# A high-order finite-volume method for hyperbolic conservation laws on locally-refined grids

Peter McCorquodale and Phillip Colella

Computational Research Division
Lawrence Berkeley National Laboratory
Berkeley, CA 94720

January 2011

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

# A high-order finite-volume method for hyperbolic conservation laws on locally-refined grids *

Peter McCorquodale and Phillip Colella
Computational Research Division
Lawrence Berkeley National Laboratory

January 28, 2011

**Abstract**

We present a fourth-order accurate finite-volume method for solving time-dependent hyperbolic systems of conservation laws on Cartesian grids with multiple levels of refinement. The underlying method is a generalization of that in [5] to nonlinear systems, and is based on using fourth-order accurate quadratures for computing fluxes on faces, combined with fourth-order accurate Runge–Kutta discretization in time. To interpolate boundary conditions at refinement boundaries, we interpolate in time in a manner consistent with the individual stages of the Runge–Kutta method, and interpolate in space by solving a least-squares problem over a neighborhood of each target cell for the coefficients of a cubic polynomial. The method also uses a variation on the extremum-preserving limiter in [8], as well as slope flattening and a fourth-order accurate artificial viscosity for strong shocks. We show that the resulting method is fourth-order accurate for smooth solutions, and is robust in the presence of complex combinations of shocks and smooth flows.

## 1 High-Order Finite-Volume Methods

In the finite-volume approach, the spatial domain in $\mathbb{R}^{\mathbf{D}}$ is discretized as a union of rectangular control volumes that covers the spatial domain. For Cartesian-grid finite-volume methods, a control volume $V_{\boldsymbol{i}}$ takes the form

$$V_{\boldsymbol{i}} = [\boldsymbol{i}h, (\boldsymbol{i}+\boldsymbol{u})h],\ \boldsymbol{i} \in \mathbb{Z}^{\mathbf{D}},\ \boldsymbol{u} = (1, 1, \ldots, 1),$$

where $h$ is the grid spacing.

*This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

A finite-volume discretization of a partial differential equation is based on averaging that equation over control volumes, applying the divergence theorem to replace volume integrals by integrals over the boundary of the control volume, and approximating the boundary integrals by quadratures. In this paper, we solve time-dependent problems that take the form of a conservation equation:

$$\frac{\partial U}{\partial t} + \nabla \cdot \vec{F}(U) = 0. \tag{1}$$

The discretized solution in space is the average of $U$ over a control volume,

$$\langle U \rangle_{\boldsymbol{i}}(t) = \frac{1}{h^{\mathbf{D}}} \int_{V_{\boldsymbol{i}}} U(\boldsymbol{x}, t) d\boldsymbol{x}. \tag{2}$$

We can compute the evolution of the spatially discretized system by a method-of-lines approach,

$$\frac{d\langle U \rangle_{\boldsymbol{i}}}{dt} = -\frac{1}{h^{\mathbf{D}}} \int_{V_{\boldsymbol{i}}} \nabla \cdot \vec{F} d\boldsymbol{x} = -\frac{1}{h} \sum_d \langle F^d \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \langle F^d \rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d} \tag{3}$$

$$\langle F^d \rangle_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d} = \frac{1}{h^{\mathbf{D}-1}} \int_{A_d^{\pm}} F^d dA, \tag{4}$$

where $A_d^{\pm}$ are the high and low faces bounding $V_{\boldsymbol{i}}$ with normals pointing in the $\boldsymbol{e}^d$ direction. In this case, the finite-volume approach computes the average of the divergence of the fluxes on the left-hand side of (4) with the sum of the integrals over faces on the right-hand side, with the latter approximated using some quadrature rule. Such approximations are desirable because they lead to conserved quantities in the original PDE satisfying an analogous conservation law in the discretized system.

The approach we take in this paper is a generalization of the method in [5] to general nonlinear systems of hyperbolic conservation laws on locally-refined grids, using fourth-order quadratures in space to evaluate the flux integrals (4) on the faces [1], and a Runge–Kutta method for evolving the ODE (3). We use this approach as the starting point for a block-structured adaptive mesh refinement method along the lines of that in [3].

# 2 Single-level algorithm

## 2.1 Temporal Discretization

Given the solution $\langle U \rangle^n \approx \langle U \rangle(t^n)$, we compute a fourth-order temporal update to $\langle U \rangle^{n+1} \approx \langle U \rangle(t^n + \Delta t)$ using the classical fourth-order Runge–Kutta (RK4) scheme

on (1). We are solving the autonomous system of ODEs

$$\frac{d\langle U\rangle}{dt} = -D\cdot\vec{F}; \tag{5}$$
$$D\cdot\vec{F} = D\cdot\vec{F}(\langle U\rangle) = \frac{1}{h}\sum_d \langle F^d\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \langle F^d\rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}.$$

Then, starting with $\langle U\rangle^{(0)} = \langle U\rangle(t^n)$, set

$$k_1 = -D\cdot\vec{F}(\langle U\rangle^{(0)})\Delta t; \tag{6}$$
$$\langle U\rangle^{(1)} = \langle U\rangle^{(0)} + \frac{k_1}{2}; \qquad k_2 = -D\cdot\vec{F}(\langle U\rangle^{(1)})\Delta t; \tag{7}$$
$$\langle U\rangle^{(2)} = \langle U\rangle^{(0)} + \frac{k_2}{2}; \qquad k_3 = -D\cdot\vec{F}(\langle U\rangle^{(2)})\Delta t; \tag{8}$$
$$\langle U\rangle^{(3)} = \langle U\rangle^{(0)} + k_3; \qquad k_4 = -D\cdot\vec{F}(\langle U\rangle^{(3)})\Delta t. \tag{9}$$

Then to integrate one time step:

$$\langle U\rangle(t^n+\Delta t) = \langle U\rangle(t^n) + \frac{1}{6}(k_1+2k_2+2k_3+k_4) + O((\Delta t)^5). \tag{10}$$

The method given above is in conservation form. That is,

$$\langle U\rangle^{n+1} = \langle U\rangle^n - \frac{\Delta t}{h}\sum_d \langle F^d\rangle^{\rm tot}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \langle F^d\rangle^{\rm tot}_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}; \tag{11}$$
$$\langle F^d\rangle^{\rm tot}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \frac{1}{6}(\langle F^d\rangle^{(0)}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} + 2\langle F^d\rangle^{(1)}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} + 2\langle F^d\rangle^{(2)}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} + \langle F^d\rangle^{(3)}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d});$$
$$\langle F^d\rangle^{(s)}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \langle F^d(\langle U^{(s)}\rangle)\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}.$$

## 2.2 Spatial Discretization

To complete the definition of the single-level algorithm, we need to specify how to compute $\langle F^d\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ as a function of $\langle U\rangle$. Our approach is a generalization of that in [5] to the case of nonlinear systems of conservation laws. Following what often is done for second-order methods, we introduce a nonlinear change of variables $W = W(U)$. In the case of gas dynamics, this is the conversion from the conserved quantities mass, momentum, and energy, $U = (\rho, \rho\vec{u}, \rho E)$, to primitive variables $W = (\rho, \vec{u}, p)$, where $\rho$ is the gas density, $\vec{u}$ is the velocity vector, $E$ is the total energy per unit mass, and $p$ is the pressure. Typically, this transformation is done to simplify the limiting process, e.g. to permit the use of componentwise limiting. Some care is required in transforming from conservative to primitive variables in order to preserve fourth-order accuracy.

1. Convert from cell-averaged conserved variables to cell-averaged primitive variables, through cell-centered values, as follows.

   Calculate a fourth-order approximation to $U$ at cell centers:

$$U_{\boldsymbol{i}} = \langle U \rangle_{\boldsymbol{i}} - \frac{h^2}{24} \Delta^{(2)} \langle U \rangle_{\boldsymbol{i}} \tag{12}$$

   where $\Delta^{(2)}$ is the second-order accurate Laplacian

$$\Delta^{(2)} q_{\boldsymbol{i}} = \sum_d \frac{1}{h^2} (q_{\boldsymbol{i}-\boldsymbol{e}^d} - 2q_{\boldsymbol{i}} + q_{\boldsymbol{i}+\boldsymbol{e}^d}). \tag{13}$$

   Then convert to primitive variables:

$$W_{\boldsymbol{i}} = W(U_{\boldsymbol{i}}); \tag{14}$$

$$\overline{W}_{\boldsymbol{i}} = W(\langle U \rangle_{\boldsymbol{i}}). \tag{15}$$

   Calculate a fourth-order approximation to cell-averaged $W$:

$$\langle W \rangle_{\boldsymbol{i}} = W_{\boldsymbol{i}} + \frac{h^2}{24} \Delta^{(2)} \overline{W}_{\boldsymbol{i}}. \tag{16}$$

2. Interpolate from cell-averaged $W$ to fourth-order face-averaged $W$ over faces in dimension $d$, by:

$$\langle W \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \frac{7}{12} (\langle W \rangle_{\boldsymbol{i}} + \langle W \rangle_{\boldsymbol{i}+\boldsymbol{e}^d}) - \frac{1}{12} (\langle W \rangle_{\boldsymbol{i}-\boldsymbol{e}^d} + \langle W \rangle_{\boldsymbol{i}+2\boldsymbol{e}^d}) \tag{17}$$

   for every $d$-face $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d$.

3. Calculate face-centered $W$:

$$W^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \langle W \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \frac{h^2}{24} \Delta^{d,2} \langle W \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} \tag{18}$$

   where the transverse Laplacian is

$$\Delta^{d,2} q^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \sum_{d' \neq d} \frac{1}{h^2} (q^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d - \boldsymbol{e}^{d'}} - 2q^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} + q^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d + \boldsymbol{e}^{d'}}). \tag{19}$$

   Then compute the face-averaged fluxes in each dimension $d$:

$$\langle F^d \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = F^d(W^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}) + \frac{h^2}{24} \Delta^{d,2} F^d(\langle W \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}) \tag{20}$$

   for every $d$-face $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d$.

   Finally, the divergence is computed as in (3).

In Step 1 above, the Laplacian is applied in (16) to $\overline{W}_{\boldsymbol{i}}$ instead of $W_{\boldsymbol{i}}$ in order to minimize the size of stencil required; this substitution makes a difference of $O(h^4)$ in (16) because the discrete Laplacian of (13) is multiplied by $h^2$. Similarly, in Step 3, $\Delta^{d,2}$ is applied in (20) to $F^d(\langle W \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d})$ instead of to $F^d(W^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d})$, in order to minimize the size of the required stencil without loss of fourth-order accuracy.

## 2.3 Modified stencils near physical boundaries

Near physical boundaries, the stencils in the algorithm of section 2.2 are modified as follows.

In Step 1, in (13), when cell $\boldsymbol{i}$ is adjacent to the physical boundary in dimension $d$, we substitute for $\boldsymbol{i}$ the appropriate formula at $\boldsymbol{i} \pm \boldsymbol{e}^d$ so that all cells in the stencil are within the domain. Likewise, in Step 3, in (19), when face $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d$ is adjacent to the physical boundary in dimension $d'$, we substitute for $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d$ the appropriate formula at $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d \pm \boldsymbol{e}^{d'}$ so that all faces in the stencil are within the domain.

In Step 2, the stencil (17) is applied only when face $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d$ is separated by at least two cells from physical boundaries along dimension $d$. In other cases:

- If face $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d$ lies on, respectively, the low or high physical boundary in dimension $d$, then:

$$\begin{aligned} \langle W \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} =& \frac{1}{12}(25\langle W \rangle_{\boldsymbol{i}+\boldsymbol{e}^d} - 23\langle W \rangle_{\boldsymbol{i}+2\boldsymbol{e}^d} + 13\langle W \rangle_{\boldsymbol{i}+3\boldsymbol{e}^d} - 3\langle W \rangle_{\boldsymbol{i}+4\boldsymbol{e}^d}) \\ & \text{or} \\ \langle W \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} =& \frac{1}{12}(25\langle W \rangle_{\boldsymbol{i}} - 23\langle W \rangle_{\boldsymbol{i}-\boldsymbol{e}^d} + 13\langle W \rangle_{\boldsymbol{i}-2\boldsymbol{e}^d} - 3\langle W \rangle_{\boldsymbol{i}-3\boldsymbol{e}^d}). \end{aligned} \tag{21}$$

- If face $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d$ is separated by a single cell from, respectively, the low or high physical boundary in dimension $d$, then:

$$\begin{aligned} \langle W \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} =& \frac{1}{12}(3\langle W \rangle_{\boldsymbol{i}} + 13\langle W \rangle_{\boldsymbol{i}+\boldsymbol{e}^d} - 5\langle W \rangle_{\boldsymbol{i}+2\boldsymbol{e}^d} + \langle W \rangle_{\boldsymbol{i}+3\boldsymbol{e}^d}) \\ & \text{or} \\ \langle W \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} =& \frac{1}{12}(3\langle W \rangle_{\boldsymbol{i}+\boldsymbol{e}^d} + 13\langle W \rangle_{\boldsymbol{i}} - 5\langle W \rangle_{\boldsymbol{i}-\boldsymbol{e}^d} + \langle W \rangle_{\boldsymbol{i}-2\boldsymbol{e}^d}). \end{aligned} \tag{22}$$

## 2.4 Limiters

For method of lines such as the one employed here, limiters are used to suppress oscillations in the presence of shocks and underresolved gradients. In one approach, the limiter takes the form of replacing the single-valued solution value at cell faces by two values, each extrapolated from each adjacent cell. This pair of values is used to compute an upwind flux of some sort, such as one obtained by solving a Riemann problem. This is the type of limiter we employ here. We use a variant of the limiter proposed in [8], which is in turn a modification that preserves extrema of the limiter for the piecewise parabolic method (PPM) in [9]. We have modified this limiter in several ways. First, we have made a small change to the method in [8] for detecting extrema that to reduce sensitivity to roundoff error. Second we have modified the limiter to eliminate difficulty that arises in multidimensional problems. To illustrate this problem, consider a solution of the form $f(x, y) = x^3 - xy^2$. This function, for

fixed $y$, has two extrema as a function of $x$ located at $x = \pm y/\sqrt{3}$. It is not difficult to see that, for any fixed $h$, and all $y$ sufficiently small, but nonzero, the limiter in [8] will be activated at those extrema, thus reducing the accuracy of the method in a region where the function is manifestly smooth enough to be discretized accurately by our underlying fourth-order method. This leads to a failure to converge at fourth-order accuracy in max norm for smooth problems. In order to eliminate this difficulty, we change the criterion by which we decide to apply the limiter in [8] at extrema, so that it is not applied to solutions that are small perturbations of a cubic profile. Finally, we have found that, in introducing the above changes, the fundamental structure of the PPM limiter, at least for the fourth-order Runge–Kutta time discretization used here, introduces too much dissipation. The PPM limiter limits the solution in two parts of the algorithm. The first is in the construction of the single value at the face, which is limited to be within a range defined by the adjacent cell values. The second step in the limiter is based on limiting parabolic profiles in the two cells adjacent to the face, leading to a potentially double-valued solution at the face. We have found that, in the present setting, the initial limiting of the face values is redundant, and in fact introduces excessive dissipation for linear advection in one dimension, and that the limiting introduced in the second step is sufficient.

We make the following additions to Step 2 in the algorithm of section 2.2, to apply limiting to $\langle W \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$. For each component $w$ of the primitive variables $W$:

1. As described in section 2.4.1 below, extrapolate $\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ to the left and right of each $d$-face to obtain $\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{L}}$ and $\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{R}}$.

2. As described in section 2.5.1, apply slope flattening to the extrapolants $\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{L}}$ and $\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{R}}$.

3. Solve the Riemann problem on faces:
   From $\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{L}}$ and $\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{R}}$, get the new $\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$.

### 2.4.1 Limiter on extrapolants

We initialize left and right extrapolated values $\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\{\mathrm{L,R}\}}$ both to $\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ . At each cell $\boldsymbol{i}$, the limiter may change either $\langle w \rangle^d_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d,\mathrm{R}}$ or $\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{L}}$ or both.

The limiter for extrapolants $\langle w \rangle^d_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d,\mathrm{R}}$ and $\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{L}}$ depends on $\langle w \rangle$ at cells $\boldsymbol{i} - 3\boldsymbol{e}^d$ through $\boldsymbol{i} + 3\boldsymbol{e}^d$, as well as the face averages $\langle w \rangle^d_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d}$.

For each cell $\boldsymbol{i}$, set the differences

$$(\delta w)^{d,\mathrm{f},-}_{\boldsymbol{i}} = \langle w \rangle_{\boldsymbol{i}} - \langle w \rangle^d_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d};$$
$$(\delta w)^{d,\mathrm{f},+}_{\boldsymbol{i}} = \langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \langle w \rangle_{\boldsymbol{i}}.$$

Also set the differences

$$(\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{f}} = 6(\langle w\rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}^d - 2\langle w\rangle_{\boldsymbol{i}} + \langle w\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}^d);$$
$$(\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{c}} = \langle w\rangle_{\boldsymbol{i}-\boldsymbol{e}^d} - 2\langle w\rangle_{\boldsymbol{i}} + \langle w\rangle_{\boldsymbol{i}+\boldsymbol{e}^d};$$

which approximate the second derivative, multiplied by $h^2$, at the center of cell $\boldsymbol{i}$.

At each cell face, $\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d$, set the difference

$$(\delta^3 w)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}^d = (\delta^2 w)_{\boldsymbol{i}+\boldsymbol{e}^d}^{d,\mathrm{c}} - (\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{c}} \tag{23}$$

which approximates the third derivative, multiplied by $h^3$, at the center of face $\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d$.

1. If, at cell $\boldsymbol{i}$, either

$$(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},-} \cdot (\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},+} \le 0 \tag{24}$$

or

$$(\langle w\rangle_{\boldsymbol{i}}^d - \langle w\rangle_{\boldsymbol{i}-2\boldsymbol{e}^d}^d) \cdot (\langle w\rangle_{\boldsymbol{i}+2\boldsymbol{e}^d}^d - \langle w\rangle_{\boldsymbol{i}}^d) \le 0 \tag{25}$$

then $w$ has an extremum on cell $\boldsymbol{i}$ along dimension $d$, and we modify $\langle w\rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d,\mathrm{R}}^d$ and $\langle w\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{L}}^d$ as follows.

If $(\delta^2 w)_{\boldsymbol{i}-\boldsymbol{e}^d}^{d,\mathrm{c}}$, $(\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{c}}$, $(\delta^2 w)_{\boldsymbol{i}+\boldsymbol{e}^d}^{d,\mathrm{c}}$, and $(\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{f}}$, all have the same sign, $s = \pm 1$, then set

$$(\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{lim}} = s \cdot \min\{|(\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{f}}|, C_2|(\delta^2 w)_{\boldsymbol{i}-\boldsymbol{e}^d}^{d,\mathrm{c}}|, C_2|(\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{c}}|, C_2|(\delta^2 w)_{\boldsymbol{i}+\boldsymbol{e}^d}^{d,\mathrm{c}}|\} \tag{26}$$

where $C_2 = 1.25$. Otherwise, set $(\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{lim}} = 0$.

If $|(\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{f}}| \le 10^{-12} \cdot \max\{|w_{\boldsymbol{i}-2\boldsymbol{e}^d}|, |w_{\boldsymbol{i}-\boldsymbol{e}^d}|, |w_{\boldsymbol{i}}|, |w_{\boldsymbol{i}+\boldsymbol{e}^d}|, |w_{\boldsymbol{i}+2\boldsymbol{e}^d}|\}$, then set $\rho_{\boldsymbol{i}} = 0$. Otherwise, set

$$\rho_{\boldsymbol{i}} = \frac{(\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{lim}}}{(\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{f}}}. \tag{27}$$

If $\rho_{\boldsymbol{i}} \ge 1 - 10^{-12}$ then a limiter is not applied. Otherwise, to check whether to apply a limiter, set

$$(\delta^3 w)_{\boldsymbol{i}}^{d,\mathrm{min}} = \min\{(\delta^3 w)_{\boldsymbol{i}-\frac{3}{2}\boldsymbol{e}^d}^d, (\delta^3 w)_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}^d, (\delta^3 w)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}^d, (\delta^3 w)_{\boldsymbol{i}+\frac{3}{2}\boldsymbol{e}^d}^d\};$$
$$(\delta^3 w)_{\boldsymbol{i}}^{d,\mathrm{max}} = \max\{(\delta^3 w)_{\boldsymbol{i}-\frac{3}{2}\boldsymbol{e}^d}^d, (\delta^3 w)_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}^d, (\delta^3 w)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}^d, (\delta^3 w)_{\boldsymbol{i}+\frac{3}{2}\boldsymbol{e}^d}^d\}.$$

A necessary condition for applying a limiter in this case is

$$C_3 \cdot \max\{|(\delta^3 w)_{\boldsymbol{i}}^{d,\mathrm{min}}|, |(\delta^3 w)_{\boldsymbol{i}}^{d,\mathrm{max}}|\} \le (\delta^3 w)_{\boldsymbol{i}}^{d,\mathrm{max}} - (\delta^3 w)_{\boldsymbol{i}}^{d,\mathrm{min}} \tag{28}$$

where $C_3 = 0.1$. If (28) holds, then:

(a) If $(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},-} \cdot (\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},+} < 0$, set

$$\langle w \rangle^d_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d,\mathrm{R}} = \langle w \rangle^d_{\boldsymbol{i}} - \rho_{\boldsymbol{i}} (\delta^2 w)_{\boldsymbol{i}}^{d,\mathrm{f},-}; \tag{29}$$

$$\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{L}} = \langle w \rangle^d_{\boldsymbol{i}} + \rho_{\boldsymbol{i}} (\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},+}. \tag{30}$$

(b) Otherwise, if $|(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},-}| \geq 2|(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},+}|$, set

$$\langle w \rangle^d_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d,\mathrm{R}} = \langle w \rangle^d_{\boldsymbol{i}} - 2(1-\rho_{\boldsymbol{i}})(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},+} - \rho_{\boldsymbol{i}} (\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},-}. \tag{31}$$

(c) Otherwise, if $|(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},+}| \geq 2|(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},-}|$, set

$$\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{L}} = \langle w \rangle^d_{\boldsymbol{i}} + 2(1-\rho_{\boldsymbol{i}})(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},-} + \rho_{\boldsymbol{i}} (\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},+}. \tag{32}$$

2. For cell indices $\boldsymbol{i}$ on which neither (24) nor (25) holds, we modify the extrapolants under the following conditions:

(a) If $|(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},-}| \geq 2|(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},+}|$, set

$$\langle w \rangle^d_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d,\mathrm{R}} = \langle w \rangle^d_{\boldsymbol{i}} - 2(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},+}. \tag{33}$$

(b) If $|(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},+}| \geq 2|(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},-}|$, set

$$\langle w \rangle^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{L}} = \langle w \rangle^d_{\boldsymbol{i}} + 2(\delta w)_{\boldsymbol{i}}^{d,\mathrm{f},-}. \tag{34}$$

The differences between this extrapolant limiter and the one in section 2.4 of [8] are as follows:

- Condition (25) tests for differences two cells away, rather than only one cell away as in [8]. This change reduces the sensitivity of the limiter to roundoff error.
- The third-derivative condition (28) is new. The purpose of this condition is to avoid applying the limiter to small perturbations of a cubic.
- There are new, smoother formulae (31)–(32) to be used instead of (29)–(30) in case (25) holds but (24) does not.
- The second term in the right-hand side of equations (33) and (34) above replaces a more complicated formula with square roots, in equation (26) of [8].

## 2.5 Dissipation Mechanisms for Strong Shocks

For the case of gas dynamics, it necessary include additional dissipation mechanisms to suppress oscillations at strong shocks. We use the approach in [9, 10] of flattening the interpolated profiles at discontinuities that are too steep, as well as the introduction of a modest artificial viscosity term in the total flux.

### 2.5.1 Flattening

In the algorithm of section 2.2, at the end of step 2 we apply slope flattening to the extrapolants. The flattening coefficients are those from [10], where the flattening coefficent for cell $\boldsymbol{i}$ is $\eta_{\boldsymbol{i}}$ (calculated from $\overline{W}$). Then the extrapolants are modified as follows:

- Replace $\langle w\rangle^{d,\mathrm{PPM}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{L}}$ by $\eta_{\boldsymbol{i}}\langle w\rangle^{d,\mathrm{PPM}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d,\mathrm{L}} + (1-\eta_{\boldsymbol{i}})\langle w\rangle_{\boldsymbol{i}}$.
- Replace $\langle w\rangle^{d,\mathrm{PPM}}_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d,\mathrm{R}}$ by $\eta_{\boldsymbol{i}}\langle w\rangle^{d,\mathrm{PPM}}_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d,\mathrm{R}} + (1-\eta_{\boldsymbol{i}})\langle w\rangle_{\boldsymbol{i}}$.

### 2.5.2 Artificial viscosity

At the end of a full iteration in the algorithm of section 2.2, we apply an artificial viscosity to $\langle F^d\rangle^{tot}$ and $\langle U\rangle$. The artificial viscosity has constant parameters $\alpha$ and $\beta$.

Take velocity $\vec{u}^n_{\boldsymbol{i}}$, pressure $p^n_{\boldsymbol{i}}$, and density $\rho^n_{\boldsymbol{i}}$, components of $\overline{W}^n_{\boldsymbol{i}}$, from (15). Calculate the face-centered divergence of the velocity:

$$\begin{aligned}\lambda^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} =& \frac{1}{h}((u_d)^n_{\boldsymbol{i}+\boldsymbol{e}^d} - (u_d)^n_{\boldsymbol{i}}) + \\ & \frac{1}{4h}\sum_{d'\neq d}((u_{d'})^n_{\boldsymbol{i}+\boldsymbol{e}^d+\boldsymbol{e}^{d'}} - (u_{d'})^n_{\boldsymbol{i}+\boldsymbol{e}^d-\boldsymbol{e}^{d'}} + (u_{d'})^n_{\boldsymbol{i}+\boldsymbol{e}^{d'}} - (u_{d'})^n_{\boldsymbol{i}-\boldsymbol{e}^{d'}}).\end{aligned} \tag{35}$$

We then compute the artificial viscosity coefficient $\nu^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ at each face by

$$\nu^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = h\lambda^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}\min\{\frac{(h\lambda^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d})^2}{(c^{\min})^2_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}\cdot\beta}, 1\} \tag{36}$$

at faces where $\lambda^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} < 0$; otherwise, $\nu^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ is set to zero. Here $c^{\min}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \min\{c_{\boldsymbol{i}}, c_{\boldsymbol{i}+\boldsymbol{e}^d}\}$ and $c_{\boldsymbol{i}} = c(\rho_{\boldsymbol{i}}, p_{\boldsymbol{i}})$ is the speed of sound. The artificial viscosity is then applied as follows

$$\langle U\rangle^{n+1}_{\boldsymbol{i}} := \langle U\rangle^{n+1}_{\boldsymbol{i}} - \frac{\Delta t}{h}\sum_d(\mu^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \mu^d_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}) \tag{37}$$

$$\mu^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \alpha\nu^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}(\langle U\rangle^n_{\boldsymbol{i}+\boldsymbol{e}^d} - \langle U\rangle^n_{\boldsymbol{i}}). \tag{38}$$

This is equivalent to incrementing the total flux $\langle F^d\rangle^{\mathrm{tot}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} := \langle F^d\rangle^{\mathrm{tot}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} + \mu^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$. In cases where we use the total flux separately as part of the refluxing algorithm to maintain conservation on locally refined grids, we must make sure that the total fluxes are incremented in such a fashion. In regions of smooth flow, $\lambda = O(1)$, and the artificial viscosity makes an $O(h^4)$ contribution to the total flux, thus preserving fourth-order accuracy. At strong shocks, where the minimum in (36) takes on the value 1, the artificial viscosity reduces to the one used in [9, 10]. In all of the calculations shown here, we have set $\alpha = \beta = .3$.

# 3 Adaptive mesh refinement

We extend the uniform grid discretization to a locally-refined, nested grid hierarchy. Our notation follows that in [14]; we review this notation only to the extent that it is needed to describe the algorithm presented here. We start with a family of nested discretizations of a rectangular domain $\{\Gamma^l\}_{l=0}^{l_{\max}}$, $\Gamma^l \subset \mathbb{Z}^{\mathbf{D}}$. Each point $\boldsymbol{i} \in \Gamma^l$ represents a control volume of the form $V_{\boldsymbol{i}} = [\boldsymbol{i}h^l, (\boldsymbol{i}+\boldsymbol{u})h^l]$ each with mesh spacing $h^l$, with $h^l = n^l_{\text{ref}} h^{l-1}$. To relate geometric regions and variables on different levels of the heirarchy to one another, we define a coarsening operator

$$\mathcal{C}_r(\boldsymbol{i}) = (\lfloor \frac{i_1}{r} \rfloor, \ldots, \lfloor \frac{i_{\mathbf{D}}}{r} \rfloor)$$

where the notation $\lfloor x \rfloor$ means the largest integer less than or equal to $x$. We assume that $\mathcal{C}^{-1}_{n^l_{\text{ref}}}(\Gamma^{l-1}) = \Gamma^l$.

At any given point in time, our computed solution will be defined using $\{\Omega^l\}_{l=0}^{l_{\max}}$, $\Omega^l = \Omega^l(t) \subset \Gamma^l$, $\mathcal{C}_{n^l_{\text{ref}}}(\Omega^l) \subset \Omega^{l-1}$, $\Omega^0 = \Gamma^0$. We also allow refinement in time, as well as in space, with the assumption that the time steps at successive levels satisfy the condition that $\Delta t^l / \Delta t^{l+1}$ is a positive integer. The sets $\Omega^l$ are assumed to satisfy the condition of *proper nesting*, meaning that $\mathcal{C}^{-1}_{n^l_{\text{ref}}}(\mathcal{C}_{n^l_{\text{ref}}}(\Omega^l)) = \Omega^l$, and that there are at least $s^l > 0$ cells in any direction in $\Omega^l$ separating $\mathcal{C}_{n^l_{\text{ref}}}(\Omega^{l+1})$ and $\mathcal{C}^{-1}_{n^{l-1}_{\text{ref}}}(\Omega^{l-1}) - \Omega^l$. In the case of periodic domains, the proper-nesting condition is assumed to hold with respect to the periodic extensions of the grids. For boundaries in non-periodic directions, we also impose the requirement that cells in $\mathcal{C}_{n^l_{\text{ref}}}(\Omega^{l+1})$ must either be adjacent to the boundary, or at least $s^l$ level-$l$ cells away from the boundary. Our choice of $s$ is based on the requirement that, in order to interpolate ghost-cell values for evaluating the spatial operators described in the previous section, only cells at the next coarser level are required. In the present work, $s^l = \lceil \frac{5}{n^{l+1}_{\text{ref}}} \rceil + 2$, where the notation $\lceil x \rceil$ means the smallest integer greater than or equal to $x$.

The primary dependent variables on each level are defined on the grids at each level,

$$\langle U \rangle^l : \Omega^l \to \mathbb{R}^M.$$

In addition to $\Omega^l$, we will also need values for $\langle U \rangle^l$ on all cells in the stencils required to compute the right-hand side of (3). We will denote the extended solution also by $\langle U \rangle^l$. To advance the solution in time on such a grid hierarchy, we use the explicit time-stepping procedure in [3] (see also [7]) as outlined in Figure 1 below for function HyperbolicAdvance($l$).

The only difference between this method and the one in [3], other than our choice of single-level integration method, is the choice of interpolation schemes that are used to compute the values that lie outside $\Omega^l$ (the "ghost-cell values" required for step 1 of HyperbolicAdvance($l$)) and are required to evaluate the right-hand side of

**HyperbolicAdvance**($l$)

1. Advance $\langle U \rangle^l$ on $\Omega^l$ from time $t^l$ to time $t^l + \Delta t^l$, using the algorithm described in section 2. For each stage of the RK4 scheme, it is necessary to interpolate a collection of values at cells in $\Gamma^l - \Omega^l$, in order to evaluate the fluxes. In the process of computing the fluxes, we accumulate values in flux registers on faces corresponding to the boundaries of $\Omega^l$ and $\Omega^{l+1}$, using the total fluxes $\vec{F}^{\text{tot}}$.

2. Call **HyperbolicAdvance** for the next finer level:

**while** $t^{l+1} < t^l$ **do**
  **call HyperbolicAdvance**($l+1$)
**end while**

3. Synchronize solution on level $l$ with the solution on the finer levels:

- Fill values of $\langle U \rangle^l$ on $\mathcal{C}_{n^l_{\text{ref}}}(\Omega^{l+1})$ with averages of the solution on the next finer level:

$$\langle U \rangle^l_{\boldsymbol{i}} = \frac{1}{(n^l_{\text{ref}})^{\mathbf{D}}} \sum_{\boldsymbol{j} \in \mathcal{C}^{-1}_{n^l_{\text{ref}}}(\{\boldsymbol{i}\})} \langle U \rangle^{l+1}_{\boldsymbol{j}}.$$

- Increment $\langle U \rangle^l$ using flux registers defined on boundary between $\Omega^{l+1}$ and $\Omega^l_{valid}$.
- Update time: $t^l := t^l + \Delta t^l$.

4. If necessary, regrid on this level and all finer levels.

**end HyperbolicAdvance**

Figure 1: Pseudocode for AMR refinement in time algorithm

equation (3), and to compute the values on newly-refined grids upon regridding in step 4. In the previous work, we use a conservative piecewise-linear interpolation in space for both tasks, along with linear interpolation in time for computing the ghost-cell values. In the present work, we use fourth-order accurate interpolation in space derived using the method of least squares, for both ghost cells and regridding. For computing ghost-cell values, this is combined with a specialized interpolation in time that is closely related to the fourth-order Runge–Kutta method we are using for our single-level time discretization.

We first discuss the computation of the ghost cell values. We assume that, from step 1 of HyperbolicAdvance($l-1$), we have sufficiently accurate estimates of $\langle U^{l-1}\rangle(t^{l-1})$ and $\langle U^{l-1}\rangle(t^{l-1}+\Delta t^{l-1})$. In order to evaluate the operator $D\cdot\vec{F}$ on $\Omega^l$ for the $s^{\text{th}}$ stage of a Runge–Kutta method beginning at time $t^l$, we first interpolate the solution in time on all cells in $\Omega^{l-1}$ that are in the spatial interpolation stencil for the ghost cells. Then we use those values on level $l-1$ to interpolate values on the level-$l$ cells in $\Gamma^l-\Omega^l$ required to evaluate the fluxes. Only the values on the coarse grid at times $t^{l-1}$ and $t^{l-1}+\Delta t^{l-1}$ are used to interpolate the ghost-cell values.

## 3.1 Coarse-Fine Interpolation in Time

For any solution of our autonomous ODE integrated using fourth-order Runge–Kutta, from $t^{l-1}$ to $t^{l-1}+\Delta t^{l-1}$, we can compute all of the derivatives through third order in terms of the stage values $k_1,\ldots,k_4$, using the formula derived by Fok and Rosales [11]. For $0\le\chi\le 1$:

$$\begin{aligned}\langle U\rangle(t^{l-1}+\chi\Delta t^{l-1}) =&\langle U\rangle^{(0)}+\chi k_1+\frac{\chi^2}{2}(-3k_1+2k_2+2k_3-k_4)\\ &+\frac{2\chi^3}{3}(k_1-k_2-k_3+k_4)+O((\Delta t^{l-1})^4)\end{aligned}\tag{39}$$

where $\langle U\rangle^{(0)}=\langle U\rangle(t^{l-1})$ is the solution at the beginning of the coarse timestep, and $k_1,k_2,k_3,k_4$ are as defined in (6)–(9).

Hence the derivatives of $\langle U\rangle$ are:

$$\begin{aligned}\frac{d\langle U\rangle}{dt}(t^{l-1}+\chi\Delta t^{l-1}) =&\frac{1}{\Delta t^{l-1}}\Big(k_1+\chi(-3k_1+2k_2+2k_3-k_4)\\ &+2\chi^2(k_1-k_2-k_3+k_4)\Big)+O((\Delta t^{l-1})^3);\end{aligned}\tag{40}$$

$$\begin{aligned}\frac{d^2\langle U\rangle}{dt^2}(t^{l-1}+\chi\Delta t^{l-1}) =&\frac{1}{(\Delta t^{l-1})^2}\Big((-3k_1+2k_2+2k_3-k_4)\\ &+4\chi(k_1-k_2-k_3+k_4)\Big)+O((\Delta t^{l-1})^2);\end{aligned}\tag{41}$$

$$\frac{d^3\langle U\rangle}{dt^3}(t^{l-1}+\chi\Delta t^{l-1}) =\frac{4}{(\Delta t^{l-1})^3}(k_1-k_2-k_3+k_4)+O(\Delta t^{l-1}).\tag{42}$$

To advance the solution on the level $l$ grid from time $t^l$ to time $t^l + \Delta t^l$, we need to interpolate in time to find fourth-order approximations to $\langle U\rangle^{(0)}, \langle U\rangle^{(1)}, \langle U\rangle^{(2)}, \langle U\rangle^{(3)}$. To compute $\langle U\rangle^{(0)}$, we evaluate (39) at $\chi = (t^l - t^{l-1})/\Delta t^{l-1}$. To find $\langle U\rangle^{(1)}$, $\langle U\rangle^{(2)}$, and $\langle U\rangle^{(3)}$ at fine timestep $s$, the simplest approach would be to substitute $\chi = (t^l + \Delta t^l/2 - t^{l-1})/\Delta t^{l-1}$, $(t^l + \Delta t^l/2 - t^{l-1})/\Delta t^{l-1}$, and $(t^l + \Delta t^l - t^{l-1})/\Delta t^{l-1}$, respectively, in (39). In the absence of limiters, we found that such a procedure gave fourth-order accurate solution errors. However, when used in conjunction with the limiters, we found that the mismatch between the interpolated values and the intermediate steps in the Runge–Kutta time discretization on the fine grid can trigger the limiters even when the solution is smooth. For that reason, we interpolate ghost values that agree with the intermediate stages of the Runge–Kutta method to $O(\Delta t)^4$.

The fourth-order Taylor expansions of $\langle U\rangle^{(1)}, \langle U\rangle^{(2)}$, and $\langle U\rangle^{(3)}$ are:

$$\langle U\rangle^{(1)} = \langle U\rangle^{(0)} + \frac{\Delta t^l}{2} f(\langle U\rangle^{(0)}); \tag{43}$$

$$\begin{aligned}\langle U\rangle^{(2)} =& \langle U\rangle^{(0)} + \frac{\Delta t^l}{2} f(\langle U\rangle^{(1)}) \\ =& \langle U\rangle^{(0)} + \frac{\Delta t^l}{2} f(\langle U\rangle^{(0)}) + \frac{(\Delta t^l)^2}{4}\frac{df}{d\langle U\rangle} f(\langle U\rangle^{(0)}) + \frac{(\Delta t^l)^3}{16}\frac{d^2 f}{d\langle U\rangle^2}(f(\langle U\rangle^{(0)}))^2 + \\ & O((\Delta t^l)^4);\end{aligned} \tag{44}$$

$$\begin{aligned}\langle U\rangle^{(3)} =& \langle U\rangle^{(0)} + \Delta t^l f(\langle U\rangle^{(2)}) \\ =& \langle U\rangle^{(0)} + \Delta t^l f(\langle U\rangle^{(0)}) + \frac{(\Delta t^l)^2}{2}\frac{df}{d\langle U\rangle} f(\langle U\rangle^{(1)}) + \frac{(\Delta t^l)^3}{8}\frac{d^2 f}{d\langle U\rangle^2}(f(\langle U\rangle^{(0)}))^2 \\ & + O((\Delta t^l)^4) \\ =& \langle U\rangle^{(0)} + \Delta t^l f(\langle U\rangle^{(0)}) + \frac{(\Delta t^l)^2}{2}\frac{df}{d\langle U\rangle} f(\langle U\rangle^{(0)}) + \frac{(\Delta t^l)^3}{4}\Big(\frac{df}{d\langle U\rangle}\Big)^2 f(\langle U\rangle^{(0)}) + \\ & \frac{(\Delta t^l)^3}{8}\frac{d^2 f}{d\langle U\rangle^2}(f(\langle U\rangle^{(0)}))^2 + O((\Delta t^l)^4).\end{aligned} \tag{45}$$

Here we use the notation $f(\langle U\rangle) = -D \cdot \vec{F}(\langle U\rangle)$, and the derivatives of the vector-valued $f$ with respect to $\langle U\rangle$ are the appropriate Jacobians and Hessians of $f$. Note that, by the chain rule,

$$\frac{d^2\langle U\rangle}{dt^2} = \frac{df}{dt} = \frac{df}{d\langle U\rangle}\frac{d\langle U\rangle}{dt} = \frac{df}{d\langle U\rangle} f; \tag{46}$$

$$\frac{d^3\langle U\rangle}{dt^3} = \frac{d}{dt}\Big(\frac{df}{d\langle U\rangle} f\Big) = \frac{d^2 f}{d\langle U\rangle^2} f^2 + \Big(\frac{df}{d\langle U\rangle}\Big)^2 f. \tag{47}$$

We can approximate these derivatives using the coarse-grid values in (40)–(42). It follows from (44) and (45) that

$$\Big(\frac{df}{d\langle U\rangle}\Big)^2 f(\langle U\rangle^{(0)}) = \frac{4(f(\langle U\rangle^{(2)}) - f(\langle U\rangle^{(1)}))}{(\Delta t^l)^2} + O(\Delta t^l), \tag{48}$$

which we can also approximate from the coarser-level data as

$$(\frac{df}{d\langle U\rangle})^2 f = \frac{4(k_3 - k_2)}{(\Delta t^{l-1})^2} + O(\Delta t^{l-1}). \tag{49}$$

In (43)–(45), the quantities $f(\langle U\rangle^{(0)})$, $\frac{df}{d\langle U\rangle} f(\langle U\rangle^{(0)})$, $\frac{d^2 f}{d\langle U\rangle^2}$ and $(f(\langle U\rangle^{(0)}))^2$ can all be expressed in terms of derivatives of $\langle U\rangle$ evaluated at $t = t^l$. These in turn are approximated with the formulas (40)–(42), while $(\frac{df}{d\langle U\rangle})^2 f(\langle U\rangle^{(0)})$ is approximated using (49). These substitutions result in fourth-order accurate formulas for $\langle U\rangle^{(1)}$, $\langle U\rangle^{(2)}$, and $\langle U\rangle^{(3)}$ in terms of $k_1$, $k_2$, $k_3$, $k_4$, and $\langle U\rangle^{(0)}$.

## 3.2 Coarse-Fine Interpolation in Space

We interpolate $\langle u\rangle^{\mathrm{c}}$, averages over coarse-level cells, to find $\langle u\rangle^{\mathrm{f}}$, averages over fine-level cells.

### 3.2.1 Notations

For each coarse cell indexed by $\boldsymbol{i} \in \mathbb{Z}^{\mathbf{D}}$, we use these notations:

- $\mathcal{F}(\boldsymbol{i})$ is the set of fine cells contained within $\boldsymbol{i}$.
- $a_{\boldsymbol{i},\boldsymbol{p}}$ (for $\boldsymbol{p} \in \mathbb{N}^{\mathbf{D}}$ such that $\|\boldsymbol{p}\|_1 = \sum_d |p_d| \le 3$) are the coefficients that will be used for interpolation to $\langle u\rangle^{\mathrm{f}}_{\boldsymbol{k}}$ for all $\boldsymbol{k} \in \mathcal{F}(\boldsymbol{i})$. These will be the coefficients of the Taylor polynomial of degree 3 for $u$ around the center of cell $\boldsymbol{i}$. The number of coefficients for each coarse cell in 2D is 10, and in 3D is 20. The coefficients will be computed from values of $\langle u\rangle^{\mathrm{c}}$.
- $\mathcal{N}(\boldsymbol{i})$ is the set of coarse cells used as a stencil from which to take $\langle u\rangle^{\mathrm{c}}$ in order to find the coefficients $a_{\boldsymbol{i},\boldsymbol{p}}$.

For $\boldsymbol{z} \in \mathbb{R}^{\mathbf{D}}$ and $\boldsymbol{p} \in \mathbb{N}^{\mathbf{D}}$, we write $\langle \boldsymbol{z}^{\boldsymbol{p}}\rangle^{\mathrm{c}}_{\boldsymbol{j}}$ or $\langle \boldsymbol{z}^{\boldsymbol{p}}\rangle^{\mathrm{f}}_{\boldsymbol{k}}$ to denote the average, respectively over coarse cell $\boldsymbol{j}$ or fine cell $\boldsymbol{k}$, of

$$\boldsymbol{z}^{\boldsymbol{p}} = \prod_d (z_d^{p_d} - K(p_d)) \tag{50}$$

where

$$K(q) = \begin{cases} \frac{1}{q+1}(\frac{1}{2})^q & \text{if } q > 0 \text{ and } q \text{ is even;} \\ 0 & \text{otherwise.} \end{cases} \tag{51}$$

This constant is included to simplify numerical calculations; the average of $\boldsymbol{z}^{\boldsymbol{p}}$ on the cube $[-\frac{1}{2}, \frac{1}{2}]^{\mathbf{D}}$ is 1 if $\boldsymbol{p} = \mathbf{0}$, and 0 otherwise.

### 3.2.2 Cells in the stencil

The stencil $\mathcal{N}(\boldsymbol{i})$ for coarse cell $\boldsymbol{i}$ depends on the number of cells between $\boldsymbol{i}$ and the boundary of the domain.

$\mathcal{N}(\boldsymbol{i})$ consists of two sets of cells: an *inner set* and an *outer set*.

- The inner set is centered on a cell $c(\boldsymbol{i})$ that is identical to $\boldsymbol{i}$ if $\boldsymbol{i}$ is separated from the boundary by at least one other cell in every dimension; or if $\boldsymbol{i}$ is adjacent to the boundary, then $c(\boldsymbol{i})$ is one cell away from the boundary in each dimension in which $\boldsymbol{i}$ is adjacent to the boundary. The inner set consists of a square or cube of $3^{\mathbf{D}}$ cells with $c(\boldsymbol{i})$ at its center.

- The outer set consists of one cell beyond the inner set in each coordinate direction from $\boldsymbol{i}$ that is in the domain. Hence in every dimension, $\mathcal{N}(\boldsymbol{i})$ contains four or five cells in a row including $\boldsymbol{i}$.

The number of cells in the outer set is at most $2\mathbf{D}$, and by the proper-nesting condition, must also be at least $\mathbf{D}+1$. Hence the total number of cells in $\mathcal{N}(\boldsymbol{i})$ in 2D is either 12 or 13, and in 3D is in the range 31 to 33. For illustrations of examples of the possible stencils $\mathcal{N}(\boldsymbol{i})$, see Figure 2 for the 2D case and Figure 3 for the 3D case.

### 3.2.3 Calculating fine-cell averages from coarse-cell averages

To obtain the coefficients $a_{\boldsymbol{i},\boldsymbol{p}}$ for coarse cell $\boldsymbol{i}$, we solve a constrained linear least-squares problem [12, pp. 585–586] for the overdetermined system

$$\sum_{\boldsymbol{p}\in\mathbb{N}^{\mathbf{D}},\|\boldsymbol{p}\|_1\leq 3} a_{\boldsymbol{i},\boldsymbol{p}}\langle(\boldsymbol{x}-\boldsymbol{x}_{\boldsymbol{i}})^{\boldsymbol{p}}\rangle_{\boldsymbol{j}}^{\mathrm{c}} = \langle u\rangle_{\boldsymbol{j}}^{\mathrm{c}} \quad \text{for all } \boldsymbol{j}\in\mathcal{N}(\boldsymbol{i})-\{\boldsymbol{i}\} \tag{52}$$

with the conservation constraint

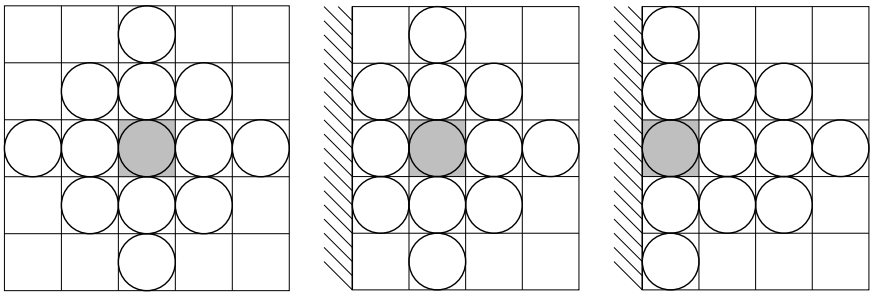
$$\sum_{\boldsymbol{p}\in\mathbb{N}^{\mathbf{D}},\|\boldsymbol{p}\|_1\leq 3} a_{\boldsymbol{i},\boldsymbol{p}}\langle(\boldsymbol{x}-\boldsymbol{x}_{\boldsymbol{i}})^{\boldsymbol{p}}\rangle_{\boldsymbol{i}}^{\mathrm{c}} = \langle u\rangle_{\boldsymbol{i}}^{\mathrm{c}} \tag{53}$$

where $\boldsymbol{x}_{\boldsymbol{i}}$ is the center of cell $\boldsymbol{i}$. We then use the coefficients $a_{\boldsymbol{i},\boldsymbol{p}}$ to interpolate for each fine cell $\boldsymbol{k}\in\mathcal{F}(\boldsymbol{i})$:

$$\langle u\rangle_{\boldsymbol{k}}^{\mathrm{f}} = \sum_{\boldsymbol{p}\in\mathbb{N}^{\mathbf{D}},\|\boldsymbol{p}\|_1\leq 3} a_{\boldsymbol{i},\boldsymbol{p}}\langle(\boldsymbol{x}-\boldsymbol{x}_{\boldsymbol{i}})^{\boldsymbol{p}}\rangle_{\boldsymbol{k}}^{\mathrm{f}}. \tag{54}$$

The conservation constraint (53) is derived as follows. The average of all interpolated $\langle u\rangle^{\mathrm{f}}$ on fine cells within coarse cell $\boldsymbol{i}$ must equal $\langle u\rangle_{\boldsymbol{i}}^{\mathrm{c}}$. Hence, using (54):

$$\frac{1}{n_{\mathrm{ref}}^{\mathbf{D}}}\sum_{\boldsymbol{k}\in\mathcal{F}(\boldsymbol{i})}\sum_{\boldsymbol{p}\in\mathbb{N}^{\mathbf{D}},\|\boldsymbol{p}\|_1\leq 3} a_{\boldsymbol{i},\boldsymbol{p}}\langle(\boldsymbol{x}-\boldsymbol{x}_{\boldsymbol{i}})^{\boldsymbol{p}}\rangle_{\boldsymbol{k}}^{\mathrm{f}} = \langle u\rangle_{\boldsymbol{i}}^{\mathrm{c}}. \tag{55}$$

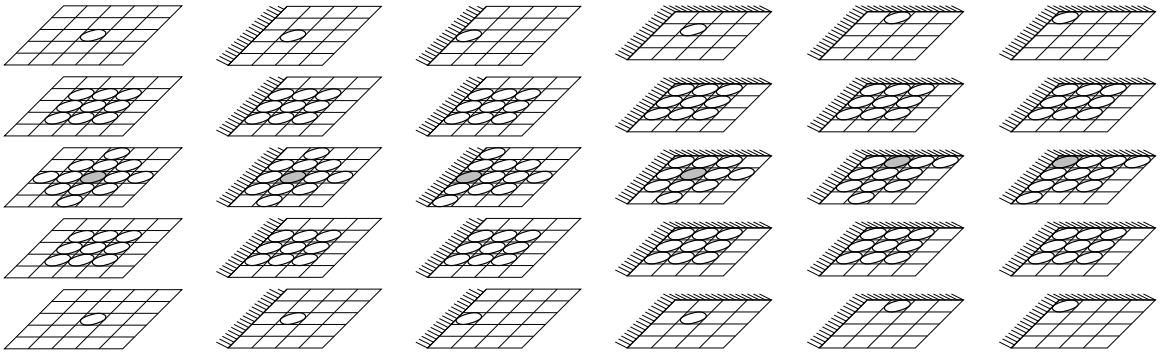


(a)
13 cells
$\kappa = 16.6$

(b)
12 cells
$\kappa = 17.3$

(c)
12 cells
$\kappa = 92.0$

Figure 2: Three examples of 2D stencils, indicated by circles, of coarse cells that are used to interpolate to the fine cells (unmarked) within the shaded coarse cell. Hash marks along an edge indicate a physical boundary on that edge. Modulo reflection and permutation of axes, these are all of the stencil possibilities that can arise in 2D. Because of the proper-nesting condition, the coarse cell containing fine ghost cells must be separated by the physical boundary by at least two other coarse cells in at least one of the dimensions. The three possible separations in the other dimension are: (a) two or more cells; (b) a single cell; (c) no separation. Figure 4 shows an instance of each of these stencils being used in a sample set of patches. In all cases, the stencil consists of a $3 \times 3$ block of cells together with the next cell beyond this block in each coordinate direction from the target cell, as long as this next cell is within the domain. Above are also shown the number of cells in each stencil, and the condition number of the matrix that converts stencil cell values to the 10 coefficients.



| (a) | (b) | (c) | (d) | (e) | (f) |
|---|---|---|---|---|---|
| 33 cells | 32 cells | 32 cells | 31 cells | 31 cells | 31 cells |
| $\kappa = 12.7$ | $\kappa = 13.3$ | $\kappa = 70.0$ | $\kappa = 14.0$ | $\kappa = 73.0$ | $\kappa = 134.$ |

Figure 3: Six examples of 3D stencils, indicated by circles, of coarse cells that are used to interpolate to the fine cells (unmarked) within the shaded coarse cell. Hash marks along an edge indicate a physical boundary on that edge. Modulo reflection and permutation of axes, these are all of the stencil possibilities that can arise in 3D. Because of the proper-nesting condition, the coarse cell containing fine ghost cells must be separated by the physical boundary by at least two other coarse cells in at least one of the dimensions. The six stencils shown here represent the possibilities in the other two dimensions for the target cell to be adjacent to the physical boundary or separated by a single cell or by two or more cells. In all cases, the stencil consists of a $3 \times 3 \times 3$ block of cells together with the next cell beyond this block in each coordinate direction from the target cell, as long as this next cell is within the domain. Above are also shown the number of cells in each stencil, and the condition number of the matrix that converts stencil cell values to the 20 coefficients.

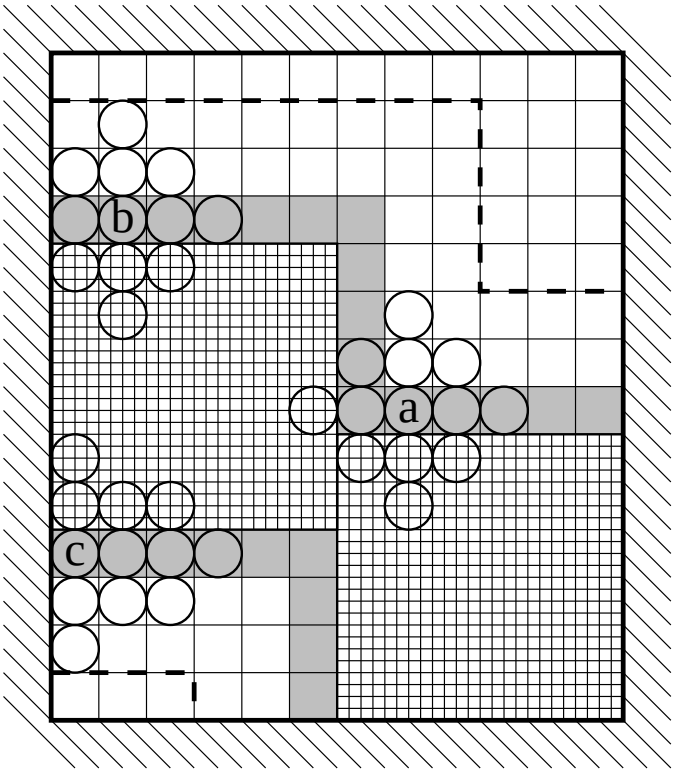



Figure 4: A 2D example of two levels with a refinement ratio of 4, the physical boundary indicated by hash marks, and the coarser level covering the whole rectangular domain. Dashed lines mark the limit of coarse cells that are used in stencils to interpolate to fine ghost cells. The shaded coarse cells contain the fine ghost cells that need to be filled in. The letters a, b, and c indicate three such coarse cells where the stencils used are respectively (a), (b), and (c) of Figure 2; the coarse cells of each stencil are marked with circles. Note that the stencil may include coarse cells that are covered by the finer level.

But splitting up coarse cell $\boldsymbol{i}$ into its fine subcells, it is also true that for each $\boldsymbol{p}$,

$$\frac{1}{n_{\mathrm{ref}}^{\mathbf{D}}} \sum_{\boldsymbol{k} \in \mathcal{F}(\boldsymbol{i})} \langle (\boldsymbol{x} - \boldsymbol{x_i})^{\boldsymbol{p}} \rangle_{\boldsymbol{k}}^{\mathrm{f}} = \langle (\boldsymbol{x} - \boldsymbol{x_i})^{\boldsymbol{p}} \rangle_{\boldsymbol{i}}^{\mathrm{c}}. \tag{56}$$

By reordering summation in (55) and making the substitution (56), then (53) follows.

In 2D, (52) has 10 variables and 11 or 12 equations. In 3D, (52) has 20 variables and 30 to 32 equations. The variables are the coefficients $a_{\boldsymbol{i},\boldsymbol{p}}$ for $\boldsymbol{p} \in \mathbb{N}^{\mathbf{D}}$ such that $\|\boldsymbol{p}\|_1 \leq 3$, and in (52) there is one equation for each $\boldsymbol{j} \in \mathcal{N}(\boldsymbol{i}) - \{\boldsymbol{i}\}$.

# 4 Results

We use this method to solve the 1D advection equation, in order to show results with the new limiter, and then to solve the equations of gas dynamics in 2D and 3D. Unless otherwise stated, the calculations are performed with the full algorithm, that is, with limiters and dissipation mechanisms turned on. For gas-dynamics problems with smooth solutions, we compare our method with that obtained without limiters, indicated here as "limiter off". We also perform a calculation of a standard shock reflection test problem.

Applying the analysis in [5] to the equations of gas dynamics gives a stability condition for time step $\Delta t$ and mesh spacing $h$, of

$$\frac{\Delta t}{h} \sum_d (|\boldsymbol{v} \cdot \boldsymbol{e}^d| + c) \lessapprox 1.3925 \tag{57}$$

where $\boldsymbol{v}$ is velocity and $c$ is the speed of sound. This condition comes from the combination of constraints for the fourth-order Runge–Kutta method in time, and first-order upwinding in space, which is the low-order scheme corresponding to the present method. Note that condition (57) is more restrictive than the one typically used in the method of [4], because there is no analogue of corner coupling that permits use of a larger time step.

## 4.1 1D advection with new limiter

We test the algorithm with limiter of section 2.4.1 on the 1D advection problem

$$\frac{\partial a}{\partial t} + u \frac{\partial a}{\partial x} = 0 \tag{58}$$

where $u$ is a constant. We can compare with the exact solution,

$$a(x, t) = a(x - ut, 0). \tag{59}$$

We use the standard 1D test problems:

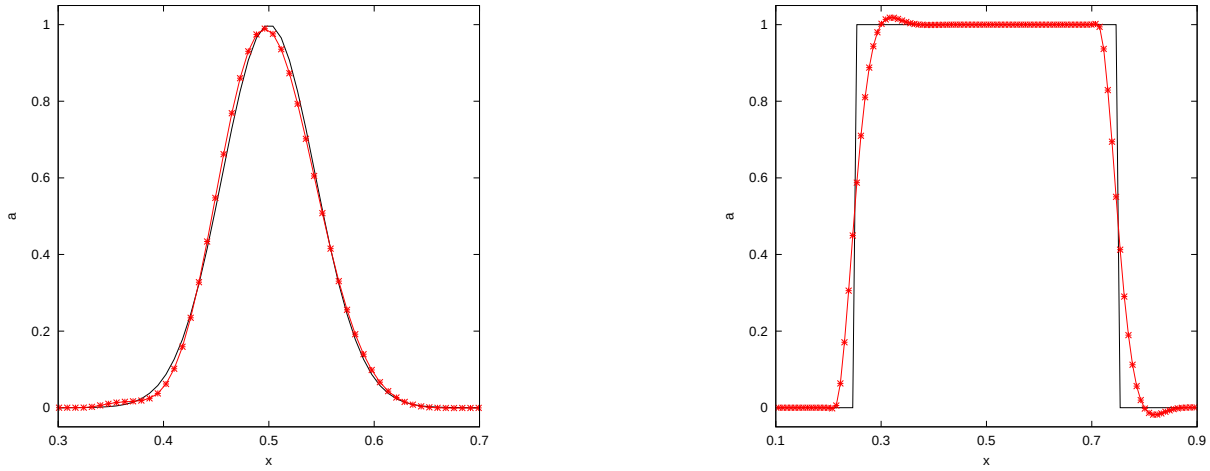
- Gaussian: $a(x,0) = e^{-256(x-\frac{1}{2})^2}$
- Square wave: $a(x,0) = 1$ if $|x - \frac{1}{2}| \leq \frac{1}{4}$, otherwise 0.

All calculations are performed on the unit interval with periodic boundary conditions, advection velocity $u = 1$, and CFL number 0.2. The dissipation mechanisms of section 2.5 do not apply. Table 1 shows errors and rates of convergence for these test problems. We find that the Gaussian problem exhibits fourth-order convergence. The square-wave problem has a convergence rate of $\frac{4}{5}$ in $L_1$-norm, as in [8].

| problem | norm | 1/128 | rate | 1/256 | rate | 1/512 | rate | 1/1024 |
|---|---|---|---|---|---|---|---|---|
| Gaussian | $L_\infty$ | 4.03e-02 | 3.91 | 2.67e-03 | 4.01 | 1.66e-04 | 4.00 | 1.04e-05 |
| Gaussian | $L_1$ | 4.75e-03 | 3.99 | 3.00e-04 | 3.99 | 1.88e-05 | 4.00 | 1.18e-06 |
| Square wave | $L_1$ | 3.26e-02 | 0.79 | 1.89e-02 | 0.79 | 1.09e-02 | 0.80 | 6.29e-03 |

Table 1: Errors and convergence rates for 1D advection tests with the limiter of section 2.4.1, at time 10., run with CFL number 0.2. The top row shows the mesh spacing.

Figure 5 shows some results for the two test problems when run with 128 cells.




Gaussian Square wave

Figure 5: Results using the limiter (red stars) and the exact solution (black curve) tested on 1D advection of a Gaussian (left) or a square wave (right). Both test problems were run on 128 cells and with CFL number 0.2; the results shown are for $a$ at time 10.

## 4.2 Gaussian acoustic pulse

Our first gas-dynamics example is of a Gaussian acoustic pulse in a polytropic gas, in a periodic domain, $[0,1]^{\mathbf{D}}$. The initial conditions at a point in this domain are

determined by the distance $r$ from the center. Initially the velocity is zero, and the density is

$$\rho(r) = \begin{cases} \rho_0 + (\delta\rho_0)e^{-16r^2}\cos^6(\pi r), & \text{if } r \le \frac{1}{2}; \\ \rho_0 & \text{otherwise;} \end{cases} \tag{60}$$

with $\rho_0 = 1.4$ and $\delta\rho_0 = 0.14$. The smoothing factor $\cos^6(\pi r)$ is present in order to ensure $\rho = \rho_0$ on the domain boundaries. For isentropicity, the initial pressure is

$$p = (\frac{\rho}{\rho_0})^\gamma \tag{61}$$

where $\gamma = 1.4$.

We run this example in 2D on a single level, with flattening and artificial viscosity, and both with and without the limiter. Throughout each run, the time step is fixed, set to $\Delta t = 0.192h$, where $h$ is the mesh spacing. The results in Table 2 show fourth-order convergence.

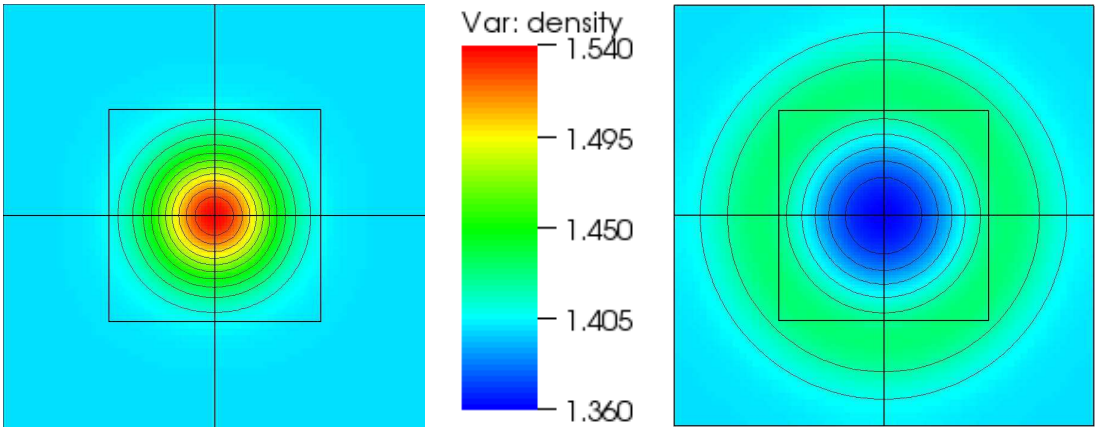
| | 1/128: | | 1/256: | | 1/512: | | 1/1024: |
|---|---|---|---|---|---|---|---|
| limiter | 1/256 | rate | 1/512 | rate | 1/1024 | rate | 1/2048 |
| on | 1.32e-06 | 4.18 | 7.28e-08 | 4.01 | 4.53e-09 | 3.99 | 2.85e-10 |
| off | 1.15e-06 | 3.99 | 7.20e-08 | 4.00 | 4.51e-09 | 4.00 | 2.82e-10 |

Table 2: Convergence of differences in calculated density at time 0.24 for 2D Gaussian acoustic pulse, run on a uniform grid, and with the limiter of section 2.4 either on or off. Columns alternate between showing the max-norm of the difference in densities between results with the indicated mesh spacings, and the convergence rate.

We also run this same problem, with and without the limiter, in 2D and 3D on two levels, with a refinement factor of 2 between the levels. Grids at the coarser level cover a cube, and grids at the finer level cover half the length of the cube in each dimension. Figure 6 shows a color plot of density at initial and final times in 2D. Tables 3 and 4 show convergence results in 2D and 3D, respectively, with the limiter either on or off, and indicate fourth-order convergence in all cases.

| | 1/64: | | 1/128: | | 1/256: | | 1/512: |
|---|---|---|---|---|---|---|---|
| limiter | 1/128 | rate | 1/256 | rate | 1/512 | rate | 1/1024 |
| on | 7.28e-06 | 3.97 | 4.66e-07 | 3.95 | 3.01e-08 | 3.99 | 1.90e-09 |
| off | 7.29e-06 | 3.97 | 4.66e-07 | 3.95 | 3.01e-08 | 3.99 | 1.90e-09 |

Table 3: Convergence of differences in calculated density at time 0.24 for 2D Gaussian acoustic pulse, run with fixed grids on two levels, and with the limiter of section 2.4 either on or off. Columns alternate between showing the max-norm of the difference in densities between results with the indicated mesh spacings at the *coarser* of the two levels, and the convergence rate.




density at time 0. density at time 0.24

Figure 6: Gaussian acoustic pulse in 2D, on two levels.

| limiter | 1/16: 1/32 | rate | 1/32: 1/64 | rate | 1/64: 1/128 | rate | 1/128: 1/256 |
|---|---|---|---|---|---|---|---|
| on | 6.84e-04 | 3.39 | 6.54e-05 | 3.69 | 5.06e-06 | 3.78 | 3.70e-07 |
| off | 7.35e-04 | 3.22 | 7.88e-05 | 3.80 | 5.66e-06 | 3.94 | 3.69e-07 |

Table 4: Convergence of differences in calculated density at time 0.24 for 3D Gaussian acoustic pulse, run with fixed grids on two levels, and with the limiter of section 2.4 either on or off. Columns alternate between showing the max-norm of the difference in densities between results with the indicated mesh spacings at the *coarser* of the two levels, and the convergence rate

Finally, we run the 2D problem, with the limiter on, on two levels such that the refinement ratio is 2 and the grids on the finer level are determined adaptively, every two coarse time steps, by refining where $|\nabla\langle\rho\rangle|/\langle\rho\rangle > 0.2h$, with $h$ the coarse-level mesh spacing. Table 5 shows the convergence of differences between results on such two-level adaptive grids and on corresponding uniform one-level grids, where the mesh spacing on the one-level grid is uniformly that on the finer of the two levels in the adaptive case. The truncation error for this method is $O(h^4)$ away from refinement boundaries, and $O(h^3)$ at refinement boundaries. Modified equation arguments would indicate that, for adaptive calculations, in which the refinement boundaries are approximately characteristic, we would see a solution error somewhere between third and fourth order in the mesh spacing, in max norm. By combining these results with those in Table 3, we obtain a convergence rate that is approximately $O(h^{15/4})$ in max norm, which is consistent with such an analysis.

| 1/128 | rate | 1/256 | rate | 1/512 | rate | 1/1024 | rate | 1/2048 |
|---|---|---|---|---|---|---|---|---|
| 8.37e-06 | 3.44 | 7.69e-07 | 3.54 | 6.59e-08 | 3.73 | 4.96e-09 | 3.75 | 3.69e-10 |

Table 5: Convergence of differences in density at time 0.24 for 2D Gaussian acoustic pulse, between results calculated on a single-level grid with the indicated uniform mesh spacing, and results calculated on *adaptive* grids on two levels with finer-level mesh spacing as indicated here and with the coarser-level mesh spacing being double that. Columns alternate between showing the max-norm of the difference in densities, and the convergence rate.

## 4.3 Shear problem

In this 2D polytropic gas problem, we start with constant density $\rho = 1.4$ and pressure $p = 7.$, with initial velocity on the unit square $[0, 1]^2$ set to

$$v_x(x, y) = \cos(2\pi y);$$
$$v_y(x, y) = \cos(2\pi x).$$

We run on the same fixed two-level hierarchy as in section 4.2. Throughout each run, the time step is fixed, with a CFL number of 0.508.

Table 6 shows convergence results with the limiters of section 2.4 turned either *off* or *on*, and with the time interpolation either as described in section 3.1 with $U^{(1)}, U^{(2)}, U^{(3)}$ from equations (43)–(45), or from substitution of $\chi = (s+\frac{1}{2})/n_{\text{ref}}, (s+\frac{1}{2})/n_{\text{ref}}, (s+1)/n_{\text{ref}}$, respectively, in (39). Note that with sufficiently high refinement, the limiter interferes with the time interpolation using substitution in (39), so that convergence is not even second order. But when using that same time interpolation with the limiter turned off, or when using the time interpolation from (43)–(45) with the limiter turned on, convergence is fourth order.

| limiter | time interp. | 1/64: 1/128 | rate | 1/128: 1/256 | rate | 1/256: 1/512 | rate | 1/512: 1/1024 |
|---|---|---|---|---|---|---|---|---|
| on | (43)–(45) | 1.32e-04 | 4.05 | 7.99e-06 | 3.95 | 5.17e-07 | 3.98 | 3.27e-08 |
| off | (39) | 1.13e-04 | 3.83 | 7.96e-06 | 3.92 | 5.24e-07 | 3.95 | 3.39e-08 |
| on | (39) | 1.32e-04 | 3.75 | 9.78e-06 | 1.39 | 3.74e-06 | 1.59 | 1.24e-06 |

Table 6: Convergence of max-norm of calculated differences in $x$-momentum for 2D shear problem at time 0.15, with limiter on or off, and time interpolation taking $U^{(1)}, U^{(2)}, U^{(3)}$ either as in equations (43)–(45) or by substitution of $\chi = (s+\frac{1}{2})/n_{\text{ref}}, (s+\frac{1}{2})/n_{\text{ref}}, (s+1)/n_{\text{ref}}$, respectively, in (39).

### 4.4 Shock-ramp problem

We implement the shock-ramp problem of Woodward and Colella [15], on two levels (refinement ratio of 4 between them), with effective resolution $1024 \times 256$. The CFL number is initially 0.3 and is kept to at most 0.8. See Figure 7 for a color plot of the whole domain and Figure 8 for a close-up. The results we obtain here show that the present method has a treatment of multidimensional time-dependent discontinuous flows that is comparable to that of the best state-of-the-art shock-capturing methods.

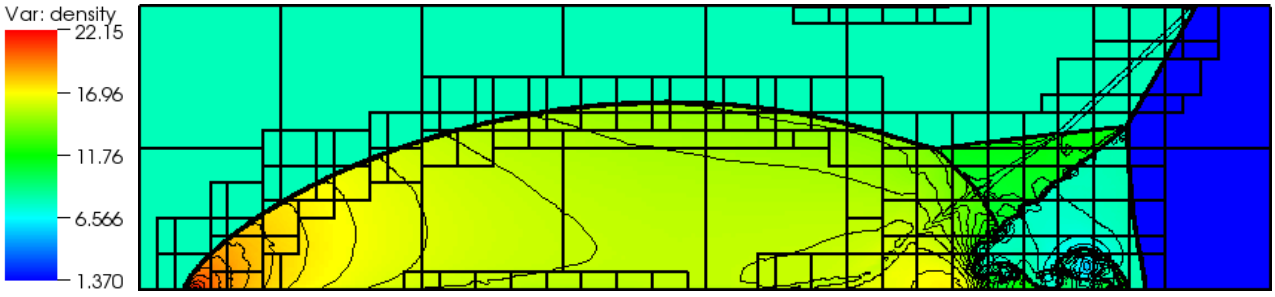



Figure 7: 2D Woodward–Colella shock-ramp problem, with a color plot and contour lines of density, and outlines of the blocks used at the two levels. Figure 8 shows a close-up of this plot.

## 5 Conclusions

In this paper, we have described an extension of the finite-volume block-structured adaptive mesh refinement algorithm for hyperbolic conservation laws in [3] that is fourth-order accurate in space and time. The underlying single-grid algorithm is an extension of the algorithm in [5] that is comparably accurate and robust to the higher-order Godunov methods for problems involving strong shocks. To achieve this combination of accuracy and robustness, we needed to modify the limiter in [8] to

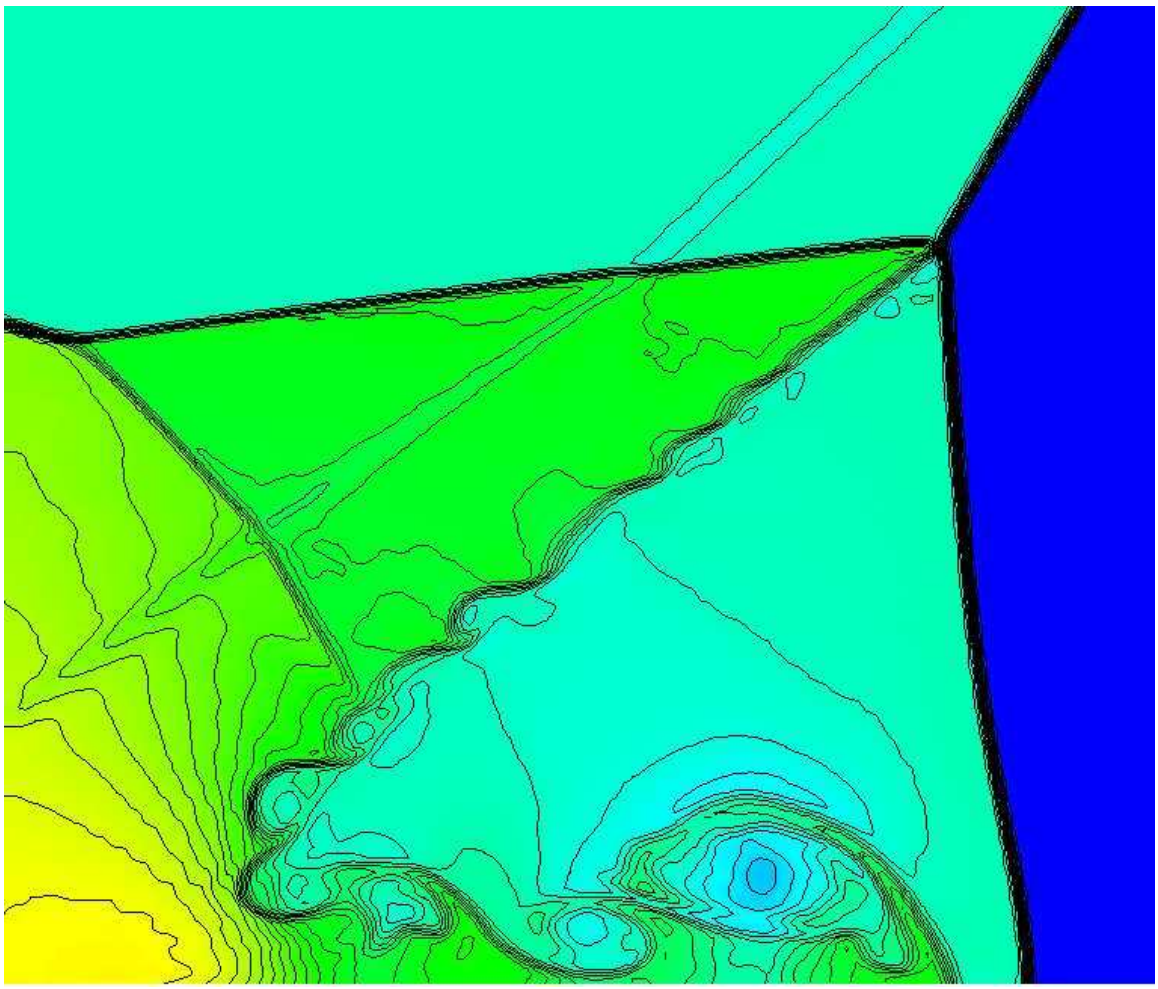


Figure 8: Close-up of Figure 7, showing a color plot and contour lines of density.

eliminate sensitivity to roundoff error, and to better distinguish smooth extrema that arise in multidimensional problems.

There are a number of directions in which it is natural to extend this algorithm. One is to combine it with the ideas in [5] to compute AMR solutions on mapped grids. This is a key step to the application of this approach to problems such as climate modeling that require mapped-multiblock grids [6]. One essential issue is the extension of the approach in [2] to higher order using the ideas in [5] so that free-stream preservation is satisfied. Another, less trivial extension is to develop a method analogous to the present one for hyperbolic-parabolic problems that is semi-implicit, treating the hyperbolic terms explicitly, and the parabolic terms implicitly. This has been done for advection-diffusion problems [16] using the the fourth-order additive Runge–Kutta method in [13], but only for refinement in space: the same time step is used on all levels. The extension to refinement in time will require the use of an appropriate version of the "dense output" representation for intermediate values described in that paper, analogous to (39) for the explicit Runge–Kutta method used here.

**Acknowledgements**.The authors would also like to thank Jeff Hittinger, Dan Martin, and Mike Minion for helpful discussions.

# References

[1] M. Barad and P. Colella. A fourth-order accurate local refinement method for Poisson's equation. *Journal of Computational Physics*, 209:1–18, 2005.

[2] J. B. Bell, P. Colella, J. A. Trangenstein, and M. Welcome. Adaptive mesh refinement on moving quadrilateral grids. In *Proceedings of the Ninth AIAA Computational Fluid Dynamics Conference*, pages 471–479. AIAA, June 1989.

[3] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82(1):64–84, May 1989.

[4] P. Colella. Multidimensional upwind methods for hyperbolic conservation laws. *J. Comput. Phys.*, 87:171–200, March 1990.

[5] P. Colella, M. Dorr, J. Hittinger, and D. F. Martin. High-order finite-volume methods in mapped coordinates. in preparation, 2009.

[6] P. Colella, M. Dorr, J. Hittinger, P. McCorquodale, and D. F. Martin. High-order finite-volume methods on locally-structured grids, proceedings. In *Proceedings, Astronum-2008 meeting, Astronomical Society of the Pacific Conference Series*, volume 406, 2008.

[7] P. Colella, D. T. Graves, N. D. Keen, T. J. Ligocki, D. F. Martin, P. W. McCorquodale, D. Modiano, P.O. Schwartz, T. D. Sternberg, and B. Van Straalen.

Chombo Software Package for AMR Applications - Design Document. 2009. `https://seesar.lbl.gov/anag/chombo/ChomboDesign-3.0.pdf`.

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

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

[10] Phillip Colella. Multidimensional upwind methods for hyperbolic conservation laws. *J. Comput. Phys.*, 87:171–200, 1990.

[11] Pak-Wing Fok and Rodolfo R. Rosales. Multirate integration of axisymmetric step-flow equations. *J. Comput. Phys.*, submitted. http://arxiv.org/abs/0810.2517v1.

[12] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, third edition, 1996.

[13] Christopher A. Kennedy and Mark H. Carpenter. Additive Runge–Kutta schemes for convection–diffusion–reaction equations. *Applied Numerical Mathematics*, 44:139–181, 2003.

[14] D. Martin, P. Colella, and D. T. Graves. A cell-centered adaptive projection method for the incompressible Navier-Stokes equations in three dimensions. *Journal of Computational Physics*, 227:1863–1886, 2008.

[15] P. R. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *J. Comput. Phys.*, 54:115–173, 1984.

[16] Qinghai Zhang, Hans Johansen, and Phillip Colella. A fourth-order accurate finite-volume method with structured adaptive mesh refinement for solving the advection-diffusion equation, 2010. submitted to *SIAM J. Sci. Comp.*