UC Berkeley

Title

A single-stage flux-corrected transport algorithm for high-order finite-volume methods

Authors

Chaplin, Christopher
Colella, Phillip

A SINGLE STAGE FLUX-CORRECTED TRANSPORT ALGORITHM FOR HIGH-ORDER FINITE-VOLUME METHODS

CHRISTOPHER CHAPLIN AND PHILLIP COLELLA

ABSTRACT. We present a new limiter method for solving the advection equation using a high-order, finite-volume discretization. The limiter is based on the flux-corrected transport algorithm. We modify the classical algorithm by introducing a new computation for solution bounds at smooth extrema, as well as improving the pre-constraint on the high-order fluxes. We compute the high-order fluxes via a method of lines approach with fourth order Runge-Kutta as the time integrator. For computing low-order fluxes, we select the corner transport upwind method due to its improved stability over donor-cell upwind. Several spatial differencing schemes are investigated for the high-order flux computation, including centered difference and upwind schemes. We show that the upwind schemes perform well on account of the dissipation of high wavenumber components. The new limiter method retains high-order accuracy for smooth solutions and accurately captures fronts in discontinuous solutions. Further, we need only apply the limiter once per complete time step.

1. Introduction

We wish to solve hyperbolic conservation laws of the form

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

where U represents a vector of conserved values and $\vec{\boldsymbol{F}}(U), [\vec{\boldsymbol{F}} = (\boldsymbol{F}^1 \dots F^D)]$ are corresponding fluxes. The discrete solution of these equations at a given time t^{n+1} and spatial location $\boldsymbol{i}$ is given by:

$$\langle U \rangle_{\boldsymbol{i}}^{n+1} = \langle U \rangle_{\boldsymbol{i}}^{n} - \frac{\Delta t}{\Delta x} \sum_d \left[(\boldsymbol{F}^d)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}^{n+\frac{1}{2}} - (\boldsymbol{F}^d)_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}^{n+\frac{1}{2}} \right] \tag{2}$$

where $\langle U \rangle^n$ and $(\boldsymbol{F}^d)^{n+\frac{1}{2}}$ approximate the average of U over a rectangular Cartesian control volumes indexed by $\boldsymbol{i}$ at time step n and the average of the fluxes in time over the respective faces, respectively.

Methods for accurately computing these fluxes $(\boldsymbol{F}^d)^{n+\frac{1}{2}}$ to obtain high-order accuracy for smooth solutions are well-understood. However, in the presence of discontinuities or underresolved gradients, such methods generate oscillatory errors, and limiters are typically introduced to selectively introduce dissipation that will damp these errors. For high-order methods based on the method of lines, the typical approach to limiters has been to apply a limiter independently at every flux evaluation in a multi-stage high-order time integration method, e.g. Runge-Kutta methods. However, such an approach is problematic, particularly for limiters that are intended to preserve high-order accuracy at smooth extrema. Such limiters typically make use of second- or higher-order spatial derivative information at extrema to determine whether the solution is smooth enough to not require limiting. This leads to the use of a larger stencil (for the limiter) at every stage of the time integration than that of the simpler high-order spatial discretization of the fluxes without limiting, thus increasing communication costs in a domain-decomposed parallel algorithm. In addition, only the linear combination of Runge-Kutta stage computations is guaranteed to produce a high-order accurate solution; the individual stage fluxes and solutions are typically lower-order accurate in time. In adaptive mesh refinement methods with refinement in time, this feature complicates the design of the time interpolation step for computing ghost cell values at refinement boundaries [4]. Finally, many of the stage-by-stage limiting schemes are geometric in nature, such that the effective fully limited method corresponds to donor-cell spatial differencing for the fluxes [7, 10]. This typically has a more restrictive CFL time step stability condition than the high-order methods.

The starting point for addressing these problems is to use a version of the classic Flux-Corrected Transport (FCT) algorithms [1, 9, 13]. These methods introduce dissipation through a nonlinear hybridization of a high-order flux with a dissipative, low-order flux. The FCT algorithm is particularly advantageous because it has a straightforward multi-dimensional expression [13] and has been used extensively with finite-volume discretizations. We will look at a family of high-order methods based on centered- and one-point-upwinded linear finite volume interpolation, following the ideas in [4], combined with the method of lines with fourth-order Runge-Kutta time discretization as the method for time discretization. All four intermediate stages of this high-order method will be computed without any limiting applied, in order to produce a high-order flux to go into the FCT hybridization. The low-order method will be the corner-transport upwind (CTU) method [2, 11]. The method for computing the hybridization coefficient will be based on the approach used in [4, 5] for

interpolation-based limiting, modified to fit into the formalism in [13]. This combination of methods addresses the issues alluded to above. The CFL stability condition for the CTU method is independent of the direction of propagation of the waves relative to the coordinate directions and of the dimensionality of the problem, and is typically much less restrictive than the CFL condition for donor-cell / method-of-lines. The limiter is applied only once – at the end of the time step – thus minimizing the impact on the communications cost.

For this study, we restrict our attention to the equation of scalar advection equation. This allows us to explore design decisions in a simple setting, but one that is still relatively unforgiving, and important for real applications (transport of scalars in the atmosphere, Vlasov equations in phase space). The extension to systems will be discussed in future work. One question we will not address here is that of positivity preservation. The reason is that, in more than one dimension for any linear first-order method that is not donor cell, it is possible to construct a discretely divergence-free advection velocity and non-negative initial data such that the solution becomes negative after one time step. Our preferred approach is to post-process the solution at the end of each time step by redistributing negative increments to nearby cells in a way that would lead to an overall non-negative solution, while still conserving [8, 12].

1.1. Advection Equation. We will consider the linear advection equation in the following form.

$$\frac{\partial q}{\partial t} + \nabla \cdot (q\vec{u}) = 0 \tag{3}$$

$$\nabla \cdot \vec{u} = 0 \tag{4}$$

on a D-dimensional square domain $\Omega = [0,1]^D$. In this case $\vec{u}$ is an advective velocity and q is a scalar field. The partial differential equation above can also be written as:

$$\frac{dq}{dt} = 0 \quad \frac{d\vec{x}}{dt} = \vec{u} \tag{5}$$

Provided that an initial condition is specified ($q_0 = q(\vec{x}(t_0), t_0)$) this system of ordinary differential equations yields a unique solution for any $q(\vec{x}(t), t)$ and $\vec{x}(t)$. The solution arrived at by integrating the equations is that q is constant along characteristic curves defined by $\vec{x}(t)$. Even though there is a simple solution to this equation, the

analysis is still quite useful since there is no diffusion or entropy condition built-in to the equation: any numerical errors introduced are propagated through the domain.

1.2. *Finite-Volume Discretization.*

Our solution approach is to use a finite-volume method to discretize the physical domain into a union of control volumes V_i (Eq. 6)

$$V_i = [ih, (i + e)h] \ , \ i \in \mathbb{Z}^D \ , \ e = (1, 1, ..., 1) \tag{6}$$

where h is the grid spacing (Δx) and i is a D-dimensional index denoting location. Values of the conserved scalar quantity q are stored as cell-averages $\langle q \rangle$ over each cell V_i (Eq. 7). The fluxes $\boldsymbol{F}^d = q\boldsymbol{u}^d$ are stored as averages $\langle \boldsymbol{F}^d \rangle_{i \pm \frac{1}{2}e^d}$ over the surface faces $A_d^{\pm}$ of each cell (Eq. 8).

$$\langle q \rangle_i(t) = \frac{1}{h^D} \int_{V_i} q(\boldsymbol{x}, t) d\boldsymbol{x} \tag{7}$$

$$\langle \boldsymbol{F}^d \rangle_{i \pm \frac{1}{2}e^d}(t) = \frac{1}{h^{D-1}} \int_{A_d^{\pm}} \boldsymbol{F}^d(\boldsymbol{x}, t) \ d\boldsymbol{x} \tag{8}$$

Applying the finite-volume discretization (Eq. 6) to Eq. 3 yields a semi-discrete system of ordinary differential equations (ODE) in time:

$$\frac{d\langle q \rangle_i}{dt} = -\frac{1}{h^D} \int_{V_i} (\nabla \cdot (\vec{\boldsymbol{F}})) d\boldsymbol{x} \tag{9}$$

$$= -\frac{1}{h} \sum_d [\langle \boldsymbol{F}^d \rangle_{i + \frac{1}{2}e^d} - \langle \boldsymbol{F}^d \rangle_{i - \frac{1}{2}e^d}] \tag{10}$$

where Eq. 10 is the result of applying the divergence theorem to Eq. 9. The integration of the above system with respect to time from t^n to t^{n+1} produces the solution given below:

$$\langle q \rangle_{\boldsymbol{i}}^{n+1} = \langle q \rangle_{\boldsymbol{i}}^{n} - \frac{\Delta t}{h} \sum_{d} \left[\langle \boldsymbol{F}^d \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}^{n+\frac{1}{2}} - \langle \boldsymbol{F}^d \rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}^{n+\frac{1}{2}} \right] \tag{11}$$

$$\langle \boldsymbol{F}^d \rangle_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d}^{n+\frac{1}{2}} = \frac{1}{\Delta t} \int_{t^n}^{t^n+\Delta t} \langle \boldsymbol{F}^d \rangle_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d}(t) dt \tag{12}$$

The resulting challenge is to accurately compute $\langle \boldsymbol{F}^d \rangle_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d}^{n+\frac{1}{2}}$. It is important to note that no approximations have been made at this point: Eq. 11 - Eq. 12 are exact relationships. However, to obtain a full discrete approximation, we need quadrature rules for the surface fluxes in Eq. 10 and for the time-averaged fluxes in Eq. 12. The quadrature rules for computing these fluxes are defined following ideas from [4]. In that work, the high-order quadratures were computed using a method of lines approach. The surface fluxes were computed using a high-order centered difference method and the temporal integration was computed using the classic fourth-order Runge-Kutta (RK4) method. We retained the use of RK4 in this study and investigated several high-order methods for computing the surface fluxes.

1.3. *Hybridization.* Returning to the flux description in Eq. 12, we may now define the hybridization

$$\langle \boldsymbol{F}^d \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}^{n+\frac{1}{2}} = (\eta_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}) \langle \boldsymbol{F}_H^d \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} + (1 - \eta_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}) \langle \boldsymbol{F}_L^d \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} \tag{13}$$

where the subscripts H and L refer to the high-order and low-order fluxes, and $\eta_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ is the hybridization coefficient.

In the following sections of the paper we will describe the design choices and procedures for computing the high-order flux, the low-order flux, and the hybridization coefficient.

2. High-Order Flux Computation

We compute the high-order fluxes using the method of lines. Two schemes must be chosen: a scheme for integrating the solution in time and a scheme for computing the spatial derivatives. High-order accuracy requires that both schemes be high-order accurate.

2.1. *High-Order Temporal Integration Scheme.* We use the RK4 scheme to advance the solution forward in time. Returning to the system of ODE (Eq. 9):

$$\frac{d\langle q \rangle}{dt} = -D \cdot \langle \vec{\boldsymbol{F}} \rangle(t) \tag{14}$$

$$= -\frac{1}{h} \sum_d \langle \boldsymbol{F}^d \rangle_{i+\frac{1}{2}e^d} - \langle \boldsymbol{F}^d \rangle_{i-\frac{1}{2}e^d}$$

we want to integrate the system from t^n to t^{n+1}. RK4 is a fourth-order integration scheme that consists of computing a linear combination of stage update variables k_s. The updates are defined below:

$$\langle q \rangle^0 = \langle q \rangle(t^n) \qquad k_1 = -D \cdot \langle \vec{\boldsymbol{F}}(\langle q \rangle^0) \rangle \Delta t \tag{15}$$

$$\langle q \rangle^1 = \langle q \rangle^0 + \frac{k_1}{2} \qquad k_2 = -D \cdot \langle \vec{\boldsymbol{F}}(\langle q \rangle^1) \rangle \Delta t \tag{16}$$

$$\langle q \rangle^2 = \langle q \rangle^0 + \frac{k_2}{2} \qquad k_3 = -D \cdot \langle \vec{\boldsymbol{F}}(\langle q \rangle^2) \rangle \Delta t \tag{17}$$

$$\langle q \rangle^3 = \langle q \rangle^0 + k_3 \qquad k_4 = -D \cdot \langle \vec{\boldsymbol{F}}(\langle q \rangle^3) \rangle \Delta t \tag{18}$$

Each update variable k_s requires computing stage fluxes $\langle \boldsymbol{F}^d \rangle^s_{i\pm\frac{1}{2}e^d} = \langle q\boldsymbol{u}^d \rangle^s_{i\pm\frac{1}{2}e^d}$. The stage fluxes are functions of the stage values $\langle q \rangle^s_i$ and $\langle \boldsymbol{u}^d \rangle^s_i$ alone and the procedure for computing the fluxes will be described in the next section.
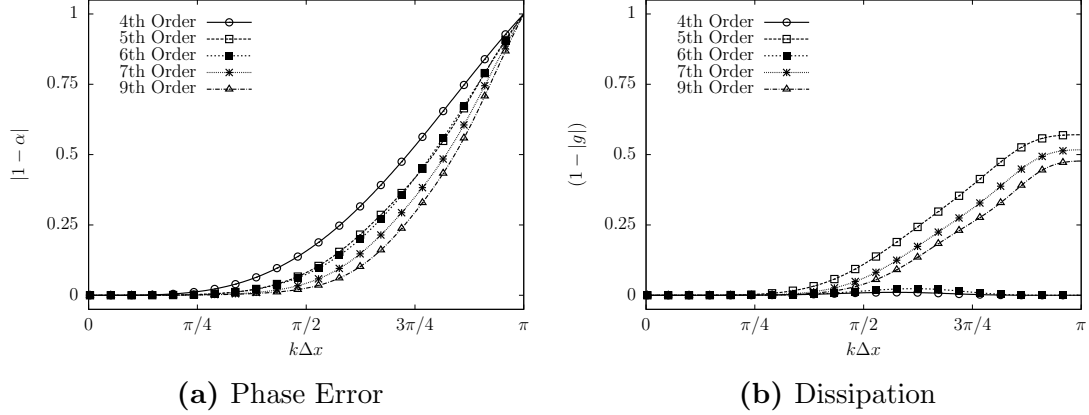
To perform the RK4 integration:

$$\langle q \rangle(t^n + \Delta t) = \langle q \rangle(t^n) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{19}$$

Returning to the conservation notation, this RK4 integration can be described by

$$\langle q \rangle_i^{n+1} = \langle q \rangle_i^n - \frac{\Delta t}{h} \sum_{d=1}^D [\langle \boldsymbol{F}_H^d \rangle_{i+\frac{1}{2}e^d} - \langle \boldsymbol{F}_H^d \rangle_{i-\frac{1}{2}e^d}] \tag{20}$$

$$\langle \boldsymbol{F}_H^d \rangle_{i\pm\frac{1}{2}e^d} = \frac{1}{6} \left[\langle \boldsymbol{F}^d \rangle_{i\pm\frac{1}{2}e^d}^{(0)} + 2\langle \boldsymbol{F}^d \rangle_{i\pm\frac{1}{2}e^d}^{(1)} + 2\langle \boldsymbol{F}^d \rangle_{i\pm\frac{1}{2}e^d}^{(2)} + \langle \boldsymbol{F}^d \rangle_{i\pm\frac{1}{2}e^d}^{(3)} \right] \tag{21}$$

(a) Phase Error **(b)** Dissipation

Figure 1. Normalized phase error and dissipation for the high-order methods ($\sigma = 0.8$)

2.2. *High-Order Spatial Difference Schemes.*

We use high-order finite-difference methods to approximate the surface fluxes associated with the spatial derivatives. The fluxes $\langle q\boldsymbol{u}^d\rangle_{i\pm\frac{1}{2}\boldsymbol{e}^d}$ for the spatial derivatives are functions only of the cell-averaged $\langle q\rangle_{\boldsymbol{i}}$ and $\langle\boldsymbol{u}^d\rangle_{\boldsymbol{i}}$ at any time. Several methods were explored in this study for computing $\langle q\rangle_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}^d}$, including high-order centered difference schemes and upwind schemes. The advantage of the upwind methods is that they have greater diffusion especially in regimes where the phase error begins to rise (Figure 1). The upwind methods only require a small additional computation and the stability between similar order centered difference and upwind methods is almost identical (Table 1).

The interpolation formulae corresponding to the spatial differencing schemes used are presented below. For compactness, the following notation will be used:

$$\langle q\rangle^n_{i+\frac{1}{2}\boldsymbol{e}^d} = \sum_{s=-S}^{S} a_s\langle q\rangle^n_{\boldsymbol{i}+s\boldsymbol{e}^d} \tag{22}$$

where S is the width of the stencil and a_s are the coefficients. The odd ordered methods use the full range of coefficients, whereas the even ordered methods have no coefficient at $s = -S$.

Fourth Order Centered Difference $(S = 2)$

$$\{a_s : s = -S + 1, \ldots, S\} = \frac{1}{12}\{-1, 7, 7, -1\} \tag{23}$$

Fifth Order Upwind $(S = 2)$

$$\{a_s : s = -S, \ldots, S\} = \frac{1}{60} \{2, -13, 47, 27, -3\} \tag{24}$$

Sixth Order Centered Difference $(S = 3)$

$$\{a_s : s = -S + 1, \ldots, S\} = \frac{1}{60} \{1, -8, 37, 37, -8, 1\} \tag{25}$$

Seventh Order Upwind $(S = 3)$

$$\{a_s : s = -S, \ldots, S\} = \frac{1}{420} \{-3, 25, -101, 319, 214, -38, 4\} \tag{26}$$

Ninth Order Upwind $(S = 4)$

$$\{a_s : s = -S, \ldots, S\} = \frac{1}{2520} \{4, -41, 199, -641, 1879, 1375, -305, 55, -5\} \tag{27}$$

2.3. Product Rule. To complete the flux computation, we must compute the average of the product of the scalar variable and the velocity $(\langle q\boldsymbol{u}^d \rangle_{i+\frac{1}{2}e^d})$. The 2D product rules for second, fourth, and sixth-order accuracy are:

$$\langle q\boldsymbol{u}^d \rangle_{i+\frac{1}{2}e^d} = \langle q \rangle_{i+\frac{1}{2}e^d} \langle \boldsymbol{u}^d \rangle_{i+\frac{1}{2}e^d} + O(h^2) \tag{28}$$

$$\langle q\boldsymbol{u}^d \rangle_{i+\frac{1}{2}e^d} = \langle q \rangle_{i+\frac{1}{2}e^d} \langle \boldsymbol{u}^d \rangle_{i+\frac{1}{2}e^d} + \frac{h^2}{12} \sum_{d' \,!= \, d} \frac{\partial q}{\partial x^{d'}} \frac{\partial \boldsymbol{u}^d}{\partial x^{d'}} + O(h^4) \tag{29}$$

$$\langle q\boldsymbol{u}^d \rangle_{i+\frac{1}{2}e^d} = \langle q \rangle_{i+\frac{1}{2}e^d} \langle \boldsymbol{u}^d \rangle_{i+\frac{1}{2}e^d} + \frac{h^2}{12} \sum_{d' \,!= \, d} \left(\frac{\partial q}{\partial x_{d'}} \frac{\partial \boldsymbol{u}^d}{\partial x_{d'}} \right) \tag{30}$$

$$+ \frac{h^4}{1440} \sum_{d' \,!= \, d} \left(3 \frac{\partial^3 q}{\partial x_{d'}^3} \frac{\partial \boldsymbol{u}^d}{\partial x_{d'}} + 3 \frac{\partial^3 \boldsymbol{u}^d}{\partial x_{d'}^3} \frac{\partial q}{\partial x_{d'}} + 2 \frac{\partial^2 \boldsymbol{u}^d}{\partial x_{d'}^2} \frac{\partial^2 q}{\partial x_{d'}^2} \right) + O(h^6)$$

The possible sources of error in the product formulae above are computing the averages $\langle q \rangle_{i+\frac{1}{2}e^d}$, $\langle \boldsymbol{u}^d \rangle_{i+\frac{1}{2}e^d}$, and computing the partial derivative sums. We have already discussed several methods and their accuracy for computing $\langle q \rangle_{i+\frac{1}{2}e^d}$. The velocity fields are analytic for advection, so $\langle \boldsymbol{u}^d \rangle_{i+\frac{1}{2}e^d}$ introduces no error. The derivative terms in the summations above were computed exclusively using centered difference approximations of appropriate accuracy. For example, the derivatives in the fourth-order accurate product formula were computed using a second-order centered difference. The derivatives in the sixth-order formula were computed to fourth-order (for the term multiplied by h^2) and to second-order (for the term multiplied by h^4).

Table 1. Stability of Method for Varying Spatial Difference Operators

Method	Stability Constraint
4th Center	$\sigma \lesssim 2.06/\mathrm{D}$
5th Upwind	$\sigma \lesssim 1.73/\mathrm{D}$
6th Center	$\sigma \lesssim 1.78/\mathrm{D}$
7th Upwind	$\sigma \lesssim 1.69/\mathrm{D}$
9th Upwind	$\sigma \lesssim 1.60/\mathrm{D}$

2.4. *Stability.* We compute the stability for each high-order scheme to determine the allowable time step size following the procedure in [3]. Stability for the method of lines requires the eigenvalues of the right hand side to lie within the stability region of the time integrator. These eigenvalues are computed by diagonalizing the the semi-discrete system (Eq. 14). For advection the eigenvalues are defined as the product of the velocity and the spatial derivative operator (Eq. 31-Eq. 32). The particular eigenvalues for each spatial differencing scheme will be presented later.

$$\frac{d\langle q \rangle}{dt} = \lambda \langle q \rangle \tag{31}$$

$$\lambda \langle q \rangle = -\vec{u} \frac{\partial}{\partial \boldsymbol{x}} \langle q \rangle \tag{32}$$

These eigenvalues must lie within the stability region of the time integrator. The stability region for RK4 is well known, and can be described by its characteristic polynomial

$$P(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} \tag{33}$$

where $z = \Delta t \lambda$. Stability for this problem requires that $|P(z)| \leq 1$. The resulting stability constraints for each spatial differencing scheme are presented in Table 1, where $\sigma = |u|\Delta t/h$.

Along with stability, the phase error and dissipation were computed (Figure 1). The dissipation was defined as $(1 - |g|)$ where:

$$|g| = \sqrt{(Re(g)^2 + Im(g)^2)} \tag{34}$$

$$Re(g) = \left(1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24}\right) - \frac{y^2}{2}\left(1 + x + \frac{x^2}{2}\right) + \frac{y^4}{24} \tag{35}$$

$$Im(g) = y\left(1 + x + \frac{x^2}{2} + \frac{x^3}{6}\right) - \frac{y^3}{6}(1 + x) \tag{36}$$

and $\lambda = x + iy$. The normalized phase error, $|1 - \alpha|$, is defined using the following computation

$$\alpha = \frac{\alpha(\beta)}{|u|} = -\frac{1}{\sigma\beta}\frac{Im(g)}{Re(g)} \tag{37}$$

2.4.1. *Spatial Differencing Eigenvalues.* The eigenvalues for each of the different high-order spatial differencing schemes are presented below. In each of the eigenvalue descriptions, β_d may range from $-\pi$ to π, and is defined as $2\pi k_d h$ with $k_d = 0, \pm 1, \pm 2, ..., \pm N/2$

Fourth Order Centered Difference:

$$\lambda_4 = \frac{i}{12h}\sum_{d=1}^{D} u^d\left[16\sin(\beta_d) - 2\sin(2\beta_d)\right] \tag{38}$$

Fifth Order Upwind:

$$\lambda_5 = \frac{1}{60h}\sum_{d=1}^{D} u^d\left[\left(-2\cos(3\beta_d) + 12\cos(2\beta_d) - 30\cos(\beta_d) + 20\right) + \right.$$
$$\left. i\left(2\sin(3\beta_d) - 18\sin(2\beta_d) + 90\sin(\beta_d)\right)\right] \tag{39}$$

Sixth Order Centered Difference:

$$\lambda_6 = \frac{i}{60h}\sum_{d=1}^{D} u^d\left[2\sin(3\beta_d) - 18\sin(2\beta_d) + 90\sin(\beta_d)\right] \tag{40}$$

Seventh Order Upwind:

$$\lambda_7 = \frac{1}{420h} \sum_{d=1}^{D} u^d \left[\left(3\cos(4\beta_d) - 24\cos(3\beta_d) + 84\cos(2\beta_d) - 168\cos(\beta_d) + 105 \right) + \right.$$
$$\left. i\left(-3\sin(4\beta_d) + 32\sin(3\beta_d) - 168\sin(2\beta_d) + 672\sin(\beta_d) \right) \right] \quad (41)$$

Ninth Order Upwind:

$$\lambda_9 = \frac{1}{2520h} \sum_{d=1}^{D} u^d \left[\left(-4\cos(5\beta_d) + 40\cos(4\beta_d) - 180\cos(3\beta_d) + \right.\right.$$
$$480\cos(2\beta_d) - 840\cos(\beta_d) + 504 \right) +$$
$$i\left(4\sin(5\beta_d) - 50\sin(4\beta_d) + 300\sin(3\beta_d) - \right.$$
$$\left.\left. 1200\sin(2\beta_d) + 4200\sin(\beta_d) \right) \right] \quad (42)$$

3. Low-Order Flux Computation

The low order fluxes are computed using the CTU method [2, 11]. CTU is a first-order time advancement scheme. The method is desirable over the simpler donor-cell upwind method because its stability is independent of dimensionality. However, this increased stability comes with a price. Instead of a single flux being defined by a single upwind value, the CTU flux is dependent upon a set of upwinded values. These values are determined by tracing the characteristic paths from the nodes that define the flux surface. This process involves an increasing number of Riemann solves as the dimensionality of the problem increases. In the 1D case, CTU is identical to donor-cell upwind.

4. Computing the Hybridization Coefficient

We compute the hybridization coefficient η using a modified multidimensional flux-corrected transport (FCT) algorithm. Note that the time superscript notation (n) for fluxes is dropped from here out, but it is implied. Our algorithm is based upon the method described first in [13]. Here is the generic FCT procedure:

(1) Compute the high-order fluxes $\langle \boldsymbol{F}_H^d \rangle_{i \pm \frac{1}{2} e^d}$ over the cell volume V_i.

(2) Compute the low-order fluxes $\langle \boldsymbol{F}_L^d \rangle_{i \pm \frac{1}{2} \boldsymbol{e}^d}$ and the corresponding low-order update $\langle q \rangle_i^{td}$.

$$\langle q \rangle_i^{td} = \langle q \rangle_i^n - \frac{\Delta t}{h} \sum_d \left[\langle \boldsymbol{F}_L^d \rangle_{i + \frac{1}{2} \boldsymbol{e}^d} - \langle \boldsymbol{F}_L^d \rangle_{i - \frac{1}{2} \boldsymbol{e}^d} \right] \tag{43}$$

(3) Compute the antidiffusive fluxes $\langle A^d \rangle_{i \pm \frac{1}{2} \boldsymbol{e}^d}$

$$\langle A^d \rangle_{i \pm \frac{1}{2} \boldsymbol{e}^d} = \langle \boldsymbol{F}_H^d \rangle_{i \pm \frac{1}{2} \boldsymbol{e}^d} - \langle \boldsymbol{F}_L^d \rangle_{i \pm \frac{1}{2} \boldsymbol{e}^d} \tag{44}$$

(4) Limit the antidiffusive fluxes

$$\langle A_\eta^d \rangle_{i \pm \frac{1}{2} \boldsymbol{e}^d} = \eta_{i \pm \frac{1}{2} \boldsymbol{e}^d}^d \langle A^d \rangle_{i \pm \frac{1}{2} \boldsymbol{e}^d} \tag{45}$$
$$0 \le \eta_{i \pm \frac{1}{2} \boldsymbol{e}^d}^d \le 1$$

(5) Update the solution with the limited antidiffusive fluxes

$$\langle q \rangle_i^{n+1} = \langle q \rangle_i^{td} - \frac{\Delta t}{h} \sum_d \left[\langle A_\eta^d \rangle_{i + \frac{1}{2} \boldsymbol{e}^d} - \langle A_\eta^d \rangle_{i - \frac{1}{2} \boldsymbol{e}^d} \right] \tag{46}$$

4.1. *Limiting the antidiffusive flux.* The primary challenge in the above formulation is comuting the hybridization coefficients $(\eta_{i \pm \frac{1}{2} \boldsymbol{e}^d})$. Following the procedure in [13], we compute the coefficients in the following manner:

Pre-constrain the high-order fluxes $\langle \boldsymbol{F}_H \rangle_{i \pm \frac{1}{2} \boldsymbol{e}^d}$. This is a pre-limiting step that seeks to keep the high-order fluxes from generating new or steeper extrema.

Compute the sum $(P_i^\pm)$ of all the antidiffusive fluxes into and out of the cell and a measure of the diffusion $(Q_i^\pm)$.

$$P_i^+ = \sum_d^D \left[\max\left(\langle A^d \rangle_{i-\frac{1}{2}e^d}, 0 \right) - \min\left(\langle A^d \rangle_{i+\frac{1}{2}e^d}, 0 \right) \right] \tag{47}$$

$$Q_i^+ = \left((q_{\max})_i - \langle q \rangle_i^{td} \right) \frac{h}{\Delta t} \tag{48}$$

$$P_i^- = \sum_d^D \left[\max\left(\langle A^d \rangle_{i+\frac{1}{2}e^d}, 0 \right) - \min\left(\langle A^d \rangle_{i-\frac{1}{2}e^d}, 0 \right) \right] \tag{49}$$

$$Q_i^- = \left(\langle q \rangle_i^{td} - (q_{\min})_i \right) \frac{h}{\Delta t} \tag{50}$$

Compute the least upper bounds $R_i^{\pm}$.

$$R_i^+ = \begin{cases} \min\left(1.0, Q_i^+/P_i^+ \right) & \text{if } P_i^+ > 0.0 \\ 0.0 & \text{otherwise} \end{cases} \tag{51}$$

$$R_i^- = \begin{cases} \min\left(1.0, Q_i^-/P_i^- \right) & \text{if } P_i^- > 0.0 \\ 0.0 & \text{otherwise} \end{cases} \tag{52}$$

Select the hybridization coefficient with the most restrictive upper bound.

$$\eta_{i+\frac{1}{2}e^d} = \begin{cases} \min\left(R_{i+e^d}^+, R_i^- \right) & \text{if } \langle A^d \rangle_{i+\frac{1}{2}e^d} > 0.0 \\ \min\left(R_i^+, R_{i+e^d}^- \right) & \text{if } \langle A^d \rangle_{i+\frac{1}{2}e^d} \leq 0.0 \end{cases} \tag{53}$$

In the above description the user is provided with two design choices: pre-constraint for the high-order flux and method of computing the solution bounds $(q_{\max})_i$ and $(q_{\min})_i$.

4.2. Computing the Solution Bounds. Compute initial estimates of the solution bounds, $(q_{\max})_i$ and $(q_{\min})_i$. First, compute the bounded solutions in a rectangular stencil (B_i) that is $[2s_i + 1]^D$ cells in size, where s_i is the stencil size. In this study, s_i was allowed to vary depending on the local velocity field in the following manner:

$$s_{\boldsymbol{i}} = \begin{cases} 2 & \text{if} \quad \sigma \max_d |(\boldsymbol{u}^d)_{\boldsymbol{i}}| \ \geq \ 0.5 \\ 1 & \text{otherwise} \end{cases} \tag{54}$$

where $\sigma = \max(|\vec{\boldsymbol{u}}|)\Delta t/h$ is the global CFL number.

The stencil was allowed to vary to ensure good performance over the spectrum of possible CFL numbers. It was found that at low CFL numbers, $s_{\boldsymbol{i}} = 2$ lead to excess diffusion near discontinuities. Alternatively at high CFL numbers, $s_{\boldsymbol{i}} = 1$ lead to excess diffusion. Allowing the stencil size to vary mitigated these issues.

After the stencil is determined, four bounds are computed: (1) max based on $\langle q \rangle^n$, (2) min based on $\langle q \rangle^n$, (3) max based on $\langle q \rangle^{td}$, and (4) min based on $\langle q \rangle^{td}$.

$$(q_{\max})_{\boldsymbol{i}}^n = \max\left(B_{\boldsymbol{i}}(\langle q \rangle^n)\right) \tag{55}$$

$$(q_{\min})_{\boldsymbol{i}}^n = \min\left(B_{\boldsymbol{i}}(\langle q \rangle^n)\right) \tag{56}$$

$$(q_{\max})_{\boldsymbol{i}}^{td} = \max\left(B_{\boldsymbol{i}}(\langle q \rangle^{td})\right) \tag{57}$$

$$(q_{\min})_{\boldsymbol{i}}^{td} = \min\left(B_{\boldsymbol{i}}(\langle q \rangle^{td})\right) \tag{58}$$

Then select the upper and lower bound of the two estimates.

$$(q_{\max})_{\boldsymbol{i}} = \max((q_{\max})_{\boldsymbol{i}}^n, (q_{\max})_{\boldsymbol{i}}^{td}) \tag{59}$$

$$(q_{\min})_{\boldsymbol{i}} = \min((q_{\min})_{\boldsymbol{i}}^n, (q_{\min})_{\boldsymbol{i}}^{td}) \tag{60}$$

4.2.1. *Accurate Solution Bounds at Smooth Extremum.* For the vast majority of cells within the domain, the previous bound computation is sufficiently accurate. However, computing bounds at extremum is more complicated. Ideally the bounds need to keep the solution monotonic and positive, but the bounds should also not "clip" the solution. There are few different methods for avoiding clipping, and we use a geometric construction that is only applied at smoothly varying extrema. It is based on the ideas in [5].

The smooth extremum criteria in 1D is:

$$(ext^d)_i \;=\; \min[\;(dq)_i \cdot (dq)_{i+e^d}\;,\;\;(dq)_{i-e^d} \cdot (dq)_{i+2e^d}\;] \leq 0.0 \;\; \&\& \qquad (61)$$
$$1.25 \cdot (dqtot)_i < (tv)_i$$

where the following definitions are used:

$$(dq)_i = \langle q \rangle_i^{td} - \langle q \rangle_{i-e^d}^{td} \qquad (62)$$
$$(dqtot)_i = |\langle q \rangle_{i+2e^d}^{td} - \langle q \rangle_{i-2e^d}^{td}| \qquad (63)$$
$$(tv)_i = |(dq)_{i+2e^d}| + |(dq)_{i+e^d}| + |(dq)_i| + |(dq)_{i-e^d}| \qquad (64)$$

This criteria has two parts. First, check for a sign change in the first derivative. The sign change will indicate either an extremum or a discontinuity in the solution. Second, ensure that the solution locally is not a perterbation of a discontinuity.

For a smooth multidimensional extremum either $(ext^d)_i$ must be true in all dimensions or it must be true for some d and the solution must remain constant along the dimensions in which $(ext^d)_i$ is not true. We use the following criterion to determine if the solution is constant:

$$\max(|(q_{\max}^d)_i^{td} - \langle q \rangle_i^{td}|, |(q_{\min}^d)_i^{td} - \langle q \rangle_i^{td}|) \leq 10^{-14} \qquad (65)$$

where the following definitions hold

$$(q_{\max}^d)_i^{td} = \max\left(\langle q \rangle_{i-e^d}^{td}, \langle q \rangle_i^{td}, \langle q \rangle_{i+e^d}^{td}\right) \qquad (66)$$
$$(q_{\min}^d)_i^{td} = \min\left(\langle q \rangle_{i-e^d}^{td}, \langle q \rangle_i^{td}, \langle q \rangle_{i+e^d}^{td}\right) \qquad (67)$$

Once we have determined that the solution at V_i is at a smooth extremum, we compute new values of $(q_{\max})_i$ and $(q_{\min})_i$. The first step is to construct a parabolic function from the local values of $\langle q \rangle$ centered at $\boldsymbol{x}_i$:

$$q^d(x) = \left(\frac{(d2q)_i^n}{2}\right) x^2 + \left(\frac{(\langle q \rangle_{i+e^d} - \langle q \rangle_{i-e^d})}{2}\right) x + \langle q \rangle_i \qquad (68)$$

where

$$(d2q)_i^n = \langle q \rangle_{i+e^d}^n + \langle q \rangle_{i-e^d}^n - 2\langle q \rangle_i^n \tag{69}$$

The location of the vertex (x_c) is given by the ratio $-b/2a$, where a and b are the quadratic and linear coefficients (Eq. 68):

$$x_c = -\frac{\langle q \rangle_{i+e^d} - \langle q \rangle_{i-e^d}}{2(d2q)_i^n} \tag{70}$$

and $-0.5 \le x_c \le 0.5$. Then, evaluate the quadratic at the vertex to find the extremum value as well as deconvolve to get an estimate of the point value:

$$(q_{ext}^d)_i = \left(\frac{(d2q)_i^n}{2} \right) x_c^2 + \left(\frac{(\langle q \rangle_{i+e^d} - \langle q \rangle_{i-e^d})}{2} \right) x_c + \langle q \rangle_i - \frac{(d2q)_i^n}{24} \tag{71}$$

Select the largest $(q_{ext}^d)_i$ or smallest $(q_{ext}^d)_i$ depending on the sign of the second derivative:

$$(q_{ext})_i = \begin{cases} \max_d((q_{ext}^d)_i, (q_{max})_i) & \text{if} \quad \text{sgn}((d2q)_i^n) \le 0.0 \\ \min_d((q_{ext}^d)_i, (q_{min})_i) & \text{otherwise} \end{cases} \tag{72}$$

Finally, compute the appropriate extremum bound by augmenting the solution value at the previous time by a scaled difference between the extremum value and the solution value:

$$(q_{max})_i = \begin{cases} q_i^n + \max(0.0, 2.0\,|(q_{ext})_i - q_i^n|) & \text{if} \quad \text{sgn}((d2q)_i^n) \le 0.0 \\ (q_{max})_i & \text{otherwise} \end{cases} \tag{73}$$

$$(q_{min})_i = \begin{cases} q_i^n + \min(0.0, 2.0\,|(q_{ext})_i - q_i^n|) & \text{if} \quad \text{sgn}((d2q)_i^n) > 0.0 \\ (q_{min})_i & \text{otherwise} \end{cases} \tag{74}$$

4.2.2. *Updating $R_i^{\pm}$ at Extrema.* We flag the extrema at which the Laplacian is changing sign. We compute the D dimensional approximation to the Laplacian over a 3-point stencil:

$$\Delta q_i = \sum_{d=1}^{D} \frac{\partial^2 q_i}{\partial x_d^2} \approx \sum_{d=1}^{D} \frac{(d2q)_i^n}{h^2} \tag{75}$$

If Δq changes sign anywhere in the 3-point vicinity of $\boldsymbol{i}$, then we flag that cell $\boldsymbol{i}$. We then update the least upper bound multiplier at the flagged cells:

$$R_{\boldsymbol{i}}^{\pm} = 0 \ \text{ if } \boldsymbol{i} \text{ flagged} \tag{76}$$

4.3. Pre-constraining the High-Order Flux. We pre-constrain the high-order flux by modifying the high-order fluxes where they would otherwise accentuate or produce a new extremum. In practice, the value of the antidiffusive flux is edited instead of the high-order flux directly. Following [13] we set the antidiffusive flux to zero in these regions.

The baseline condition for applying the pre-constraint:

$$\langle A^d \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} (\langle q \rangle_{\boldsymbol{i}+\boldsymbol{e}^d}^{td} - \langle q \rangle_{\boldsymbol{i}}^{td}) \leq 0.0 \tag{77}$$

However, this condition alone is not sufficient. This condition will occasionally be satisfied at smooth areas in the solution. We add the following conditions to make sure we only apply this condition away from smooth areas:

$$\min \left[(d2q)_{\boldsymbol{i}+\boldsymbol{e}^d}^n \cdot (d2q)_{\boldsymbol{i}}^n \ , \ (d2q)_{\boldsymbol{i}}^n \cdot (d2q)_{\boldsymbol{i}-\boldsymbol{e}^d}^n \ , \ (d2q)_{\boldsymbol{i}+\boldsymbol{e}^d}^n \cdot (d2q)_{\boldsymbol{i}+2\boldsymbol{e}^d}^n \right] < 0.0 \tag{78}$$

$$|\langle A^d \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}| \leq \frac{|(\boldsymbol{u}^d)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}|h}{2} \left(1 - \sigma_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} \right) \frac{|(d2q)_{\boldsymbol{i}} + (d2q)_{\boldsymbol{i}+\boldsymbol{e}^d}|}{2} \tag{79}$$

where $\sigma_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = |(\boldsymbol{u}^d)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}|\Delta t/h$.

The first constraint above attempts to detect a discontinuity in the solution. Discontinuities are natural places for high-order fluxes and consequently antidiffusive fluxes to generate a new extremum. However, there are smooth multidimensional solutions in which the second derivative naturally changes sign. The second constraint seeks to preclude this case. The term on the right hand side of the inequality (Eq. 79) is the d-directional dissipation term, scaled by the cell size, in the modified equation analysis (Eq. 80) of CTU applied to the advection equation

$$\frac{\partial q}{\partial t} + \sum_d^D \left(\boldsymbol{u}^d \frac{\partial q}{\partial x_d} \right) = \sum_d^D \left(\frac{\boldsymbol{u}^d h}{2} (1 - \sigma_d) \frac{\partial^2 q}{\partial x_d^2} \right) + O(h^2) \tag{80}$$

We are interested in comparing the magnitude of this dissipative term to the antidiffusive flux. While it is true that the second derivative takes on a zero value right at a discontinuity, the second derivative takes on large values immeadiately surrounding this point. This means that the dissipative term is large in the neighborhood of discontinuities. If the magnitude of the disspative term is large relative to the the antidiffusive term, then we assume we are near a discontinuity and allow the pre-constraint.

5. Results

Results in one and two dimensions are presented. A total of four initial conditions were investigated. Of the four, one initial condition was smooth and the others contained a discontinuity. For the two dimensional tests, we used two different velocity fields (Eq. 81-Eq. 82): constant diagonal and solid body rotation.

$$\boldsymbol{u} = [1, 1] \tag{81}$$

$$\boldsymbol{u} = 2\pi[y - 0.5, 0.5 - x] \tag{82}$$

The center for the constant velocity initial condition was in the middle of the domain, whereas it was offset by 0.25 of the grid height for the solid body rotation examples

$$\boldsymbol{x}_c^{\text{const}} = (0.5, 0.5)$$

$$\boldsymbol{x}_c^{\text{solid}} = (0.5, 0.75)$$

5.1. *Initial Conditions.* The smooth initial condition was constructed as a power of cosines.

$$q_i(t_0) = \begin{cases} \cos^8 \left(\frac{\pi}{2} \cdot \frac{R}{R_0} \right) & \text{if } R \leq R_0 \\ 0 & \text{otherwise} \end{cases} \tag{83}$$

$$\boldsymbol{x}_i \in [0, 1]$$

$$R = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_c)^2}$$

For this initial condition, $R_0 = 15$ was chosen.

Three different discontinuous initial conditions were investigated. The first was a square and is described as:

$$q_i(t_0) = \begin{cases} 1 & \text{if for each } D: |x_i^D - x_c^D| \leq 0.15 \\ 0 & \text{otherwise} \end{cases} \tag{84}$$
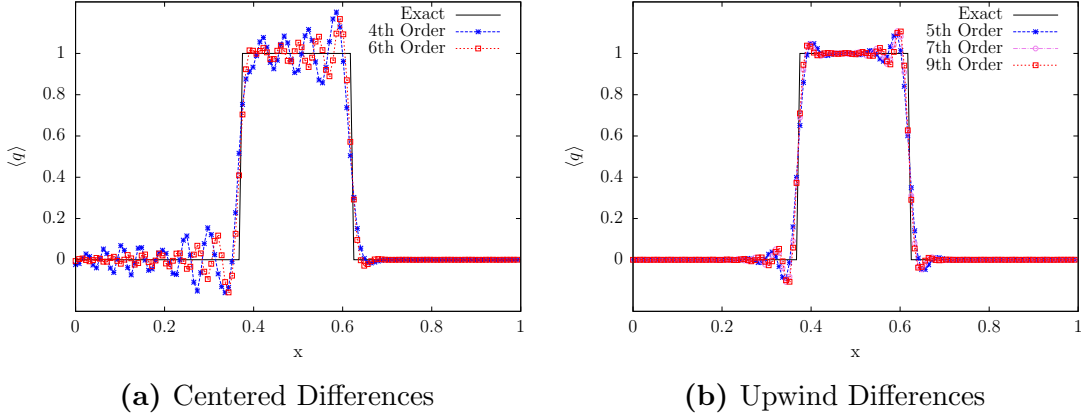
The next is a semi-ellipse:

$$q_i(t_0) = \begin{cases} \sqrt{1.0 - \left(\frac{R}{R_0}\right)^2} & \text{if } R \leq R_0 \\ 0 & \text{otherwise} \end{cases} \tag{85}$$

In this case $R_0 = 0.25$.

The last test case is the classic slotted cylinder in two dimensions [13]. For this paper we used $N = 256$ to represent the cylinder. This is roughly twice as resolved as the original.

5.2. One-Dimensional Tests. The first requirement for the limiter method is that it reduces to the high-order scheme away from discontinuities. All of the high-order schemes achieved similar errors for smooth solutions (Figure 2). The rate of convergence for each method was 4.0, which is the expected rate due to the usage of RK4 for the time integration scheme. This implies that the limiter is not being activated in smooth regions, and the first requirement is met.

The second requirement is that the limiter method accurately represent discontinuities. To determine if this requirement is met, we analyzed the performance of the high-order schemes with and without the limiter. The first test was a comparison between the high-order schemes without the limiter (Figure 3). In the presence of a discontinuity, the centered difference solutions produced more pronounced oscillations than the upwind solutions. This result is consistent with the amplitude and phase error analysis presented earlier.

(a) Solution Error without Limiter   **(b)** Solution Error with Limiter
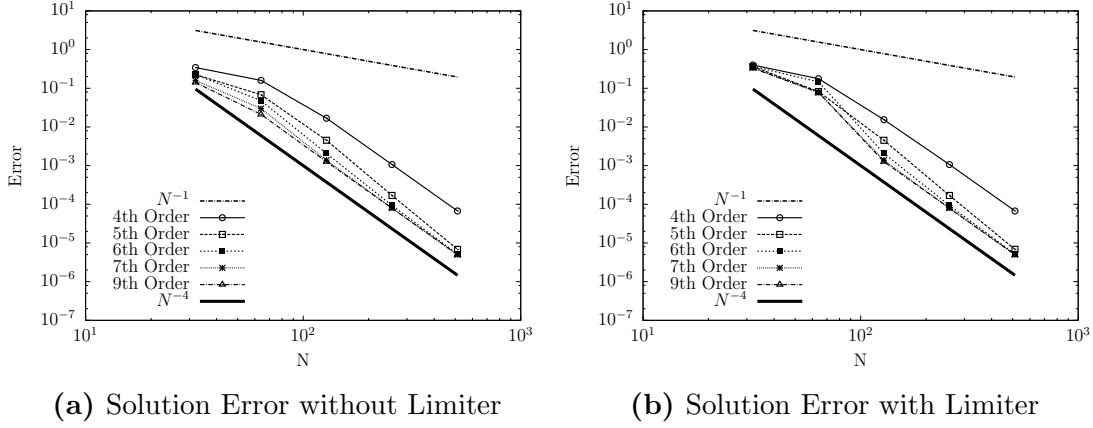
Figure 2. Max Norm Errors in 1D

(a) Centered Differences   **(b)** Upwind Differences

Figure 3. Comparison of Square Solutions with Different High-Order Spatial Differencing
Schemes ($\sigma = 0.8$, $t = 1.0$, $N = 128$)

The more important comparisons are how these high-order schemes perform with
the limiter on. For the square initial condition, there is very little difference in the
solution (Figure 4). The only noticeable difference is some additional diffusion on
the tailing edge of the centered difference solution. However, for the semi-ellipse,
the centered difference schemes produce oscillations even in the limited solution near
the discontinuity (Figure 5). The oscillations are bounded by FCT; however, they are
clearly undesirable. The upwind schemes also produce some oscillations, but again the
magnitude of the oscillations is smaller. This is motivation for selecting the upwind
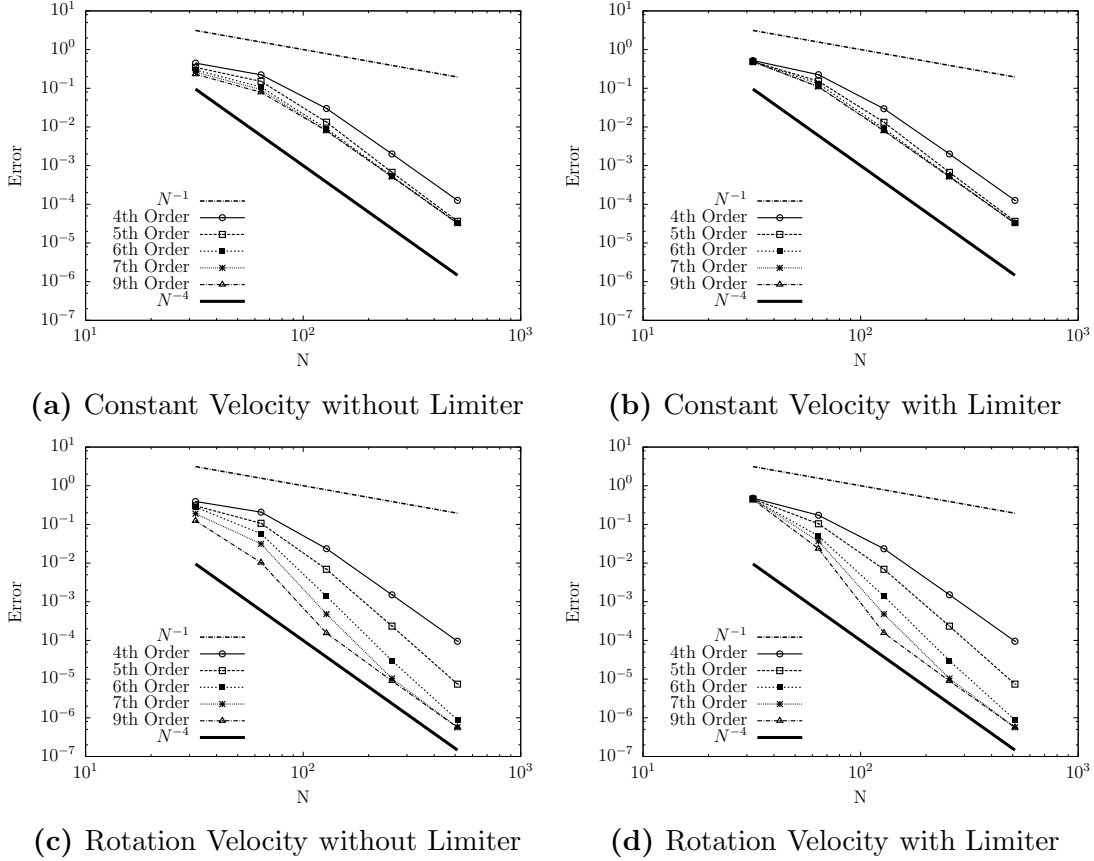methods moving forward.

(a) Centered Differences   **(b)** Upwind Differences
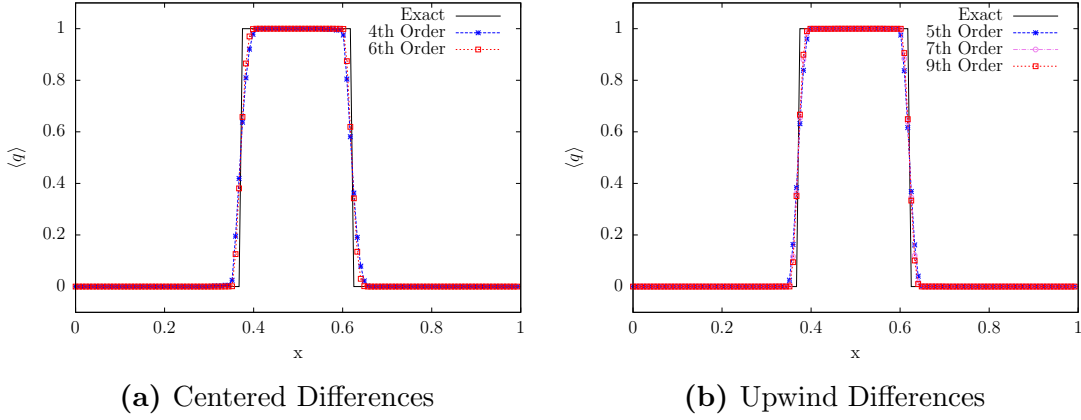
Figure 4. Comparison of Limited Square Solutions with Different High-Order Spatial Differencing Schemes ($\sigma = 0.8$, $t = 1.0$, $N = 128$)

(a) Centered Differences   **(b)** Upwind Differences
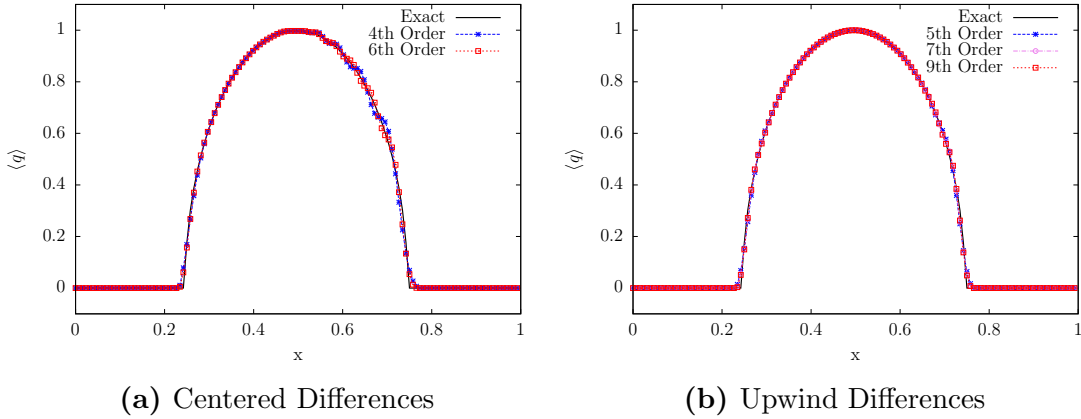
Figure 5. Comparison of Limited Semi-Ellipse Solutions with Different High-Order Spatial Differencing Schemes ($\sigma = 0.8$, $t = 1.0$, $N = 128$)
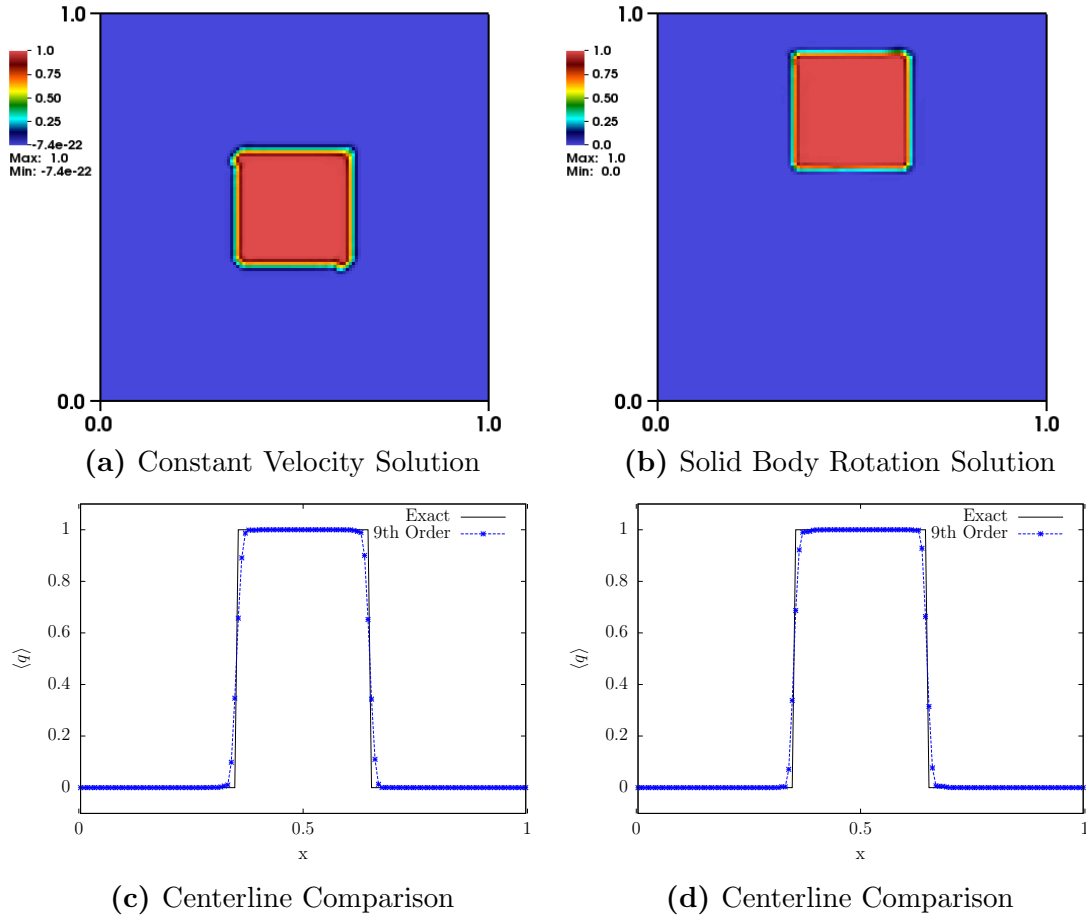
5.3. *Two Dimensional Tests.* The smooth test errors in two dimensions are reported with the limiter for both velocity fields (Figure 6). As in the one dimensional case, the high-order solution accuracy requirement is met. Each method achieves fourth order accuracy, at a minimum. Interestingly, for the solid body rotation solution, the error reduction is greater than fourth order. Since the spatial differencing error is sixth order accurate, this suggests that the time discretization error is smaller than expected even for large values of σ. Also the ninth order scheme is still convergent right at its theoretical stability limit ($\sigma \sim 0.8$).

(a) Constant Velocity without Limiter

(b) Constant Velocity with Limiter

(c) Rotation Velocity without Limiter

(d) Rotation Velocity with Limiter

Figure 6. Max Norm Errors in 2D

The limiter also performs quite well at representing discontinuities in two dimensions. Various solution plots for discontinuous initial conditions are presented (Figure 7 - Figure 9). All of the two dimensional plots were generated using the ninth order scheme in space, running right at the method's theoretical stability limit ($\sigma = 0.8$ in 2D). The square solution under constant velocity has few, if any, ripples and is nicely bounded (Figure 7). There is some distortion of the corners, particularly at the top-left and bottom-right. The square solution under solid body rotation looks similar to the constant velocity solution and the corner issue is mitigated.

The semi-ellipse solution is likewise well resolved (Figure 8). As with the one dimensional case there are some dispersive errors on the leading edge, but they are small. The solutions under both velocity fields are accurately bounded. The semi-ellipse solution under solid body rotation was centered at $\boldsymbol{x}_c^{\text{solid}} = (1.0, 1.5)$ to keep the edge

(a) Constant Velocity Solution

(b) Solid Body Rotation Solution

(c) Centerline Comparison

(d) Centerline Comparison

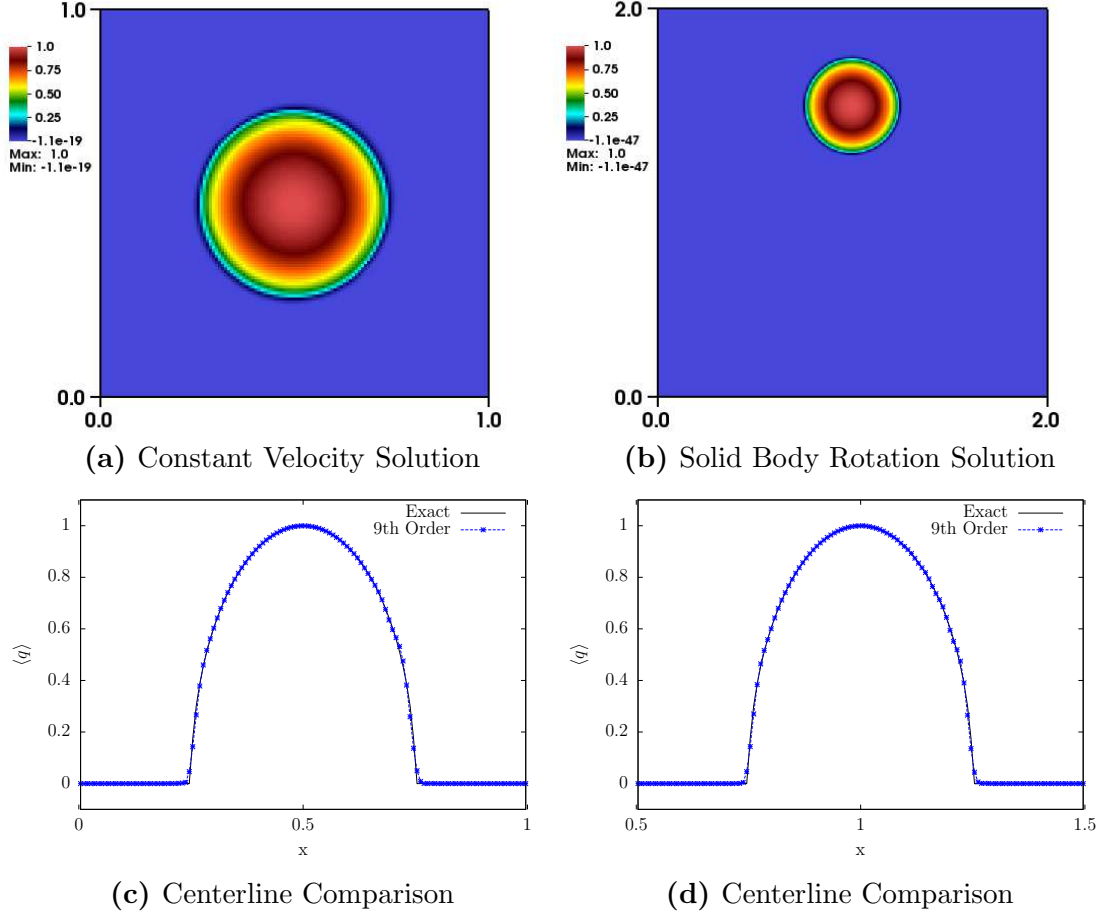**Figure 7.** Square Solutions ($t = 1.0$, $N = 128$, $\sigma = 0.8$)

of the conditon away from the domain boundary. The domain was also expanded to $\boldsymbol{x_i} \in [0, 2]$.

The final test was the slotted cylinder (Figure 9). The limiter method keeps the solution bounded and resolves the fronts quite nicely. At lower grid resolutions the slot can fill in and the bounds may not be enforced. But as the grid is refined, both of these issues are resolved.

6. Conclusions

We presented a new flux limiter based upon FCT that retains high-order accuracy for smooth solutions and captures fronts well. Our algorithm presented here uses

(a) Constant Velocity Solution

(b) Solid Body Rotation Solution

(c) Centerline Comparison

(d) Centerline Comparison

Figure 8. Semi-ellipse Solutions ($t = 1.0$, $N = 128$, $\sigma = 0.8$)

CTU for low-order fluxes, upwind schemes for high-order fluxes, and RK4 for time integration. Our additions to the previous FCT method included a new computation for the extrema and an expanded pre-constraint on the high-order fluxes. This new pre-constraint is more restrictive than the original one, and seeks to only constrain the fluxes near discontinuities in the solution.

Extensions for this work are applying the limiter to systems of hyperbolic conservation laws, possibly developing new high-order upwind methods with corner-coupling, and further improving the pre-constraint on the high-order fluxes. Applying the limiter to hyperbolic systems is the most straightforward extension of this work. It would be relatively simple and informative to apply this limiter in a gas dynamics solver. High-order, corner-coupled upwind methods for use with general multi-stage time

(a) Solid Body Rotation Solution   **(b)** Centerline Comparison
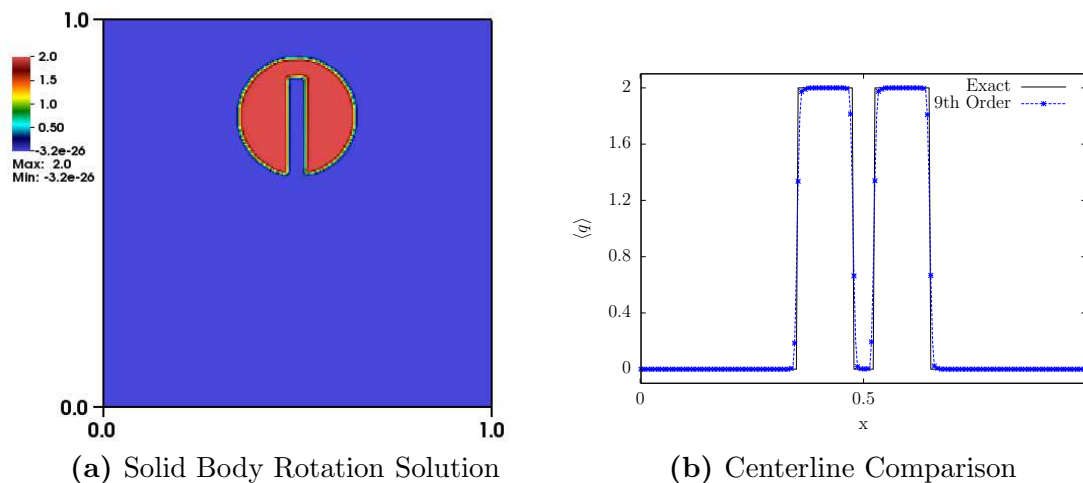
Figure 9. Slotted Cylinder under Solid Body Rotation ($t = 1.0$, $N = 256$, $\sigma = 0.8$)

integrators could remove the dimensional dependence of the stability. However, no upwind methods of this nature currently exists. The pre-constraint on the high-order fluxes is another area where additional study could pay off. In this work we found that the pre-constraint affects a delicate balance between effectively representing discontinuities and retaining high-order accuracy in smooth areas. It was relatively simple to get one or the other. Ensuring both required testing many versions of the pre-constraint. Different versions of the pre-constraint have been proposed [6], but they did not produce good results with the algorithm presented here.

Other versions of FCT [9] have implemented high-order centered difference schemes with hyper-diffusive fluxes. These schemes, like the high-order upwind schemes, seek to add diffusion to the large wave-number solution components. These centered difference schemes were not examined in this study, but they do offer a possible alternative to the upwind ones used here.

References

[1] J.P. Boris and D.L. Book. Flux-corrected transport. I. SHASTA, a fluid transport algorithm that works. *Journal of Computational Physics*, 11(1):38–69, 1973.

[2] P. Colella. Multidimensional upwind methods for hyperbolic conservation laws. *Journal of Computational Physics*, 87(1):171–200, 1990.

[3] P. Colella, M.R. Dorr, J.a.F. Hittinger, and D.F. Martin. High-Order, Finite-Volume Methods in Mapped Coordinates. *Journal of Computational Physics*, 230(8):2952–2976, 2011.

[4] P. Colella and P. McCorquodale. A High-Order Finite-Volume Method for Hyperbolic Conservation Laws on Locally-Refined Grids. *Communications in Applied Mathematics and Computational Science*, 6(1):1–25, 2011.

[5] P. Colella and M. Sekora. A Limiter for PPM that Preserves Accuracy at Smooth Extrema. *Journal of Computational Physics*, 227(15):7069–7076, 2008.

[6] C.R. DeVore. An Improved Limiter for Multidimensional Flux-Corrected Transport. Technical Report Naval Research Laboratory, Washington, DC, 1998.

[7] A. Harten, B. Engquist, S. Osher, and S.R. Chakravarthy. Uniformly High Order Accurate Essentially Non-oscillatory Schemes. III. *Journal of Computational Physics*, 303(71):231–303, 1987.

[8] J. Hilditch and P. Colella. A Front Tracking Method for Compressible Flames in One Dimension. *SIAM Journal on Scientific Computing*, 16(4):755–772, 1995.

[9] D. Kuzmin, R. Löhner, and S. Turek, editors. *Flux-Corrected Transport: Principles, Algorithms, and Applications*. Springer, 1st edition, 2005.

[10] X. Liu, S. Osher, and T. Chan. Weighted Essentially Non-oscillatory Schemes. *Journal of Computational Physics*, 115(1):200–212, 1994.

[11] J. Saltzman. An Unsplit 3D Upwind Method for Hyperbolic Conservation Laws. *Journal of Computational Physics*, 115(1):153–168, 1994.

[12] B. Wang, G.H. Miller, and P Colella. A Particle-in-Cell Method with Adaptive Phase-Space Remapping for Kinetic Plasmas. *SIAM Journal on Scientific Computing*, 33(6):3509–3537, 2011.

[13] S.T. Zalesak. Fully Multidimensional Flux-Corrected Transport Algorithms for Fluids. *Journal of Computational Physics*, 31(3):335–362, 1979.