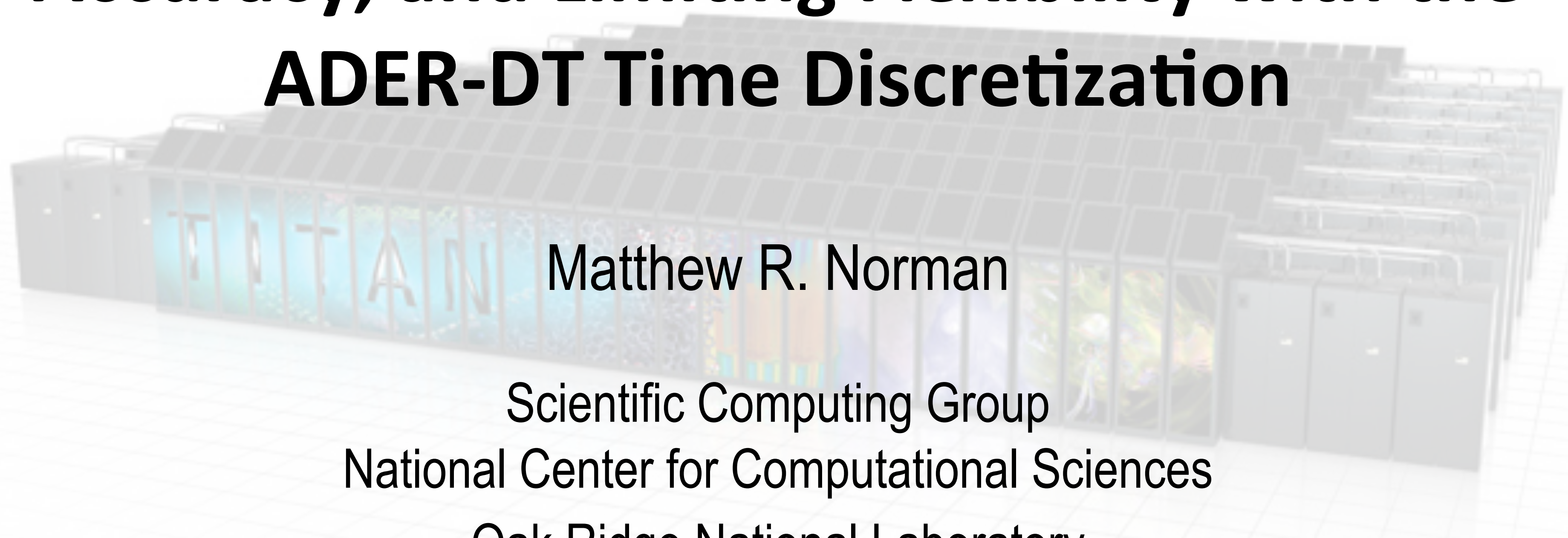
Improving Dynamical Core Scalability, Accuracy, and Limiting Flexibility with the ADER-DT Time Discretization



Matthew R. Norman

Scientific Computing Group
National Center for Computational Sciences
Oak Ridge National Laboratory

PDEs On The Sphere 2014

OAK RIDGE National Laboratory

OLCF
OAK RIDGE LEADERSHIP COMPUTING FACILITY

Some Properties Of A "Good" Algorithm

- Accuracy
  - Higher-order generally gives bigger bang for your buck
  - High-order coupling of PDE terms & dimensions
- Speed
  - Larger time step
  - Lower cost per time step
  - Scalability
    - Lower MPI Overhead
    - Computing scales faster than communication
- Robustness
  - Limited, non-amplifying oscillations, positivity, monotonicity

What Is ADER-DT?

- ADER-DT = <u>A</u>britrary <u>DER</u>ivatives with <u>D</u>ifferential <u>T</u>ransforms
- ADER: (Spatial derivatives)+(the PDE itself)→(time derivatives)

$$\frac{\partial q}{\partial t}+\frac{\partial f\left(q\right)}{\partial x}=0 \quad \Rightarrow \quad \frac{\partial q}{\partial t}=-\frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

$$\frac{\partial^2 q}{\partial x\partial t}=-\frac{\partial^2 f}{\partial q^2}\left(\frac{\partial q}{\partial x}\right)^2-\frac{\partial f}{\partial q}\frac{\partial^2 q}{\partial x^2} \quad \Rightarrow \quad \frac{\partial^2 q}{\partial t^2}=-\frac{\partial^2 f}{\partial q^2}\frac{\partial q}{\partial t}\frac{\partial q}{\partial x}-\frac{\partial f}{\partial q}\frac{\partial^2 q}{\partial t\partial x}$$

- Temporal order of accuracy matches the spatial order
- A Differential Transform (DT) is just a Taylor Series coefficient

$$Q\left(k_x,k_t\right)=\frac{1}{k_x!k_t!}\frac{\partial^{k_x+k_y}q\left(x,t\right)}{\partial x^{k_x}\partial t^{k_t}} \qquad q\left(x,t\right)=\sum_{k_x=0}^{N-1}\sum_{k_t=0}^{N-1}Q\left(k_x,k_t\right)x^{k_x}t^{k_t}$$

DTs Make ADER Cheaper & Simpler

Burger's Equation

$$\frac{\partial q}{\partial t}+\frac{\partial f(q)}{\partial x}=0$$

$$f(q)=\frac{q^2}{2}$$

DT of Burger's Equation

$$Q(k_x,k_t+1)=-\frac{k_x+1}{k_t+1}F(k_x+1,k_t)$$

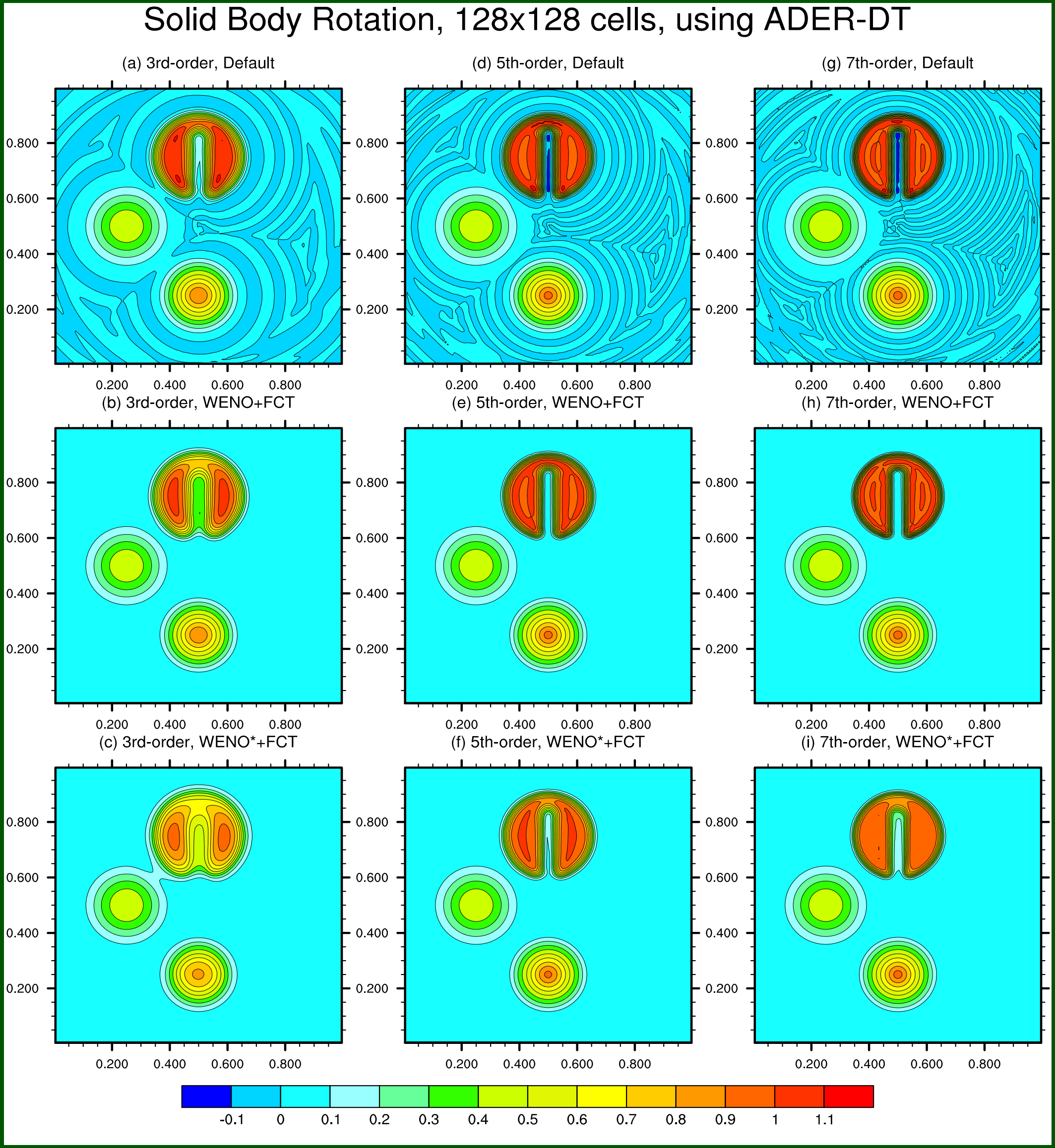
$$F(k_x,k_t)=\frac{1}{2}\sum_{r_x=0}^{k_x}\sum_{r_t=0}^{k_t}Q(r_x,r_t)Q(k_x-r_x,k_t-r_t)$$

- All PDE terms are space-time polynomials (no quadrature)
- Non-linearly coupled, high-order accuracy w/ no stages (scalable)
- Any order of accuracy by changing just one line of code
- Automatically preserves non-oscillatory properties
- Easily adapted to any grid, spatial operator, or PDE set
- When using WENO, only one limiting applied per time step
- p-refinement happens in time as well, not only in space

The Gist of Why I Like FV and ADER-DT

- FV + ADER-DT + WENO + FCT positivity + Half-tensor
  - Means a positive, limited, high-order, cheap, & large time step with only 1 data transfer per time step
- **MM**FV + ADER-DT + **H**WENO + FCT + Half-tensor
  - Same as before, same time step, and you get multi-moment

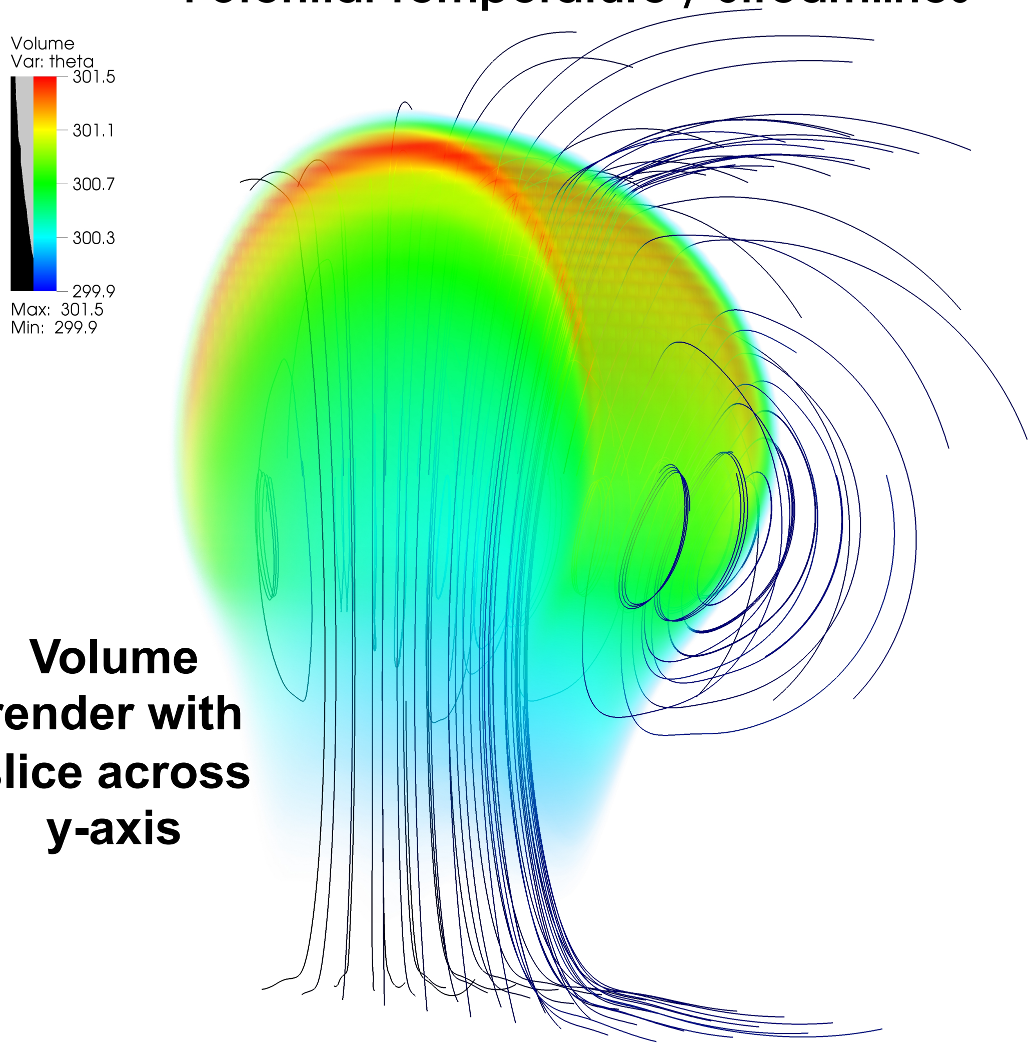
Is ADER-DT Viable For Transport?



- Left to Right: 3rd, 5th, 7th
- Top panel: no limiting
- Middle: "light WENO" & FCT positivity
- Bottom: "heavy WENO" & FCT positivity
- Even 7th-order can be quite smooth
  - But you pay in accuracy
- ADER-DT + WENO offers a range of limiting
  - Two tunable parameters
  - Can be smooth or sharp

3-D Cartesian Euler Model: Rising Thermal

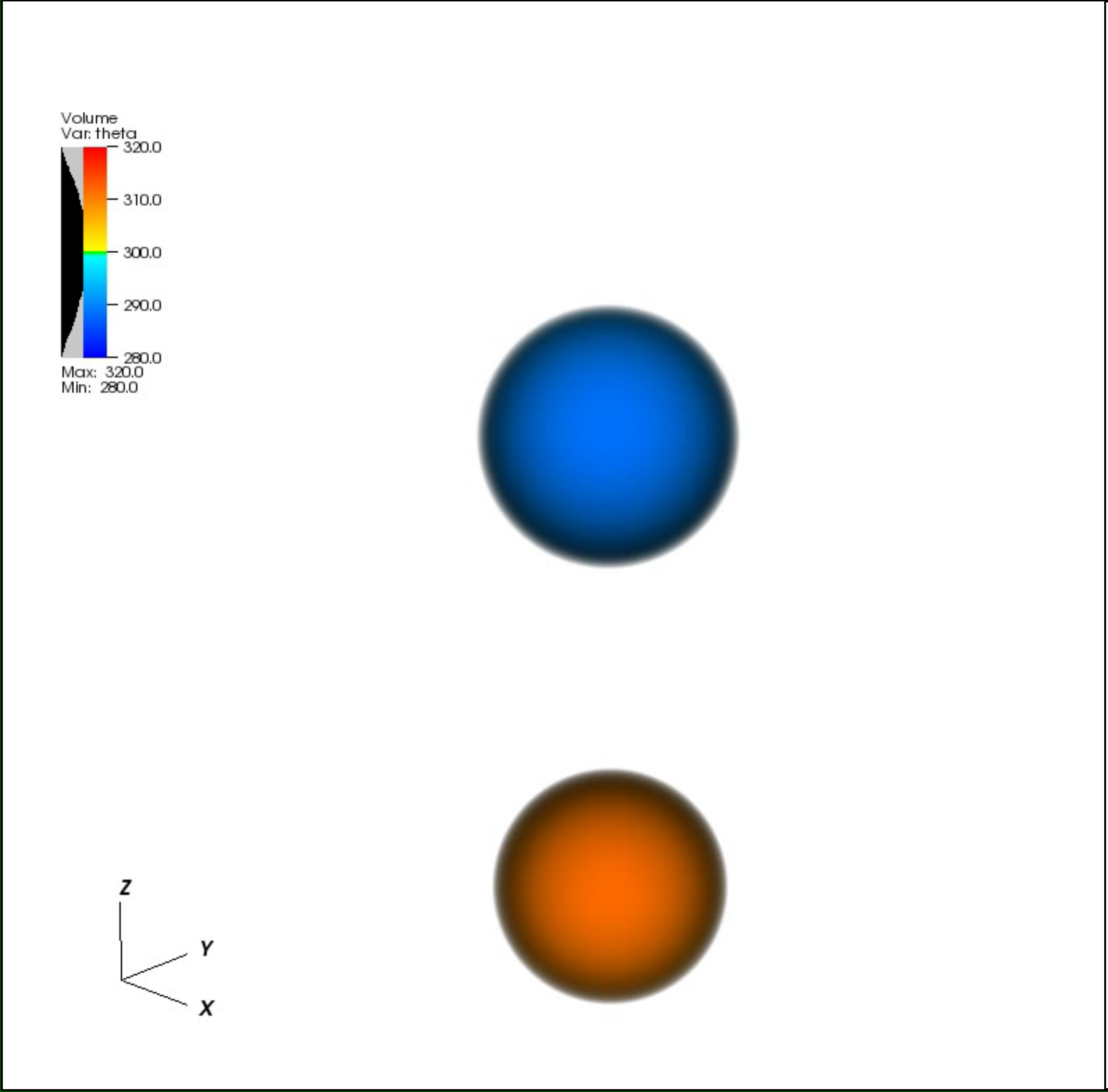
Potential Temperature / Streamlines



Volume render with slice across y-axis

- Genuinely 3-D Finite-Volume using ADER-DT
- ADER-DT is 5x, 4.5x, and 2x faster than SSP-RK4
  - 5th-, 7th-, & 9th-ord ADER versus 4th-ord RK
  - Using RK max stable time step, not SSP
- OpenMP + 3-D MPI
- 3rd, 5th, 7th, & 9th-order accuracies so far
- 3-D WENO limiting
- HLLC, LLF, or Upwind flux
- Uses "half-tensor" for up to 24x less cost

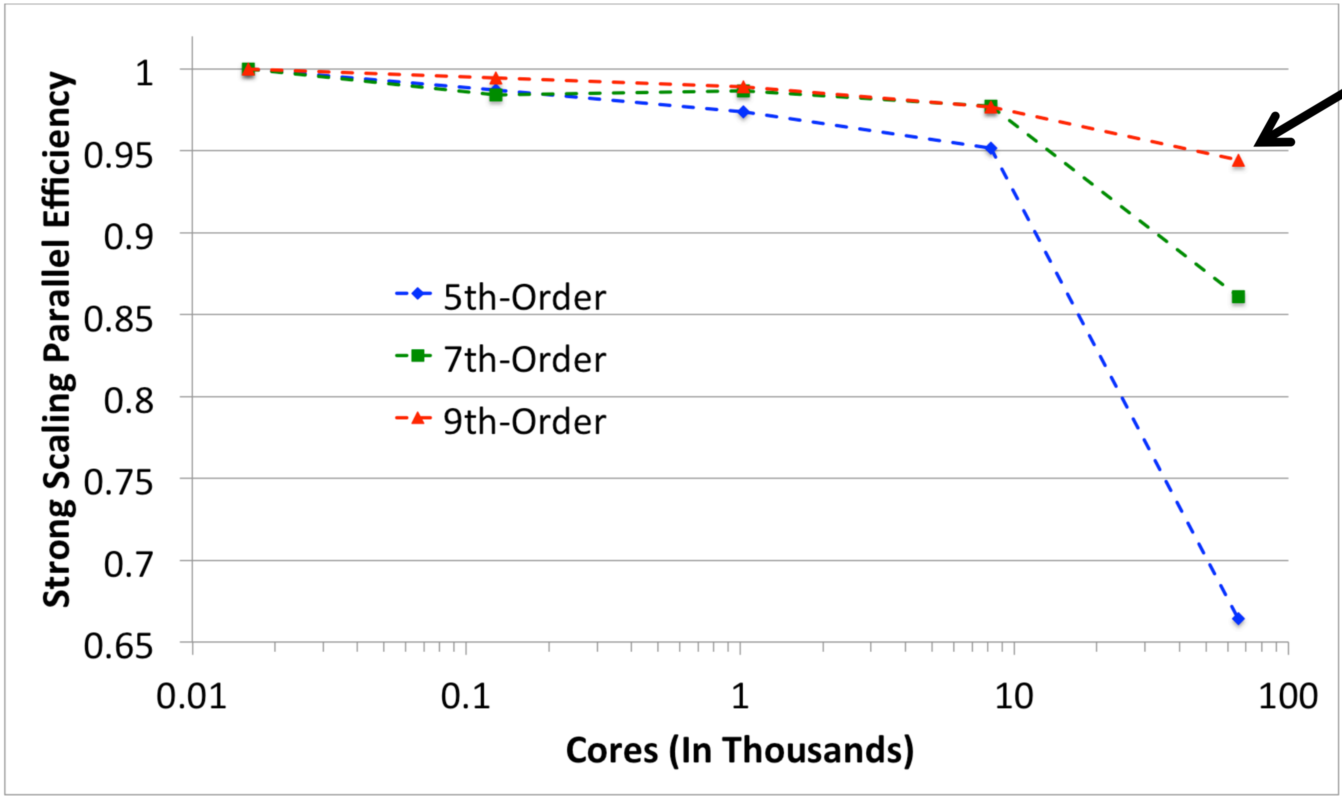
3-D Cartesian Euler Model: Colliding Thermals



- 20K perturbation bubbles collide
- Sharp discontinuities
- WENO successfully limits to provide stability
- Uses "sub-cell" method for multi-dim WENO
- Larger WENO coef causes more damping
- WENO overhead only 30% to 60%
  - Overhead relatively less for higher-order

ADER-DT + FV Scaling: 3-D Compressible Euler

- Strong scaling: 256^3 cells from 1 node to 4096 nodes on Titan
- MPI+OpenMP; Fortran 90; Cray compiler; CPU-only for now
- On-node: 20% peak flops on Interlagos & Sandybridge



Only 16^3 cells per node

No WENO Limiting Used

Simple 3-D MPI decomp; no overlapping

Why Use A Half-Tensor?

Full Tensor

```
for kt=0,N-1
  for kz=0,N-1
    for ky=0,N-1
      for kx=0,N-1
        for rt=0,kt
          for rz=0,kz
            for ry=0,ky
              for rx=0,kx
```

Half Tensor

```
for kt=0,N-1
  for kz=0,N-1-kt
    for ky=0,N-1-kt-kz
      for kx=0,N-1-kt-kz-ky
        for rt=0,kt
          for rz=0,kz
            for ry=0,ky
              for rx=0,kx
```

- Innermost loop body executed 102x, 205x, & 319x fewer times at 5th-, 7th-, and 9th-order accuracies
- Space-time polynomial contains 8.9x, 11.4x, & 13.3x fewer terms
- Half-tensor make genuinely multi-dim simulation more feasible

Handling Quasi-Steady Balances

- Example: Hydrostasis $\partial_z p_H\left(\vec{x},t\right) = -g\rho_H\left(\vec{x},t\right)$
- Already have true pressure & density as space-time polynomials
- Equate "hydrostatic pressure" with true pressure
- Diagnose "hydrostatic density" with Differential Transforms

$$R_H\left(k_x,k_y,k_z,k_t\right) = -\frac{k_z+1}{g}P_H\left(k_x,k_y,k_z+1,k_t\right)$$

- Removing hydrostasis removes vertical pressure term entirely
- If there's a balance, you can recast a flux term as a source term
  - Or vice versa

Portability For Accelerators

- Enough cells per node = efficient port to GPUs
- How many cells? Generally ≈32K, preferably many more
  - Equivalent to CAM-SE, 4th-order, 64 elements per node
- CPU implementation computes each operation on blocks
  - Blocks of 16 (32) for double (single) precision perform best
  - Must be known at compile time for vectorization to be successful
- GPU implementation would just increase block size to 32K
  - No re-organization of data structures needed
- <u>May</u> be opportunity for more threading over DT procedure
  - 8 nested triangular loops for a 3-D simulation
  - If surface area of “front of dependence” is large, that means threads
  - Likely viable only at higher-order (7th and up, perhaps)

Implementation In Finite-Volume Framework

- Start with PDE: $\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = S$ and cell averages: $\bar{U}_{i,j,k}$
- Reconstruct spatial expansion at cell centroid: $U_{i,j,k}(x,y,z)$
- Perform ADER-DT to form space-time expansion:

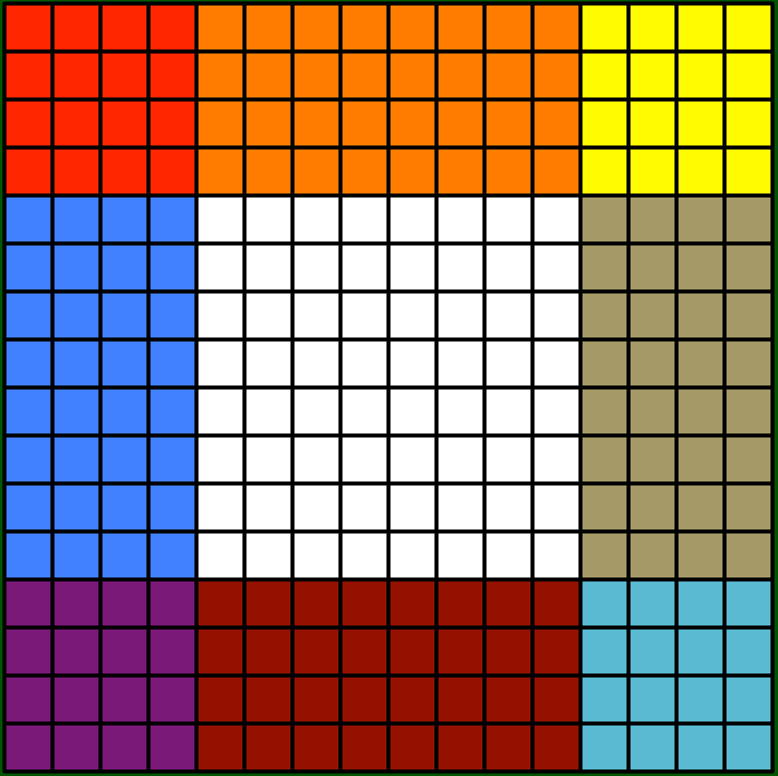
  $U_{i,j,k}(x,y,z,t) \quad F_{i,j,k}(x,y,z,t) \quad G_{i,j,k}(x,y,z,t) \quad H_{i,j,k}(x,y,z,t) \quad S_{i,j,k}(x,y,z,t)$
- Compute space-time averages of fluxes at cell faces: $\hat{F}_{i,j,k} \quad \hat{G}_{i,j,k} \quad \hat{H}_{i,j,k}$
- Compute space-time average of source over cell body: $\hat{S}_{i,j,k}$
- Apply <u>one</u> Riemann solve per face per time step using space-time-averaged limits at the interface
  - Any linearized Riemann solver will do: Upwind, LLF, HLLC
- For general spatial operators
  - Compute time-average and apply operator like normal

Why I Prefer Finite-Volume

- Order of accuracy doesn’t affect the time step
- Multi-moment doesn’t affect the time step (MMFV + HWENO)
- WENO is a given: no added transfers, no load imbalance
- (FV) + (ADER-DT) + ([H]WENO) + (FCT positivity) means a positive, limited, high-order, large time step w/ only 1 transfer
- Can do genuinely multi-dimensional with less work
  - Half-tensor of derivatives is up to 24x smaller than full-tensor in 3D
- Some Galerkin schemes are hardwired to quadrature

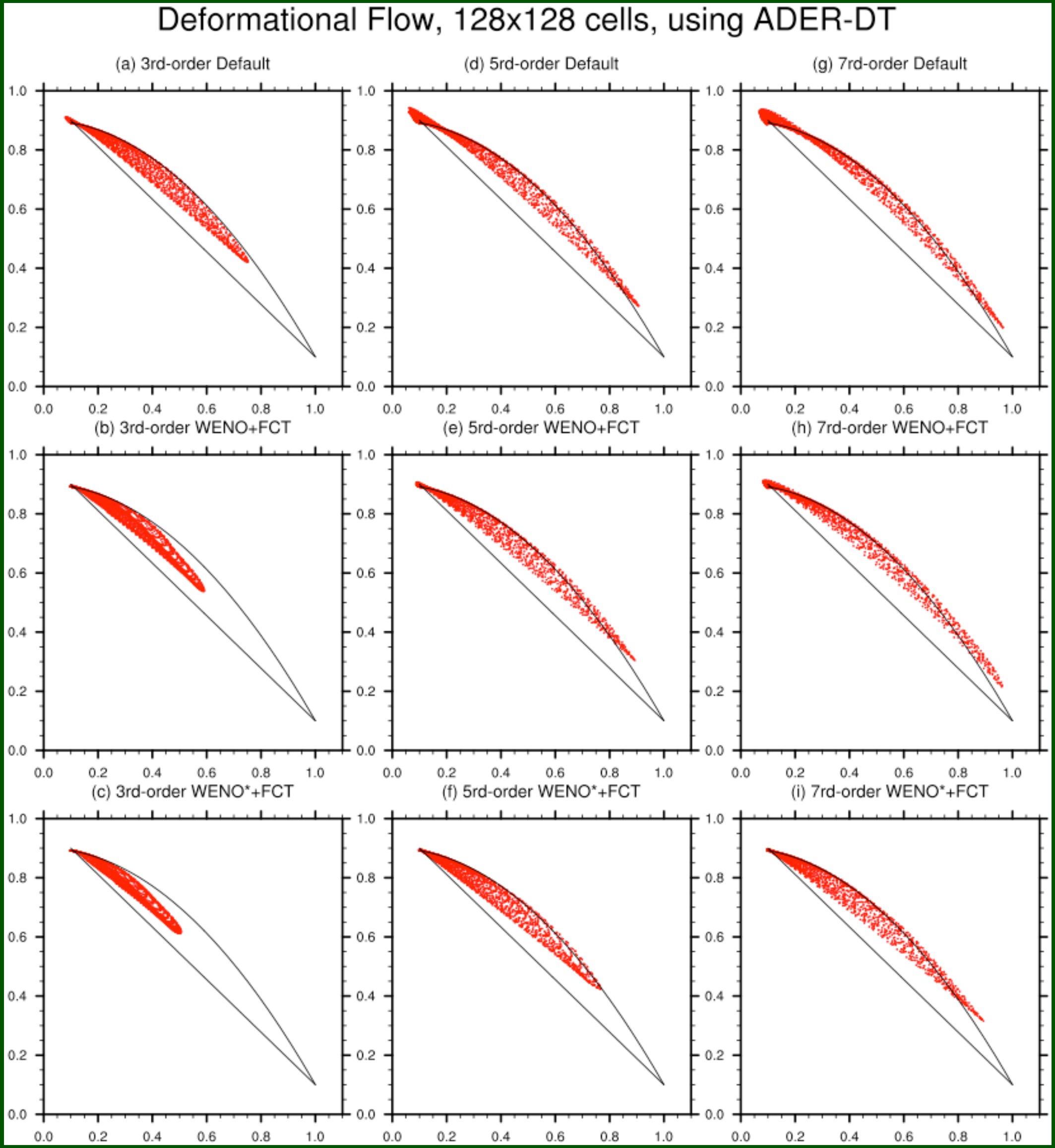
The 3-D High-Order FV Halo: 2-D Slice

- Shaded regions depend on halo updates (16^3 cells at 9th-order)
- Only 12.5% of the domain is free of halo dependence
- Won't the size of the transfers kill you at scale?



- Computation increases much faster than communication
- Increase in work per node outweighs increased transfer cost
- WENO improves this by adding computation w/o communication
- Ameliorates needing to overlap computation & communication

Is ADER-DT Viable For Transport?



- Left to Right: 3rd, 5th, 7th
- Top panel: no limiting
- Middle: “light WENO” & FCT positivity
- Bottom: “heavy WENO” & FCT positivity

- WENO successfully removes bouds violating unmixing
- Still some unmixing at the unresolved tail

ADER-DT Verses Runge-Kutta

- Multi-stage time discretizations most common (Runge-Kutta)
  - Multiple copies of the fluid state (taxing on memory)
  - Multiple data transfers per time step / small effective time step
  - Higher than 4th-order is difficult and expensive to obtain
  - Maintaining non-oscillatory properties reduces time step even further
  - WENO limiting typically applied at each stage
- ADER-DT improves upon this
  - Only one copy of the fluid state is needed, more work per byte
  - Only one data transfer per time step / much larger effective time step
  - Any order of accuracy is as easy as changing one line of code
  - Non-oscillatory properties automatically maintained, same time step
  - Only one WENO procedure per large time step