A high-order WENO-limited finite-volume algorithm for atmospheric flow using the ADER-differential transform time discretization

Matthew R. Norman

Abstract
A high-order-accurate weighted essentially non-oscillatory (WENO) limited upwind finite-volume scheme is detailed for the compressible, nonhydrostatic, inviscid Euler equations using an arbitrary derivatives (ADER) time-stepping scheme based on differential transforms (DTs). A second-order-accurate alternating Strang dimensional splitting is compared against multidimensional simulation with 2D transport using solid body rotation of various data. The two were found to give nearly identical accuracy in orthogonal, Cartesian coordinates. Orders of convergence are demonstrated at up to ninth-order accuracy with 2D transport. 1D transport is used to confirm that error decreases monotonically with increasing order of accuracy with WENO limiting even for discontinuous data. Further, WENO limiting always decreased the error compared with simulation without limiting in the $L_1$ norm. A series of standard 2D compressible nonhydrostatic Euler equation test cases were validated against previous results from literature. Finally, it was demonstrated that increasing the order of accuracy led to better resolved features and increased power for kinetic energy at small wavelengths.

KEYWORDS
ADER, Finite-Volume, high-order, WENO

INTRODUCTION

1.1 A brief survey of common discretizations for the Euler equations

1.1.1 Spatial discretizations

Multi-moment/element-based schemes
Spatial discretizations used for the Euler equations (or any hyperbolic equation system) could be grouped into...
two main categories: single-moment and multi-moment. Multi-moment schemes treat groups of degrees of freedom (DOFs), usually called “elements,” all together at once. Multi-moment schemes for the Euler equations include modal and nodal, continuous and discontinuous Galerkin methods Nair et al. (2005); Fournier et al. (2004); Giraldo et al. (2002); Marras et al. (2016); non-variational element-based methods Yabe et al. (2001); Li and Xiao (2009); Norman (2015a), mixed finite-element methods Cotter and Shipton (2012), and elements that also use reconstruction Dumbser et al. (2008).

Single-moment schemes

Single-moment schemes evolve each DOF individually using surrounding data to reconstruct variation in the vicinity of the DOF being updated. Single-moment schemes are largely divided into two camps: finite differences and finite volumes. Finite differences use the strong form of the PDE and require a well-posed flux derivative. Finite volumes (FVs) use the weak (integrated, not variational) form of the PDE, removing the flux term spatial derivative via the Gauss divergence theorem.

Finite volumes can be decomposed into two main camps based on how fluxes are computed: (a) direct flux computations and (b) reconciled discontinuous fluxes. Directly reconstructed fluxes (e.g., Skamarock and Klemp (2008)) add less inherent dissipation to the FV scheme, and in these methods, cell-edge fluxes are directly reconstructed from surrounding information. Reconciled discontinuous fluxes Toro (2013); Rider (1994) allow the scheme developer to introduce a varying range of dissipation. In these methods, one typically uses cell-centered recovery operators to provide two different flux estimates per cell edge. Then, these fluxes are reconciled using a numerical flux function that could be some approximation to the upwind Riemann state or some other perhaps less dissipative flux function that may also be valid for non-hyperbolic systems of PDEs Rider and Lowrie (2002); Kurganov and Lin (2007).

The discretization presented in this paper is a single-moment finite-volume scheme that uses a reconciled discontinuous flux via an upwind linear Riemann solver for inherent dissipation.

1.1.2 Oscillation limiting

Instabilities often develop while numerically integrating the Euler equations from accumulation of energy at small wavelengths. This is related to “Runge” oscillations that develop in the presence of discontinuities with polynomial approximations of higher than first-order accuracy. Even if the simulation is stable without explicit dissipation, oscillations will still develop, which can contaminate certain added physical approximations such as microphysics and chemistry Lauritzen et al. (2012). Discontinuities need to be limited to remove oscillations while preferably still keeping the gradients as steep as possible.

Element-based limiting typically treats all DOFs in an element together at once. There are variable second-order viscosity approaches Yu et al. (2015), higher-order “hyper”-viscosity approaches Skamarock and Klemp (2008); Dennis et al. (2012), optimization-based approaches that limit element extrema over a time step to that of the neighborhood Guba et al. (2014), WENO-based limiters Qiu and Shu (2004); Guo et al. (2016), modal high-frequency limiters Michoski et al. (2016), and flux-correction approaches Löhner et al. (1987); Kuzmin et al. (2012). Most if not all element-based limiting requires added parallel communication compared with the underlying scheme.

With FV schemes in particular, since a halo has already been transferred for reconstruction, that same halo can also be used for high-resolution limiting techniques that include essentially non-oscillatory (ENO) Shu and Osher (1988), weighted ENO (WENO) Liu et al. (1994), piecewise parabolic method (PPM) Colella and Woodward (1984), and non-polynomial recovery operators Xiao et al. (2002); Norman and Nair (2008). Other FV limiting mechanisms typically require additional parallel communication, including flux correction, viscosity, and hyperviscosity. This study uses WENO limiting.

1.1.3 Grid staggering

Grid staggering places different variables on cell centers, faces, edges, and corners (with implied directionality where appropriate) Skamarock et al. (2012); Arakawa and Lamb (1981). Certain aspects of staggering have also been generalized using differential forms in compatible finite-element techniques Natale et al. (2016) to exactly mimic certain properties of the underlying PDEs in a discretized setting. Notably, mimetic techniques can conserve quantities that are not in the vector of conserved variables.

In the present study, a co-located grid (“A”-grid) is used. Many atmospheric discretizations rely on staggering to achieve better stability. The upwind finite volumes used here have enough inherent dissipation to achieve stability without explicitly added dissipation.

1.1.4 Dimensional splitting

Dimensional splitting, often expressed as a formal operator splitting, is the act of treating each dimension separately,
and it can be done in different ways. One method is to apply a traditional operator splitting technique such as Strang splitting, which computes $x, y,$ and $z$-direction tendencies in certain sequences, using updates from the previous direction before computing tendencies in the next MacNamara and Strang (2016). Another approach is to compute tendencies in all directions simultaneously using the same forcing, and to couple them within each stage of a multistage operator, relying on the stages to generate cross terms automatically in the discretization Katta et al. (2015). There also exist splitting schemes that couple cheap nonconservative Lagrangian updates with more expensive flux-form updates (e.g., Leonard et al. (1996)).

Dimensional splitting for finite-volume methods is desirable computationally because it typically reduces the computational complexity of the scheme by a factor of $N^{(D-1)}$, where $N$ is the order of accuracy of the scheme, and $D$ is the number of spatial dimensions. Most splittings will suffice in an orthogonal coordinate system as shown in section 3.2.1, but in nonorthogonal coordinates, a straightforward Strang splitting gives very poor accuracy Norman and Nair (2018); Chen et al. (2017), though coupling dimensions through a multistage time integrator is still accurate in nonorthogonal coordinates Katta et al. (2015). However, for ADER-DT, there is no existing implementation that uses stages. For this reason, ADER-DT as currently implemented must be fully multidimensional in nonorthogonal coordinates as demonstrated in Norman and Nair (2018). However, since the present study only considers Cartesian coordinates, a second-order-accurate Strang splitting is used, and the errors associated with this approach are investigated in section 3.2.1.

### 1.2 Temporal discretizations

Unlike time-implicit discretizations Barros et al. (1995); Wood et al. (2014); Evans et al. (2019), time-explicit discretizations Zängl et al. (2015); Dennis et al. (2012); Skamarock and Klemp (2008) do not set up global dependence within each time step. Therefore, time-explicit time steps will be limited by stability ultimately because of the need to resolve finite propagation of information. Some applications require time steps even smaller than stability dictates to control errors. Time-explicit discretizations are generally categorized as multistep, multistage, or fully discrete Durran (2010); Norman et al. (2011a). The semi-discrete schemes (e.g., multistage or multistep) treat the spatially discretized (yet continuous in time) ODE with an ODE solver such as Runge-Kutta or Adams-Bashforth (or other varieties as well). Multistage and multistep methods must be strong stability preserving (SSP) to avoid having the temporal discretization itself lead to undesired new extrema Gottlieb (2005).

Fully discrete schemes operate by propagating known spatial information into the temporal domain, the high-order generalization of which will either be Lagrangian Staniforth and Côté (1991) or arbitrary derivative (ADER) in nature Titarev and Toro (2002); Toro (2013). Lagrangian methods use characteristic trajectories of hyperbolic waves to link temporal and spatial information. ADER methods use the PDE definition itself, which directly relates spatial and temporal variation through derivatives. For instance, with linear uniform advection, $u_t = -u_x$, if we know the first spatial derivative, we can compute the first temporal derivative. By further differentiating in space and time, higher-order time derivatives can be computed from higher-order spatial derivatives. For a more in-depth description, please see section 2.5.

This study uses a time-explicit ADER scheme. Efficiently implemented, ADER methods have the desirable computational complexity of $O(N^{D+2})$ for the complete space–time operator Norman and Nair (2018).

### 1.3 Goal of the proposed method

The main hypothesis being explored in this study is that high-order accuracy, when properly limited, can provide increasing resolution even for complex and discontinuous flows. Here, we look at a chaotic nonlinear context with the Euler equations. The emphasis of this scheme is improving the accuracy per DOF for realistic flows and using a large effective CFL value (compared with existing time-explicit solvers).

### 1.4 New contributions of this study

This study is the first application of the ADER-DT time discretization to the 3D Euler equations. It is the first application of a new, optimized function-based WENO limiting scheme to the 3D Euler equations as well. The author is not aware of another publication that details the temporal-only convergence of an ADER algorithm by using a temporal order of accuracy much less than the spatial order to force temporal error to dominate the total error. Also, this paper performs an investigation into the errors associated with dimensional splitting in Cartesian geometry with an “A”-grid or collocated grid. This study also uses a new time-averaging of the flux time derivatives before the Riemann solver to reduce the number of Riemann solves and the volume of MPI data transfers in an ADER time discretization.
2 | MATHEMATICAL FORMULATION AND IMPLEMENTATION

2.1 | Algorithmic summary

For each dimension, the numerical method presented in this paper is summarized in Algorithm 1, where \( N \) is the spatial order of accuracy and \( N_T \) is the temporal order of accuracy. Details of each step in the algorithm will be expanded throughout the rest of this section.

2.2 | Inviscid Euler equations

The 3D compressible nonhydrostatic Euler equations, though inviscid, form the basis of atmospheric simulation and a good testbed for prospective numerical methods. The equation set describing this flow is:

\[
\begin{align*}
\frac{\partial}{\partial t} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho \theta \\ \rho \theta \end{bmatrix} &+ \frac{\partial}{\partial x} \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho u \theta \\ \rho u \theta \end{bmatrix} + \frac{\partial}{\partial y} \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ \rho v \theta \\ \rho v \theta \end{bmatrix} + \frac{\partial}{\partial z} \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p - p_H \\ \rho w \theta \\ 0 \end{bmatrix} = 0 \\
\rho_H &- \frac{1}{g} \frac{\partial p}{\partial z} = 0
\end{align*}
\]

where \( \rho \) is density, \( u, v, \) and \( w \) are winds in the \( x, y, \) and \( z \)-directions, respectively, \( \theta \) is temperature related to temperature, \( T \), by \( \theta = T(P_0/P)^{\gamma/\gamma'} \), \( P_0 = 10^5 \) Pa, \( g = 9.8 \text{ m} \cdot \text{s}^{-2} \) is acceleration due to gravity, \( p = C_0(\rho \theta)' \) is pressure, \( C_0 = R_g P_0 / c_p \), \( R_g = 287 \text{ J} \cdot \text{kg}^{-1} \cdot \text{K}^{-1} \), \( \gamma = c_p / c_v \), \( c_p = 1,004 \text{ J} \cdot \text{kg}^{-1} \cdot \text{K}^{-1} \), \( c_v = 717 \text{ J} \cdot \text{kg}^{-1} \cdot \text{K}^{-1} \), and \( P_0 = 10^5 \) Pa. The nature of the constant, \( C_0 \), is to provide a convenient proportionality between \( p \) and \( (\rho \theta)' \) in the equation of state. The hydrostatic density and hydrostatic potential temperature are related to hydrostatic pressure through the following relation:

\[
\frac{dp_H}{dz} \equiv C_0 \frac{d(\rho_H \theta_H)'}{dz} = -\rho_H g
\]

Equation (1) can be cast in vector form as \( \partial_t \mathbf{q} + \partial_x \mathbf{f} + \partial_y \mathbf{g} + \partial_z \mathbf{h} = \mathbf{s} \), where bold-faced variables represent vector quantities.

Algorithm 1: Update the fluid state over a full time step. \( N \) is the spatial order of accuracy, and \( N_T \) is the temporal order of accuracy.

1: for each direction (\( x, y, \) and \( z \)) do
2:     — Parallel communication: If this is the \( x \) or \( y \) direction, exchange stencil halo values with neighboring MPI tasks.
3:     for each fluid variable do
4:         — Transform a stencil of \( N \) cell averages into \( N \) WENO-limited polynomial coefficients with Algorithm 3.
5:         — Sample the \( N \) limited polynomial coefficients at \( N_T \) spatial Gauss–Legendre–Lobatto (GLL) points. This can be done with a nonsquare matrix–vector product.
6:         — Compute \( N_T - 1 \) time derivatives of the flux and state at each of the \( N_T \) spatial GLL points using differential transforms of the PDE using Algorithm 4 (see section 2.5.3).
7:         — Integrate state and flux time derivatives into a time-averaged state and flux using Algorithm 5.
8:         — Store cell-edge–time averages of the state and flux vectors into large arrays for all cell edges in this MPI task.
9:     — Parallel communication: Exchange flux and state cell-edge estimates with neighboring MPI tasks.
10:    for each cell interface in this direction do
11:        — Compute a single-valued numerical flux from discontinuous estimates from adjacent cells using the linear upwind Godunov state using Algorithm 2.
12:        — Store single-valued cell-edge fluxes into large arrays for all cell edges in this MPI task.
13:    for each cell do
14:        — Update the fluid state using the time-averaged tendency for the cell as shown below:
15:        x-direction:
16:            \( \tilde{q}_{n+1} = \tilde{q}_n - \Delta t \left( \hat{f}_{n+1/2} - \hat{f}_{n-1/2} \right) / \Delta x \)
17:        y-direction:
18:            \( \tilde{q}_{n+1} = \tilde{q}_n - \Delta t \left( \hat{g}_{n+1/2} - \hat{g}_{n-1/2} \right) / \Delta y \)
19:        z-direction:
20:            \( \tilde{q}_{n+1} = \tilde{q}_n - \Delta t \left( \hat{h}_{n+1/2} - \hat{h}_{n-1/2} \right) / \Delta z + \Delta t \hat{\mathbf{s}}_k \)
21:        where \( \hat{f} \) is the time-averaged flux, \( \tilde{q} \) is the cell-averaged fluid state, and \( \hat{\mathbf{s}} \) is the time-averaged and cell-averaged source term.
2.3 Spatial discretization

2.3.1 Dimensional splitting

Fully multidimensional FV requires $N^{D-1}$ times more computations than dimensionally split FV. While a dimensionally split simulation can use a CFL value of one, a fully multidimensional simulation must use a smaller time step for stability. These two situations motivate the use of dimensional splitting where possible.

A second-order-accurate alternating Strang splitting is used in this study, which reverses the order of the dimensional solves for each time step. If the update in a given direction can be cast as $q_{n+1} = RHS(q_n)$, where $n$ is the time step index, then the update proceeds as follows:

$$q^*_n = RHS_x(q_n)$$

$$q_{n+1} = RHS_x(q^*_n)$$

$$q^*_n = RHS_y(q_{n+1})$$

$$q_{n+1} = RHS_y(q^*_n)$$

$$q^*_n = RHS_z(q_{n+2})$$

$$q_{n+2} = RHS_z(q^*_n)$$

(4)

To be more precise, the $RHS_x, RHS_y, and RHS_z$ terms are describing the full finite-volume updates in each of those directions, respectively, described in Equation (7). The same time step of $\Delta t$ is used in all $RHS$ updates.

For the dynamics solve, the equations in the $x, y,$ and $z$-directions are: $\partial_t q + \partial_x f = 0, \partial_t q + \partial_y g = 0,$ and $\partial_t q + \partial_z h = s,$ respectively. For sake of simplicity, in the rest of this article each will be described in general as

$$\partial_t q + \partial_x f = s$$

(6)

except for the $z$-direction, $s \equiv 0.$

2.3.2 Finite-volume semi-discretized equations

For the Finite-volume discretization, the 1D Equation (6) is integrated over a 1D local cell domain, $\Omega_i \in [x_{i-1/2}, x_{i+1/2}]$, where $x_{i \pm 1/2} = x_i \pm \Delta x_i / 2$, and $\Delta x_i$ is the grid spacing for the cell of index $i$. This gives rise to the following semi-discretized equations:

$$\frac{\partial \tilde{q}_i}{\partial t} + \frac{1}{\Delta x_i} (F_{i+1/2} - F_{i-1/2}) = \frac{1}{\Delta x_i} \int_{x_{i-1/2}}^{x_{i+1/2}} s_i(x, t) dx$$

(7)

$$F_{i-1/2}(t) = R \left[ \hat{f}_{i-1,j} \left( x_{i+1/2}, t \right), \hat{f}_{i,j} \left( x_{i-1/2}, t \right) \right]$$

(8)

where $\tilde{q}_i \equiv \frac{1}{\Delta x_i} \int_{x_{i-1/2}}^{x_{i+1/2}} q(x, t) dx$ and $\hat{f}_i(x, t)$ and $\hat{s}_i(x, t)$ are spatial reconstructions of the vectors $f$ and $s$, respectively, over the $i$th cell domain, $\Omega_i$, (the time dimension is left continuous for now), and $R \left( f^+, f^- \right)$ is a Riemann solver to reconcile the multivalued flux at a cell interface due to samples from disparate reconstructions at the same location in space (Algorithm 2). Equation (7) forms the basis of step 15 of Algorithm 1.

2.3.3 Spatial recovery operator

Nth-order-accurate FV methods recover intra-cell variation by projecting a stencil of $N$ surrounding cell averages onto a polynomial of order $N - 1$. In the present case, and for reasons that will be more fully discussed, the stencil is projected onto $N_T$ spatial GLL points that span the domain $\Omega_i,$ where $N_T$ is the temporal order of accuracy. The polynomial coefficients are computed using the constraints of Equation (12), and the resulting polynomial is sampled at the GLL points.

The vertical profiles of density and potential temperature are dominated by hydrostatic balance, upon which relevant motions result from relatively small perturbations. Even small deviations from hydrostatics will lead to significant changes in the flow. Because of this and the fact that hydrostatics is not polynomial in nature, directly reconstructing hydrostatic variables with polynomials often leads to spurious motions that contaminate the simulation. Hydrostatics is defined by:

$$\frac{d\rho_H(z)}{dz} = -\rho_H(z) g \Rightarrow C_0 \frac{d\rho_H(z)'}{dz} = -q_{H,1}(z) g \quad (9)$$

where the $H$ subscript denotes the hydrostatically balanced profile that depends only on the vertical dimension. Therefore, before reconstruction, hydrostatics is removed from $\rho$ and $\rho_H: \tilde{\rho}'(x, y, z, t) = \tilde{\rho}(x, y, z, t) - \rho_H(z)$ and $\tilde{\rho}'(x, y, z, t) = \tilde{\rho}(x, y, z, t) - \tilde{\rho}_H(z).$ After the perturbations are projected onto GLL points, hydrostatics (which has been precomputed at GLL points) is added back: $\rho_{GLL,j} = \rho_{GLL,j}' + \rho_{H,GLL,j}$ and $\rho_{H,GLL,j} = \rho_{H,GLL,j}' + \rho_{H,H,GLL,j},$ where $j$ is the GLL point index within the cell.

2.3.4 Linear upwind Riemann solver

This corresponds to step 12 in Algorithm 1. For the dynamics, a characteristics-based flux vector splitting is used to compute an upwind flux using linear characteristics from a locally “frozen” background state. Consider a PDE: $\partial_t q + \partial_x f = 0$ with a flux Jacobian: $A = \partial f / \partial q$.

Left-multiplying the PDE by the flux Jacobian and assuming that it is locally constant in space and uniform in time gives $\partial_t f + A \partial_x f = 0.$ The flux Jacobian can then
be diagonalized into $A = RAR^{-1}$, where $R$ is a matrix whose columns are right eigenvectors, and $\Lambda$ is a diagonal matrix whose diagonal components are eigenvalues, $\lambda_p$. Using this diagonalization gives $\partial_t f + RAR^{-1} \partial_x f = 0$. Left-multiplying by $R^{-1}$ gives $\partial_t w + \Lambda \partial_x w = 0$, where $w = R^{-1} f$ is the vector of five flux-based characteristic variables that each travel at a finite speed of $\lambda_p$. The diagonalizations for each direction are given in Appendix A.

To compute the locally frozen flux vector, the linear average at the interface is used: $\hat{q} = (q^- + q^+) / 2$. Then, the locally frozen eigenmatrices, $\hat{R}$, $\hat{\Lambda}$, and $\hat{L} = \hat{R}^{-1}$ are built on the averaged state, $\hat{q}$. Next, the flux-based characteristic variable vector, $w_{upw}$, is computed as:

$$w_{upw,j} = \begin{cases} \hat{\epsilon}_j \cdot \hat{f} & \text{if } \lambda_j > \epsilon \\ \hat{\epsilon}_j \cdot (\hat{f}^+ + \hat{f}^-) / 2 & \text{otherwise} \end{cases}$$

where $\hat{\epsilon}_j$ is the $j$th row of $\hat{L}$, $\lambda_j$ is the $j$th diagonal component of $\hat{\Lambda}$, and $\epsilon$ is a small tolerance to handle characteristic speeds very close to zero. Finally, the upwind flux is obtained as $R(f^+ - f^-) = \hat{R}w_{upw}$. The Riemann solver for a single cell interface is detailed in Algorithm 2.

### 2.4 Weighted essentially non-oscillatory (WENO) limiting

Weighted essentially non-oscillatory (WENO) limiting corresponds to step 5 in Algorithm 1. WENO interpolation uses multiple lower-order-accurate candidate polynomial approximations of the solution (e.g., at a cell edge in finite-volume schemes) Liu et al. (1994). In these schemes, the polynomial coefficients (which require a matrix–vector product) are never actually computed. Rather, a simpler dot product against stencil values gives the interpolation for each polynomial at a single point, $x_i$. Next, a linear system is formed such that a weighted combination of lower-order interpolations at $x_i$ gives the high-order interpolation at $x_i$. This system is solved to determine the ideal weights, and therefore, the ideal weights are determined by the interpolants and the grid geometry.

This approach is not directly usable by ADER schemes because ADER needs higher-order variation in space so that it can translate that variation into the temporal domain via the PDE’s relation of space and time (see section 2.5 for more information). Thus, for use with ADER, we need the actual polynomial coefficients. This study adopts the function-based WENO approach

| Algorithm 2: Compute the upwind fluxes at an interface |
|-------------------------------------------------------|
| **Input:** $q_L, q_R, f_L, f_R$ (left and right state and fluxes, each vectors of size 5) |
| **Output:** $f_U$ (upwind fluxes, vector of size 5) |
| 1 Compute $q_i = (q_L + q_R) / 2$ // the interface average |
| 2 Compute the left eigenmatrix, $L$, using $q_i$ and Appendix A. |
| 3 Compute left and right flux-based characteristic variables: $w_L = Lf_L$; $w_R = Lf_R$ (matrix multiplication) |
| /* Compute the upwind characteristic variables below, Equation (10): */ |
| 4 for each of the five characteristic variables, $j$, do |
| /* See Appendix A and section 2.3.4 for details on $\lambda_j$ below. */ |
| /* $w_U$ is a vector of size 5, and $\epsilon$ is a small number such as $10^{-6}$ or smaller. */ |
| 5 if $\lambda_j > \epsilon$ then |
| 6 $w_{U,j} = w_{L,j}$ |
| 7 else if $\lambda_j < \epsilon$ then |
| 8 $w_{U,j} = w_{R,j}$ |
| 9 else |
| 10 $w_{U,j} = (w_{R,j} + w_{L,j}) / 2$ |
| 11 Compute the right eigenmatrix, $R$, using $q_i$. |
| 12 Compute the upwind flux as $f_U = R_i w_U$ (matrix multiplication) |
FIGURE 1  Example of WENO limiting, assuming a grid spacing of 1, where \( P_L(x), P_C(x), \) and \( P_R(x) \) are the left-biased, centered, and right-biased lower-ordered polynomials, respectively; and \( P_H(x) \) and \( P_W(x) \) are the high-order and WENO polynomials over the center cell’s domain, respectively. Thick black lines denote the stencil average values for this example. Note the domains of \( P_L, P_C, \) and \( P_R \) all overlap in the center cell. Note this figure also appears in Norman and Larkin (2020) [Colour figure can be viewed at wileyonlinelibrary.com]

of Norman and Nair (2018), which is similar in nature to the WENO implementations in (Capdeville, 2008a; 2008b; Norman, 2015b). In this approach, all of the polynomial coefficients are explicitly formed, and the WENO-limited polynomial is a weighted combination of the lower-order polynomials. The WENO-limited polynomial is valid over the center cell domain of \( N \) stencil cell averages.

In this study, for an \( N \)th-order-accurate WENO reconstruction (where \( N \) is an odd integer), the lower-order polynomials are constrained according to Equation (11):

\[
\frac{1}{\Delta x} \int_{\Omega_{i+j, h-1+s}} p_{L,j}(x) \, dx = \bar{q}_{i+j, h-1+s} \; ; \quad \forall j \in \{1, \ldots, h\}, \quad s \in \{0, \ldots, h\}
\]  

where \( h = (N - 1)/2 \) is the size of the stencil “halo,” \( \bar{q}_{i,j} \) is the average of the quantity being reconstructed in cell \( i \), and \( i \) is the index of the stencil’s center cell. What this describes is a moving window within the stencil to constrain the lower-order polynomials (see Figure 1 as an example). Next, the high-order-accurate polynomial is constrained according to Equation (12):

\[
\frac{1}{\Delta x} \int_{\Omega_{i+j, h}} p_{H}(x) \, dx = \bar{q}_{i+j, h} \; ; \quad \forall s \in \{-h, \ldots, h\}
\]  

Once these are computed, a “bridge” polynomial, \( p_{L,B}(x) \), is created such that in a weighted combination with the lower-order polynomials, we retrieve the high-order polynomial as shown in Equation (13):

\[
p_{H}(x) = \sum_{j \in \{1, \ldots, h\}} \gamma_j p_{L,j}(x) \quad \Rightarrow \quad p_{L,B}(x) = \frac{1}{\gamma_B} \left( p_{H}(x) - \sum_{j \in \{1, \ldots, h\}} \gamma_j p_{L,j}(x) \right)
\]  

While the bridge polynomial has higher-order derivatives, its accuracy is still lower-order. In this case, any set of ideal weights, \( \gamma_j \), will produce the high-order polynomial. Therefore, the ideal weights are free parameters that can be optimized to achieve desired properties in the resulting WENO interpolation. The only constraints placed on the ideal weights is that they be spatially symmetric, all positive, and that they sum to one. Larger weights for the bridge polynomial and the lower-order polynomials near the stencil center will result in a more oscillatory WENO interpolation.

Next, the total variation of each lower-order polynomials is computed according to Equation (14):

\[
TV_j = \frac{1}{\Delta x} \int_{\Omega_{i+j, h-1}} \left( \sum_{k=1}^{N-1} \left| \frac{\partial^k p_{L,j}(x)}{\partial x^k} \right| (\Delta x)^k \right)^2 dx \; ; \quad \forall j \in \{1, \ldots, h, B\}
\]  

The TV estimate is the integral of the squared derivative summed across all available derivatives and normalized by the grid spacing. The TV of the bridge polynomial has higher-order derivatives, and it is natural to assume it will be inherently larger than the other TV estimates.
Therefore, a handicapping parameter, $\sigma \in (0, 1]$, is introduced to reduce $TV_B$ to a lower value closer to the average of the other TV estimates. This is shown in Equation (15):

$$TV_B = \frac{1}{h} \sum_{j=1}^{h} TV_{Lj} + \sigma \left( TV_B - \frac{1}{h} \sum_{j=1}^{h} TV_{Lj} \right)$$ (15)

The smaller the value of $\sigma$, the closer $TV_B$ becomes to the other TV estimates, and the more oscillatory the WENO interpolation becomes. The value for $\sigma$ is also a free parameter that can be optimized to achieve desired properties in the WENO interpolation.

Next, the WENO weights are computed as the inverse square of the TV estimates so that the lower-order polynomials with the most variation are weighted the lowest. This is described by Equation (16):

$$\omega_j^* = \frac{\gamma_j}{TV_j^2 + \epsilon}$$ (16)

where $\epsilon$ is a small number used to avoid division by zero. This code uses $\epsilon = 10^{-20}$. Even in single precision (32-bit IEEE floating point), the smallest number that can be represented is $10^{-38}$, and this choice of $\epsilon$ has not caused any problems in single precision in any of the experiments. Next, the WENO-limited weights are normalized into a convex set of weights according to Equation (17):

$$\omega_j = \frac{\omega_j^*}{\sum_k \omega_k^* + \epsilon}$$ (17)

From (16) and (17), it is evident that if all low-order polynomials have the same variation (i.e., the flow is “smooth”), the ideal weights are recovered. Finally, the WENO-limited interpolation is given by Equation (18) below, where the only difference from Equation (13) is that the WENO-limited weights are used in place of the ideal weights.

$$p_{\text{WENO}}(x) = \sum_{j \in \{1, \ldots, h, B\}} \omega_j p_{Lj}(x)$$ (18)

The WENO limiting process is detailed in Algorithm 3.

Algorithm 3: Compute WENO-limited polynomial coefficients from a stencil of values

**Input:** A stencil of $N$ cell averages

- Ideal weights ($\gamma_{Lj}$) for $j \in \{1, \ldots, h, B\}$ where $h = (N - 1)/2$
- Bridge polynomial TV reduction parameter ($\sigma$)

**Output:** $N$ WENO-limited polynomial coefficients valid over the center cell

1. Reconstruct the $N$ high-order polynomial coefficients using the constraints from Equation (12).
2. Reconstruct the coefficients of $(N + 1)/2$
   lower-ordered polynomials using the constraints defined in Equation (11). These polynomials are labeled $p_{Lj}(x)$, where $j \in \{1, \ldots, h\}$.
3. Compute the coefficients of the “bridge” polynomial, $p_{L,B}(x)$ using Equation (13).
4. Compute the TV of the lower-ordered polynomials, $TV_j$ for $j \in \{1, \ldots, h, B\}$, using Equation (14).
5. Reduce $TV_B$ according to Equation (15).
6. Compute WENO-limited weights, $\omega_j$, using Equations (16) and (17).
7. Map the WENO weights, $\omega_j$, according to Henrick et al. (2005).
8. Compute the WENO-limited polynomial using Equation (18).

This optimization are given in Table 1 along with the maximum overshoot in the 1D transport experiment with square wave data. The table shows that in all cases, optimizing for minimal $L_1$ error also happened to reduce the maximum overshoot in the simulations to values $5 \times 10^{-3}$ or lower for all orders of accuracy. This is a stark reduction in overshoot magnitude compared with the unlimited cases, which all had oscillations on the order of 0.1 or larger.

2.5 ADER-DT time discretization

This section corresponds to step 7 in Algorithm 1.

2.5.1 The ADER approach

The ADER time discretization Titarev and Toro (2002); Toro and Titarev (2005); Toro and Titarev (2006) can be thought of as a higher-order extension of the Lax–Wendroff Roe (1984) approach. Typical fluid
2.5.2 | ADER with differential transforms

Another implementation called ADER-DT relies on differential transforms (DTs) to transform the PDE into a recurrence relation to generate higher-order time and space–time derivatives Norman and Finkel (2012); Norman (2013); Norman (2014). The temporal polynomial coefficients can be integrated to a time average and inserted into the spatial operator as if it were semi-discrete, requiring no changes to the original semi-discretized operator. The downside to the implementation in Norman and Finkel (2012); Norman (2013) is that the computational complexity is $\mathcal{O}(N^{2(D+1)})$, which is incredibly expensive, compared with the expected efficient computational complexity of a space–time operator of $\mathcal{O}(N^{D+2})$. In Norman and Nair (2018), however, the ADER-DT approach was significantly improved by only computing DTs in the temporal dimension on a set of GLL points in space. This study uses the improved implementation in Norman and Nair (2018) expanded to the Euler equations.

The $k$th differential transform of a function, $f$, is the $k$th term in the Taylor series of $f$. The inverse transform is the Taylor series itself.

$$F(k) = \frac{1}{k!} \frac{\partial^k f(t)}{\partial t^k}; f(t) = \sum_{k=0}^{N-1} F(k) t^k \quad (19)$$

Using these two definitions, the rules to transform various nonlinear operators can be derived, and transform rules for most common operators can be found in Tsukanov and Hill (2000).

First, the spatial reconstruction procedure transforms a stencil of $N$ cell averages of the model state, $\mathbf{q}$, into $N_T$ GLL points, where $N$ and $N_T$ are the spatial and temporal orders of accuracy, respectively. Next, the PDE system (6) is transformed into nodal derivative space only in the temporal dimension using DTs, leaving the spatial dimension in nodal space on GLL points.

2.5.3 | Temporal differential transforms of the Euler equations

Since the DTs are only being performed in time for a given GLL point, the spatial dimensions are neglected for simplicity, and the transformed PDEs become in the $x$, $y$,
and z-directions, respectively:

\[
Q(k_i + 1) = -\frac{1}{k_i + 1} \partial_z F(k_i) \tag{20}
\]

\[
Q(k_i + 1) = -\frac{1}{k_i + 1} \partial_j G(k_i) \tag{21}
\]

\[
Q(k_i + 1) = -\frac{1}{k_i + 1} \left[ \partial_z H(k_i) + S(k_i) \right] \tag{22}
\]

where a capital letter denotes the transformed function in time, and \(k_i\) denotes the \((k_i)\)th-order DT in time, which is also the \((k_i)\)th term in the temporal Taylor series.

Before computing the DTs of the flux and source terms, the following auxiliary functions are defined: \(\phi_{ij}(t) = q_i(t) q_j(t) / q_1(t)\) and \(p^{\ast}(t) = q_3(t)\). The symmetry \(\phi_{ij} = \phi_{ji}\) can be used to reduce computations. Given these functions, the following are the temporal DTs of the flux and source vectors in the Euler equations.

\[
F(k_i) = \begin{bmatrix} Q_2(k_i) \\ \Phi_{22}(k_i) + P(k_i) \\ \Phi_{23}(k_i) \\ \Phi_{24}(k_i) \\ \Phi_{25}(k_i) \end{bmatrix}
\]

\[
G(k_i) = \begin{bmatrix} Q_3(k_i) \\ \Phi_{32}(k_i) + P(k_i) \\ \Phi_{34}(k_i) \\ \Phi_{35}(k_i) \end{bmatrix}
\]

\[
H(k_i) = \begin{bmatrix} Q_4(k_i) \\ \Phi_{42}(k_i) \\ \Phi_{43}(k_i) + P(k_i) \\ \Phi_{45}(k_i) \end{bmatrix}
\]

\[
S(k_i) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -gQ_1(k_i) \\ 0 \end{bmatrix} \tag{24}
\]

The flux terms, \(q_i q_j / q_1\), have the following differential transforms:

\[
\Phi_{ij}(k_i) = \frac{1}{Q_1(0)} \sum_{r_i=0}^{k_i} \left[ Q_1(r_i) Q_j(k_i - r_i) - Q_1(r_i) \Phi_{ij}(k_i - r_i) \right]; \forall i \in \{2, 3, 4\}, j \in \{2, 3, 4, 5\}
\]

Pressure has the following differential transform described in two steps in Equations (26 and 27), where \(P^{\ast}\) is the differential transform of the expression \(p^{\ast} = (\rho \theta)^{\gamma} = q_5(t)\):

\[
P(k_i) = \frac{C_0}{2} p^{\ast}(k_i); \forall k_i > 0 \tag{26}
\]

\[
P^{\ast}(k_i) = \frac{1}{Q_5(0)} \left[ \gamma P^{\ast}(0) Q_5(k_i) + \frac{1}{k_i} \sum_{r_i=0}^{k_i-1} [(k_i - r_i) \times (\gamma P^{\ast}(r_i) Q_5(k_i - r_i) - Q_5(r_i) P^{\ast}(k_i - r_i))] \right] \tag{27}
\]

These functions are initialized as follows: \(\Phi_{ij}(0) = Q_i(0) Q_j(0) / Q_1(0); P(0) = C_0 P^{\ast}(0);\) and \(P^{\ast}(0) = Q_5(0)^{\gamma}\).

The summations used to compute the DTs for \(\Phi_{ij}(k_i)\) and \(P^{\ast}(k_i)\) include the actual values \(\Phi_{ij}(k_i)\) and \(P^{\ast}(k_i)\) themselves, even though they are not technically defined by that point. In these cases when a function’s DT depends on itself, it is assumed to be zero in the summation. This can be implemented in practice by initializing each of these functions to zero before entering the loops to compute the DTs (see steps 1–7 in Algorithm 4).

### 2.5.4 Implementation specifics

To compute the spatial derivative of \(F(k_i)\), an \(N_T \times N_T\) matrix, \(M_D\), is created that transforms the \(N_T\) GLL points over the cell into \(N_T\) polynomial coefficients, differentiates the polynomial, and then samples the differentiated polynomial at the same \(N_T\) GLL points across the cell. This can be done with a single matrix–vector multiplication.

### 2.5.5 Algorithmic simplifications

Traditionally, ADER methods perform a Riemann solve not only for the flux (using more traditional solvers such as HLLC Toro et al. (1994)), but also for the temporal derivatives as well. This sets up derivative Riemann problems that are typically solved using the locally constant linear upwind state. In this implementation, however, the algorithm is simplified by computing a single, traditional Riemann solve on the time-averaged state. This has two advantages over traditional ADER approaches: (a) only one Riemann solve needs to be performed per cell edge per time step, and (b) less data needs to be communicated between parallel nodes. The errors associated with this simplification are investigated in section 3.3. This corresponds to step 8 in Algorithm 1.
2.5.6 Algorithm for computing time derivatives with DTs in the \( x \) direction

Algorithm 4 details how to compute the time derivatives of the state and fluxes efficiently in the \( x \) direction. The \( y \) and \( z \) directions are analogous. Local arrays of size \( (\text{ngll}, \text{ngll}) \) are needed, where the first index is for the modal time \( \text{DT} \) and the second index is for the nodal spatial GLL point) for the following variables:

\[ r, \rho, \rho u, \rho v, \rho w, \rho \theta, \theta, \theta \psi, \psi \]

Once this algorithm is finished, \( \text{DTs} \) of all of the variables from order 0 to \( \text{ngll}-1 \) have been computed. Recall that the \( n \)th-order \( \text{DT} \) is the coefficient to the \( n \)th term in the Taylor series in time according to Equation (19).

Once the \( \text{DTs} \) of the variables have been computed, the time average of each of these variables can be computed. It is often most convenient to store the time average in the zeroth-order \( \text{DT} \) index as follows, and Algorithm 5 shows how to do this. The time-averaged state can be immediately stored for the Riemann solver. The time-averaged fluxes for \( \rho, \rho u, \rho v, \rho w, \text{ and } \rho \theta \) are computed as follows (using the variables used in Algorithm 4):

\[
\begin{align*}
\rho - \text{flux} & \quad \rho u - \text{flux} & \quad \rho v - \text{flux} & \quad \rho w - \text{flux} & \quad \rho \theta - \text{flux} \\
ru(0,:) & \quad ruu(0,:) & \quad ruv(0,:) & \quad ruw(0,:) & \quad rut(0,:) \\
& + C0\times rt\_gamma(0,:) \\
rt\_gamma & \\
\end{align*}
\]

2.6 Boundary conditions

In all test cases in this study, the vertical boundaries are assumed to be no-slip solid walls, and the horizontal boundaries are assumed to be periodic. Since perturbations are stored for \( \rho \) and \( \rho \theta \), the ghost cells for the solid wall are the same as the nearest physical domain value (mimicking a zero gradient) for everything but the normal (vertical) velocity, which is set to zero in the ghost cells. When GLL points of the state and flux are computed, the vertical velocity coinciding with the vertical boundaries is set to zero to enforce a no-flux boundary condition. Setting the vertical wind to zero in the vertical ghost cells acts to help the recovery operators near the vertical boundaries to approach zero naturally so that setting the actual boundary samples to zero does not create oscillations. The Straka density current test case is quite sensitive to boundary condition implementation, and it gives the expected answers with this approach.
Algorithm 5: Compute the time average. This algorithm uses zero-based indexing.

**Input:** \( DTs(:, :) \) The first index is the time DT order, and the second index is the spatial GLL point.

**Output:** \( DTs(0,:) \) The time average is stored in the zeroth time DT index.

```plaintext
1. \( dtmult = dt \) // \( dt \) is the time step
2. for \( kt \) from 1 to \( ngl-1 \)
3. \( DTs(0,:) = DTs(0,:) + DTs(kt,:) * dtmult / (kt+1) \)
4. \( dtmult = dtmult * dt \)
```

## 2.7 2D option

Since the vast majority of test cases are in literature in only two spatial dimensions, the code can switch to 2D mode with a few simple modifications. First, only one cell is used in the \( y \) dimension, and the \( y \) location is assumed to be the middle of the \( y \) direction domain. Next, the \( v \) velocity is initialized to zero. Finally, the \( y \) direction update in the dimensionally split time-stepping is not performed. Therefore, the state vector never feels the \( y \) direction forcing, and \( v \) remains zero for all time.

## 3 NUMERICAL TEST CASES AND VALIDATION

This study uses literature standard test cases to validate the expected behavior of the nonhydrostatic model. All experiments use an initial CFL value of 0.8 unless otherwise specified and use a constant time step throughout the simulation. All experiments use a temporal order of accuracy that matches the spatial order of accuracy unless otherwise indicated. Only the convergence studies are performed in double precision. All other simulations are performed in single precision. All error norms are computed at the final time, and they are defined as follows:

\[
L_1 = \frac{1}{N_{cells}} \sum |q_{\text{exact}} - q_{\text{numer}}| \quad (28)
\]

\[
L_2 = \sqrt{\frac{1}{N_{cells}} \sum |q_{\text{exact}} - q_{\text{numer}}|^2} \quad (29)
\]

\[
L_\infty = \max |q_{\text{exact}} - q_{\text{numer}}| \quad (30)
\]

### 3.1 1D transport

High-order convergence will be demonstrated in section 3.2 in a 2D transport setting with smooth data. However, in realistic atmospheric simulations, the data are rarely if ever smooth enough for high-order convergence to be observed in practice. It begs the question as to why one should bother with high-order accuracy. The answer is that even though high-order convergence is not observed, the constant on the error term still goes down with higher-order accuracy, particularly if suitable limiting is used. To demonstrate that the solution improves even with highly discontinuous data, a “square” wave, a “triangle” wave, and a cosine bell are advected in a 1D transport setting with uniform winds. The governing PDE is \( \partial_t q + \partial_x q = 0 \) on a domain of \([0, 1]\), and \( q \) is initialized as follows:

\[
q(x, t = 0) =\begin{cases} 
1 & \text{if } 0.1 \leq x \leq 0.3 \\
10(x - 0.4) & \text{if } 0.4 \leq x \leq 0.5 \\
10(0.6 - x) & \text{if } 0.5 < x \leq 0.6 \\
\cos(10\pi(x - 0.8)) + 1 & \text{if } 0.7 \leq x \leq 0.9 \\
0 & \text{otherwise}
\end{cases} \quad (31)
\]

Figure 2 plots the initial data and final solution after one revolution about the domain with 50 cells and a CFL of 0.8 with ADER time stepping and WENO limiting over a variety of orders of accuracy. There is also a zoom of the square wave at different orders of accuracy along with a comparison of ninth-order accuracy with and without WENO limiting. The most notable aspect is that the jump in resolution from third order to fifth order is quite large, which seems to suggest that fifth order might be considered a minimum desired order of accuracy for a simulation with discontinuities and WENO limiting. Also, note that the resolution of the square wave’s jump increases monotonically with increasing order, as does the resolution of the triangle wave. The cosine bell is a bit smoother, and as expected, all orders of accuracy greater than third order resolve it quite well. Note that with only 50 cells, even the cosine bell is not all that smooth as it is expressed on the coarse grid, but visual comparison is clearer at coarser resolution. Finally, note the significant reduction in oscillations at ninth-order accuracy with WENO limiting.

Table 2 gives a more quantitative view of this test case with 100 cells and the same CFL value of 0.8. In all cases, the \( L_1 \) error is lower with WENO limiting than without. The WENO \( L_\infty \) error is better than the unlimited \( L_\infty \) error at third and fifth orders of accuracy but worse at seventh and ninth orders of accuracy. This suggests that WENO experiences some degradation in the steepness of discontinuities at the higher orders of accuracy relative to unlimited reconstruction. As the order of accuracy is increased, WENO limiting improves the error monotonically for all error norms. While unlimited reconstruction improves errors monotonically with increasing...
order for the $L_2$ and $L_\infty$ errors, this is not the case in the $L_1$ error, which increases from seventh to ninth order. This demonstrates that with WENO limiting, high-order accuracy still reduces the error, even if the error is not converging in a high-order fashion. Finally, without limiting, the overshoots were all $O(10^{-1})$, but with WENO limiting, the overshoots were $O([10^{-5}, 10^{-3}])$, depending on the order of accuracy.

3.2  2D transport

3.2.1  Dimensionally split versus fully multidimensional

Since this method uses dimensional splitting, it is worthwhile to investigate how this affects the accuracy in a context that easily allows exact error norms: 2D transport. To get an idea of how the second-order-accurate alternating Strang splitting affects the accuracy, solid body rotation (SBR) of a cone, a cosine bell, and a so-called “slotted cylinder” is performed for one full rotation with a multidimensional approach and a dimensionally split approach. In general, the multidimensional simulation must use a CFL value less than 1/2 while the dimensionally split can use a CFL value up to one. The domain is assumed to be $[0, 1] \times [0, 1]$ subdivided uniquely and equally by $100 \times 100$ equally spaced finite-volume cells in Cartesian coordinates.

The 2D transport PDEs are:

$$\frac{\partial q}{\partial t} + \frac{\partial qu}{\partial x} + \frac{\partialqv}{\partial y} = 0.$$  (32)
They are solved in the same manner as the 3D Euler solver, with the upwind Riemann solver being defined as using the upwind value according to the wind velocities at an interface. For 2D WENO reconstruction, this is performed in a tensored manner for fully multidimensional reconstruction. SBR winds do not change in time, and therefore, the temporal differential transform is simple and linear, for example:

$$ Q(k_i + 1) = -\frac{1}{k_i + 1} \delta_x [uQ(k_i)] - \frac{1}{k_i + 1} \delta_y [vQ(k_i)] \quad (33) $$

For solid body rotation, the winds are initialized as $a = -\omega (y - 0.5)$ and $v = \omega (x - 0.5)$, where $\omega = 2\pi$. The tracer field is initialized, as shown in Figure 3, with a linear cone ($\psi_1$), a cosine bell ($\psi_2$), and a slotted cylinder ($\psi_3$). They are defined as follows:

$$ \psi_1(x, y, 0) = \max(1 - r_1, 0) \quad (34) $$

$$ \psi_2(x, y, 0) = \begin{cases} \frac{1}{4} \cos(\pi r_2) + 1 & \text{if } r_2 \leq 1 \\ 0 & \text{otherwise} \end{cases} \quad (35) $$

$$ \psi_3(x, y, 0) = \begin{cases} 1 & \text{if } r_3 \leq 1 \text{ and } |x - x_{C,3}| \geq \frac{2x_3}{6} \text{ or } y - y_{C,3} \geq \frac{2y_3}{1.5} \\ 0 & \text{otherwise} \end{cases} \quad (36) $$

$$ r_i = \sqrt{\left(\frac{x - x_{C,i}}{x_{R,i}}\right)^2 + \left(\frac{y - y_{C,i}}{y_{R,i}}\right)^2}, \quad i \in \{1, 2, 3\} \quad (37) $$

where $x_{C,1} = 0.50$, $y_{C,1} = 0.25$, $x_{R,1} = y_{R,1} = 0.15$, $x_{C,2} = 0.25$, $y_{C,2} = 0.50$, $x_{R,2} = y_{R,2} = 0.2$, $x_{C,3} = 0.50$, $y_{C,3} = 0.75$, and $x_{R,3} = y_{R,3} = 0.15$.

For this test, ninth-order accuracy in space and time is used to ensure the operators are as accurate as they can be to best highlight any dimensional splitting errors in time.

Table 3 shows error norms for solid body rotation of a cosine bell, cone, and slotted cylinder using 100 $\times$ 100 cells at varying CFL values with and without WENO limiting for a fully multidimensional discretization and a dimensionally split discretization. When both are run at their largest stable CFL values, the dimensionally split simulation actually performs slightly better, presumably because there are fewer spatial reconstructions required with the larger CFL value. When run at the same CFL value, the error norms are nearly identical.

A plot of the solution with ninth-order accuracy and WENO limiting is given in Figure 3. Notice that small oscillations develop at the zero contour. None of these is greater than order $10^{-2}$. One interesting aspect of WENO limiting is that the algorithm sometimes senses that an extremum is “smooth”. This happens inside the slot of the cylinder and at the top as well, where the number of grid points is small enough that the extremum looks smooth. A positivity filter will fix any negative values, and

### Table 2
Error norms for 1D transport of a square wave, a triangle wave, and a cosine bell with 100 cells at a CFL value of 0.8 with ADER time stepping

| Order | Limiter | L1 error | L2 error | Linf error | Max overshoot |
|-------|---------|----------|----------|------------|--------------|
| 3     | WENO    | 0.0237284| 0.0671192| 0.331407   | 4.79E-05     |
| Nolim | 0.0298798| 0.0701363| 0.350141 | 6.02E-02   |
| 5     | WENO    | 0.0174282| 0.0630102| 0.315265   | 1.65E-05     |
| Nolim | 0.0226469| 0.0629407| 0.319557 | 8.54E-02   |
| 7     | WENO    | 0.0158469| 0.0598307| 0.306018   | 1.05E-05     |
| Nolim | 0.0192598| 0.0576667| 0.288819 | 9.23E-02   |
| 9     | WENO    | 0.0146989| 0.0581678| 0.29946    | 9.17E-04     |
| Nolim | 0.0194524| 0.0538259| 0.263825 | 7.69E-02   |

### Table 3
Error norms for a 2D transport test case with ADER-DT time stepping, an upwind finite-volume spatial discretization, and solid body rotation of a slotted cylinder, a cone, and a cosine hill for one rotation about the domain with ninth-order accuracy in space and time with and without a second-order alternating dimensional splitting at varying CFL values using 100 $\times$ 100 cells with and without WENO limiting

| Limiter | Norm $\text{MD,CFL=0.45}$ | $\text{DS,CFL=0.45}$ | $\text{DS,CFL=0.9}$ |
|---------|-----------------------------|------------------------|---------------------|
| WENO    | $L_1$ 0.00824706            | 0.00822985             | 0.00789167          |
|         | $L_2$ 0.0379321             | 0.0379126              | 0.0367557           |
|         | $L_{\infty}$ 0.448695      | 0.438675               | 0.422399            |
| None    | $L_1$ 0.0095333             | 0.00951727             | 0.00913494          |
|         | $L_2$ 0.0348188             | 0.0347791              | 0.0338221           |
|         | $L_{\infty}$ 0.398708      | 0.400849               | 0.399101            |

Note: ‘‘DS’’ means dimensionally split, and “MD” means multidimensional.
FIGURE 3  Initial data and final solution for transporting a square wave, a triangle wave, and a cosine bell about the domain once using ADER time stepping and a CFL value of 0.8 [Colour figure can be viewed at wileyonlinelibrary.com]

the extremum at the top of the cylinder is smooth without $2\Delta x$ oscillations. With more grid cells, the top of the cylinder will be better resolved, and the smooth extremum will go away as it does in the rest of the cylinder.

This demonstrates that, in Cartesian geometry, dimensional splitting is introducing negligible error into the overall scheme over a range of CFL values up to the maximum CFL value. This is not necessarily sufficient to guarantee the same is true for the Euler equations, but it gives confidence toward that expectation. It is more difficult to judge this in the full Euler equations without any means of providing exact error norms. It is also not sufficient to demonstrate that the temporal error convergence is higher than second order. For that demonstration, please see the next section.

3.2.2  High-order convergence

To demonstrate high-order convergence for a sufficiently smooth function, 2D transport with uniform winds, $u = v = 1$, of a sine wave profile, $q(x, t = 0) = \frac{\sin(2\pi x) \sin(2\pi y) + 1}{2}$, is performed with a CFL value of 0.8 in all simulations with ADER-DT and WENO limiting at varying orders of accuracy and grid spacings. The data are revolved once about the domain. The results of this experiment are given in Table 4, giving the $L_1$ norms and orders of convergence. The dimensionally split ADER-DT WENO-limited scheme achieves the desired order of accuracy for each order.

It was mentioned earlier that 3.2.1 was not sufficient to demonstrate that the temporal error is converging at high-order accuracy. Rather, it only demonstrates that dimensional splitting error does not dominate the overall error over a range of CFL values. Table 5 performs the same experiment performed in Table 4 except with a fixed spatial order of accuracy of ninth order with varying temporal orders of accuracy. This is done so that temporal error is more likely to dominate the overall error. Indeed, in Table 5, with a temporal order of accuracy significantly lower than the spatial order, the rate of convergence is exactly what is expected—that is, second, third, fourth, and fifth orders in time converge at exactly second, third, fourth, and fifth orders of accuracy, respectively. This demonstrates that, even though the dimensional splitting is second-order accurate, the constant on the splitting error term is low enough in Cartesian geometry that it does not show up in the overall temporal error. Further, Table 5 also demonstrates that at sixth-order accuracy in time, the spatial error begins to dominate, and the overall error converges at ninth order. Thus, with ninth-order accuracy in space, no benefit is gained from temporal orders of accuracy greater than six.

3.3  1D smooth Euler simulation to investigate flux time-averaging error

The algorithm in this paper also performs a time-averaging of the flux vector time derivatives before performing the Riemann solver. This introduces error because the integral of a nonlinear function is in general not the same as the nonlinear function of the integral. Due to the nonlinearity, the two operators do not commute without incurring error.
TABLE 4 $L_1$ errors and orders of convergence ("ord") for 2D transport of a sine wave with uniform winds and a CFL value of 0.8 at varying numbers of cells and orders of accuracy

| Cells | 40 | 50 | 60 |
|-------|----|----|----|
| Order | L1 | L1 | ord | L1 | ord |
| 3     | 1.77E-04 | 9.23E-05 | 2.92 | 5.26E-05 | 3.09 |
| 5     | 8.96E-07 | 2.99E-07 | 4.93 | 1.18E-07 | 5.09 |
| 7     | 4.79E-09 | 1.02E-09 | 6.91 | 2.81E-10 | 7.09 |
| 9     | 2.64E-11 | 3.61E-12 | 8.92 | 6.84E-13 | 9.12 |

In more precise terms:

$$\sum_{i=0}^{N-1} R \left( \frac{\partial q}{\partial t}, \frac{\partial q}{\partial t} \right)^+ \frac{1}{(i+1)!} (t - t_n)^{i+1}$$

Note: The spatial order of accuracy is fixed at ninth order to cause temporal error to dominate.

FIGURE 4 Contour plots of density, $u$ velocity, and pressure for the smooth 1D Euler test case using 1,000 cells at ninth-order accuracy with ADER-DT time stepping and a maximum CFL value of 0.8 [Colour figure can be viewed at wileyonlinelibrary.com]
of this density, $u$, and pressure up to time $t = 1$ with 1,000 cells at ninth-order accuracy with ADER-DT time stepping and WENO limiting as a grid-converged reference solution.

The linearized upwind Godunov state Riemann solver uses the following flux Jacobian diagonalization and is otherwise treated exactly the same as the 3D Euler model (see section 2.3.4):

$$\frac{\partial f}{\partial q} = \begin{bmatrix} 0 & 1 & 0 \\ -u^2 & 2u & c_s \\ -\theta u & \theta & u \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & -c_s + u & c_s + u \\ 0 & \theta & \theta \end{bmatrix}$$

$$\times \begin{bmatrix} u & 0 & 0 \\ 0 & -c_s + u & 0 \\ 0 & 0 & c_s + u \end{bmatrix} \begin{bmatrix} 1 & 0 & -1/\theta \\ \frac{u}{2c_s} & \frac{1}{2\theta} & \frac{1}{2\theta} \\ -\frac{u}{2c_s} & \frac{1}{2\theta} \end{bmatrix}$$

(40)

To gauge the impact of averaging before the Riemann solver is applied rather than after, this model is implemented in two ways: (a) integrating the time derivatives of the fluxes before calling the Riemann solver (labeled as “flux-averaged”), and (b) performing the Riemann solver on all time derivatives of the flux and then integrating the time derivatives of the tendencies (labeled as “tendency-averaged”). The tendency-averaged approach does not incur the error that the flux-averaged approach does and should theoretically be the more accurate discretization of the underlying PDEs.

To compare the flux-averaged and tendency-averaged approaches, Tables 6 and 7 give the errors and convergence rates for both approaches using 100 and 200 cells, with a maximum CFL value of 0.8, at two different times: $t = 0.2$ (a very smooth solution) and $t = 0.9$ (just before the fluid develops shocks). This gives a sense of the behavior for smooth and nonsmooth solutions. In all cases, the differences between the two approaches are never more than 1%. There are cases where the flux-averaging approach is ≤1% more accurate, and there are cases where the tendency-averaging approach is ≤1% more accurate. Also, for the smoother solution at $t = 0.2$, the numerical methods all achieve the desired convergence rates in the $L_1$ error norm for the full nonlinear equations.

3.4 | 2D nonhydrostatic Euler test cases

For the 2D nonhydrostatic Euler test cases, only the fifth-order-accurate 2D version of the model with WENO limiting is used to validate the model against expected literature results (e.g., Norman et al. (2011b) and references therein). Section 4 will investigate the effects of higher-order accuracy and WENO limiting in the context of 2D nonhydrostatic Euler flow.

3.4.1 | Hydrostatic initialization for Euler simulations

To initialize a hydrostatic atmosphere, $N$-point Gauss–Legendre quadrature is used to initialize cell means for an $N$th-order-accurate method to ensure the atmosphere is initially balanced. For cell boundaries, exact estimates of the hydrostatic background state are sampled. Hydrostatics is a function of the vertical dimension only.

**Neutral atmosphere**

To initialize hydrostatic balance for a neutral atmosphere with constant potential temperature, $\theta_0$, it is easiest to obtain a vertical profile for Exner pressure, $\pi$, rather than pressure directly. Exner pressure is a function of pressure

| TABLE 6 | Errors and convergence rates for density with the smooth 1D Euler test case at time $t = 0.2$ using ADER-DT time stepping and a maximum CFL value of 0.8 |

| Order | Method | Cells | $L_1$ Error | Conv. | $L_2$ Error | Conv. | $L_{\infty}$ Error | Conv. |
|-------|--------|-------|-------------|------|-------------|------|-------------------|------|
| 3     | Flux   | 100   | 1.82E-05    | —    | 3.41E-05    | —    | 1.87E-04          | —    |
|       |        | 200   | 2.16E-06    | 3.07 | 5.66E-06    | 2.59 | 5.04E-05          | 1.89 |
|       | Tend   | 100   | 1.82E-05    | —    | 3.41E-05    | —    | 1.87E-04          | —    |
|       |        | 200   | 2.16E-06    | 3.07 | 5.66E-06    | 2.59 | 5.04E-05          | 1.89 |
| 9     | Flux   | 100   | 3.16E-11    | —    | 6.00E-11    | —    | 2.32E-10          | —    |
|       |        | 200   | 7.57E-14    | 8.71 | 1.32E-13    | 8.83 | 5.26E-13          | 8.78 |
|       | Tend   | 100   | 3.16E-11    | —    | 6.00E-11    | —    | 2.32E-10          | —    |
|       |        | 200   | 7.49E-14    | 8.72 | 1.32E-13    | 8.83 | 5.19E-13          | 8.80 |

Note: The baseline for error norms is a grid-converged solution at ninth-order accuracy using 1,000 cells averaged to a 100- or 200-cell grid for comparison.
TABLE 7 Errors and convergence rates for density with the smooth 1D Euler test case at time $t = 0.9$ using ADER-DT time stepping and a maximum CFL value of 0.8

| Order | Method | Cells | $L_1$ Error | Conv. | $L_2$ Error | Conv. | $L_\infty$ Error | Conv. |
|-------|--------|-------|-------------|-------|-------------|-------|------------------|-------|
| 3     | Flux   | 100   | 6.84E-04    | —     | 2.18E-03    | —     | 1.22E-02         | —     |
|       |        | 200   | 2.32E-04    | 1.56  | 1.05E-03    | 1.06  | 8.95E-03         | 0.45  |
| Tend  | 100    | 6.84E-04 | —     | 2.18E-03    | —     | 1.22E-02         | —     |
|       | 200    | 2.32E-04 | 1.56  | 1.05E-03    | 1.06  | 8.95E-03         | 0.45  |
| 9     | Flux   | 100   | 2.56E-04    | —     | 6.61E-04    | —     | 3.58E-03         | —     |
|       |        | 200   | 4.70E-05    | 2.45  | 1.88E-04    | 1.82  | 1.47E-03         | 1.28  |
| Tend  | 100    | 2.53E-04 | —     | 6.53E-04    | —     | 3.37E-03         | —     |
|       | 200    | 4.77E-05 | 2.41  | 1.92E-04    | 1.77  | 1.52E-03         | 1.15  |

Note: The baseline for error norms is a grid-converged solution at ninth-order accuracy using 1,000 cells averaged to a 100- or 200-cell grid for comparison.

only, given by:

$$\pi = \left(\frac{p}{p_0}\right)^{\gamma/c_p}$$  \hspace{1cm} (41)

And its hydrostatic balance equation is given in terms of only potential temperature:

$$\frac{d\pi}{dz} = -\frac{g}{c_p\theta}$$  \hspace{1cm} (42)

The first two test cases assume a constant potential temperature basic state. Therefore, the hydrostatically balanced Exner pressure profile is trivial:

$$\pi(z) = \pi_{sfc} - \frac{gz}{c_p\theta_0}$$  \hspace{1cm} (43)

where it is assumed that $\pi_{sfc} = 1$ meaning $p = p_0$ at the surface.

**Constant Brunt–Vaisala frequency**

To initialize a stable atmosphere, a constant Brunt–Vaisala frequency, $N_0$, is specified. The Brunt–Vaisala frequency is given in terms of fractional vertical gradient of potential temperature:

$$\sqrt{\frac{g}{\partial \theta / \partial z}} = N_0$$  \hspace{1cm} (44)

Therefore,

$$\theta(z) = \theta_{sfc} \exp\left(-\frac{N_0^2 z}{g}\right)$$  \hspace{1cm} (45)

Plugging this into (42), one eventually obtains the following vertical profile:

$$\pi(z) = \pi_{sfc} - \frac{g^2}{c_p N_0^2} \left(\frac{\theta(z) - \theta_{sfc}}{\theta(z) \theta_{sfc}}\right)$$  \hspace{1cm} (46)

The constants are set as follows: $\pi_{sfc} = 1$, $\theta_{sfc} = 300$ K, and $N_0 = 10^{-2}$ s$^{-1}$.

Pressure can be obtained from (43) or (46) by the Exner pressure equation, (41). Then, density is determined from the pressure by the equation of state.

### 3.4.2 | Rising thermal

The rising thermal test case Wicker and Skamarock (1998) perturbs the potential temperature of a neutrally stratified atmosphere as follows:

$$\theta(x, y, z) = \theta_H(z) + \hat{\theta} \max(0, 1 - d)$$  \hspace{1cm} (47)

$$d = \sqrt{\left(x - x_0\right)^2 + \left(y - y_0\right)^2 + \left(z - z_0\right)^2}$$  \hspace{1cm} (48)

where $\hat{\theta} = 2$ K is the amplitude of the perturbation, $(x_0, y_0, z_0) = (10, 10, 2$ km) is the center of the thermal, and $x_R = y_R = z_R = 2$ km is the radius of the thermal. This test case is simulated on a domain of $[0, 20] \times [0, 20] \times [0, 10]$ km in the $x$, $y$, and $z$ directions for 1,000 s. Winds are initially at rest. For a 2D domain, the $y$ coordinate is always set to $y_0$.

Plots of the solution are given in Figure 5 using a fifth-order-accurate WENO ADER-DT scheme with 100 m grid spacing and a CFL value of 0.8 giving a time step of 0.23 s. To compare against literature results, please see Figure 5 of Ahmad and Lindeman (2007) and Figure 2 of Norman et al. (2011b). Quantitative comparison is difficult since it is a nonlinear flow regime wherein small initial condition differences and numerical differences eventually lead to large differences in the flow. However, the structure matches other studies well. Figure 6 also plots...
traces of the domain maximum potential temperature and vertical velocity magnitude over the course of the simulation, and it also compares well against fig. 6 of Ahmad and Lindeman (2007) and fig. 3 of Norman et al. (2011b).

3.4.3 | Straka density current

The Straka density current test case Straka et al. (1993) perturbs the potential temperature of a neutrally stratified atmosphere as follows:

\[
\theta(x, y, z) = \theta_H(z) + \hat{\theta} \begin{cases} 
(\cos(\pi d) + 1) / 2 & \text{if } d \leq 1 \\
0 & \text{otherwise}
\end{cases} 
\]

(49)

\[
d = \sqrt{\left(\frac{x - x_0}{x_R}\right)^2 + \left(\frac{y - y_0}{y_R}\right)^2 + \left(\frac{z - z_0}{z_R}\right)^2} 
\]

(50)

where \(\hat{\theta} = -15\) K is the amplitude of the perturbation, \((x_0, y_0, z_0) = (26.5, 26.5, 3)\) km is the center of the cold bubble, and \(x_R = y_R = 4\) km and \(z_R = 2\) km are the radii of the cold bubble in the horizontal and vertical directions, respectively. This test case is simulated on a domain of \([0, 53] \times [0, 53] \times [0, 6.4]\) km in the \(x, y,\) and \(z\) directions for 900 s. For a 2D domain, the \(y\) coordinate is always set to \(y_0\).

This test case is traditionally run with a dynamic viscosity, which is defined by adding the following to the source term vector in Equation (??):

\[
\mathbf{s} = \mathbf{s} + \rho K \begin{bmatrix} 0 \\
\partial_x u + \partial_y u + \partial_z u \\
\partial_x v + \partial_y v + \partial_z v \\
\partial_x w + \partial_y w + \partial_z w \\
\partial_x \theta + \partial_y \theta + \partial_z \theta
\end{bmatrix}
\]

(51)

where \(K = 75\) m\(^2\) \cdot s\(^{-1}\). However, it can be run without viscosity as well. Winds are initially at rest. The test run here
includes the prescribed dynamics viscosity for comparison against literature results. The potential temperature is plotted in Figure 7 using a 100 m grid spacing with a CFL value of 0.8 giving a time step of 0.23 s. The minimum and maximum potential temperature perturbations at 900 s were -8.65 and 0.015 K, respectively. For comparison against other literature simulations of this test case (aside from the original Straka paper), please see fig. 8 of Ahmad and Lindeman (2007), figs. 7 and 8 of Giraldo and Restelli (2008), and fig. 5 of Norman et al. (2011b). These match literature expectations in both the contour plot and the cross section plot. The minimum and maximum potential temperature perturbation at 900 s are also similar to those reported in other studies.

3.4.4 | Inertia–gravity waves

The inertia–gravity wave test case Skamarock and Klemp (1994) perturbs the potential temperature of a stably stratified atmosphere (with $N_0 = 10^{-2}$) as follows:

$$\theta (x, y, z) = \theta_H (z) + \hat{\theta} \frac{\sin (\pi z / z_T)}{\sqrt{1 + (x - x_0)^2 / h_R^2 + (y - y_0)^2 / h_R^2}}$$

(52)

where $\hat{\theta} = 10^{-2}$ K is the amplitude of the perturbation, $(x_0, y_0) = (100, 100$ km is the center of the perturbation, and $h_R = 5$ km is the horizontal radius. This test case is simulated on a domain of $[0, 300] \times [0, 300] \times [0, 10]$ km in the $x$, $y$, and $z$ directions for 3,000 s. For a 2D domain, the $y$ coordinate is always set to $y_0$. The horizontal winds are initialized to $u = v = 20$ m s$^{-1}$ (where $v$ is set to zero in 2D simulations). Typically, this test case is solved with the horizontal to vertical grid spacing ratio set to 10:1. Plots of the potential temperature are given in Figure 8 after 3,000 s using 100 m and 1 km grid spacing in the vertical and horizontal directions, respectively. With a CFL value of 0.8, the time step was 0.23 s. To compare against previous literature solutions for this test case, please see figs. 1–4 of Ahmad and Lindeman (2007); fig. 2 of Giraldo and Restelli (2008); and figs. 7 and 8 of Norman et al. (2011b).

To keep these plots from being noisy, this particular simulation was performed in double precision, since the thermal perturbation has a relative magnitude of $10^{-5}$ compared with the background potential temperature.

3.4.5 | Colliding thermals

The colliding thermals test case to the author’s knowledge has not been presented in literature before. It exists to create stronger discontinuities, stronger winds, and more significant turbulence than the other simulations. The colliding thermals test case perturbs the potential temperature of a neutrally stratified atmosphere as follows:

$$\theta (x, y, z) = \theta_H (z) + \hat{\theta} (\max (1 - d_1, 0) + \max (1 - d_2, 0))$$

(53)
Potential temperature contour plot
(a) (b)

Potential temperature at $z = 1,200$ m

FIGURE 7  Potential temperature after 900 s for the 2D viscous Straka density current test case with a fifth-order-accurate WENO ADER-DT scheme using $530 \times 64$ cells with dynamic viscosity. This is for visual comparison against literature-standard test cases and for validation of the vertical boundary condition implementation [Colour figure can be viewed at wileyonlinelibrary.com]

Solution plots for the 2D inertia–gravity wave test case after 3,000 s with a fifth-order-accurate WENO ADER-DT scheme using $300 \times 100$ cells. This is for visual comparison against literature-standard test cases [Colour figure can be viewed at wileyonlinelibrary.com]

\[ d_1 = \sqrt{\left(\frac{x-x_1}{r}\right)^2 + \left(\frac{y-y_1}{r}\right)^2 + \left(\frac{z-z_1}{r}\right)^2} \]  \hspace{1cm} (54)

\[ d_2 = \sqrt{\left(\frac{x-x_2}{r}\right)^2 + \left(\frac{y-y_2}{r}\right)^2 + \left(\frac{z-z_2}{r}\right)^2} \]  \hspace{1cm} (55)

where $\hat{\theta} = 20$ K is the amplitude of the perturbation, $(x_1, y_1, z_1) = (10, 10, 2$ km) is the center of the warm bubble, $(x_2, y_2, z_2) = (10, 10, 8$ km) is the center of the cold bubble, and $r = 2$ km is the radius of the bubbles. This test case is simulated on a domain of $[0, 20] \times [0, 20] \times [0, 10]$ km in the $x$, $y$, and $z$ directions for 700 s. Winds are initially at rest. For a 2D domain, the $y$ coordinate is always set to $y_0$.

Plots of potential temperature after 0, 200, 400, and 700 s are given in Figure 9 using 100 m grid spacing. With a CFL value of 0.8, the time step was 0.23 s.

4 | DISCUSSION OF LIMITING AND ORDER OF ACCURACY

In this section, only odd orders of accuracy are used (i.e., third, fifth, seventh, and ninth order). This is because a cell-centered approach naturally contains an odd number of cells in a stencil to use for recovery. It would, similarly, be more natural for a cell-edge-centered
Potential Temperature, initial

(a) (b)

Potential Temperature, 200s

(c) (d)

Potential Temperature, 400s

Potential Temperature, 700s

FIGURE 9 Solution plots for the 2D colliding thermals test case with a fifth-order-accurate WENO ADER-DT scheme using 200 x 100 cells. This is to demonstrate stability with large discontinuities and to generate turbulence for later results [Colour figure can be viewed at wileyonlinelibrary.com]

Also note that when this paper mentions “order of accuracy,” there is no implication that the results for multidimensional Euler dynamics are actually converging to high-order accuracy. That would be impossible for any method because the underlying data are not smooth enough for well-posed high-order derivatives. Rather, the implication is that the recovery operators are high-order accurate if the data are smooth enough (see section 3.2). Further, the hypothesis being tested is that even though high-order convergence is impossible in realistic atmospheric flow, the constant of error can still be reduced via higher-order-accurate recovery operators that are appropriately limited with high-resolution WENO limiting (see section 3.1).

4.1 WENO versus no limiter

Figure 10 shows a plot of potential temperature perturbation for the inertia–gravity waves test case at fifth-order accuracy with and without WENO limiting at z = 5 km. The overlapping lines demonstrate that for smooth flows there is no effect when applying the WENO limiter, a designed property of WENO limiting. This is expected because it was demonstrated in the 2D transport context in section 3.2 in the order of convergence study. However, it is helpful to confirm that this is still true in the context of the multidimensional Euler equations.

Figures 11 and 12 show contour plots and line plots at x = 10 km of potential temperature perturbations with and without WENO limiting after 270 s of simulation at ninth-order accuracy to visually show the effects of WENO
limiting for flows that become discontinuous at the front of the colliding thermals. WENO limiting slightly reduces the extrema and significantly reduces the visible $2\Delta x$ oscillations. Yet, the larger scale flow remains undamped in the cross section plot.

4.2 Order of accuracy

Figure 13 shows the vertical wind for the rising thermal test case over varying orders of accuracy from third to ninth order, all using WENO limiting. Notice the increasing detail in the thermal center in particular. Of course, there is a limit to how much order of accuracy can help, particularly for smoother flows, since they are by definition already fairly well resolved. The differences are visible up to seventh-order accuracy, but for this test case, the differences between seventh order and ninth order are minimal.

Figure 14 shows plots of potential temperature for the collision test case after 700 s of simulation. Again, notice the increasing amount of visible detail at smaller scales as the order of accuracy increases. Since this is a chaotic fluid that is now in a turbulent regime, the results are expected to diverge because of different numerical properties. But the general scale of the turbulent eddies present in the potential temperature plot becomes finer with higher-order reconstruction. Still, it would be helpful to take a more quantitative approach to this.

Figure 15 plots the kinetic energy spectral power density as a function of signal frequency for the 2D collision test case with $400 \times 200$ cells and a CFL value of 0.8 with ADER time stepping. Spectral power density (SPD) is a measure of the squared amplitude of a signal at different frequencies. SPD, $P$, of a signal is defined as: $P(\omega) = |c(\omega)|^2$, where $\omega$ is the frequency, and $c(\omega)$ is the the Fourier coefficient for that frequency. Practically, it is computed by squaring the coefficients of an FFT (in python, this is spd=numpy.abs(numpy.fft.rfft(data[:]))**2). To create the plot, the SPDs using only $x$-direction FFTs are averaged over 100 time steps throughout the simulation and over the 200 vertical levels to create something smooth enough for easy visual comparison between different orders of accuracy and different limiting mechanisms. Further, a moving average of SPD over four frequencies is used to create the final plot. Kinetic energy is calculated as $KE = (u^2 + v^2 + w^2) / 2$. Note that a similar study is performed in Norman and Larkin (2020), but that plot was too noisy to draw robust conclusions. Here, the results are averaged over many time steps and vertical levels to produce something smooth enough for clear comparison.
When looking at Figure 15, the wavelengths get smaller as the frequencies get larger, meaning the smallest wavelength admitted in the flow ($2\Delta x$) is at the farthest right in the plot. One would expect that if the higher-order-accurate simulations really are resolving finer-scale structures, then there will be more power in the kinetic energy at the smaller wavelengths. Note that while the $k^{-3}$ slope is plotted, it is not clear that this type of flow must follow that particular spectral slope. Further, this simulation has a relatively small inertial range for the energy cascade. In 2D, to create an inertial range, energy must be injected at the smallest scale because it will cascade upscale, while, in 3D, energy must be injected at the largest scales. With the collision of thermals, energy is created at small scales, but it is not a sustained injection of kinetic energy.

The most obvious result from this experiment is that the power at small wavelengths (large frequencies) increases as the order of accuracy is increased. This seems to confirm that the higher-order-accurate reconstructions are indeed resolving finer-scale structures in the flow. There is some overlap at seventh and ninth orders of accuracy between $4\Delta x$ and $8\Delta x$ grid spacings, but between $4\Delta x$ and $2\Delta x$ grid spacings, the ninth-order scheme clearly has more power. Also, for each case, the WENO limiting decreases power in the spectra after about $16\Delta x$ grid spacing. The loss in power associated with WENO limiting is quite small at $16\Delta x$ and gradually increases as the grid spacing increases. From this plot, it seems there is relatively little value in moving from seventh-order accuracy to ninth-order accuracy in the unlimited regime. However, when WENO limiting is turned on, there is much more to be gained from going to ninth-order accuracy. Clearly, WENO limiting is not necessary for stability because there is no appreciable energy accumulation at the $2\Delta x$ scale.

Therefore, it would be worth looking into limiting only potential temperature and density with WENO and leaving the wind velocities unlimited.

In summary, this experiment clearly demonstrates that at higher orders of accuracy, there is greater power in
FIGURE 14 Plots of potential temperature for the colliding thermals test case with $200 \times 100$ cells at different orders of accuracy. Note in particular how smaller scale eddies are resolved by the higher-order schemes with the same grid spacing [Colour figure can be viewed at wileyonlinelibrary.com]

FIGURE 15 Plot of the simulation time-average of the kinetic energy spectral power density for the 2D collision test case run for 1,000 s at a CFL value of 0.8 with $400 \times 200$ cells. A running mean of four frequencies was used to smooth the spectra for easier visual comparison between different limiting mechanisms and orders of accuracy [Colour figure can be viewed at wileyonlinelibrary.com]

the kinetic energy field at smaller wavelengths. Further, WENO limiting induces a gradual reduction in power starting at $16\Delta x$ wavelengths.

4.3 Performance on graphics processing units (GPUs)

While a detailed description of the code’s C++ performance portable implementation and analysis of this code’s performance on GPU accelerators is given in Norman and Larkin (2020), it is worth mentioning a few interesting computational findings. GPUs typically have memory bandwidth rates (the amount of data that can be transferred per second) that are much higher than those of CPUs. For this reason alone, most applications enjoy a speed-up on GPUs. However, GPUs are also capable of significantly more computations per second than CPUs. Low-order-accurate methods do not use GPUs efficiently because they do not perform many floating
point operations per byte of data fetched from memory. High-order-accurate methods and WENO limiting improve on this because they increase computations without significantly increasing data movement. This is true even for small GPU workloads (see section 5.1.4 of Norman and Larkin (2020)).

Table 8 shows the number of cells updated per second for varying orders of accuracy in space and time using $80 \times 80 \times 40$ cells on one V100 GPU using single-precision floating-point arithmetic. At third order of accuracy in time, without WENO limiting, all of the spatial orders of accuracy are within 5% of one another in terms of runtime. At third-order accuracy in time, the overhead of WENO limiting at third, fifth, seventh, and ninth-order accuracy is 6, 28, 37, and 77%, respectively. There is a notable difference in performance moving from third-order accuracy to fourth-order accuracy in time. This is actually due more to how the kernel is compiled by the Nvidia nvcc compiler (register usage and GPU occupancy) than it is due to the number of computations. At fourth-order accuracy in time, all experiments are within 3% of one another, regardless of order of spatial accuracy or limiting. It might seem unexpected for performance to modestly improve with increasing spatial order of accuracy at fourth-order accuracy in time. While this is common with GPU kernels, the details that lead to these unexpected anomalies will be investigated in a future study. Note that Table 8 is related to table 4 in Norman and Larkin (2020), but in this case, an improvement to the ADER-DT algorithm has been made, the result of which is detailed in Algorithm 4.

The ADER-DT scheme has a maximum stable CFL value of one, which is roughly $2 \times$ the size of the maximum effective SSP CFL value of most (but not all) existing strong stability preserving Runge–Kutta time integrators Gottlieb et al. (2011). Please note that this is not an attempt at an overall comparison between numerical methods, as that would involve many more considerations. Also, the cells updated per second metric neglects accuracy, limiting of oscillations, and time-step size.

5 | CONCLUSIONS AND FUTURE WORK

The hypothesis being explored in this study is that high-order accuracy, when properly limited, can provide increasing resolution even for complex and discontinuous flows. This is furthering the study of Norman and Nair (2018) that explores this hypothesis in the context of transport. A brief survey of common discretizations was given in section 1. The mathematical discretization was described in detail in section 2. In section 3, a number of aspects of the discretization were validated using standard test cases. The effects of increasing order of accuracy and WENO limiting were explored in section 4.

It was found in 1D transport that the new algorithm lowers error monotonically with increasing order of accuracy up to ninth order even with highly discontinuous data. Notably WENO gave lower errors than unlimited reconstruction in every experiment in the $L_1$ norm, though this is not true in the $L_\infty$ norm. In 2D transport, solid body rotation of various data demonstrated that dimensional splitting errors are negligible in Cartesian geometry even at a CFL value of 0.8. Also in 2D transport, high-order convergence up to ninth order is confirmed by transporting a sine wave. It was also found in 2D transport that, when the temporal order of accuracy is much lower than the spatial order of accuracy, time error dominates the overall error, and the solution converges at the expected temporal order of accuracy. A 1D Euler model was used in smooth and nonsmooth contexts to demonstrate that averaging the flux time derivatives before the Riemann solver has negligible effects on accuracy.

Next, 2D test cases including a rising thermal, a density current, and inertia–gravity waves provide validation of the new algorithm with ADER time stepping and WENO limiting using previous studies of discretizations of this equation set. It was found that structures in the flow were better resolved with increasing order of accuracy even with WENO limiting enabled. Spectral power density plotted as a function of frequency for kinetic energy confirmed that higher-order accuracy had more power at smaller wavelengths than lower-order accuracy. Finally, WENO limiting begins to lower the kinetic energy’s spectral power at a grid spacing of about $16\Delta x$ with effects that gradually increase up to the $2\Delta x$ scale.
Future directions planned for this research include investigating a multistage implementation of the ADER-DT algorithm in hopes of a simple dimensional splitting in nonorthogonal coordinates similar to Katta et al. (2015). The computational expense of an unsplit approach will also be investigated. Next, the inevitable expansion into global spatial scales will require implementing either HEVI or vertically sound-proof techniques, much of which has already been investigated in various contexts Weller et al. (2013); Gardner et al. (2018). This will also require further investigation into how to couple a diagonally implicit Runge–Kutta (DIRK) scheme Alexander (1977) with an ADER-DT explicit discretization, if the ADER approach proves possible and worthwhile in that context.

ACKNOWLEDGMENTS

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

ORCID

Matthew R. Norman https://orcid.org/0000-0003-4764-3348

REFERENCES

Ahmad, N. and Lindeman, J. (2007) Euler solutions using flux-based wave decomposition. *International Journal for Numerical Methods in Fluids*, 54, 47–72.

Alexander, R. (1977) Diagonally implicit Runge–Kutta methods for stiff ode’s. *SIAM Journal on Numerical Analysis*, 14, 1006–1021.

Arakawa, A. and Lamb, V.R. (1981) A potential enstrophy and energy conserving scheme for the shallow water equations. *Monthly Weather Review*, 109, 18–36.

Balsara, D.S., Rumpf, T., Dumbser, M. and Munz, C.-D. (2009) Efficient, high accuracy ADER-WENO schemes for hydrodynamics and divergence-free magnetohydrodynamics. *Journal of Computational Physics*, 228, 2480–2516.

Barros, S.R.M., Dent, D., Isaksen, L., Robinson, G., Mozdzynski, G. and Wollenweber, E. (1995) The ifs model: a parallel production weather code. *Parallel Computing*, 21, 1621–1638.

Capdeville, G. (2008a) A central WENO scheme for solving hyperbolic conservation laws on non-uniform meshes. *Journal of Computational Physics*, 227, 2977–3014.

Capdeville, G. (2008b) A Hermite upwind WENO scheme for solving hyperbolic conservation laws. *Journal of Computational Physics*, 227, 2430–2454.

Chen, Y., Weller, H., Pring, S. and Shaw, J. (2017) Comparison of dimensionally split and multi-dimensional atmospheric transport schemes for long time steps. *Quarterly Journal of the Royal Meteorological Society*, 143, 2764–2779.

Colella, P. and Woodward, P.R. (1984) The piecewise parabolic method (ppm) for gas-dynamical simulations. *Journal of Computational Physics*, 54, 174–201.

Cotter, C.J. and Shipton, J. (2012) Mixed finite elements for numerical weather prediction. *Journal of Computational Physics*, 231, 7076–7091.

Dennis, J.M., Edwards, J., Evans, K.J., Guba, O., Lauritzen, P.H., Mirin, A.A., St-Cyr, A., Taylor, M.A. and Worley, P.H. (2012) Cam-se: a scalable spectral element dynamical core for the community atmosphere model. *The International Journal of High Performance Computing Applications*, 26, 74–89.

Dumbser, M., Balsara, D.S., Toro, E.F. and Munz, C.-D. (2008) A unified framework for the construction of one-step finite volume and discontinuous Galerkin schemes on unstructured meshes. *Journal of Computational Physics*, 227, 8209–8253.

Durran, D.R. (2010) *Numerical Methods for Fluid Dynamics: With Applications to Geophysics* Vol. 32. New York: Springer Science and Business Media.

Evans, K.J., Archibald, R.K., Gardner, D.J., Norman, M.R., Taylor, M.A., Woodward, C.S. and Worley, P.H. (2019) Performance analysis of fully explicit and fully implicit solvers within a spectral element shallow-water atmosphere model. *The International Journal of High Performance Computing Applications*, 33, 268–284.

Fournier, A., Taylor, M.A. and Tribbia, J.J. (2004) The spectral element atmosphere model (SEAM): high-resolution parallel computation and localized resolution of regional dynamics. *Monthly Weather Review*, 132, 726–748.

Gardner, D.J., Guerra, J.E., Hamon, F.P., Reynolds, D.R., Ullrich, P.A. and Woodward, C.S. (2018) Implicit-explicit (imex) Runge-Kutta methods for non-hydrostatic atmospheric models. *Geoscientific Model Development*, 11, 1497–1515.

Giraldo, F.X., Hesthaven, J.S. and Warburton, T. (2002) Nodal high-order discontinuous Galerkin methods for the spherical shallow water equations. *Journal of Computational Physics*, 181, 499–525.

Giraldo, F.X. and Restelli, M. (2008) A study of spectral element and discontinuous Galerkin methods for the Navier–Stokes equations in nonhydrostatic mesoscale atmospheric modeling: equation sets and test cases. *Journal of Computational Physics*, 227, 3849–3877.

Gottlieb, S. (2005) On high order strong stability preserving Runge–Kutta and multi step time discretizations. *Journal of Scientific Computing*, 25, 105–128.

Gottlieb, S., Ketcheson, D.I. and Shu, C.-W. (2011) *Strong Stability Preserving Runge-Kutta and Multistep Time Discretizations*. Hackensack, NJ: World Scientific.

Guba, O., Taylor, M. and St-Cyr, A. (2014) Optimization-based limiters for the spectral element method. *Journal of Computational Physics*, 267, 176–195.

Guo, W., Nair, R.D. and Zhong, X. (2016) An efficient WENO limiter for discontinuous Galerkin transport scheme on the cubed sphere. *International Journal for Numerical Methods in Fluids*, 81, 3–21.

Henrick, A.K., Aslam, T.D. and Powers, J.M. (2005) Mapped weighted essentially non-oscillatory schemes: achieving optimal order near critical points. *Journal of Computational Physics*, 207, 542–567.

Li, S. and Xiao, F. (2009) High order multi-moment constrained finite volume method. Part I: basic formulation. *Journal of Computational Physics*, 228, 3669–3707.

Kutta, K.K., Nair, R.D. and Kumar, V. (2015) High-order finite-volume transport on the cubed sphere: comparison
between 1D and 2D reconstruction schemes. *Monthly Weather Review*, 143, 2937–2954. https://doi.org/10.1175/MWR-D-13-00176.1.

Kurganov, A. and Lin, C.-T. (2007) On the reduction of numerical dissipation in central-upwind schemes. *Computer Physics Communications*, 2, 141–163.

Kuzmin, D., Löhner, R. and Turek, S. (2012) *Flux-Corrected Transport: Principles, Algorithms, and Applications*. New York: Springer.

Lauritzen, P.H., Skamarock, W.C., Prather, M.J. and Taylor, M.A. (2012) A standard test case suite for two-dimensional linear transport on the sphere. *Geoscientific Model Development*, 5.

Leonard, B., Lock, A. and MacVean, M. (1996) Conservative explicit unrestricted-time-step multidimensional constancy-preserving advection schemes. *Monthly Weather Review*, 124, 2588–2606.

Liu, X.-D., Osher, S. and Chan, T. (1994) Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115, 200–221.

Löhner, R., Morgan, K., Peraire, J. and Vahdati, M. (1987) Finite element flux-corrected transport (FEM–FCT) for the Euler and Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, 7, 1093–1109.

MacNamara, S. and Strang, G. (2016). Operator splitting, Splitting Methods in Communication, Imaging, Science, and Engineering, pp. 95–114: Cham, Switzerland: Springer.

Marras, S., Kelly, J.F., Moragues, M., Müller, A., Kopera, M.A., Vázquez, M., Giraldo, F.X., Houzeaux, G. and Joba, O. (2016) A review of element-based Galerkin methods for numerical weather prediction: finite elements, spectral elements, and discontinuous Galerkin. *Archives of Computational Methods in Engineering*, 23, 673–722.

Michoski, C., Dawson, C., Kubatko, E.J., Wirasaet, D., Brus, S. and Westerink, J.J. (2016) A comparison of artificial viscosity, limiters, and filters, for high order discontinuous Galerkin solutions in nonlinear settings. *Journal of Scientific Computing*, 66, 406–434.

Nair, R.D., Thomas, S.J. and Loft, R.D. (2005) A discontinuous Galerkin transport scheme on the cubed sphere. *Monthly Weather Review*, 133, 814–828.

Natale, A., Shipton, J. and Cotter, C.J. (2016) Compatible finite element spaces for geophysical fluid dynamics. *Dynamics and Statistics of the Climate System*, 1.

Norman, M. and Larkin, J. (2020) A holistic algorithmic approach to improving accuracy, robustness, and computational efficiency for atmospheric dynamics. *SIAM Journal on Scientific Computing*, 42, B1302–B1327.

Norman, M.R. (2013) Algorithmic improvements for schemes using the ADER time discretization. *Journal of Computational Physics*, 243, 176–178.

Norman, M.R. (2014) A WENO-limited, ADER-DT, finite-volume scheme for efficient, robust, and communication-avoiding multi-dimensional transport. *Journal of Computational Physics*, 274, 1–18.

Norman, M.R. (2015a) Arbitrarily high-order-accurate, hermite WENO limited, boundary-averaged multi-moment constrained finite-volume (BA-MCV) schemes for 1-D transport. *Procedia Computer Science*, 51, 2688–2697.

Norman, M.R. (2015b) Hermite WENO limiting for multi-moment finite-volume methods using the ADER-DT time discretization for 1-D systems of conservation laws. *Journal of Computational Physics*, 282, 381–396.

Norman, M.R. and Finkel, H. (2012) Multi-moment ADER-Taylor methods for systems of conservation laws with source terms in one dimension. *Journal of Computational Physics*, 231, 6622–6642.

Norman, M.R. and Nair, R.D. (2008) Inherently conservative non-polynomial-based remapping schemes: application to semi-Lagrangian transport. *Monthly Weather Review*, 136, 5044–5061.

Norman, M.R. and Nair, R.D. (2018) A positive-definite, WENO-limited, high-order finite volume solver for 2-D transport on the cubed sphere using an ADER time discretization. *Journal of Advances in Modeling Earth Systems*, 10, 1587–1612.

Norman, M.R., Nair, R.D. and Semazzi, F.H. (2011a) A low communication and large time step explicit finite-volume solver for non-hydrostatic atmospheric dynamics. *Journal of Computational Physics*, 230, 1567–1584.

Norman, M.R., Nair, R.D. and Semazzi, F.H. (2011b) A low communication and large time step explicit finite-volume solver for non-hydrostatic atmospheric dynamics. *Journal of Computational Physics*, 230, 1567–1584. http://linkinghub.elsevier.com/retrieve/pii/S00219991110006376.

Qiu, J. and Shu, C.-W. (2004) Hermite WENO schemes and their application as limiters for Runge–Kutta discontinuous Galerkin method: one-dimensional case. *Journal of Computational Physics*, 193, 115–135.

Rider, W.J. (1994) A review of approximate Riemann solvers with Godunov’s method in Lagrangian coordinates. *Computers & fluids*, 23, 397–413.

Rider, W.J. and Lowrie, R.B. (2002) The use of classical Lax–Friedrichs Riemann solvers with discontinuous Galerkin methods. *International Journal for Numerical Methods in Fluids*, 40, 479–486.

Roe, P.L. (1984) Generalized Formulation of TVD Lax–Wendroff Schemes.

Shu, C.-W. and Osher, S. (1988) Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77, 439–471.

Skamarock, W.C. and Klemp, J.B. (1994) Efficiency and accuracy of the Klemp–Wilhelmson time-splitting technique. *Monthly Weather Review*, 122, 2623–2630.

Skamarock, W.C. and Klemp, J.B. (2008) A time-split nonhydrostatic atmospheric model for weather research and forecasting applications. *Journal of Computational Physics*, 227, 3465–3485.

Skamarock, W.C., Klemp, J.B., Duda, M.G., Fowler, L.D., Park, S.-H. and Ringler, T.D. (2012) A multiscale nonhydrostatic atmospheric model using centroidal Voronoi tessellations and C-grid staggering. *Monthly Weather Review*, 140, 3090–3105.

Staniforth, A. and Côté, J. (1991) Semi-Lagrangian integration schemes for atmospheric models—a review. *Monthly Weather Review*, 119, 2206–2223.

Straka, J.M., Wilhelmson, R.B., Wicker, L.J., Anderson, J.R. and Droegemeier, K.K. (1993) Numerical solutions of a non-linear density current: a benchmark solution and comparisons. *International Journal for Numerical Methods in Fluids*, 17, 1–22.

Titarev, V.A. and Toro, E.F. (2002) ADER: arbitrary high order Godunov approach. *Journal of Scientific Computing*, 17, 609–618.
APPENDIX A. EULER EQUATION CHARACTERISTICS

Here, the flux Jacobians and their diagonalizations in each coordinate direction are given, where \( c_s = \sqrt{\gamma p/\rho} \) is the speed of sound. In each direction, the flux Jacobian is diagonalized into \( R \Lambda L \), where \( L = R^{-1} \).

\[
\begin{align*}
\frac{\partial \mathbf{f}}{\partial \mathbf{q}} &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\
-u^2 & 2u & 0 & 0 & \frac{\gamma}{\gamma-1} \\
-uv & v & u & 0 & 0 \\
-uv & w & u & 0 & 0 \\
-\theta u & \theta & 0 & 0 & u \end{bmatrix}, \\
\Lambda_x &= \begin{bmatrix} -c_s + u & 0 & 0 & 0 & 0 \\
0 & c_s + u & 0 & 0 & 0 \\
0 & 0 & u & 0 & 0 \\
0 & 0 & 0 & u & 0 \\
0 & 0 & 0 & 0 & u \end{bmatrix}, \\
R_x &= \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\
-c_s + u & c_s + u & u & 0 & 0 \\
v & v & 0 & 1 & 0 \\
w & w & 0 & 0 & 1 \\
\theta & \theta & 0 & 0 & 0 \end{bmatrix}, \\
L_x &= \begin{bmatrix} \frac{u}{2c_s} - \frac{1}{2} & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\
1 & 0 & 0 & 0 & -\frac{1}{\theta} \\
0 & 0 & 1 & 0 & -\frac{v}{\theta} \\
0 & 0 & 0 & 1 & -\frac{w}{\theta} \end{bmatrix}, \\
\frac{\partial \mathbf{g}}{\partial \mathbf{q}} &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\
-uv & v & u & 0 & 0 \\
-u^2 & 0 & 2v & 0 & \frac{\gamma}{\gamma-1} \\
-ww & w & v & 0 & 0 \\
-\theta v & 0 & \theta & 0 & v \end{bmatrix}, \\
\Lambda_y &= \begin{bmatrix} -c_s + v & 0 & 0 & 0 & 0 \\
0 & c_s + v & 0 & 0 & 0 \\
0 & 0 & v & 0 & 0 \\
0 & 0 & 0 & v & 0 \\
0 & 0 & 0 & 0 & v \end{bmatrix}
\end{align*}
\]
\[ R_y = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ u & u & 0 & 1 & 0 \\ -c_s + v & c_s + v & v & 0 & 0 \\ w & w & 0 & 0 & 1 \\ \theta & \theta & 0 & 0 & 0 \end{bmatrix} ; \]

\[ L_y = \begin{bmatrix} \frac{v}{2c_s} & 0 & -\frac{1}{2c_s} & 0 & \frac{1}{2\theta} \\ -\frac{v}{2c_s} & 0 & \frac{1}{2c_s} & 0 & \frac{1}{2\theta} \\ 1 & 0 & 0 & 0 & -\frac{1}{\theta} \\ 0 & 1 & 0 & 0 & -\frac{w}{\theta} \\ 0 & 0 & 0 & 1 & -\frac{w}{\theta} \end{bmatrix} \]

\[ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ -uw & w & 0 & u & 0 \\ -vw & 0 & w & v & 0 \\ -w^2 & 0 & 0 & 2w & \frac{c_s^2}{\theta} \\ -\theta w & 0 & 0 & \theta & w \end{bmatrix} ; \]

\[ \frac{\partial \mathbf{h}}{\partial \mathbf{q}} = \begin{bmatrix} -c_s + w & 0 & 0 & 0 & 0 \\ 0 & c_s + w & 0 & 0 & 0 \\ 0 & 0 & w & 0 & 0 \\ 0 & 0 & 0 & w & 0 \\ 0 & 0 & 0 & 0 & w \end{bmatrix} \]

\[ \Lambda_z = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ u & u & 0 & 1 & 0 \\ v & v & 0 & 0 & 1 \\ -c_s + w & c_s + w & w & 0 & 0 \\ \theta & \theta & 0 & 0 & 0 \end{bmatrix} ; \]

\[ L_z = \begin{bmatrix} \frac{w}{2c_s} & 0 & 0 & -\frac{1}{2c_s} & \frac{1}{2\theta} \\ -\frac{w}{2c_s} & 0 & 0 & \frac{1}{2c_s} & \frac{1}{2\theta} \\ 1 & 0 & 0 & 0 & -\frac{1}{\theta} \\ 0 & 1 & 0 & 0 & -\frac{u}{\theta} \\ 0 & 0 & 1 & 0 & -\frac{v}{\theta} \end{bmatrix} \]

(A4)

(A6)

\[ \frac{\partial \mathbf{h}}{\partial \mathbf{q}} = \begin{bmatrix} \frac{v}{2c_s} & 0 & -\frac{1}{2c_s} & 0 & \frac{1}{2\theta} \\ -\frac{v}{2c_s} & 0 & \frac{1}{2c_s} & 0 & \frac{1}{2\theta} \\ 1 & 0 & 0 & 0 & -\frac{1}{\theta} \\ 0 & 1 & 0 & 0 & -\frac{w}{\theta} \\ 0 & 0 & 0 & 1 & -\frac{w}{\theta} \end{bmatrix} \]