TWO-DIMENSIONAL SLOPE LIMITERS FOR FINITE VOLUME SCHEMES ON NON-COORDINATE-ALIGNED MESHES*

SANDRA MAY† AND MARSHA BERGER†

Abstract. In this paper we develop a new limiter for linear reconstruction on non-coordinate-aligned meshes in two space dimensions, with focus on Cartesian embedded boundary grids. Our limiter is inherently two-dimensional and linearity preserving. It separately limits the x and y components of the gradient, as opposed to a scalar limiter which limits all components simultaneously with one scalar. The limiter is based on solving a tiny linear program (LP) on each cell, using a very efficient version of the simplex method. A variety of computational results on triangular and embedded boundary meshes are presented. They demonstrate that the LP limiter successfully removes oscillations and significantly increases solution accuracy compared to a scalar limiter.

Key words. slope limiter, Cartesian cut cell method, finite volume scheme, linear programming

AMS subject classifications. 65M08, 65K99

1. Introduction. The goal of this research is to develop a robust and accurate limiter for finite volume schemes on non-coordinate-aligned meshes, with focus on Cartesian embedded boundary meshes. These meshes have cells that are *cut* by a solid body at the edge of the domain in an essentially arbitrary fashion (see Fig. 3.1). They are much more irregular than the triangles typically found in unstructured mesh methods. For example, cut cells can have 3, 4 or 5 faces and neighboring cells can differ by several orders of magnitude in size. A *scalar* limiter is typically used on these cells. We will generalize this to compute a *vector* limiter suitable for use on cut cells. In fact, we develop a limiter *framework* that can handle different types of constraints while retaining the vector property.

The idea of scalar limiting for unstructured meshes was introduced by Barth and Jespersen [5]: compute a trial gradient, then reduce it by a scalar $\phi \in [0, 1]$ applied to all components of the gradient. Several authors have since developed variants of scalar limiting to reduce diffusion. For example, Batten et al [6] construct several trial gradients, limit each using a scalar limiter, and then choose the one with the largest norm. Park et al [28] extended the multi-dimensional limiting process (MLP) introduced in [24, 33] from structured to unstructured meshes, and use an enlarged stencil in the limiting process.

For unstructered triangular grids, several approaches have been suggested to replace the scalar limiter with a more multi-dimensional approach, rather than simply reducing the length of an initially constructed gradient [21, 22, 30, 12, 25]. Additionally, ENO/WENO-type schemes have been used successfully on unstructured grids [1, 14, 19, 27]. However, none of these methods can be easily extended to cut cells without compromising the resolution of the solution or at great expense. To date, the scalar limiter is typically used on cut cells, as well as remaining the most commonly used limiter in practice for triangular grids.

In this paper we follow the suggestion in [8] and develop a two-dimensional limiter suitable for cut cells and, more generally, any type of distorted cells. It limits the components of the gradients in the x and y directions separately, leading to a less

*This work was supported in part by the DOE office of Advanced Scientific Computing Research under grant DE-FG02-88ER25053 and by AFOSR grant FA9550-10-1-0158

†Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012, USA

diffusive method than a scalar limiter provides. Note that such a *vector* limiter (ϕ_x, ϕ_y) cannot be implemented using the standard method of sequentially limiting each edge of a cell (or face in three dimensions), since changing the components of the gradient individually *rotates* the gradient, whereas a scalar limiter just shortens its length. After rotation the previously limited faces may no longer satisfy the monotonicity requirements. Of course on the orthogonal (Cartesian) part of an embedded boundary mesh the coordinate directions decouple, and limiting one direction has no effect on the other.

To solve this problem we formulate the slope limiter as the solution to a linear program (LP). In words, we try to retain as much of the initial gradient as possible while satisfying monotonicity constraints. This is made precise in section 3. Overall, this leads to many tiny LPs, whose efficient solution will be critical for this method to be practical. The traditional LP usually has a very large number of unknowns. In our case, i.e., in two space dimensions, the LP has only two unknowns ϕ_x and ϕ_y, but we need to solve many of these small problems. We use the little known all-inequality simplex algorithm[1] that solves these tiny problems much more efficiently than the simplex method for LPs in standard form, and is mathematically equivalent to it. This more efficient variant is described in detail in Appendix B.

The idea of formulating a limiter as an optimization problem also shows up in the work of others dealing with triangular and quadrilateral grids. Hubbard [20] suggests using a quadratic program (QP) for limiting on triangular grids, enforcing the reconstruction to stay within a designated 'maximum principle region'. He finds the optimal solution by evaluating all constraint intersections defining his maximum principle region. In [21] Hubbard uses a similar approach of projecting the initial gradient estimate onto a defined maximum principle region, but again he does not provide a general solution algorithm.

Buffard & Clain [10] consider a least squares problem with the solution fulfilling TVD constraints. They develop a solution strategy specific to their constraints on a triangular grid which satisfies some minor geometric conditions. In [18], Hoteit et al. develop a limiter for Discontinuous Galerkin methods on arbitrary unstructured quadrangular and triangular grids based on [11]. They make use of least squares problems with linear equality and inequality constraints and (unlike [20, 10]) suggest a general optimization algorithm for their solution. However, solving a least squares problem with (in-)equality constraints is significantly more expensive than solving an LP – especially if the solution of the LP is achieved in the very efficient way suggested in this work.

In this paper we reformulate and extend this optimization approach to slope limiting to the more complicated geometry of a cut cell, and provide an efficient solution algorithm. The paper is organized as follows. In section 2 we briefly review the finite volume scheme and the cut cell method that provide the motivation for this work. In section 3, we completely characterize the LPs used to enforce the limiting and discuss how to solve them. We actually describe a *family* of limiters, depending on the chosen monotonicity constraints. Sections 4 and 5 present computational experiments in two space dimensions. We first show experiments for linear advection on triangular meshes. The purpose of these results is to confirm general accuracy and monotonocity properties and to be able to compare to well-known methods from the literature. Then, in section 5 we present results solving the Euler equations on cut cell meshes. The tests include a problem with an exact solution to be able to measure

[1] thanks to Margaret Wright for pointing us to it

the error, and a non-linear problem with shocks to show robustness. For the problems we tested, our method is two to five times more accurate than using a scalar limiter, depending on the specific problem and the specific constraints used. We conclude with some thoughts on extending this approach to three dimensions.

2. Embedded Boundary Finite Volume Schemes. Cut cells can have arbitrarily irregular shapes and aspect ratios, depending on how they are cut by a solid object. Figure 3.3 shows cut cells with 2, 3, or 4 edge neighbors in addition to a boundary segment. This makes finite volume schemes the method of choice, since they are easily formulated independent of the shape of a cell, and so are often used for meshes with heterogeneous cell types. The slope limited (rather than flux limited) finite volume schemes are also preferred, since slope reconstruction and limiting has a straightforward geometric interpretation: find a gradient that reconstructs a linear function through the cell centroid without creating new extrema.

Many second-order finite volume schemes are based on the following steps:

1. Construct an initial gradient (D_x, D_y) on cell M.
2. Reduce the gradient to enforce monotonicity constraints (the limiting step):
   - Scalar limiters reduce the gradient using the form $\phi(D_x, D_y)$.
   - We will use a vector limiter giving a gradient of the form $(\phi_x D_x, \phi_y D_y)$.
3. Compute left and right states at the midpoint of each edge, needed to solve the Riemann problem.
4. Compute the numerical flux by solving the Riemann problem using these states.
5. Evolve the solution to the next timestep (typically using a Runge-Kutta method).

The particular details vary. For example, one can use a least squares approach or a Green-Gauss method for the initial gradient construction. The gradient can be based on the solution using face neighbors, vertex neighbors, or a combination of both, usually depending on one's choice of data structures and whether the scheme is cell-centered or node-based. The choice of numerical flux function often depends on the problem (van Leer is more robust, HLLC is less dissipative). TVD Runge-Kutta schemes [16] have favorable non-linear stability properties and are finding increased use. For a multi-stage Runge-Kutta scheme, steps 1-4 above are generally repeated at every stage, but shortcuts can be taken where gradients and limiters are evaluated less frequently (e.g. [21]). For steady-state solutions this shortcut is often taken.

If steady-state solutions are the goal, then the choice of integration scheme in time is not as significant, since the order of accuracy in time will not matter. Furthermore, one can use local time-stepping for steady-state flows, which allows a different Δt to be used in each cell, greatly accelerating the convergence to steady-state. This is especially advantageous on a cut cell mesh. The straightforward scheme outlined above can be used on the cut cells choosing Δt_j proportional to each cell volume V_j, and stability is retained without any extra work.

For time-dependent flows additional work must be done to preserve accuracy and stability in the cut cells. The time-dependent computations with cut cells in section 5 use the h-box method [9] to retain stability for a full-sized Δt based on the mesh size h of the regular mesh cells. In other words, the CFL number is based on the stability limit of the regular explicit finite volume scheme. This method uses special linear combinations of the solution and gradient on the Cartesian mesh to populate the so-called h-boxes. The h-box values are fed to the Riemann solvers for the cut cells, and replace the standard left and right states usually taken from the cells adjacent to a

cell edge.

For both the h-box simulations and the tests on triangular meshes, we need a time-stepping scheme, for which we use a simple second-order TVD Runge-Kutta method due to Gottlieb & Shu [16]. To solve $u_t = Lu$, the two stage method is given by:

$$\begin{aligned} u^{(1)} &= u^n + \Delta t L(u^n), \\ u^{n+1} &= \frac{1}{2}u^n + \frac{1}{2}\left(u^{(1)} + \Delta t L(u^{(1)})\right). \end{aligned} \tag{2.1}$$

In all these variations of the basic finite volume method, our limiter works without modification, once the choice of monotonicity constraints is made.

3. LP Limiting. This section contains the main contribution of this paper: we formulate the two-dimensional LPs used for limiting and discuss their efficient solution. Furthermore, we present two versions of the monotonicity constraints for cut cells, which we call the *standard formulation*, and the *relaxed formulation*, and discuss their pros and cons. We also present a positivity result. In the computational experiments we also show results for some additional possibilities, demonstrating that the LP limiter is really a limiter *framework* that can easily accomodate a variety of constraints.

For concreteness, however, before we discuss the monotonicity constraints we discuss the formation of the initial pre-limited gradient. We use a standard least squares approach [4], and solve an overdetermined linear system of equations. Consider a cut cell M with centroid (x_M, y_M) and cell value u_M that has 3 neighbors with centroids (x_i, y_i) and cell values u_i, $i = 1, \ldots, 3$, as shown in Figure 3.1. Then, the gradient (D_x, D_x) is given by finding the least squares solution to $\min \|\mathbf{r}\|_2$ where

$$\mathbf{r} = \begin{bmatrix} x_1 - x_M & y_1 - y_M \\ x_2 - x_M & y_2 - y_M \\ x_3 - x_M & y_3 - y_M \end{bmatrix} \begin{bmatrix} D_x \\ D_y \end{bmatrix} - \begin{bmatrix} u_1 - u_M \\ u_2 - u_M \\ u_3 - u_M \end{bmatrix}. \tag{3.1}$$

This formulation reconstructs linear functions exactly, and appears to be more accurate on distorted meshes than the Green-Gauss approach [3, 2]. As noted above, we would like to preserve the linearly exact property through the limiting step.

All edge neighbors of a cut cell are used in (3.1). However, if a cut cell has only two edge neighbors, the diagonal neighbor is also used to compute the gradient.

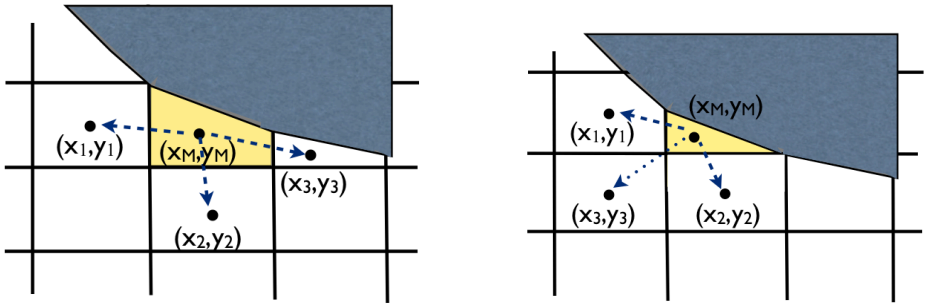


FIG. 3.1. *Slope reconstruction on yellow cell. Triangular cut cells with only two edge neighbors use diagonal cell too.*

3.1. LP Formulation. To improve accuracy and reduce the numerical diffusion, we limit the x and y components of the gradient (D_x, D_y) separately using scalars $\phi_x, \phi_y \in [0,1]$. The reconstructed solution in cell M can be written as linear function

$$u(x,y) = u_M + \begin{bmatrix}\phi_x D_x \\ \phi_y D_y\end{bmatrix} \cdot \begin{bmatrix}x - x_M \\ y - y_M\end{bmatrix}. \tag{3.2}$$

Following [8], the limiter is formulated as a constrained optimization problem with the goal of retaining as much of the unlimited gradient as possible while fulfilling the monotonicity conditions. Using the l^1-norm to measure the difference between the limited and unlimited gradient, the objective function is given by

$$\begin{aligned}\text{minimize } & |D_x - \phi_x D_x| + |D_y - \phi_y D_y| \\ & = (1-\phi_x)|D_x| + (1-\phi_y)|D_y| \\ & = -|D_x|\phi_x - |D_y|\phi_y + \text{ constant term},\end{aligned} \tag{3.3}$$

where the constant term is $|D_x| + |D_y|$. Formulating the constraints as linear inequalities in (ϕ_x, ϕ_y), we very naturally end up with an LP of the form

$$\min_{\phi_x,\phi_y} \; -|D_x|\phi_x - |D_y|\phi_y \qquad \text{subject to} \qquad A\begin{bmatrix}\phi_x \\ \phi_y\end{bmatrix} \geq b, \tag{3.4}$$

with $A \in \mathbb{R}^{k\times 2}$ and $b \in \mathbb{R}^k$.

We investigate two different formulations for the constraints in the following.

3.1.1. Monotonicity Constraints – Standard Formulation. The *standard* formulation can be viewed as a two-dimensional generalization of the *minmod* limiter. In one space dimension *minmod* has the nice geometric interpretation of limiting the gradient so that when reconstructing the solution in a cell to the neighboring centroid, it should not exceed the neighboring cell average (for increasing data, or lie below it for decreasing data).

Let $j = 1, \ldots, N$ be the neighbors of cell M where monotonicity conditions should be enforced. These include all the edge neighbors, and in the case of the h-box scheme on triangular cells, the diagonals too. Denote by (x_j, y_j) their centroid and by u_j the cell value. We want to enforce that the limited reconstruction when evaluated at the neighboring centroid be bounded by u_M and that centroid's value, i.e.

$$\min(u_M, u_j) \leq u_M + \begin{bmatrix}(x_j - x_M)D_x \\ (y_j - y_M)D_y\end{bmatrix} \cdot \begin{bmatrix}\phi_x \\ \phi_y\end{bmatrix} \leq \max(u_M, u_j), \text{ for } j = 1, \ldots, N. \tag{3.5}$$

This situation is shown in Figure 3.1(left). This is equivalent to

$$\text{if } u_M \leq u_j: \qquad -\begin{bmatrix}(x_j - x_M)D_x \\ (y_j - y_M)D_y\end{bmatrix} \cdot \begin{bmatrix}\phi_x \\ \phi_y\end{bmatrix} \geq -(u_j - u_M), \tag{3.6a}$$

$$\begin{bmatrix}(x_j - x_M)D_x \\ (y_j - y_M)D_y\end{bmatrix} \cdot \begin{bmatrix}\phi_x \\ \phi_y\end{bmatrix} \geq 0, \tag{3.6b}$$

$$\text{if } u_M > u_j: \qquad -\begin{bmatrix}(x_j - x_M)D_x \\ (y_j - y_M)D_y\end{bmatrix} \cdot \begin{bmatrix}\phi_x \\ \phi_y\end{bmatrix} \geq 0, \tag{3.6c}$$

$$\begin{bmatrix}(x_j - x_M)D_x \\ (y_j - y_M)D_y\end{bmatrix} \cdot \begin{bmatrix}\phi_x \\ \phi_y\end{bmatrix} \geq u_j - u_M. \tag{3.6d}$$

In other words, for every neighbor j we evaluate whether $u_M \leq u_j$ or whether $u_M > u_j$ and then add either inequalities (3.6a) and (3.6b) or inequalities (3.6c) and (3.6d) to the constraints used in the LP (3.4). For example, if we assume $u_M \leq u_1$, one row of A is given by $[-(x_1 - x_M)D_x, -(y_1 - y_M)D_y]$ with the corresponding entry in b equal to $-(u_1 - u_M)$. In addition, A and b also capture the constraints $0 \leq \phi_x, \phi_y \leq 1$. The dimension of A and b in (3.4) depends on the number of constraints one wants to enforce. Typically, k is on the order of 10 to 15.

3.1.2. Monotonicity Constraints – Relaxed Formulation. In one space dimension, the *monotonized central difference* (MC) limiter [31] is less diffusive than *minmod*. Geometrically, it reconstructs only to the cell edges, but still limits so that the solution does not exceed the neighboring centroid. A two-dimensional generalization of this would reconstruct to the midpoint of a cell edge and then limit using the neighboring centroid value. This is done e.g. in [6]. Note, however, on a non-coordinate-aligned grid, this approach can potentially limit even linear solutions. For example this can happen if the contour lines of the solution go from one cut cell center to the neighboring centroid. On a non-aligned mesh the edge midpoint does not lie on this line and is thus an extremum. Other situations where this can also happen are depicted in Fig. 3.2.

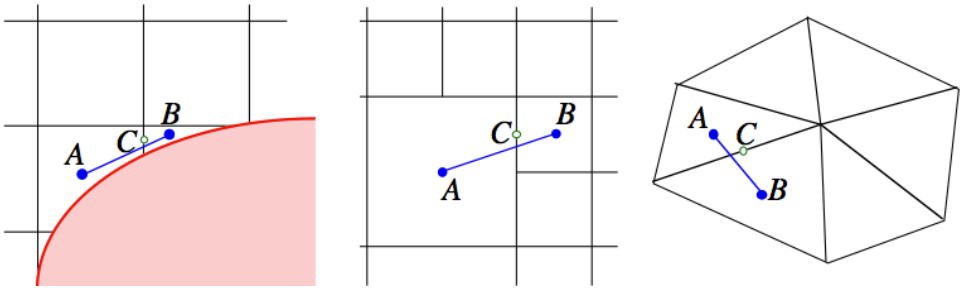


FIG. 3.2. In the three situations depicted here, assuming linear data, if points $\boldsymbol{A}$ and $\boldsymbol{B}$ lie on the same contour line, the edge midpoint $\boldsymbol{C}$ is an extremal point and will be limited (taken from [8]).

To reduce the likelihood of limiting a linear function, one could reconstruct to the edge midpoints but limit using the max and min over all neighboring cells, not just the solution from that edge neighbor as, e.g. suggested in [5]. Although this variant can be less diffusive, it can still lead to limiting of a linear solution if the edge midpoint does not lie in the convex hull spanned by the neighboring centroids. This typically occurs for example in cut cells with triangular shape.

Therefore, instead of reconstructing to the edge midpoint, for the *relaxed* formulation we will still reconstruct to the neighboring centroid but limit using the max and min over all neighboring cells. Contrasting with (3.5) the constraints are

$$\min(u_M, u_1, \ldots\ldots, u_N) \leq u_M + \begin{bmatrix}(x_j - x_M)D_x \\ (y_j - y_M)D_y\end{bmatrix} \cdot \begin{bmatrix}\phi_x \\ \phi_y\end{bmatrix} \leq \max(u_M, u_1, \ldots, u_N). \quad (3.7)$$

These monotonicity constraints again lead to an LP of the form (3.4) with a different matrix A of constraints. We will compare these two constraint formulations in the computational experiments in section 5.

3.1.3. Limiting at the boundary edge midpoint. In both formulations there is still the question about whether to limit at the boundary segment. By definition, a cut cell is at a boundary, so one has to deal with whether to apply a monotonicity constraint and limit at the boundary edge midpoint. One possible approach is to request that the reconstruction evaluated at the boundary edge midpoint lies in the range of values given by the neighboring cells and the central cell, as in the relaxed formulation. However, as succintly pointed out in [29], since the boundary edge midpoint does not lie in the convex hull of these points this will surely lead to limiting of linear functions.

We prefer not to put any constraints on the boundary edge midpoints other than ensuring positivity for certain variables such as density and pressure when simulating the Euler equations. For this purpose, the following inequality is added:

$$u_M + \begin{bmatrix} (x_{\text{bdy}} - x_M)D_x \\ (y_{\text{bdy}} - y_M)D_y \end{bmatrix} \cdot \begin{bmatrix} \phi_x \\ \phi_y \end{bmatrix} \geq 0, \tag{3.8}$$

where $(x_{\text{bdy}}, y_{\text{bdy}})$ denotes the boundary edge midpoint. Later in this section and in Appendix A we will discuss positivity more generally, and present a positivity result for a two-dimensional problem with a planar boundary that relies on adding this positivity constraint.

3.2. Solvability of LP (3.4)**.** Following standard slope limiting practice, all our LPs are formulated such that $(\phi_x, \phi_y) = (0, 0)$ is a feasible point, i.e., that the zero gradient fulfills all constraints. Therefore, there exists at least one feasible point. Together with the fact that the feasible domain is bounded due to $0 \leq \phi_x, \phi_y \leq 1$, there must exist a bounded optimal solution to our LPs.

3.3. LP Solution. There is a considerable amount of literature on solving a linear program. Most of it is aimed at LPs with hundreds of thousands of variables. Our LP problems are small, but need to be solved fast, since they are solved on every cut cell at every time step, or several times per step for a multi-stage scheme. Methods designed for large problems may not be the best here.

Two standard methods for solving LPs are the simplex method and the interior point method (see e.g. Nocedal & Wright [26]). To be thorough we tried both methods, but it is clear that the simplex method is more suitable, since:

- *Starting point:* For all monotonicity constraints considered here, the point $(\phi_x, \phi_y) = (0,0)$ is a corner of the admissible set and therefore a suitable starting point for the simplex method. It is usually nontrivial, however, to find a truly interior feasible starting point to be used for the interior point method.
- *Cost:* Each iteration of interior point is considerably more expensive than an iteration of the simplex method, and the interior point method did not take fewer iterations than the simplex method in our tests.

The standard form for the simplex method in most textbooks is given by

$$\min_x d^T x, \quad \text{subject to } Fx = g, \ x \geq 0. \tag{3.9}$$

Eq. (3.4) can be transformed to this standard form by means of 'slack variables' and the algorithm described in [26] can be used. However, this computation is fairly slow.

As suggested to us by Margaret Wright, there exists another version of the simplex algorithm that can be used only if the constraints consist exclusively of inequality

constraints. This version is especially useful if the number of inequality constraints is much larger than the number of variables, which is the case here. Mathematically, the all-inequality form (3.4) and the standard form (3.9) are equivalent, but the algorithm based on the all-inequality form is much faster for our problems. This version is apparently not well known – the only description we know of is in Gill et al. [15], which is currently out of print. Thus, we present the algorithm in Appendix B, along with specific choices and pricing strategies used in our simulations.

3.4. Computational Cost of LP Limiter. The cost of the LP limiter is essentially determined by the number of iterations needed to solve the linear program. In all the two-dimensional cases we have run, including the standard and relaxed formuations as well as those in the next two sections, the average number of iterations was between 2 and 3.5, and 2.5 iterations was typical. The maximum was always 6 or less, and we did not encounter any cycling issues for these small two-dimensional problems. For both the scalar and LP limiter one first computes a least squares gradient, so this part of the work is the same. For the problems we need to solve, one iteration of the simplex method is very inexpensive: it requires the solution of two two-by-two linear systems to determine the Lagrange parameter λ and the search direction, and then computes the step length by checking each constraint for a possible violation by the proposed step to be taken. This is more floating point computation but is approximately equivalent in terms of data motion to the scalar limiter. Also, the LP limiter is very cache friendly. Therefore we count one iteration of the simplex method as roughly equivalent to the computational cost of the scalar limiter, and the net cost of the LP limiter is 2.5 times the cost of the scalar limiter. Since our main goal is to apply the LP limiter to cut cells, which are usually a small fraction of the grid, this results in only a small increase in total computational time.

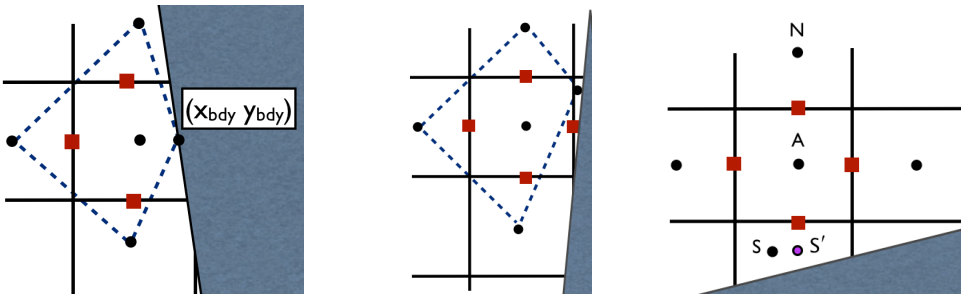
3.5. A Positivity Result. Since both the standard and the relaxed formulation use the neighboring centroids for limiting and not the edge midpoints, we can not conclude anything about the reconstructed value at the cell edges. One might worry that the value at the edge midpoint might be lower than at the adjacent centroids, and could therefore become negative. However, for cut cells in two dimensions, assuming a planar surface for the solid body, a straightforward (but very tedious) geometric analysis (not included here) shows that:

Lemma 1: All edge midpoints of a cut cell lie in the convex hull spanned by the neighboring centroids and the cut cell's boundary edge midpoint.

The setup for Lemma 1 is illustrated in Fig. 3.3(a). Since we use linear reconstruction, the maximum and minimum value over this convex hull must be attained at the corners. So for density and pressure for example, if all neighboring centroids have positive values, and the boundary edge midpoint is positive due to constraint (3.8), then the edge midpoints contained in the convex hull must also have positive values.

The corners of the cut cells may not be positive. Edge midpoint positivity, however, is sufficient for the positivity result proved in Appendix A. There, if the cell values at time t^n are positive we show that under certain conditions the value in a cut cell M is positive at time t^{n+1}.

Note that the convex hull used in Lemma 1 includes the boundary edge midpoint, since a positive constraint is applied there. Therefore this convex hull is different from the one considered in section 3.1.2.



(a) (b) (c)

FIG. 3.3. (a) shows a cut cell where the edge midpoints (marked in red squares) lie in the convex hull spanned by neighboring centroids and the boundary edge midpoint (marked with black circles). (b) shows a full Cartesian cell adjacent to a cut cell where the edge midpoint is not guaranteed to be positive without special limiting. (c) illustrates the re-centering idea for limiting these cells.

3.6. Treatment of Neighboring Cartesian Cells. The first uncut Cartesian cell *adjacent* to a cut cell has an irregular stencil and also needs special attention. Using the LP limiter on these cells is possible but not optimal. First, the same geometric analysis as used for Lemma 1 shows that for uncut Cartesian cells the edge midpoints are not always contained in the convex hull of neighbors. An example is given in Fig. 3.3(b). Therefore, additional positivity constraints would be needed. Second, solving an LP on these cells would roughly double the number of LPs to solve, since each cut cell is adjacent to a full Cartesian cell.

Instead, we use a *re-centering* idea [8, 23] to limit these cells and maintain positivity. We limit the cut cells first, and use the limited gradient in formulating the constraints for the adjacent full cell. We refer now to Fig. 3.3(c). To limit the y component of the gradient we treat the North and South face neighbors separately. Since the North face is uncut the centroids align, and we can build the standard one-sided difference quotient $(u_N - u_A)/\Delta y$. On the South face, however, the centroid S of the cut cell is not aligned with A. Instead of using S for limiting we would like to use the point S′ which has the same x coordinate as A. To recenter from S to S′ we make use of the already known and fully limited reconstruction in the cut cell. Then we can use the one-sided difference quotient $(u_A - u_{S'})/(y(A) - y(S'))$ and apply the *minmod* limiter to define the limited y-slope as

$$minmod\left(\frac{u_N - u_A}{\Delta y}, \frac{u_A - u_{S'}}{y(A) - y(S')}\right).$$

Even in the most extreme case of the slimmest triangle, the location of the re-centered point S' that aligns with the edge midpoint is inside the triangle and in particular within the convex hull that guarantees positivity (again assuming a planar boundary). Thus the solution at S′ is positive, since the cut cells have already been limited. Since A and S′ are both positive, the edge midpoint will be too. Note that this procedure does not guarantee that the solution at A when reconstructed to S will not overshoot, only that the reconstructed edge value is positive. In this way it is akin to the MC limiter at the cell edge rather than minmod. This procedure is also linearity preserving.

3.7. Monotonicity Constraints for the H-Box Method. The h-box method [17, 9] is an explicit time-dependent, fully second order finite volume scheme that

allows the use of a time step appropriate for the regular cells even when updating an arbitrarily small cut cell volume. Essentially it enlarges the domain of dependence by increasing the stencil of a cut cell update in a very particular way to maintain stability. Because of this increase in stencil, a cut cell can be used to compute the flux for a cell that is not an edge neighbor but is diagonally adjacent on a regular grid. Triangular cells in particular have only one neighbor in the x and y direction, so it often happens that the cut cell has not been limited in the direction in which it is used.

For example, in Fig. 3.4 the edge marked with an x uses the h-box state Q_ξ^L in solving the Riemann problem to compute the flux. This state is computed from a linear combination of the solution on the underlying Cartesian grid, in this case from cells A and M. The gradient ∇Q_ξ^L is also determined from a linear combination of the gradients in A and M. The problem is that the gradient in cell M only has one x face (shared with cell C) in the negative x direction that provides (most of) the limiting for the x component of the gradient. This is not sufficient to guarantee positivity at the flux edge between cells A and E, which is further away and in the positive x direction.

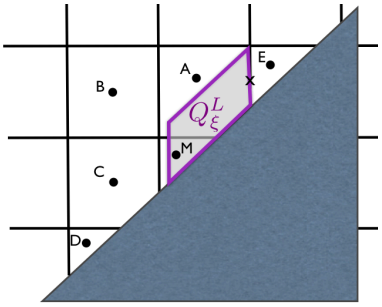
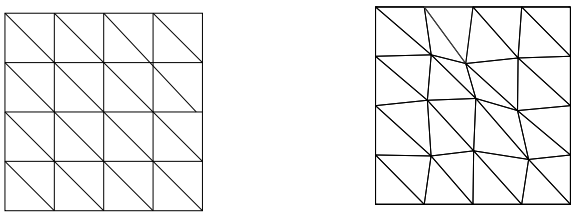


FIG. 3.4. The stencil for limiting the triangular cut cell M in the h-box method is enlarged to include neighbors D and E, even though they are not edge neighbors. This is because M and its gradient is used to compute the tangential component of the flux at the edge marked with x in cell E, and so more effective limiting than what is provided by cells A and C needs to be performed in that direction.

As a consequence, when using the h-box method on triangular cells in conjunction with the LP limiter and the standard formulation constraints, some of our tests did not maintain positivity. We note that the much more diffusive scalar limiter did maintain positivity at the edge midpoints of cell E. To ensure positivity at x, we add monotonicity constraints to the LP at triangular cut cells for any neighbor in the 3 by 3 neighborhood of cells surrounding cell M. In Fig. 3.4 this means that monotonicity equations for cells B, D and E are added to the constraints for cell M. The additional equations are of the same form as (3.6), exactly analogous to the conditions used for neighbors A and C. Note that cells D and E do not participate in the initial computation of the least squares gradient. This extra limiting for triangular cells works robustly in practice and was very easy to incorporate into our flexible LP framework.

4. Computational Results on Triangular Meshes. In this section we present numerical results on triangular meshes. The purpose of these tests is to confirm general accuracy and monotonicity properties of the LP limiter as well as to compare to some well-known methods from the literature [6, 21, 28]. The tests in section 4.1 demonstrate how the LP limiter can be viewed as a framework for a class of limiters rather than as one specific limiter. For the case of cut cell constraints, we have already discussed in sections 3.1.1 and 3.1.2 what choice we consider most suitable. Although we have not done as extensive tests, we believe that for triangular meshes a suitable choice of constraints depends strongly on the regularity of the grid. Thus, in this section we show the performance of the LP limiter for a variety of constraints.



(a) Grid A (b) Perturbed grid

FIG. 4.1. Two different grid types used in the triangular mesh tests - Grid A (from [21]), and a randomized version with interior vertices perturbed by $15\%\times$ the mesh width.

4.1. Smooth Test Case. We first examine the accuracy of the LP limiter. We solve the linear advection equation $u_t + \lambda_1 u_x + \lambda_2 u_y = 0$ on the unit square $[0,1]^2$ with periodic boundary conditions using the smooth test function

$$u(x,y) = \sin(2\pi x)\sin(2\pi y). \tag{4.1}$$

on what is called Grid A in the literature [21] (see Figure 4.1(a)). Following these references, we choose $\lambda = (-1, 2)$ and integrate until time $T = 1$. We use $\Delta t = 0.16\Delta x$ with Δx denoting the length of a horizontal or vertical triangle edge. This time step constraint does not satisfy the assumptions for the maximum principles in [6, 28], but it is stable and works fine for this smooth test case. (Results for $\Delta t = 0.05\Delta x$ are qualitatively very similar). We use the second order TVD RK scheme (2.1) for all tests, but point out that the Hancock scheme was used in [21] and a third order TVD RK scheme was used in [28]. In tests with the Hancock scheme we had the same qualitative results, but the latter results were more accurate since that scheme is less diffusive. In all calculations we use the least squares approach (3.1) to calculate the unlimited gradient.

We compare several variations of the scalar and LP limiter. To generalize our notation slightly, we test both the standard and relaxed formulation of the limiter using the adjacent centroid. In addition, we also test limiting at the edge midpoints, using both standard and relaxed formulations, since this is a commonly used limiting option. For this option the constraints are

$$u_{\text{lower}} \le u_M + \begin{bmatrix}(x_{j,edgemid} - x_M)D_x \\ (y_{j,edgemid} - y_M)D_y\end{bmatrix} \cdot \begin{bmatrix}\phi_x \\ \phi_y\end{bmatrix} \le u_{\text{upper}} \tag{4.2}$$

where triangle j shares an edge with triangle M, and $(x_{j,edgemid}, y_{j,edgemid})$ are the coordinates of the edge midpoint. For the standard formulation $u_{\text{lower}} = \min(u_M, u_j)$, and for the relaxed formulation $u_{\text{lower}} = \min(u_M, u_1, u_2, u_3)$ for all 3 neighbors. The upper bound takes the max instead of the min.

Finally, we include the MLP limiter (version u1) [28] as a representative example of newer limiters geared to triangles that fulfills a maximum principle. Using the above terminology, the MLP limiter evaluates the linear reconstruction at the 3 vertices of a triangle, and limits at each vertex using a scalar limiter so that it does not exceed the max or min of the solution in all triangles sharing that vertex. The final scalar limiter for the triangle is then chosen as the min of the three vertex limiters. This results in a relatively large neighborhood used in the limiting process.

TABLE 4.1

Results for grid A. Methods are ordered by performance. 'Sc' stands for scalar. 'Peak' of the solution is given as an indication of accuracy and numerical diffusion.

Scheme	Grid	L^1	order	L^∞	order	Peak
Sc-standard-edgemid	$64^2 \times 2$	4.07e-02	0.95	1.59e-01	0.66	0.8745
	$128^2 \times 2$	2.17e-02	0.91	1.07e-01	0.56	0.9372
	$256^2 \times 2$	1.19e-02	0.86	8.19e-02	0.39	0.9669
LP-standard-edgemid	$64^2 \times 2$	5.62e-03	1.86	4.20e-02	1.25	0.9634
	$128^2 \times 2$	1.42e-03	1.98	1.66e-02	1.34	0.9856
	$256^2 \times 2$	3.54e-04	2.01	6.42e-03	1.37	0.9944
MLP	$64^2 \times 2$	4.23e-03	2.10	3.40e-02	1.46	0.9648
	$128^2 \times 2$	9.78e-04	2.11	1.25e-02	1.44	0.9878
	$256^2 \times 2$	2.31e-04	2.08	4.38e-03	1.51	0.9958
Sc-relaxed-edgemid	$64^2 \times 2$	3.55e-03	1.97	1.54e-02	1.54	0.9842
	$128^2 \times 2$	8.85e-04	2.01	5.32e-03	1.53	0.9947
	$256^2 \times 2$	2.19e-04	2.01	1.90e-03	1.48	0.9982
LP-relaxed-edgemid	$64^2 \times 2$	3.53e-03	1.97	1.44e-02	1.54	0.9849
	$128^2 \times 2$	8.82e-04	2.00	5.05e-03	1.51	0.9948
	$256^2 \times 2$	2.19e-04	2.01	1.81e-03	1.48	0.9982
unlimited	$64^2 \times 2$	3.42e-03	1.99	5.52e-03	2.01	0.9979
	$128^2 \times 2$	8.57e-04	2.00	1.38e-03	2.00	0.9996
	$256^2 \times 2$	2.14e-04	2.00	3.45e-04	2.00	0.9999

Table 4.1 shows the results for various limiters on grid A. We only show results for reconstruction to the edge midpoint (except for the MLP limiter) since on this regular grid all edge midpoints lie on the lines connecting neighboring centroids. We note:

- 'LP-standard-edgemid' is significantly more accurate than 'Sc-standard-edgemid'. In particular the order of convergence differs,
- 'LP-relaxed-edgemid' is only slightly better than 'Sc-relaxed-edgemid' – but the latter is already very close to the unlimited case,
- the LP limiter is at least as accurate as the MLP limiter.

To make the grid less regular we randomly perturb all interior nodes: we shift their x- and y-coordinates by $0.15 \cdot h \cdot$ rand[-1,1] with rand[-1,1] denoting a random uniformly distributed number from the interval [-1,1]. A very coarse example of the resulting grid is shown in Figure 4.1(b). As a result of the pertubation, the edge midpoints typically no longer lie on the lines connecting neighboring centroids.

The results of all the same experiments on the perturbed grid are shown in Table 4.2. We note:

- 'LP-standard-edgemid' is no longer 2^{nd} order, since the limiter is no longer linearity-preserving. But it is still significantly more accurate than 'Sc-standard-edgemid',
- 'LP-standard-centroid' is linearity-preserving and shows close to 2^{nd} order convergence on finer grids. Note that on grid A, reconstructing to the centroid to limit was worse than the edge midpoint since the distances were about twice as large. Here, on the finer grids it is better to reconstruct to neighboring centroids and preserve linearity,
- both 'Sc-standard-edgemid' and 'Sc-standard-centroid' are only 1^{st} order,
- both 'Sc-relaxed-edgemid' and 'LP-relaxed-edgemid' are 2^{nd} order indicating

TABLE 4.2

Results for randomly perturbed grid. Methods are ordered by performance. 'Sc' stands for scalar. 'Peak' of the solution is given as an indication of accuracy and numerical diffusion.

Scheme	Grid	L^1	order	L^∞	order	Peak
Sc-standard-centroid	$64^2 \times 2$	6.60e-02	0.84	2.01e-01	0.76	0.8145
	$128^2 \times 2$	3.70e-02	0.83	1.12e-01	0.85	0.8947
	$256^2 \times 2$	2.07e-02	0.84	8.11e-02	0.46	0.9471
Sc-standard-edgemid	$64^2 \times 2$	4.60e-02	0.83	1.52e-01	0.72	0.8732
	$128^2 \times 2$	2.55e-02	0.85	1.17e-01	0.38	0.9295
	$256^2 \times 2$	1.44e-02	0.82	7.80e-02	0.59	0.9661
LP-standard-edgemid	$64^2 \times 2$	7.40e-03	1.60	4.25e-02	1.22	0.9625
	$128^2 \times 2$	3.08e-03	1.26	1.99e-02	1.09	0.9841
	$256^2 \times 2$	1.56e-03	0.98	1.29e-02	0.62	0.9936
	$512^2 \times 2$	8.49e-04	0.87	8.99e-03	0.52	0.9972
LP-standard-centroid	$64^2 \times 2$	2.03e-02	1.47	8.50e-02	1.18	0.9170
	$128^2 \times 2$	6.74e-03	1.59	3.64e-02	1.23	0.9645
	$256^2 \times 2$	1.96e-03	1.78	1.51e-02	1.27	0.9853
	$512^2 \times 2$	5.95e-04	1.72	6.20e-03	1.29	0.9940
MLP	$64^2 \times 2$	4.37e-03	2.10	3.54e-02	1.45	0.9637
	$128^2 \times 2$	1.00e-03	2.12	1.33e-02	1.42	0.9874
	$256^2 \times 2$	2.37e-04	2.08	4.83e-03	1.46	0.9957
Sc-relaxed-edgemid	$64^2 \times 2$	3.64e-03	1.96	1.67e-02	1.57	0.9833
	$128^2 \times 2$	9.07e-04	2.01	6.01e-03	1.47	0.9943
	$256^2 \times 2$	2.24e-04	2.02	2.07e-03	1.54	0.9980
LP-relaxed-edgemid	$64^2 \times 2$	3.64e-03	1.95	1.61e-02	1.59	0.9837
	$128^2 \times 2$	9.06e-04	2.01	5.71e-03	1.50	0.9945
	$256^2 \times 2$	2.24e-04	2.01	2.00e-03	1.51	0.9981
unlimited	$64^2 \times 2$	3.50e-03	1.98	6.21e-03	1.98	0.9983
	$128^2 \times 2$	8.76e-04	2.00	1.57e-03	1.98	0.9997
	$256^2 \times 2$	2.19e-04	2.00	4.10e-04	1.94	1.0000

that all edge midpoints are contained in the convex hull of the neighboring triangles' centroids. The results are again very similar for these two limiters,

- MLP has a similar place in the ranking as for grid A.

In sum, these tests confirm that the LP limiter results in 2^{nd} order convergence for suitable choices of constraints. It gives significantly better results than the scalar limiter for the relatively restrictive constraints (compare the 'standard'-versions). For the relaxed constraints, for which the scalar limiter is already relatively close to the unlimited case on these relatively regular grids, we only see a slight improvement using the LP limiter.

4.2. Discontinuous Test Case. For a discontinuous test case we advect a two-dimensional square wave function

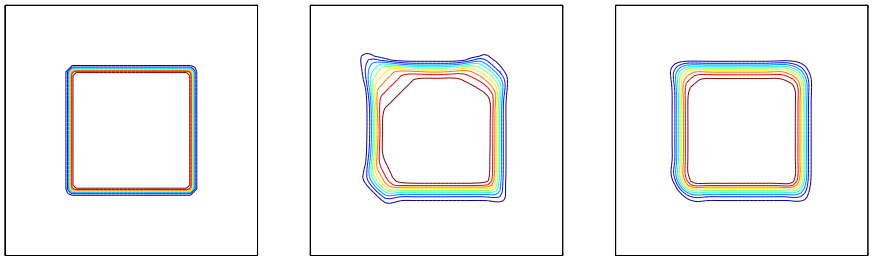
$$u(x,y) = \begin{cases} 1 & \text{if } (0.25 \le x \le 0.75) \ \& \ (0.25 \le y \le 0.75), \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

The test setup is exactly the same as for the smooth test function. We use a time step that satisfies the maximum principle in [6] of $\Delta t = 0.08\Delta x$. We only show results for grid A here. The results for the randomly perturbed grid are qualitatively very similar.

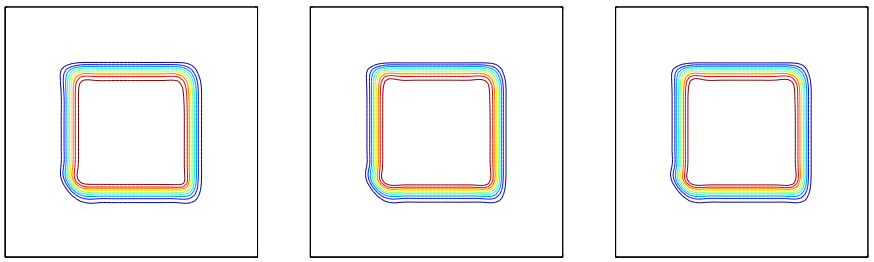
TABLE 4.3

Results for grid A. Methods are ordered according to performance. 'Sc' stands for scalar. 'Min/Max' of the solution is given as an indication of numerical diffusion and robustness.

	$16^2 \times 2$		$32^2 \times 2$	
Scheme	min	max	min	max
unlimited	-0.1030	1.1727	-0.1139	1.1678
Sc-standard-edgemid	0.0004	0.8867	0.0000	0.9948
MLP	0.0003	0.9628	0.0000	0.9998
LP-standard-edgemid	0.0000	0.9836	0.0000	1.0000
Sc-relaxed-edgemid	0.0000	0.9984	0.0000	1.0000
LP-relaxed-edgemid	0.0000	0.9990	0.0000	1.0000



(a) exact solution (b) sc-strict-edgemid (c) LP-strict-edgemid



(d) MLP (e) sc-mm_neigh-edgemid (f) LP-mm_neigh-edgemid

FIG. 4.2. Solutions for discontinuituous test case on grid A with $2 \cdot 64^2$ grid cells. Contour lines go from .1 to .9 by .1.

Table 4.3 shows the minimum and maximum value we observed for a variety of limiters on grid A. All limiters satisfy (as expected) the maximum principle (except of course the unlimited case which has large overshoots). The 'LP-standard-edgemid' leads to slightly better extrema than 'MLP'. These results support the same ordering of the limiters by performance as in the previous experiments.

Finally, Fig. 4.2 shows contour lines of the solution at $T = 1$ on a $64^2 \times 2$ grid for the same set of limiters. As expected, 'LP-standard-edgemid' is significantly better than 'Sc-standard-edgemid'. The plots for 'MLP', 'Sc-relaxed-edgemid', 'LP-relaxed-edgemid' are very comparable.

5. Computational Results on Cut Cell Meshes. This section presents tests using the embedded boundary method described in section 2, solving the two-dimensional Euler equations. First we show results for a smooth steady-state test case, again comparing the accuracy of the LP limiter to the scalar limiter. We then use the LP limiter in a time-dependent simulation of shocked flow diffracting around a cylinder. For this case we use the h-box method in the cut cells, with the LP limiter applied on the underlying Cartesian grid cut cells. In both cases the limiting is performed on the primitive variables, not the conserved variables, and the van Leer Riemann solver is used.

5.1. Smooth Test Case: Supersonic Vortex. We consider the case of inviscid, isentropic, supersonic flow between concentric arcs as presented in Aftosmis et al. [2]. Since the flow is shock free and there is an analytic solution, we use this problem to measure the accuracy of the LP limiter in the cut cells.

The cut cell grid is shown in Fig. 5.1. The simulations are initialized with the exact solution and run to steady state using a multi-stage Runge-Kutta scheme with local time stepping, so the scheme is stable in the cut cells. We use the same parameters as [2], $M_i = 2.25$, inner radius 1.0, outer radius 1.43, and CFL number .9.

This problem does not actually need to be limited. We use it to measure how much the solution accuracy degrades when limiters are turned on. We only limit at the cut cells. If all the Cartesian cells were limited the interior error would dominate, and we would not be able see the effect of using different limiters in the cut cells. Also, since the solution is determined completely from the inflow conditions, we specify the exact fluxes into the first column of cells, and do not limit these either. For the gradient reconstruction on fully regular Cartesian cells with only Cartesian neighbors we use a central difference formula. On Cartesian cells adjacent to cut cells the least squares formulation is used.

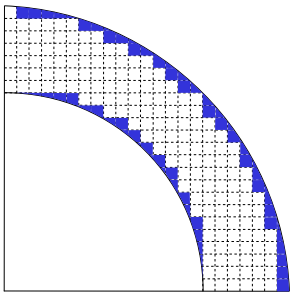


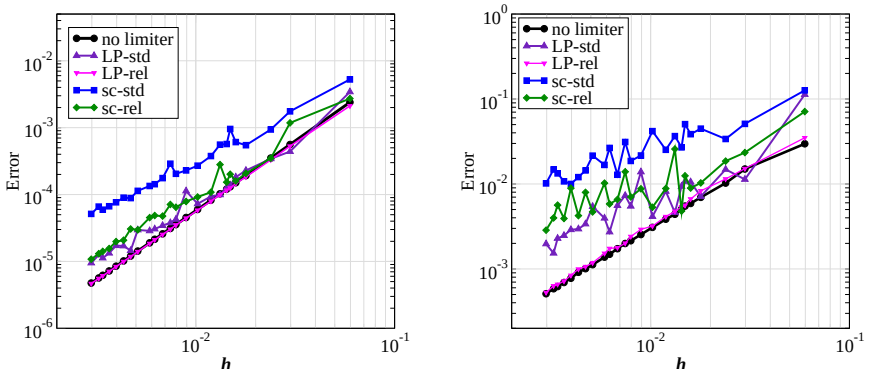
FIG. 5.1. The circular boundary is approximated by a single line segment in each cut cell. Only the interior cut cells are limited in the following tests, using either scalar or LP limiting.

For the shaded cut cells of Fig. 5.1 we use either the scalar or LP limiter, and compare to the unlimited results. Two different sets of constraints are again tested for both the scalar and LP limiter:

- the 'standard' formulation, defined by (3.5),
- the 'relaxed' formulation, defined by (3.7).

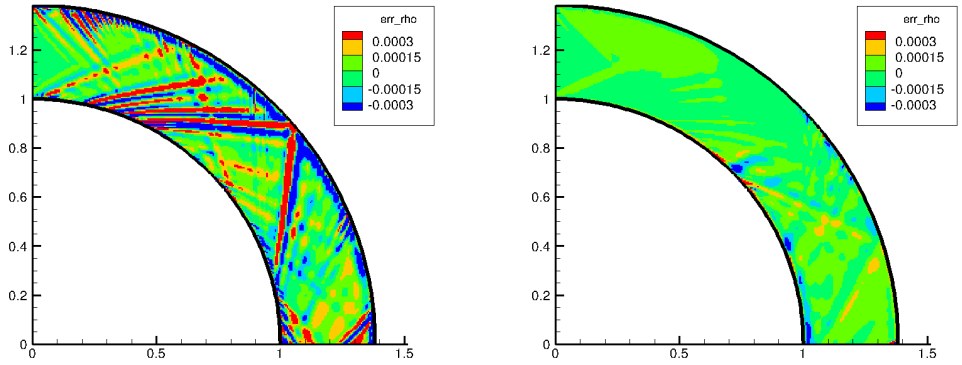
We also include a positivity constraint at the boundary segment. Thus, when applying the LP limiter, each LP has $4 + 2N + p$ constraints, where N is the number of neighboring cells used for limiting and p is 1 for density and pressure and 0 for the velocity variables.

Figure 5.2 shows the results in the L^1 norm and L^∞ norm. In both norms, the LP limiter is considerably better than the scalar limiter using the same set of constraints. The 'LP-standard' version is about 4-5 times more accurate than the 'scalar-standard' version. The 'LP-relaxed' version is on average about twice as accurate as the 'scalar-relaxed' version in L^1 and 4 times more accurate in L^∞. Figure 5.2 also shows that the 'LP-relaxed' version is very comparable to the unlimited version. Note in the error



(a) L^1 error (b) L^∞ error

FIG. 5.2. Error in density over the domain for standard and relaxed formulation of LP limiter. The mesh width h on the horizontal axis denotes the length/height of a Cartesian cell. The unlimited results show second order convergence in L^1, and $h^{3/2}$ in the max norm.



(a) scalar limiter (b) LP limiter

FIG. 5.3. Error in density using the scalar limiter (a) and LP limiter (b) in the standard formulation. Only the cut cells (except first inflow cell) are limited. The grid uses $h = (1.43/244) \approx 0.0059$. Red represents overshoot, blue undershoot and green negligible error.

plots how the error is not smooth, which is expected since the grid is not smooth, and there is no asymptotic error expansion for cut cells.

Figure 5.3 shows the actual error distribution for density for the scalar and LP limiters, both using the 'standard' formulation. Red represents overshoots, blue undershoots and green is negligible error with the same scale being used for both pictures. As one can clearly see, the errors bounce between the two arcs with the error for the scalar being considerably bigger than for the LP limiter version.

A frequently discussed issue with limiters for steady-state problems is the problem of limiter chatter. When limiters start rattling convergence is stalled. Unfortunately, the LP limiter still suffers from limiter chatter, although it kicks in later than for the scalar limiter. (In previous work [8] we also found that more accurate numerics somewhat improves convergence and delays the onset of chatter). The paper by Venkatakrishnan [32] attributes limiter chatter to a conflict between monotonicity and convergence to a steady-state. We see some of this with the LP limiter. If we loosen

TABLE 5.1
Error in density using reconstruction to either the neighboring centroid or edge midpoint, both using the 'standard' formulation.

	$h = 1.43/48$		$h = 1.43/60$	
Scheme	L^1	L^∞	L^1	L^∞
LP – recon. to neighboring centroid	5.08e-04	1.27e-02	3.97e-04	1.55e-02
LP – recon. to edge midpoint	1.21e-03	8.30e-02	5.78e-04	5.65e-02

the constraints and allow some relative overshoot of the order of 10^{-3} convergence improves, but at the cost of allowing new extrema. Also the LP limiter is not smooth.

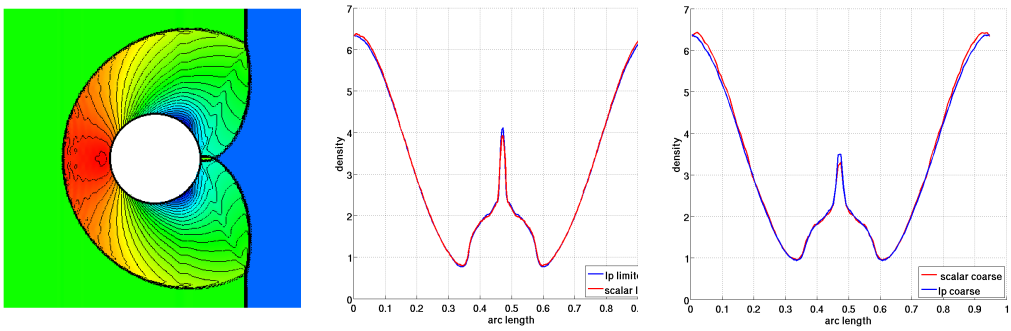
We perform one final test comparing reconstruction to neighboring centroids vs. edge midpoints. Table 5.1 shows results on two grid sizes, both showing that reconstructing to centroids is somewhat better than reconstructing to edge midpoints. On the coarser grid the difference is actually more pronounced.

5.2. Discontinuous Test Case: Shock Diffraction from a Cylinder. We next consider the behavior of the LP limiter with discontinuities. We consider shock reflection from a cylinder using the same setup described in [17]. The domain is $[0, 1]^2$, the cylinder has a radius of $r = 0.15$ and is centered at $(0.5, 0.5)$. A Mach 2 shock starts at $x = 0.2$. The state in front of the shock is given by $\rho = 1.4$, $u = v = 0$, $p = 1$. The mesh for this computation was 302 by 302, and there were 364 cut cells around the cylinder. For this problem all cells must be limited. We use MC limiter in the regular Cartesian cells, and compare the use of scalar and LP limiter in the cut cells employing the 'standard' constraints for both. The positivity constraint is included for density and pressure in the cut cells.

Figure 5.4 shows contour lines of density at t=0.30 using the LP limiter. Overall, the flow field looks identical since the MC Leer limiter is used over most of the domain, and only the cut cell limiting differs. For a more detailed comparison, we plot the values of density as a function of arc length around the boundary of the cylinder, on two grid sizes. The curve starts at the high density region in front of the cylinder and goes clockwise until it loops back to the starting cell. The fine grid results are shown in the middle of Fig. 5.4, where both the LP and scalar limiter are plotted. The maximum value at the peak in the LP solution is 4.11. For the scalar limiter it is 3.90. This is an improvement of over 5%, for this small highly-peaked feature. The figure on the right shows results from a coarser grid with mesh width $h \approx .008$. Here the difference between the limiters is even greater, and is slightly over 6%.

We also notice that the minimum density in the LP computations is also slightly lower than the scalar versions. The high density in front of the cylinder would appear to be the only place that the scalar limiter attains a higher value. However, comparison to results from [17] as well as with results using even finer grids shows that the LP solution is more accurate in this region too, and the scalar version converges to the lower value of the maximum density here, with the single peak symmetrically located.

6. Conclusions. We have developed a vector limiter in two space dimensions for non-coordinate-aligned meshes based on solving a tiny LP in every cell. The all-inequality simplex method suggested in this paper is a very efficient way of solving LPs with this structure. Overall, the LP limiter is roughly 2-3 times as expensive as the scalar limiter. Limiting the x and y slope independently instead of using a scalar limiter significantly increases the accuracy of the solution. In our tests, the errors using the LP limiter are a factor of 2-5 times smaller than when using the



(a) density contours (b) fine grid (c) coarse grid

FIG. 5.4. Simulation of shock diffraction around a cylinder: (a) shows density contours for the LP limiter on a fine grid. (b) and (c) show the density as a function of arc length around the cylinder on a fine and coarse grid, respectively. The LP limiter gives density at the sharp peak that is 5% higher than the scalar limiter on the fine grid, and 6% higher on the coarser grid.

scalar limiter depending on the specific test and the monotonicity conditions chosen. The main application for the LP limiter is to the highly irregular cut cells arising for Cartesian embedded boundary meshes. For this case, we also proved a positivity result for a planar boundary. Nevertheless, we have shown that the LP limiter works well on triangular grids, too. One of the main strengths of the new LP limiter is its flexibility in choice of monotonicity constraints, since it easily permits changing constraints and adjusting the degree of limiting to the specific problem at hand.

The next step is to extend this to three space dimensions. In principle this should be straightforward since we only increase the number of variables from two to three in our LPs and add constraints for the additional neighbors. We can still use the same all-inequality simplex method for their solution. Initial tests have found occasional problems with cycling, however, which did not arise in two dimensions. The literature has strategies (such as 'Bland's rule' [13, 15]) for dealing with this, which we will investigate. Also, since cut cell geometry in three dimensions is significantly more complicated than in two dimensions, we might need to add additional constraints to guarantee positivity. Fortunately, our framework is general enough to do this at little additional cost. We expect the resulting improvement in accuracy, due to limiting each of the three coordinate directions separately, to easily compensate for these complications.

Acknowledgments. The authors would like to thank Margaret Wright for very helpful discussions and introducing us to the all-inquality simplex method.

Appendix A. Sketch of Positivity Proof for Cut Cells.

Following the approach of Batten et al. [6] showing positivity for linear advection on triangular grids, we sketch a positivity result for cut cells with LP limiting. We consider the two-dimensional linear advection equation $u_t + \lambda_1 u_x + \lambda_2 u_y = 0$. Let M be a cut cell, where

(i) the gradient on cell M is limited using:
- the positivity constraint (3.8) at the boundary edge midpoint,
- neighboring centroids (not edge midpoints) using either the 'standard' or the 'relaxed' version discussed in sections 3.1.1 and 3.1.2,

(ii) the embedded boundary is planar,
(iii) the time step satisfies the CFL-like condition

$$\Delta t \leq \frac{V_M}{6 \sum_{j:\lambda \cdot n_j \geq 0} \lambda \cdot n_j \Delta e_j}, \tag{A.1}$$

where V_M denotes the volume of cell M, n_j denotes the unit normal vector of the j-th edge of cell M and Δe_j the length of that edge.

Then we can show:

Lemma 2: If the data at time t^n are positive, i.e. $u_i^n \geq 0 \, \forall \, i$, then $u_M^{n+1} \geq 0$.

Proof. We show this in two parts, and make use of Lemma 1, which states that the LP limiting procedure results in positive values at the edge midpoints for the limited linear reconstruction $\hat{u}_M(x, y) = u_M + (x - x_M)\phi_x D_x + (y - y_m)\phi_y D_y$.

Part 1: We show

$$M_L := \max_i \hat{u}_M(\mathbf{m_i}) \leq 6 \, u_M^n, \quad i = 1, \ldots, k, \tag{A.2}$$

where $\mathbf{m_i}$ is the midpoint of the i^{th} edge, and cell M has k edges including the boundary edge.

The main idea for deducing this (non-optimal) bound becomes clear when considering Fig. A.1: We know by assumption and Lemma 1 that the reconstruction $\hat{u}_M$ is positive at all neighboring centroids and all edge midpoints, including the boundary edge midpoint. If $\hat{u}_M(\mathbf{m_i})$ grew too big for one of the edge midpoints, then $\hat{u}_M$ would become negative on the 'opposite' side and would violate a positivity constraint. For example, we can deduce from centroids u_2 and u_3 that the reconstruction is positive at the left upper corner of the cut cell. Together with the values at points $\mathbf{m_2}$ and $\mathbf{m_3}$ being positive, this implies that the reconstruction is positive at the line marked in yellow. For a 5-sided cell as shown in the figure (3- and 4-sided cells are treated similarly) $x_M - x(\mathbf{m_2}) \geq \frac{1}{3}\Delta x$, where x_M and $x(\mathbf{m_2})$ are the x-coordinates of the cut cell centroid and the point $\mathbf{m_2}$ respectively. Furthermore, $y(\mathbf{m_3}) - y_M \leq \frac{1}{2}\Delta y$, i.e., when projecting the cut cell centroid onto the left cell edge, it would lie on the yellow line. Therefore, the x-component of the gradient is bounded above by $3u_M/\Delta x$, since otherwise the positivity on the yellow line would be violated. Similarly, we can bound the y-component of the gradient from below by $-3u_M/\Delta y$. Getting a lower bound on the x slope and upper bound on the y slope is more tricky and usually requires considering several constraints at the same time (e.g., $\mathbf{m_1}$ and $\mathbf{m_5}$). Continuing in this fashion with a detailed analysis of the 3-, 4-, and 5-sided cut cells, the

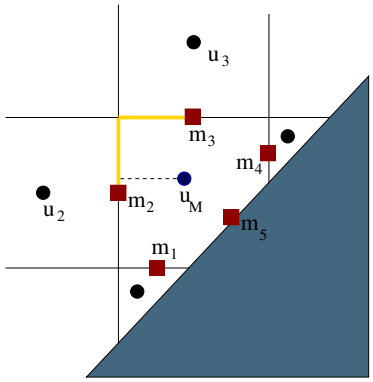


FIG. A.1. Geometric arguments lead to a bound for $\max_i \hat{u}_M(\mathbf{m_i})$. The linear function $\hat{u}_M$ is non-negative at all red squares and black circles by assumption. Therefore its gradient cannot be too step or it will violate one of the positivity constraints in the other direction.

bound (A.2) can be derived.

Part 2: The standard finite volume update for scalar advection is

$$u_M^{n+1} = u_M^n - \frac{\Delta t}{V_M} \sum_{j=1}^{k} F_j(\hat{u}_M(\mathbf{m_j}), \hat{u}_j(\mathbf{m_j}))$$

where $\hat{u}_M$ and $\hat{u}_j$ denote the limited reconstruction on cell M and neighboring cell j, and

$$F_j(u_L, u_R) = \begin{cases} u_L \, \lambda \cdot n_j \Delta e_j & \text{if } \lambda \cdot n_j \geq 0, \\ u_R \, \lambda \cdot n_j \Delta e_j & \text{otherwise.} \end{cases}$$

Using that $\hat{u}_j(\mathbf{m_j}) \geq 0, j = 1, \ldots, k$, and the bound (A.2) on M_L we get

$$\begin{aligned} u_M^{n+1} &\geq u_M^n - \frac{\Delta t}{V_M} \sum_{j=1}^{k} F_j(\hat{u}_M(\mathbf{m_j}), 0) \\ &= u_M^n - \frac{\Delta t}{V_M} \sum_{j:\lambda \cdot n_j \geq 0} \hat{u}_M(\mathbf{m_j}) \, \lambda \cdot n_j \Delta e_j \\ &\geq u_M^n - M_L \frac{\Delta t}{V_M} \sum_{j:\lambda \cdot n_j \geq 0} \lambda \cdot n_j \Delta e_j \\ &\geq u_M^n \left[1 - 6 \frac{\Delta t}{V_M} \sum_{j:\lambda \cdot n_j \geq 0} \lambda \cdot n_j \Delta e_j \right]. \end{aligned}$$

The time step constraint (A.1) on Δt then guarantees that $u_M^{n+1} \geq 0$. □

The theorem given in [6] for triangles is overall very similar to Lemma 2 but uses limiting at the edge midpoints and has the time step constraint

$$\Delta t \leq \frac{V}{3 \max_j |\lambda \cdot n_j \Delta e_j|}. \tag{A.3}$$

The main differences in the proofs are:

- Bound for $\max_i \hat{u}(\mathbf{m_i})$: [6] exploits the midpoint quadrature rule for triangles (i.e., that for a linear function the average of the values at edge midpoints equals the cell average, and the fact that $\hat{u}(\mathbf{m_i}) \geq 0$ for all i) to deduce $\max_i \hat{u}(\mathbf{m_i}) \leq 3 \, u_M^n$. For cut cells we have to go through a complicated geometric analysis;
- Sum: by the divergence theorem, the term $\sum_{j:\lambda \cdot n_j \geq 0} \lambda \cdot n_j \Delta e_j$ in the time step constraint can be replaced by $\max_j |\lambda \cdot n_j \Delta e_j|$ for triangular cells. This is not possible for cut cells, which can have 4 or 5 sides as well.

Appendix B. All-inequality Simplex Method. A wonderful, detailed description of the all-inequality simplex method is found in Gill et al. [15]. Since that book is out of print, and since this version of the simplex algorithm is not well known, we summarize the steps of the method and give specifics about our implementation.

The idea is the same as in the regular simplex algorithm: If there is a (bounded) solution to the linear program, then there must exist a vertex in the feasible region that achieves that solution. We start examining a vertex, and go from vertex to

vertex until we end up at the optimal one. The objective function value has to be non-increasing during that procedure. The all-inequality form (3.4) can be mathematically transformed to the standard form (3.9) using slack variables, but this greatly increases the size of the matrix to be inverted at each iteration. It is this smaller matrix size that makes the all-inequality version more efficient.

Consider an LP in the all-inequality standard form:

$$\min_x \; c^T x \quad \text{subject to} \quad Ax \geq b,$$

where $c, x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $n \leq m$, and $\text{rank}(A) = n$. In our case n=2 and m is typically 10–15. The simplex method is an iterative algorithm with

$$x_{k+1} = x_k + \alpha_k \, p_k, \tag{B.1}$$

where the step length α_k and the descent direction p_k need to be determined. To explain the algorithm, we introduce some notation:

- *Working set $\mathcal{W}_k$:* Set of (exactly) n indices, each of which is the index of a constraint that is active at x_k.
- *Working set matrix $\mathcal{A}_k$:* $n \times n$ matrix consisting of rows numbered $\omega_1, \ldots, \omega_n$ of A, where $\mathcal{W}_k = \{\omega_1, \ldots, \omega_n\}$. We denote this

$$\mathcal{A}_k = \begin{bmatrix} a_{\omega_1}^T \\ \vdots \\ a_{\omega_n}^T \end{bmatrix}, \quad \text{and} \quad \mathbf{b}_k = \begin{bmatrix} b_{\omega_1} \\ \vdots \\ b_{\omega_n} \end{bmatrix},$$

  where a_i^T denotes the i^{th} row of A. $\mathcal{A}_k$ needs to be nonsingular.
- *Set of decreasing constraints:* For a given descent direction p_k, the set of decreasing constraints $\mathcal{D}_k$ is defined by

$$\mathcal{D}_k = \{i : a_i^T p_k < 0\}.$$

A vertex of the feasible set is *nondegenerate* if exactly n constraints are active at this vertex. In that case, the working set $\mathcal{W}_k$ is uniquely determined. If more than n constraints are active the vertex is degenerate. This is the case for our LPs, and it can lead to problems. We discuss this issue later.

One iteration of the all-inequality simplex algorithm is given by:

ALGORITHM B.1. *Let x_k be a vertex of the feasible set satisfying $Ax_k \geq b$ and let $\mathcal{W}_k$ be a working set such that $\mathcal{A}_k$ is nonsingular and $\mathcal{A}_k x_k = \mathbf{b}_k$.*

1. *Calculate the Lagrange multipliers $\lambda_k \in \mathbb{R}^n$ by solving $\mathcal{A}_k^T \lambda_k = c$.*
2. *If $\lambda_k \geq 0$, STOP. In this case, the point x_k is optimal.*
3. *Otherwise select q such that $(\lambda_k)_q < 0$. This constraint will be removed from $\mathcal{W}_k$. (Details on which q to choose are given below.)*
4. *Calculate the descent direction p_k from $\mathcal{A}_k p_k = e_q$, $e_q = q^{th}$ coordinate vector.*
5. *Choose the step length α_k to be taken along p_k:*
   (a) *Find the set of decreasing constraints along p_k*

$$\mathcal{D}_k \leftarrow \{i : a_i^T p_k < 0\}$$

   (Note that by the definition of p_k : $a_i^T p_k = 0 \;\; \forall i \in \mathcal{W}_k$, $i \neq q$.)
   (b) *If $\mathcal{D}_k = \emptyset$, STOP. In this case, the problem is unbounded. (This will never be the case for our special LPs, since the constraints $0 \leq \phi_x, \phi_y \leq 1$ guarantee a bounded feasible domain.)*

(c) *For all $i \in \mathcal{D}_k$ calculate the maximum step length γ_i one can take before violating constraint i:*

$$\gamma_i \leftarrow \frac{a_i^T x_k - b_i}{-a_i^T p_k}.$$

(d) *Calculate the largest step length possible that doesn't violate any constraints as*

$$\alpha_k \leftarrow \min_{i \in \mathcal{D}_k}(\gamma_i)$$

6. *Update the working set*
   (a) $x_{k+1} \leftarrow x_k + \alpha_k p_k$
   (b) *Determine the set of blocking constraints $\mathcal{S}_k$ which contains all indices i for which $\gamma_i = \alpha_k$.*
   (c) *From $\mathcal{S}_k$, choose a constraint t to be added to $\mathcal{W}_k$*
   (d) *Update the working set: $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k - \{q\} + \{t\}$*
   (e) *Update the working set matrix: $\mathcal{A}_{k+1} \leftarrow \mathcal{A}_k$ with row t replacing row q of A.*
   (f) *Update iteration count: $k \leftarrow k + 1$*

To fully specify the algorithm we need to include:

- *Starting point:* Usually, it is non-trivial to find a starting point for this iteration. For our specific LPs, however, we know that $x_0 = (0, 0)$, corresponding to the fully limited, zero gradient, (i.e. a first order update), is a valid starting point. We initialize $\mathcal{W}_0$ with the indices corresponding to the conditions $x_0^1, x_0^2 \geq 0$. The corresponding matrix $\mathcal{A}_0$ is therefore the identity matrix and in particular regular. Note that depending on how we define the monotonicity constraints, we might start out at a degenerate vertex. Any constraint of the form (assume WLOG $u_M \leq u_j$)

$$\begin{bmatrix}(x_j - x_M)D_x \\ (y_j - y_M)D_y\end{bmatrix} \begin{bmatrix}\phi_x \\ \phi_y\end{bmatrix} \geq 0 \tag{B.2}$$

  will be active at this starting point.
- *Constraint deletion/addition:* For the constraint deletion, we take the index corresponding to the most negative component of λ_k. The constraint addition is uniquely defined if the vertex x_{k+1} is non-degenerate. However, this is in general not the case for us. So if several constraints (that are currently not in $\mathcal{W}_k$) happen to block the step at exactly the same maximum step length α_k, we choose the one with the lowest index to enter the working set for $\mathcal{W}_{k+1}$.
- *Cycling/Degenerate vertices:* The main problem with degenerate LPs is that they could encounter 'cycling': The search direction is computed such that it does not violate any of the constraints in the working set $\mathcal{W}_k \setminus q$. However, if a constraint j is active but not in the working set (which happens for a degenerate vertex), one can end up with a search direction violating that constraint. Therefore α_k will be determined to be zero, j will be added to the working set and one index will be removed. This can lead to a cycle. We note that even though our LPs are degenerate, we have not seen any instances of cycling in our two-dimensional test problems.
- *Numerical issues:* The search direction in Step 4 of Algorithm B.1 mathematically guarantees that $a_i^T p_k = 0$ for $i \in \mathcal{W}_k$, $i \neq q$. Consequently,

$i \in \mathcal{W}_k, i \neq q$, should never qualify for the set $\mathcal{D}_k$ calculated in Step 5a. Numerically, however, $a_i^T p_k = \delta$ with δ small but not identically zero. When calculating the set $\mathcal{D}_k$, we therefore don't check whether $a_i^T p_k < 0$ but whether $a_i^T p_k < -10^{-11}$. For all two-dimensional test problems, this was sufficient to ensure that an index $i \in \mathcal{W}_k$ that was supposed to stay in the working set does not qualify for the set $\mathcal{D}_k$.

REFERENCES

[1] R. Abgrall. On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation. *J. Comput. Phys*, 114:45–58, 1994.

[2] M. Aftosmis, D. Gaitonde, and T. Sean Tavares. On the accuracy, stability and monotonicity of various reconstruction algorithms for unstructured meshes. In *32nd AIAA Aerospace Sciences Meeting, Reno, NV*, 1994. Paper AIAA 94-0415.

[3] T. J. Barth. Numerical aspects of computing viscous high Reynolds number flows on unstructured meshes. 1991. Paper AIAA 91-0721.

[4] T. J. Barth. Numerical methods and error estimation for conservation laws on structured and unstructured meshes. *Van Karman Institute Computational Fluid Dynamics Lecture Notes*, 2003.

[5] T. J. Barth and D. Jespersen. The design and application of upwind schemes on unstructured meshes. In *27th AIAA Aerospace Sciences Meeting, Reno, NV*, 1989. Paper AIAA 89-0366.

[6] P. Batten, C. Lambert, and D. M. Causon. Positively conservative high-resolution schemes for unstructured elements. *Int. J. Numer. Methods Eng.*, 39:1821–1838, 1996.

[7] 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.*, 74:362–397, 1988.

[8] M. Berger, M. J. Aftosmis, and S. M. Murman. Analysis of slope limiters on irregular grids. In *43rd AIAA Aerospace Sciences Meeting, Reno, NV*, 2005. Paper AIAA 2005-0490.

[9] M. Berger and C. Helzel. A simplified h-box method for embedded boundary grids. *Siam J. Sci. Comput.*, 34, 2012.

[10] T. Buffard and S. Clain. Monoslope and multislope muscl methods for unstructured meshes. *J. Comput. Phys.*, 229:3745–3776, 2010.

[11] G. Chavent and J. Jaffré. *Mathematical Models and Finite Elements for Reservoir Simulation.* North-Holland, Amsterdam, 1986.

[12] I. Christov and B. Popov. New non-oscillatory central schemes on unstructured triangulations for hyperbolic systems of conservation laws. *J. Comput. Phys.*, 227:5736–5757, 2008.

[13] V. Chvatal. *Linear Programming.* W. H. Freeman and Company, 1983.

[14] O. Friedrich. Weighted essentially non-oscillatory schemes for the interpolation on unstructured grids. *J. Comput. Phys.*, 144:194–212, 1998.

[15] P. E. Gill, W. Murray, and M. H. Wright. *Numerical Linear Algebra and Optimization*, volume 1. Addison-Wesley Publishing Company, 1991.

[16] S. Gottlieb and C. Shu. Total variation diminishing Runge-Kutta schemes. *Math. Comput.*, 67:73–85, 1998.

[17] C. Helzel, M. Berger, and R. Leveque. A high-resolution rotated grid method for conservation laws with embedded geometries. *Siam J. Sci. Comput.*, 26:785–809, 2005.

[18] H. Hoteit, P. Ackerer, R. Mose, J. Erhel, and B. Philippe. New two-dimensional slope limiters for discontinuous Galerkin methods on arbitrary meshes. *International Journal for Numerical Methods in Engineering*, 61(14):2566–2593, 2004.

[19] C. Hu and C.-W. Shu. Weighted essentially non-oscillatory schemes on triangular meshes. *J. Comput. Phys.*, 150:97–127, 1999.

[20] M. E. Hubbard. Multidimensional slope limiters for muscl-type finite volume schemes. Numerical analysis report 2/98, Department of Mathematics, University of Reading, 1998.

[21] M. E. Hubbard. Multidimensional slope limiters for muscl-type finite volume schemes on unstructured grids. *J. Comput. Phys.*, 155:54–74, 1999.

[22] P. Jawahar and H. Kamath. A high-resolution procedure for Euler and Navier-Stokes computations on unstructured grids. *J. Comput. Phys.*, 164:165–203, 2000.

[23] H. Ji, F-S. Lien, and E. Yee. An efficient second-order accurate cut-cell method for solving the variable coefficient poisson equation with jump conditions on irregular domains. *Intl. J. Num. Methods in Fluids*, 52, 2006.

[24] K. H. Kim and C. Kim. Accurate, efficient and monotonic numerical methods for multi-dimensional compressible flows Part II: multi-dimensional limiting process. *J. Comput.*

Phys., 208:570–615, 2005.

[25] W. Li, Y.-X. Ren, G. Lei, and H. Luo. The multi-dimensional limiters for solving hyperbolic conservation laws on unstructured grids. *J. Comput. Phys.*, 230:7775–7795, 2011.

[26] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[27] C. F. Ollivier-Gooch. Quasi-ENO schemes for unstructured meshes based on unlimited data-dependent least-squares reconstruction. *J. Comput. Phys.*, 133:6–17, 1997.

[28] J. S. Park, S.-H. Yoon, and C. Kim. Multi-dimensional limiting process for hyperbolic conservation laws on unstructured grids. *J. Comput. Phys.*, 229:788–812, 2010.

[29] B. Swartz. Good neighborhoods for multidimensional van Leer limiting. *J. Comput. Phys.*, 154:237–241, 1999.

[30] S. Tu and A. Aliabadi. A slope limiting procedure in discontinuous Galerkin Finite Element method for gasdynamics applications. *International Journal of Numerical Analysis and Modeling*, 2(2):163–178, 2005.

[31] B. van Leer. Towards the ultimate conservative difference scheme, V. a second order sequel to Godunov's methods. *J. Comput. Phys.*, 32:101–136, 1979.

[32] V. Venkatakrishnan. Convergence to steady state solutions of the Euler equations on unstructured grids with limiters. *J. Comput. Phys.*, 118:120–130, 1995.

[33] S.-H. Yoon, C. Kim, and K. H. Kim. Multi-dimensional limiting process for three-dimensional flow physics analyses. *J. Comput. Phys.*, 227:6001–6043, 2008.