $\mathcal{D}^{xy}$. We compute Γ^{y+} by first forming an expression for s_{CDGH} by integrating (3.17) over $\mathcal{D}^{xy}$:

$$\int_{t^n}^{t^{n+1}} \int\int\int_{CD\overline{GHIJ}} \left[s_t + u_{i+1/2,j,k} s_x + v_{i,j+1/2,k} s_y + (ws)_z + s u_{x,ijk} + s v_{y,ijk} \right] dx\ dy\ dz\ dt = 0. \tag{3.18}$$

Integrating by parts, noting that $v_{i,j+1/2,k}$ is constant over the time step and that the integrals along time-varying surfaces vanish because the surfaces are characteristic, (3.18) simplifies to

$$\begin{aligned} 0 = &- \int\int\int_{CDGHIJ} s\ dx\ dy\ dz \Big|_{t^n} + v_{i,j+1/2,k} s_{CDGH} \frac{u_{i+1/2,j,k}\Delta t}{2} \Delta z \Delta t \\ &+ \int_{t^n}^{t^{n+1}} \int\int_{C\overline{GI}} ws\ dx\ dy\ dt - \int_{t^n}^{t^{n+1}} \int\int_{D\overline{HJ}} ws\ dx\ dy\ dt \\ &+ \int_{t^n}^{t^{n+1}} \int\int\int_{CD\overline{GHIJ}} (s u_{x,ijk} + s v_{y,ijk})\, dx\ dy\ dz\ dt. \end{aligned} \tag{3.19}$$

The integrals containing ws represent corner fluxes, each of which can correspond to one of two possible cases, depending on the sign of $w_{i,j,k\pm 1/2}$. In the first case ($w_{i,j,k+1/2} > 0$), the integral eliminates the effects of characteristics that originate from volume $CDGHIJ$ at t^n but do not pass through face $CD\overline{GH}$ during the time step. See the upper right of Figure 3.2 for a depiction of such a spatial region at t^n. In the second case ($w_{i,j,k+1/2} < 0$), the integral accounts for the effects of characteristics that do not originate from volume $CDGHIJ$ at t^n but do pass through face $CD\overline{GH}$ during the time step. See the lower right of Figure 3.2 for a depiction of such a spatial region at t^n. In the latter case, these external characteristics originate from red cell Z $(i, j, k+1)$ in Figure 3.1. All characteristics traced backward in time from face $CD\overline{GH}$ for $t \in [t^n, t^{n+1}]$ intersect either volume $CDGHIJ$ at t^n or one of the two triangles, $C\overline{GI}$ or $D\overline{HJ}$ at some $t \in [t^n, t^{n+1}]$. Furthermore, any characteristic traced forward in time from volume $CDGHIJ$ at t^n intersects either face $CD\overline{GH}$ or one of the triangles $C\overline{GI}$ or $D\overline{HJ}$ at some $t \in [t^n, t^{n+1}]$.

The first term on the right-hand side of (3.19) can be simplified by analytically integrating the trilinear approximation for s over $CDGHIJ$ at t^n:

$$\int\int\int_{CDGHIJ} s\ dx\ dy\ dz \Big|_{t^n} = s_{M2} \frac{(u_{i+1/2,j,k}\Delta t)(v_{i,j+1/2,k}\Delta t)}{2} \Delta z, \tag{3.20}$$

where s_{M2} is the average value of s over $CDGHIJ$. Since we are dealing with trilinear polynomials and faces CGI and DHJ are normal to the $\pm z$-direction, s_{M2} is equal to the average of the trilinear function evaluations at the centroids of faces $CDGH$, $GHIJ$, and $CDIJ$ in cell R. We now rewrite (3.19) to obtain the transverse predictor equation:

$$s_{CDGH} = s_{M2} - \frac{\Delta t}{3\Delta z} \left(\Gamma^{y+,z+} - \Gamma^{y+,z-} \right) - \frac{\Delta t}{3} \left(s_{M2} u_{x,ijk} + s_{M2} v_{y,ijk} \right), \tag{3.21}$$

where $\Gamma^{y+,z+}$ ($\Gamma^{y+,z-}$) are the corner flux corrections that represent the average value of the flux ws over faces $C\overline{GI}$ ($D\overline{HJ}$) over $t \in [t^n, t^{n+1}]$. The factors of $\Delta t/3$ in (3.21)

come from the fact that we are evaluating the last three integrals on the right-hand side of (3.19) over space-time regions that are shrinking with respect to x and y. In other words, the space-time integral of these characteristic regions is equal to the corresponding area (or volume) at t^n multiplied by $\Delta t/3$. After computing s_{CDGH}, we set $\Gamma^{y+} = v_{i,j+1/2,k} s_{CDGH}$. We will now complete the description of Γ^{y+} when $v_{i,j+1/2,k} > 0$ by describing how to compute the corner flux corrections. Then we will describe the computation of Γ^{y+} when $v_{i,j+1/2,k} < 0$.

To compute the corner fluxes, we restrict the development to the integral over $C\overline{GI}$ to compute $\Gamma^{y+,z+}$; $\Gamma^{y+,z-}$ is computed in an analogous way. The integrand for this term represents the flux in the z-direction. For this reason we estimate an average value of s on face $C\overline{GI}$ using linearization from the upwind side of face $(i, j, k + 1/2)$ and then compute the flux.

To compute $\Gamma^{y+,z+}$, first consider the case where $w_{i,j,k+1/2} > 0$. We linearize (1.1) about cell R and expand the derivatives with respect to all three directions to obtain

$$s_t + u_{i+1/2,j,k} s_x + v_{i,j+1/2,k} s_y + w_{i,j,k+1/2} s_z + s u_{x,ijk} + s v_{y,ijk} + s w_{z,ijk} = 0. \tag{3.22}$$

Referring to the upper right of Figure 3.2, the characteristic domain of dependence in all dimensions of face $C\overline{GI}$ over $t \in [t^n, t^{n+1}]$ is given by the space-time region bounded by volume $C\overline{GIL}$ over $t \in [t^n, t^{n+1}]$. This volume changes shape over time such that

- $C\overline{G}$ and $\overline{GI}$ change as described before.
- $\overline{IL}$ has length $|w_{i,j,k+1/2}\Delta t|$ at t^n, length zero at t^{n+1}, with the length changing as a linear function of time.

In other words, volume $C\overline{GIL}$ remains geometrically similar to $CGIL$. We denote this space-time region as $\mathcal{D}^{xyz}$. All characteristics traced backward in time from face $C\overline{GI}$ for $t \in [t^n, t^{n+1}]$ intersect volume $CGIL$ at t^n. Furthermore, any characteristic traced forward in time from volume $CGIL$ at t^n intersects face $C\overline{GI}$ at some $t \in [t^n, t^{n+1}]$. We compute $\Gamma^{y+,z+}$ by first forming an expression for s_{CGI} by integrating (3.22) over $\mathcal{D}^{xyz}$:

$$\int_{t^n}^{t^{n+1}} \int\int\int_{C\overline{GIL}} [s_t + u_{i+1/2,j,k} s_x + v_{i,j+1/2,k} s_y + w_{i,j,k+1/2} s_z + s u_{x,ijk} + s v_{y,ijk} + s w_{z,ijk}] dx\, dy\, dz\, dt = 0. \tag{3.23}$$

Integrating (3.23) by parts and simplification lead to the corner predictor equation:

$$s_{CGI} = s_{M3} - \frac{\Delta t}{4} \left(u_{x,ijk} s_{M3} + v_{y,ijk} s_{M3} + w_{z,ijk} s_{M3} \right), \tag{3.24}$$

where s_{M3} is the average value of s over volume $CGIL$. We evaluate s_{M3} using the five-point Gaussian quadrature rule given in [15], which is accurate for cubic polynomials over three-dimensional tetrahedra using the trilinear polynomial representation in cell R. The factor of $\Delta t/4$ in (3.24) comes from the fact that we are evaluating the last three integrals on the right-hand side of (3.23) over a space-time region that is shrinking with respect to x, y, and z. In other words, the space-time integral of this characteristic region is equal to the volume at t^n multiplied by $\Delta t/4$. Finally, we set $\Gamma^{y+,z+} = w_{i,j,k+1/2} s_{CGI}$.

The other case to consider when computing $\Gamma^{y+,z+}$ is when $w_{i,j,k+1/2} < 0$. The resulting developments are similar to the case where $w_{i,j,k+1/2} > 0$. In particular, we now linearize (1.1) about red cell Z from Figure 3.1 rather than cell R to compute s_{CGI}. The resulting volume $CGIM$, which we integrate over, is shown in the lower-

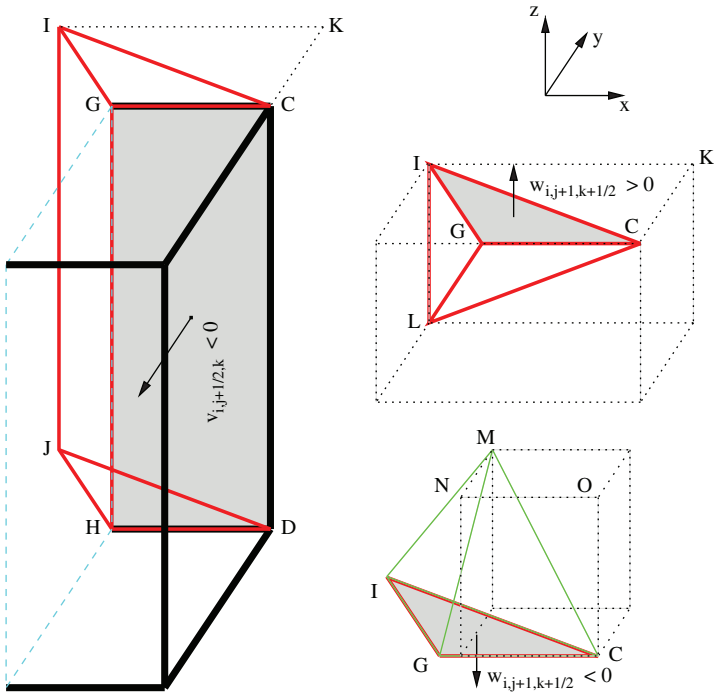


FIG. 3.3. *(Left) Volume $CDGHIJ$ is the characteristic domain of dependence in x and y of face $CD\overline{GH}$ at t^n for the case when $v_{i,j+1/2,k} < 0$. The red volume outline indicates that this region lies in red cell T from Figure* 3.1. *(Top right) Volume $CGIL$ is the characteristic domain of dependence of face $C\overline{GI}$ in all spatial dimensions at t^n for the case when $w_{i,j,k+1/2} > 0$. The red volume outline indicates that this region lies in red cell T from Figure* 3.1. *(Bottom right) Volume $CGIM$ is the characteristic domain of dependence of face $C\overline{GI}$ in all spatial dimensions at t^n for the case when $w_{i,j,k+1/2} < 0$. The green volume outline indicates that this region lies in green cell S from Figure* 3.1. *(A color version of this figure is available online.)*

right of Figure 3.2. However, the shape of volume $CGIM$ is determined by face velocities that do not lie on one of the six faces of cell R. In this case

- NO has length $|u_{i+1/2,j,k+1}|\Delta t$, whereas CG has length $|u_{i+1/2,j,k}|\Delta t$.
- MN has length $|v_{i,j+1/2,k+1}|\Delta t$, whereas GI has length $|v_{i,j+1/2,k}|\Delta t$.
- CO has length $|w_{i,j,k+1/2}|\Delta t$, which is analogous to IL.

Also, $\overline{G}$, $\overline{I}$, $\overline{M}$, $\overline{N}$, and $\overline{O}$ move such that volume $C\overline{GIM}$ remains geometrically similar to volume $CGIM$. Using these guidelines, it is possible for point M to lie outside of red cell Z from Figure 3.1. This can happen if $u_{i+1/2,j,k+1} < 0$ or $v_{i,j+1/2,k+1} < 0$. In this case, we simply construct our tetrahedral regions as if the velocity component(s) that violates this condition was equal to zero. In other words, we do not allow the characteristic region to extend across multiple cells. We enforce this idea for the remainder of the algorithm. Then, we set $\Gamma^{y+,z+} = w_{i,j,k+1/2} s_{CGI}$. The description of Γ^{y+} when $v_{i,j+1/2,k} > 0$ is now complete.

To consider the computation of Γ^{y+} when $v_{i,j+1/2,k} < 0$, refer to the left of Figure 3.3. The derivation follows as before, noting that we linearize (1.1) about red cell T from Figure 3.1 to compute s_{CDGH} and that

- CG and DH have length $|u_{i+1/2,j,k}|\Delta t$.
- IK has length $|u_{i+1/2,j+1,k}|\Delta t$.
- CK has length $|v_{i,j+1/2,k}|\Delta t$.

To compute $\Gamma^{y+,z+}$ when $w_{i,j+1,k+1/2} > 0$, refer to the upper-right of Figure 3.3. We linearize (1.1) about red cell T from Figure 3.1, noting that IL has length $|w_{i,j+1,k+1/2}|\Delta t$. To compute $\Gamma^{y+,z+}$ when $w_{i,j+1,k+1/2} < 0$, refer to the lower-right of Figure 3.3. We linearize (1.1) about green cell S from Figure 3.1, noting that

- NO has length $|u_{i+1/2,j+1,k+1}|\Delta t$.
- MN has length $|v_{i,j+1/2,k+1}|\Delta t$.
- CO has length $|w_{i,j+1,k+1/2}|\Delta t$.

This completes the description of the construction of edge states.

4. Results. In this section, we perform a series of tests to examine the overshoot, shape tracking, error, and convergence rates for our algorithm. All simulations are performed on a unit cube with $x, y, z \in [0, 1]$, periodic boundary conditions, a CFL number of 0.9, and a final time of $t = 1$. We perform each simulation using a domain with 64^3, 128^3, and 256^3 cells. We compare results for our new scheme, denoted as "BDS," to Saltzman's unsplit piecewise-linear method [30], referred to as "PLM", and to the unsplit PPM method of Miller and Colella [26]. In the PLM method, the piecewise-linear representation is computed using fourth-order limited slopes in each coordinate direction; PPM uses a limited quadratic profile in each coordinate direction. We consider two limiters for PPM: a recent version designed to avoid limiting at smooth extrema, referred to as "PPM2," [10, 25], and the original PPM limiter (labeled as "PPM1") [11]. We also compare our results to the three-dimensional advection algorithm available in the CLAWPACK package [17, 18, 19], referred to as "CLAW," with the full corner coupling and van Leer limiting [20] options enabled. We also present some results without limiters. For the BDS algorithm, turning off limiters is achieved by using the unlimited corner values as defined in (3.3) and (3.4) to compute slopes for the trilinear polynomial representation using (3.9)–(3.11). For the PLM algorithm, turning off limiters is achieved by using an unlimited fourth-order slope formula. For the PPM algorithm (PPM1 and PPM2 reduce to the same algorithm in the absence of limiters), turning off limiters is achieved by using the one-dimensional quadratic profile in each coordinate direction in each cell with no limiting. For the CLAW algorithm, we simply disable the van Leer limiters. In our tables, we define the L^1 and L^2 error as

$$L^1 = \frac{1}{N}\sum_{ijk} |s_{ijk} - s^{\text{exact}}_{ijk}|, \qquad L^2 = \sqrt{\frac{1}{N}\sum_{ijk} |s_{ijk} - s^{\text{exact}}_{ijk}|^2}, \tag{4.1}$$

where N is the number of cells in the domain and s^{exact} is the exact solution. We define the convergence rate between adjacent resolutions as

$$p^{\text{coarser/finer}} = \log_2\left(\frac{L_{\text{coarser}}}{L_{\text{finer}}}\right). \tag{4.2}$$

4.1. Constant velocity advection. Our first test is for constant-velocity, off-diagonal advection using $\mathbf{u} = (1.0, 0.5, 0.25)$. We advect a spherical step function with an initial minimum of 0 and maximum of 1:

$$s_0(x, y, z) = \begin{cases} 1, & r \le 0.1, \\ 0, & r > 0.1; \end{cases} \qquad r = \sqrt{(x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2}. \tag{4.3}$$

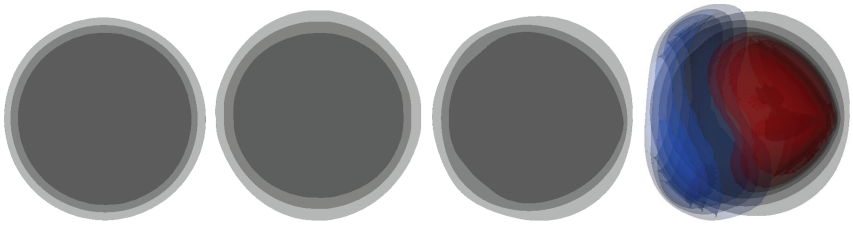
We also advect a Gaussian profile that has an initial analytical range of $s_0 \in [0, 1]$:

$$s_0(x, y, z) = e^{-300r^2}, \tag{4.4}$$

where r is given in (4.3).

TABLE 4.1
*Minimum and maximum scalar values for advection of a spherical step function at $t = 1$ with $\mathbf{u} = (1, 0.5, 0.25)$. *The actual maximum for the 256^3 BDS simulation is $1 - \epsilon$, where $\epsilon = 7.95e-11$.*

Scheme	64^3 min	64^3 max	128^3 min	128^3 max	256^3 min	256^3 max
Exact	0.00e-00	1.00000	0.000e-00	1.00000	0.000e-00	1.00000
BDS	−1.37e-11	0.99754	−3.11e-12	0.99999	−4.80e-11	1.00000*
PPM2	−2.17e-01	1.32268	−2.58e-01	1.29997	−2.62e-01	1.25976
PPM1	−2.13e-01	1.26217	−2.52e-01	1.27265	−2.61e-01	1.25617
PLM	−2.08e-01	1.24458	−2.59e-01	1.26629	−2.70e-01	1.24739
CLAW	−1.53e-03	0.99826	−3.04e-03	1.00211	−3.99e-03	1.00410



(a) BDS - Contours: 0.1, 0.5, 0.9

(b) CLAW - Contours: 0.1, 0.5, 0.9

(c) PPM2 - Contours: 0.1, 0.5, 0.9

(d) PPM2 - Contours: (Blue) -0.15, -0.1, -0.05 (Red) 1.05, 1.1, 1.15

FIG. 4.1. *Results for advection of a spherical step function with $\mathbf{u} = (1, 0.5, 0.25)$ at 256^3 resolution.* 4.1(a)–4.1(c): *The gray contours lie at scalar values of* 0.1, 0.5, *and* 0.9. 4.1(d): *We have added additional contours to the PPM*2 *solution. The blue contours show where the method undershoots, corresponding to scalar values of* −0.15, −0.1, *and* −0.05, *and the red contours show where the method overshoots, corresponding to scalar values of* 1.05, 1.1, *and* 1.15.

4.1.1. Constant velocity advection: Step function. In Table 4.1 we report the minimum and maximum scalar values of the computed solutions for the spherical step function at $t = 1$. CLAW shows a modest amount of over/undershoot that becomes worse with resolution (with a maximum of 0.4% on a 256^3 grid). The PPM2, PPM1, and PLM algorithms show considerable overshoot that does not appreciably improve with resolution. The PPM2 algorithm is the worst offender, with overshoot/undershoot of 22%/32% in the coarsest simulation and 26%/26% in the finest simulation. The BDS algorithm does a remarkable job of preserving the peak value of the step function without any overshoot. At the coarsest resolution, the peak value is 0.99754 and at the finest resolution, the peak value is $1 - \epsilon$, where $\epsilon = 7.95e-11$. The undershoot in the BDS algorithm is $\mathcal{O}(10^{-11})$ and can be directly attributed to roundoff error and the choice of ϵ in the slope limiting algorithm in section 3.1.

In Figure 4.1, we show contours of the BDS, CLAW, and PPM2 solutions at 256^3 resolution. Figures 4.1(a)–4.1(c) show contours at 0.1, 0.5, and 0.9. We see that the BDS contours are tighter than the CLAW and PPM2 contours, indicating less smearing of the front. We also note some slight shape distortion in the PPM2 method. In Figure 4.1(d), we have added additional contours to the PPM2 solution indicating regions where the method undershoots (in blue) and overshoots (in red).

In Table 4.2, we report the L^1 error and convergence rates for the spherical step function for each method. Due to the discontinuous initial data, all of the methods show less than first-order convergence. The BDS solution has the lowest L^1 error of all the methods, such that the error has been reduced by 23% as compared to the CLAW

TABLE 4.2
L^1 error and convergence rates for advection of a spherical step function with $\mathbf{u} = (1, 0.5, 0.25)$.

Scheme	64^3 error	$p^{64/128}$	128^3 error	$p^{128/256}$	256^3 error
BDS	1.26e-03	0.74	7.54e-04	0.70	4.63e-04
PPM2	1.94e-03	0.58	1.30e-03	0.55	8.86e-04
PPM1	1.92e-03	0.58	1.28e-03	0.56	8.67e-04
PLM	1.92e-03	0.57	1.29e-03	0.57	8.70e-04
CLAW	1.64e-03	0.72	9.93e-04	0.68	6.20e-04
BDS, no limiting	1.56e-03	0.69	9.64e-04	0.66	6.11e-04
PPM, no limiting	2.27e-03	0.61	1.49e-03	0.52	1.04e-03
PLM, no limiting	2.25e-03	0.57	1.52e-03	0.60	1.00e-03
CLAW, no limiting	3.07e-03	0.59	2.04e-03	0.56	1.38e-03

TABLE 4.3
Minimum and maximum scalar values for advection of a Gaussian profile at $t = 1$ *with* $\mathbf{u} = (1, 0.5, 0.25)$.

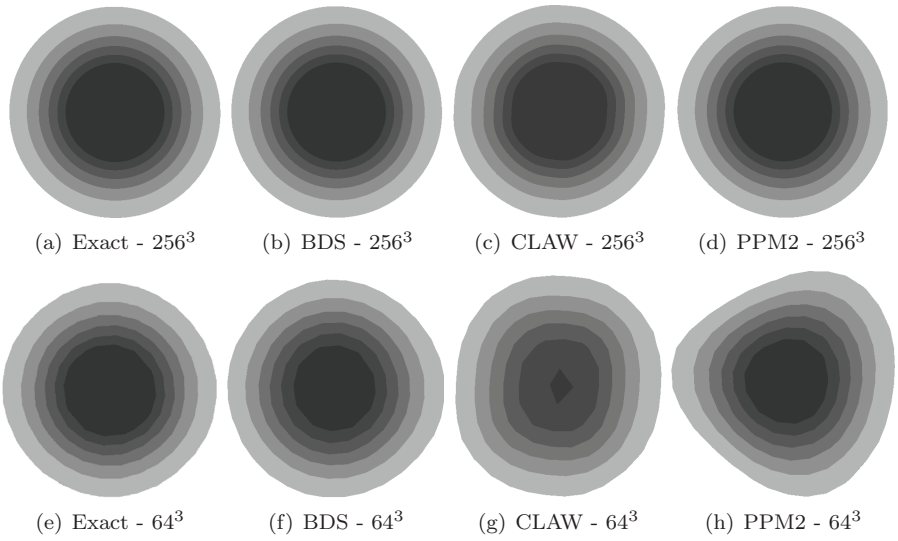
Scheme	64^3 min	64^3 max	128^3 min	128^3 max	256^3 min	256^3 max
Exact	0.00e-00	0.93000	0.00e-00	0.98190	0.00e-00	0.99544
BDS	−2.47e-11	0.74179	−4.95e-11	0.91376	−1.10e-10	0.97265
PPM2	−3.19e-02	0.84358	−1.40e-04	0.97404	−6.34e-11	0.99497
PPM1	−3.73e-02	0.74702	−3.91e-04	0.91641	−4.95e-10	0.97272
PLM	−3.01e-02	0.71146	−2.44e-04	0.89800	−6.07e-10	0.96421
CLAW	−1.86e-06	0.61655	−6.51e-15	0.83346	−1.19e-20	0.93453

solution in the coarsest simulation and by 25% in the finest simulation. Compared to the PPM2 solution, the BDS method has 35% less error in the coarsest simulation and 48% less error in the finest simulation. Disabling the limiters does lead to a minor increase in error for all the schemes (particularly for CLAW, where the error increases by approximately a factor of two) but does not significantly change the convergence rate of any of the methods. In summary, for this test we conclude

- BDS satisfies a maximum principle, whereas CLAW exhibits mild over/undershoot, and the PPM methods exhibit significant over/undershoot.
- BDS has the best shape preservation.
- All methods are less than first-order, with BDS having the lowest error, followed by CLAW and then by the PPM methods.

4.1.2. Constant velocity advection: Gaussian. In Table 4.3, we report the minimum and maximum scalar values of the computed solutions for the Gaussian profile at $t = 1$. We note that the exact analytical maximum is 1.0, but due to numerical quadrature the exact numerical maximum is slightly less than 1.0, as indicated in the table. Even for smooth initial data in this test, the PPM2, PPM1, and PLM algorithms do not satisfy a maximum principle. At the coarsest resolution, these methods exhibit approximately 3% undershoot, which tends to zero with increasing resolution. We also note that at the coarsest resolution, the CLAW method undershoots slightly. By contrast, the undershoot in the BDS algorithm is $\mathcal{O}(10^{-10})$ at any resolution, which again is due to roundoff error and the choice of ϵ in the slope limiting. We note that the PPM2 algorithm preserves the peak better than the BDS method, and the BDS method preserves the peak better than the CLAW method. It is not unexpected that the PPM2 algorithm retains the peak value best since it is specifically designed to avoid limiting at smooth extrema.

In Figure 4.2, we show contours of the exact, BDS, CLAW, and PPM2 solutions at 256^3 and 64^3 resolution. For the 256^3 simulations, all methods do a good job at



(a) Exact - 256^3 (b) BDS - 256^3 (c) CLAW - 256^3 (d) PPM2 - 256^3

(e) Exact - 64^3 (f) BDS - 64^3 (g) CLAW - 64^3 (h) PPM2 - 64^3

FIG. 4.2. *Results for advection of a Gaussian profile with* $\mathbf{u} = (1, 0.5, 0.25)$. *The contours lie at scalar values of* 0.1 *through* 0.6. 4.2(a)–4.2(d): 256^3 *resolution.* 4.2(e)–4.2(h): 64^3 *resolution.*

TABLE 4.4
L^1 *error and convergence rates for advection of a Gaussian profile with* $\mathbf{u} = (1, 0.5, 0.25)$.

Scheme	64^3 error	$p^{64/128}$	128^3 error	$p^{128/256}$	256^3 error
BDS	9.21e-05	2.35	1.81e-05	2.10	4.23e-06
PPM2	2.56e-04	1.97	6.55e-05	1.99	1.65e-05
PPM1	2.62e-04	1.90	7.00e-05	1.98	1.77e-05
PLM	2.70e-04	1.86	7.43e-05	1.96	1.91e-05
CLAW	1.80e-04	1.83	5.08e-05	1.81	1.44e-05
BDS, no limiting	8.46e-05	2.27	1.76e-05	2.07	4.18e-06
PPM, no limiting	2.57e-04	1.97	6.55e-05	1.99	1.65e-05
PLM, no limiting	2.67e-04	1.97	6.81e-05	1.99	1.72e-05
CLAW, no limiting	4.68e-04	1.94	1.22e-04	2.00	3.06e-05

preserving the shape of the Gaussian, with only a slight deformation seen in the CLAW simulation. For the 64^3 simulations, the CLAW and PPM2 contours are clearly more distorted than the BDS contours, and the smearing of the peak is very noticeable in the CLAW method.

In Table 4.4, we report the L^1 error and convergence rates for the Gaussian profile. Each of the methods is second-order accurate and the BDS algorithm has the lowest error of all of the methods. Specifically, the BDS algorithm reduces the L^1 error by 49% (for the 64^3 case) and 71% (for the 256^3 case) compared to CLAW. Also, the BDS algorithm reduces the L^1 error by 64% (for the 64^3 case) and 74% (for the 256^3 case) compared to PPM2. Although BDS does not do as well as PPM2 in preserving the peak, the reduced distortion of BDS leads to an overall reduction in error. Table 4.4 also shows that the presence of limiters does little to affect the error or the convergence rates for any of the algorithms (with the exception being the CLAW algorithm, where the error increases by approximately a factor of two). In Table 4.5, we report the L^2 error and convergence rates for the Gaussian profile. We see very similar behavior as in the L^1 error case, where each of the methods is second-order accurate and the BDS

TABLE 4.5
L^2 error and convergence rates for advection of a Gaussian profile with $\mathbf{u} = (1, 0.5, 0.25)$.

Scheme	64^3 error	$p^{64/128}$	128^3 error	$p^{128/256}$	256^3 error
BDS	1.82e-03	2.40	3.46e-04	2.22	7.43e-05
PPM2	3.40e-03	1.82	9.65e-04	1.97	2.47e-04
PPM1	3.58e-03	1.73	1.08e-03	1.92	2.86e-04
PLM	3.82e-03	1.67	1.20e-03	1.87	3.29e-04
CLAW	3.80e-03	1.66	1.20e-03	1.73	3.61e-04
BDS, no limiting	1.38e-03	2.23	2.94e-04	2.08	6.93e-05
PPM, no limiting	3.40e-03	1.82	9.65e-04	1.97	2.47e-04
PLM, no limiting	3.64e-03	1.81	1.04e-03	1.96	2.68e-04
CLAW, no limiting	6.07e-03	1.67	1.91e-03	1.95	4.96e-04

algorithm has the lowest error of all of the methods. Specifically, the BDS algorithm reduces the L^2 error by 52% (for the 64^3 case) and 79% (for the 256^3 case) compared to CLAW. Also, the BDS algorithm reduces the L^2 error by 46% (for the 64^3 case) and 70% (for the 256^3 case) compared to PPM2. In summary, for this test case we conclude

- BDS satisfies a maximum principle, whereas the other methods show mild undershoot at low resolution.
- The PPM methods preserve the peak value the best, followed by BDS and then CLAW.
- BDS has the best shape preservation, particularly at low resolution.
- All methods are second-order, with BDS having the lowest error, followed by either CLAW or the PPM methods (depending on the choice of norm).

4.2. Minimum and maximum values as a function of angle. Here we examine the overshoot properties of the PPM2, CLAW, and BDS methods for a spherical step function advected at a constant velocity at several different angles. Specifically, we set $u = 1$ and vary v and w. Here we choose to compare against PPM2 rather than PPM1 or PLM since PPM2 is considered to be more state of the art. Table 4.6 shows the minimum and maximum scalar values at $t = 1$ for a domain with 128^3 cells. For any angle, the BDS method does not overshoot, retaining the peak value within 2e-05, while the undershoot is $\mathcal{O}(10^{-12})$. PPM2 violates a maximum principle for every case, and the undershoot is most severe for $\mathbf{u} = (1, 0.75, 0.50)$ (27%) and the overshoot is the most severe for $\mathbf{u} = (1, 0.5, 0.5)$ (35%). PPM performs reasonably well for the $\mathbf{u} = (1, 0, 0)$ case, with an overshoot of only 0.1%, which is expected since this corresponds to a one-dimensional problem. CLAW performs very well in the $\mathbf{u} = (1, 0, 0)$ case with no overshoot, but violates a maximum principle for all other cases. The overshoot/undershoot is modest compared to PPM2, with the most severe case occurring for $\mathbf{u} = (1, 0.25, 0.25)$ (0.4%).

4.3. Variable velocity advection. We advect the same step function and Gaussian profiles from section 4.1, but now we use a velocity field that varies in space, $\mathbf{u} = [1, 0.5 + 0.5\sin(2\pi x), 0.25 + 0.25\cos(2\pi x)]$. Thus, we are no longer guaranteed to satisfy a maximum principle. We use this test to demonstrate that, even for nonconstant velocity fields, the BDS method exhibits significantly less overshoot than the other methods, has reduced error, and is second-order accurate for smooth data.

4.3.1. Variable velocity advection: Step function. In Table 4.7 we report the minimum and maximum scalar values of the exact and computed solutions for the

TABLE 4.6

*Minimum and maximum scalar values for advection of a spherical step function with a constant velocity field at several different angles at $t = 1$ using a domain with 128^3 cells. We use $u = 1$ and (v, w) as given below. * The actual maxima is $1-\epsilon$, where $\epsilon <$ 5e-06, and thus there is no overshoot.*

(v, w)	PPM2 min/max	CLAW min/max	BDS min/max
(0.00,0.00)	0.00e-00 / 1.00143	0.00e-00 / 1.00000	−3.02e-12 / 1.00000*
(0.25,0.00)	−2.11e-01 / 1.18439	−1.21e-03 / 1.00055	−4.22e-12 / 1.00000*
(0.25,0.25)	−2.45e-01 / 1.24855	−3.92e-03 / 1.00396	−3.22e-12 / 1.00000*
(0.50,0.00)	−2.45e-01 / 1.22161	−5.35e-04 / 1.00003	−3.23e-12 / 0.99999
(0.50,0.25)	−2.58e-01 / 1.29997	−3.04e-03 / 1.00211	−3.11e-12 / 0.99999
(0.50,0.50)	−2.63e-01 / 1.34643	−2.13e-03 / 1.00064	−2.68e-12 / 0.99998
4 (0.75,0.00)	−2.35e-01 / 1.19973	−9.28e-04 / 1.00024	−3.91e-12 / 1.00000*
(0.75,0.25)	−2.57e-01 / 1.26605	−3.64e-03 / 1.00326	−3.14e-12 / 0.99999
(0.75,0.50)	−2.68e-01 / 1.31557	−2.67e-03 / 1.00136	−3.00e-12 / 0.99999
(0.75,0.75)	−2.51e-01 / 1.28507	−3.22e-03 / 1.00235	−3.00e-12 / 0.99999
(1.00,0.00)	−1.90e-01 / 1.11843	−1.18e-03 / 1.00073	−2.95e-12 / 1.00000*
(1.00,0.25)	−2.37e-01 / 1.22304	−3.46e-03 / 1.00386	−3.02e-12 / 1.00000*
(1.00,0.50)	−2.57e-01 / 1.26227	−2.65e-03 / 1.00205	−2.55e-12 / 0.99999
(1.00,0.75)	−2.56e-01 / 1.24886	−3.14e-03 / 1.00290	−2.85e-12 / 1.00000*
(1.00,1.00)	−1.74e-01 / 1.15754	−3.03e-03 / 1.00383	−2.78e-12 / 1.00000*

TABLE 4.7

*Minimum and maximum scalar values for advection of a spherical step function at $t = 1$ with $\mathbf{u} = [1, 0.5 + 0.5\sin(2\pi x), 0.25 + 0.25\cos(2\pi x)]$. * The maximum for the 256^3 BDS solution is $1 + \epsilon$, where $\epsilon = 3.99e$-11.*

Scheme	64^3 min	64^3 max	128^3 min	128^3 max	256^3 min	256^3 max
Exact	0.00e-00	1.00000	0.00e-00	1.00000	0.00e-00	1.00000
BDS	−1.25e-11	0.99887	−2.00e-11	0.99999	−4.76e-11	1.00000*
PPM2	−1.88e-01	1.22082	−2.27e-01	1.19627	−2.45e-01	1.20707
PPM1	−1.99e-01	1.19684	−2.32e-01	1.20626	−2.44e-01	1.21534
PLM	−1.82e-01	1.19884	−2.23e-01	1.20309	−2.42e-01	1.20577
CLAW	−1.40e-03	1.00001	−1.69e-03	1.00139	−1.83e-03	1.00177

spherical step function at $t = 1$. Similar to the constant velocity case, PPM2, PPM1, and PLM show considerable overshoot that does not improve with resolution. The overshoot for the CLAW method is modest and becomes a bit worse with resolution (up to 0.2% on a 256^3 grid). The overshoot/undershoot for the PPM2 simulation is 19%/22% in the coarsest simulation and 25%/21% in the finest simulation. The undershoot for the BDS simulation is $\mathcal{O}(10^{-11})$ at all resolutions and the overshoot at the finest resolution is $\mathcal{O}(10^{-11})$, which again is due to roundoff error and the choice of ϵ in the slope limiting. The BDS algorithm still does a good job at retaining the peak value, with a maximum of 0.99887 at the coarsest resolution.

In Table 4.8, we report the L^1 error and convergence rates. As in the constant velocity case, the convergence rate for each algorithm is less than first order. The BDS error is 20% less than the CLAW error at the coarsest resolution, and 23% less at the finest resolution. Also, the BDS error is 28% less than PPM2 at the coarsest resolution and 43% less at the finest resolution. Again, disabling the limiters does not significantly affect the error or convergence of the methods (with the exception being CLAW). In summary, this test confirms the conclusions from the constant velocity advection test case in section 4.1.1.

4.3.2. Variable velocity advection: Gaussian. In Table 4.9 we report the minimum and maximum scalar values of the exact and computed solutions for the

TABLE 4.8

L^1 error and convergence rates for advection of a spherical step function with $\mathbf{u} = [1, 0.5 + 0.5\sin(2\pi x), 0.25 + 0.25\cos(2\pi x)]$.

Scheme	64^3 error	$p^{64/128}$	128^3 error	$p^{128/256}$	256^3 error
BDS	1.26e-03	0.74	7.52e-04	0.71	4.61e-04
PPM2	1.74e-03	0.56	1.18e-03	0.53	8.15e-04
PPM1	1.76e-03	0.55	1.20e-03	0.55	8.17e-04
PLM	1.80e-03	0.56	1.22e-03	0.55	8.36e-04
CLAW	1.57e-03	0.71	9.58e-04	0.67	6.01e-04
BDS, no limiting	1.53e-03	0.69	9.50e-04	0.66	6.00e-04
PPM, no limiting	1.98e-03	0.57	1.33e-03	0.56	9.05e-04
PLM, no limiting	2.11e-03	0.55	1.44e-03	0.53	9.94e-04
CLAW, no limiting	2.78e-03	0.56	1.89e-03	0.56	1.28e-03

TABLE 4.9

Minimum and maximum scalar values for advection of a Gaussian profile at $t = 1$ *with* $\mathbf{u} = [1, 0.5 + 0.5\sin(2\pi x), 0.25 + 0.25\cos(2\pi x)]$.

Scheme	64^3 min	64^3 max	128^3 min	128^3 max	256^3 min	256^3 max
Exact	0.00e-00	0.93000	0.00e-00	0.98190	0.00e-00	0.99544
BDS	−2.57e-11	0.74172	−5.79e-11	0.91194	−1.16e-10	0.97175
PPM2	−3.58e-02	0.84890	−5.35e-04	0.97406	−1.89e-09	0.99493
PPM1	−4.12e-02	0.77632	−7.45e-04	0.93978	−3.86e-08	0.98471
PLM	−4.26e-02	0.74555	−1.36e-03	0.93019	−2.16e-09	0.98171
CLAW	−3.24e-06	0.64455	−1.46e-16	0.84826	−2.01e-21	0.94080

TABLE 4.10

L^1 error and convergence rates for advection of a Gaussian profile with $\mathbf{u} = [1, 0.5 + 0.5\sin(2\pi x), 0.25 + 0.25\cos(2\pi x)]$.

Scheme	64^3 error	$p^{64/128}$	128^3 error	$p^{128/256}$	256^3 error
BDS	9.12e-05	2.31	1.84e-05	2.11	4.26e-06
PPM2	2.27e-04	1.88	6.18e-05	1.98	1.57e-05
PPM1	2.33e-04	1.89	6.30e-05	1.97	1.61e-05
PLM	2.54e-04	1.86	7.01e-05	1.95	1.82e-05
CLAW	1.57e-04	1.95	4.06e-05	1.91	1.08e-05
BDS, no limiting	8.87e-05	2.30	1.80e-05	2.09	4.22e-06
PPM, no limiting	2.30e-04	1.90	6.18e-05	1.98	1.57e-05
PLM, no limiting	2.64e-04	1.89	7.11e-05	1.98	1.80e-05
CLAW, no limiting	3.95e-04	1.86	1.09e-04	1.92	2.89e-05

Gaussian profile at $t = 1$. The undershoot in the BDS algorithm is in the range of numerical roundoff. There is only minor undershoot in the CLAW algorithm at the coarsest resolution. The undershoot for the remaining methods is 4% at the coarsest resolution and tends to zero with increasing resolution. As in the constant velocity case, we observe that the PPM2 solution preserves the peak value better than the BDS algorithm, and the BDS algorithm preserves the peak better than the CLAW algorithm.

In Table 4.10, we report the L^1 error and convergence rates. Each algorithm is second order accurate, but the error of the BDS method is lower than the error of the other methods, with a 42% decrease as compared to CLAW at the coarsest resolution, and 61% decrease at the finest resolution. Also, the BDS error is 60% less than PPM2 at the coarsest resolution and 73% less at the finest resolution. Again, disabling the limiters does not significantly affect the error or convergence of the methods (with

TABLE 4.11

*Minimum and maximum scalar values for advection of a spherical step function at $t = 1$ in a cylindrical shear layer. *The actual maxima for the 128^3 and 256^3 BDS simulations are $1 + \epsilon$, where $\epsilon = 3.17e\text{-}09$ and $\epsilon = 2.24e\text{-}07$, respectively.*

Scheme	64^3 min	64^3 max	128^3 min	128^3 max	256^3 min	256^3 max
Exact	0.00e-00	1.00000	0.00e-00	1.00000	0.00e-00	1.00000
BDS	−4.26e-10	0.99999	−3.69e-09	1.00000*	−7.49e-11	1.00000*
PPM2	−1.16e-01	1.10916	−1.83e-01	1.15142	−2.60e-01	1.20912

the exception being CLAW). In summary, this test confirms the conclusions from the constant velocity advection test case in section 4.1.2.

4.4. Cylindrical shear layer. We now test the advection of a spherical step function tracer profile in a three-dimensional flow that varies in space and time to ensure that our algorithm satisfies a maximum principle. We use a constant density of $\rho = 1$ and let the velocity evolve using the algorithm of Almgren et al. [2], which is an incompressible Navier–Stokes solver based on the projection method of Bell, Colella, and Glaz [4]. In this approach, velocities are guaranteed to be divergence-free to the tolerance of the linear solvers, so the exact solution should not have significant new maxima or minima relative to the initial data. The initial velocity field is given by a cylindrical shear layer subject to a small transverse perturbation in the form of a wide, flat Gaussian bump as originally used in [7]. We add an additional transverse motion in the form of an initially constant y-velocity, so that the initial velocity is completely specified as $u = \tanh\{[0.15 - \sqrt{(y-0.5)^2 + (z-0.5)^2}]/0.333\}$, $v = 0.25$, and $w = 0.05\exp\{-15[(x-0.5)^2 + (y-0.5)^2]\}$. The initial tracer profile is a spherical step function

$$s_0(x,y,z) = \begin{cases} 1, & r \le 0.1, \\ 0, & r > 0.1; \end{cases} \qquad r = \sqrt{(x-0.375)^2 + (y-0.5)^2 + (z-0.5)^2}. \tag{4.5}$$

As before, we run the simulation to $t = 1$ using a CFL number of 0.9 at three different resolutions. Here we compare BDS to PPM2 (the CLAWPACK libraries are not directly compatible with our incompressible Navier–Stokes libraries). In Table 4.11, we report the minimum and maximum scalar values of our computed solutions at $t = 1$. We do not have an exact solution available, but since the flow is divergence-free, we know that the exact minimum and maximum bounds are 0 and 1 at any later time. The BDS algorithm captures the maximum remarkably well, even for the coarsest resolution where a peak value of 0.99999 is observed. The BDS algorithm undershoots by at most $\mathcal{O}(10^{-9})$ and overshoots by $\mathcal{O}(10^{-7})$ at the finest resolution. The overshoot behavior is slightly worse than we observed our previous test problems, but is reflective of the tolerances in the multigrid solver (additional testing suggests that changes to the solver tolerance lead to commensurate changes in the undershoot and overshoot). By contrast, the PPM2 algorithm has a severe overshoot/undershoot of 12%/11% at the coarsest resolution and 26%/21% at the finest resolution. Figure 4.3 shows contours of the computed PPM2 solutions. We have highlighted the regions where the PPM2 solution violates a maximum principle by using blue and red contours. Note that the overshoot and undershoot occur over large, continuous regions.

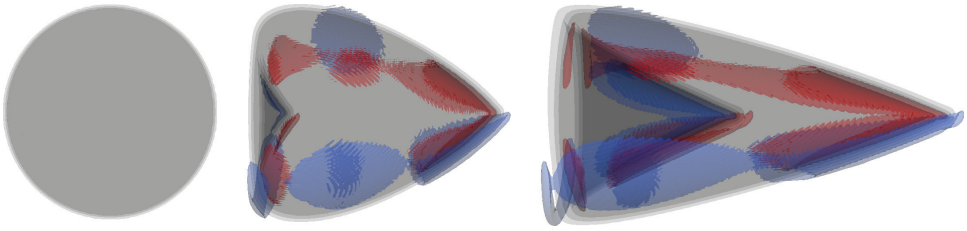


FIG. 4.3. *Results for advection of a spherical step function in a cylindrical shear layer using* 256^3 *cells and the PPM2 algorithm at (left)* $t = 0$*, (middle)* $t = 0.3$*, and (right)* $t = 1$*. The gray contours lie at scalar values of* 0.1*,* 0.5*, and* 0.9*. The blue contour shows where the method undershoots, encapsulating regions where* $s < -0.001$*, and the red contour show where the method overshoots, encapsulating regions where* $s > 1.001$*. There are large continuous regions of both undershoot and overshoot. The BDS results (not pictured) look very similar in overall shape, except that there are no regions of overshoot/undershoot.*

5. Conclusions. We have developed an unsplit, three-dimensional Godunov-type method for linear advection based on an extension of an algorithm of Bell, Dawson, and Shubin. The method is based on constructing a limited trilinear approximation of the solution on each grid cell. Fluxes for a conservative update are constructed by integrating this trilinear approximation over the characteristic domain of dependence of each face in three dimensions. The resulting method has the property that for constant coefficient advection it is analytically equivalent to exactly advecting the reconstructed profiles and averaging them onto the grid at the new time. As a consequence of this observation, the method satisfies a discrete maximum principle for constant coefficient advection. Comparison of the new method with standard unsplit approaches for general systems of conservation laws show that the new method has lower errors and better shape-preservation properties than the more general schemes and avoids the significant undershoot and overshoot that can plague those schemes for non-grid–aligned advection.

The extension of the method presented here to a more general scalar conservation law as done in [5] is straightforward. Of more interest would be the inclusion of quadratic terms in the approximation, similar to the PPM methods. Including quadratic terms in the present framework would allow us to improve the accuracy of the scheme, particularly for smooth flows, without introducing issues of distortion and the undershoot/overshoot observed in the more general methods. A first step in that direction has been taken in May et al. [24], which considers a quadratic version of the BDS scheme in two dimensions. Additional improvements to the scheme could be made by introducing new limiting ideas discussed in the introduction that avoid limiting at smooth extrema. These extensions will be discussed in future work. Finally, we plan to incorporate the three-dimensional BDS scheme into large-scale applications codes for combustion, astrophysics and subsurface flow in which advection by a specified velocity field places a central role.

Acknowledgments. The simulations in this paper used resources at National Energy Research Scientific Computing Center (NERSC), which is supported by the Office of Science of the U.S. Department of Energy under Contract DE-AC02-05CH11231 and at the Oak Ridge Leadership Computational Facility (OLCF), which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725. Visualization was performed using the VisIt visualization software [35].

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), pp. 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), pp. 1–46.

[3] J. Balbás and X. Qian, *Non-oscillatory central scheme for 3d hyperbolic conservation laws*, Proc. Sympos. Appl. Math., 67 (2009), pp. 389–398.

[4] J. B. Bell, P. Colella, and H. M. Glaz, *A second-order projection method for the incompressible Navier-Stokes equations*, J. Comput. Phys., 85 (1989), pp. 257–283.

[5] 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), pp. 1–24.

[6] 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), pp. 677–694.

[7] J. B. Bell and D. L. Marcus, *A second-order projection method for variable-density flows*, J. Comput. Phys., 101 (1992), pp. 334–348.

[8] S. J. Billett and E. F. Toro, *On WAF-type schemes for multidimensional hyperbolic conservation laws*, J. Comput. Phys., 130 (1997), pp. 1–24.

[9] B. Cockburn and C.-W. Shu, *Foreward for the special issue on discontinuous Galerkin methods*, J. Sci. Comput., 22-23 (2005), pp. 1–3.

[10] P. Colella and M. D. Sekora, *A limiter for PPM that preserves accuracy at smooth extrema*, J. Comput. Phys., 227 (2008), pp. 7069–7076.

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

[12] P. Colella, *Multidimensional upwind methods for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 171–200.

[13] C. Dawson, *Foreword for the special issue on discontinuous Galerkin methods*, Comput. Methods Appl. Mech. Engrg., 195 (2006), p. 3183.

[14] M. S. Day and J. B. Bell, *Numerical simulation of laminar reacting flows with complex chemistry*, Combust. Theory Modelling, 4 (2000), pp. 535–556.

[15] Y. Jinyun, *Symmetric Gaussian quadrature formulae for tetrahedronal regions*, Comput. Methods Appl. Mech. Engrg., 43 (1984), pp. 349–353.

[16] 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), pp. 683–729.

[17] J. O. Langseth and R. J. LeVeque, *A wave propagation method for three-dimensional hyperbolic conservation laws*, J. Comput. Phys., 165 (2000), pp. 126–166.

[18] R. J. LeVeque, *High-resolution conservative algorithms for advection in incompressible flows*, SIAM J. Numer. Anal., 33 (1996), pp. 627–665.

[19] R. J. LeVeque, *Wave propagation algorithms for multi-dimensional hyperbolic systems*, J. Comput. Phys., 131 (1997), pp. 327–353.

[20] R. J. LeVeque, *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press, Cambridge, UK, 2002.

[21] M. Lukáčová-Medvid'ová, K. W. Morton, and G. Warnecke, *Finite volume evolution Galerkin methods for hyperbolic systems*, SIAM J. Sci. Comput., 26 (2004), pp. 1–30.

[22] M. Lukáčová-Medvid'ová, J. Saibertová, and G. Warnecke, *Finite volume evolution Galerkin methods for nonlinear hyperbolic systems*, J. Comput. Phys., 183 (2002), pp. 533–562.

[23] M. Lukáčová-Medvid'ová, G. Warnecke, and Y. Zahaykah, *Finite volume evolution Galerkin (FVEG) methods for three-dimensional wave equation system*, Appl. Numer. Math., 57 (2007), pp. 1050–1064.

[24] S. May, A. J. Nonaka, A. S. Almgren, and J. B. Bell, *An unsplit, higher order Godunov method using quadratic reconstruction for advection in two dimensions*, Commun. Appl. Math. Comput. Sci., submitted.

[25] P. McCorquodale and P. Colella, *A high-order finite-volume method for hyperbolic conservation laws on locally-refined grids*, Commun. Appl. Math. Comput. Sci., submitted.

[26] G. H. Miller and P. Colella, *A conservative three-dimensional Eulerian method for coupled solid-fluid shock capturing*, J. Comput. Phys., 183 (2002), pp. 26–82.

[27] S. Noelle, *The MoT-ICE: A new high resolution wave-propagation algoirthm for multidimensional systems of conservation laws based on Fey's method of transport*, J. Comput. Phys., 164 (2000), pp. 283–334.

[28] 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. Supplement, 188 (2010), pp. 358–383.

[29] G. S. H. Pau, A. S. Almgren, J. B. Bell, and M. J. Lijewki, *A parallel second-order adaptive mesh algorithm for incompressible flow in porous media*, Philos. Trans. R. Soc. A, 367 (2009), pp. 4633–4654.

[30] J. Saltzman, *An unsplit 3D upwind method for hyperbolic conservation laws*, J. Comput. Phys., 115 (1994), pp. 153–168.

[31] C.-W. Shu, *High order weighted essentially nonoscillatory schemes for convection dominated problems*, SIAM Rev., 51 (2009), pp. 82–126.

[32] P. K. Smolarkiewicz and L. G. Margolin, *MPDATA: A finite difference solver for geophysical flows*, J. Comput. Phys., 140 (1998), pp. 459–480.

[33] P. K. Smolarkiewicz, *Multidimensional positive definite advection transport algorithm: An overview*, Internat. J. Numer. Methods Fluids, 50 (2006), pp. 1123–1144.

[34] E. F. Toro and V. A. Titarev, *ADER scheme for scalar non-linear hyperbolic conservation laws with source terms in three-space dimensions*, J. Comput. Phys., 202 (2005), pp. 196–215.

[35] VisIt User's Manual. https://wci.llnl.gov/codes/visit/home.html, 2010.