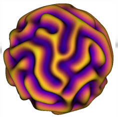
High-Order Genuinely Multidimensional Finite Volume Methods via Kernel-Based WENO

May 5, 2023

Ian May, Dongwook Lee

Department of Applied Mathematics
University of California Santa Cruz

Santa Cruz, CA



Introduction

Goal

Solve systems of hyperbolic conservation laws

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = \mathbf{S}(\mathbf{U})$$

with an *accurate* and *robust* finite volume method

$$\frac{\partial \langle \mathbf{U} \rangle_\Omega}{\partial t} + \frac{1}{||\Omega||} \int\limits_{\partial\Omega} \hat{\mathbf{F}} \left(\mathbf{U}^-(\mathbf{x}), \mathbf{U}^+(\mathbf{x}) \right) \cdot \mathbf{n} dx = \frac{1}{||\Omega||} \int\limits_{\Omega} \mathbf{S}(\mathbf{U}) d\boldsymbol{x}$$

in *multiple* dimensions.

Systems of interest

Compressible Euler equations

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial}{\partial x}\mathbf{F}(\mathbf{U}) + \frac{\partial}{\partial y}\mathbf{G}(\mathbf{U}) + \frac{\partial}{\partial z}\mathbf{H}(\mathbf{U}) = 0$$

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{pmatrix} \quad \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u w \\ u(E+p) \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ \rho v w \\ v(E+p) \end{pmatrix} \quad \mathbf{H} = \begin{pmatrix} \rho w \\ \rho u w \\ \rho v w \\ \rho w^2 + p \\ w(E+p) \end{pmatrix},$$

for a calorically ideal gas,

$$p = (\gamma - 1)\rho\epsilon, \quad \rho\epsilon = E - \frac{1}{2}\rho\mathbf{v}\cdot\mathbf{v}.$$

Systems of interest

Ideal GLM-MHD equations

$$
\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \\ B_x \\ B_y \\ B_z \\ \psi \end{pmatrix} \quad
\mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p_* - B_x^2 \\ \rho u v - B_x B_y \\ \rho u w - B_x B_z \\ F_E \\ C_h \psi \\ u B_y - v B_x \\ u B_z - w B_x \\ C_h B_x \end{pmatrix} \quad
\mathbf{S} = \begin{pmatrix} 0 \\ -B_x \nabla \cdot \mathbf{B} \\ -B_y \nabla \cdot \mathbf{B} \\ -B_z \nabla \cdot \mathbf{B} \\ -S_E \\ -u \nabla \cdot \mathbf{B} \\ -v \nabla \cdot \mathbf{B} \\ -w \nabla \cdot \mathbf{B} \\ -\mathbf{v} \cdot \nabla \psi \end{pmatrix},
$$

where,

$$
\begin{aligned}
p_* &= p_{\text{gas}} + p_{\text{mag}} \\
F_E &= u(E + p_g) + B_x \left(C_h \psi - \mathbf{v} \cdot \mathbf{B} \right) \\
S_E &= (\mathbf{v} \cdot \mathbf{B}) \nabla \cdot \mathbf{B} + \psi \mathbf{v} \cdot \nabla \psi
\end{aligned}
$$

Quick overview of FVM

Abstract formulation

Partition full domain Ω into *finite volumes* Ω_i such that $\Omega = \bigcup\limits_i \Omega_i$, and $\Omega_i \cap \Omega_j = \varnothing,\ i \neq j$. Denote

$$\langle \cdot \rangle_i = \frac{1}{||\Omega_i||} \int\limits_{\Omega_i} \cdot d\mathbf{x},$$

then for (systems of) hyperbolic conservation laws

$$\frac{\partial}{\partial t} \langle \mathbf{U} \rangle_i = -\frac{1}{||\Omega_i||} \oint\limits_{\partial \Omega_i} \hat{\mathbf{F}} \left(\mathbf{U}^-, \mathbf{U}^+ \right) \cdot \mathbf{n} ds$$

for numeric flux $\hat{\mathbf{F}}$, and states $\mathbf{U}^-$ and $\mathbf{U}^+$ inside and outside Ω_i.

Quick overview of FVM

Uniform 2D Cartesian grids

Let $\Omega_{i,j} = \left[x_i - \frac{\Delta x}{2}, x_i - \frac{\Delta x}{2}\right] \times \left[y_i - \frac{\Delta y}{2}, y_j - \frac{\Delta y}{2}\right]$, then

$$\frac{\partial}{\partial t}\langle \mathbf{U} \rangle_{i,j} = -\frac{1}{||\Omega_{i,j}||} \oint\limits_{\partial \Omega_{i,j}} \hat{\mathbf{F}}\left(\mathbf{U}^-, \mathbf{U}^+\right) \cdot \mathbf{n} ds$$

$$= -\frac{1}{\Delta x}\left(\langle \hat{\mathbf{F}} \rangle_{i+\frac{1}{2},j} - \langle \hat{\mathbf{F}} \rangle_{i-\frac{1}{2},j}\right) - \frac{1}{\Delta y}\left(\langle \hat{\mathbf{G}} \rangle_{i,j+\frac{1}{2}} - \langle \hat{\mathbf{G}} \rangle_{i,j-\frac{1}{2}}\right)$$

where half-indices indicate integration over faces.

Quick overview of FVM

Uniform 2D Cartesian grids

Let $\Omega_{i,j} = \left[x_i - \frac{\Delta x}{2}, x_i - \frac{\Delta x}{2}\right] \times \left[y_i - \frac{\Delta y}{2}, y_j - \frac{\Delta y}{2}\right]$, then

$$\frac{\partial}{\partial t}\langle \mathbf{U} \rangle_{i,j} = -\frac{1}{||\Omega_{i,j}||} \oint\limits_{\partial \Omega_{i,j}} \hat{\mathbf{F}}\left(\mathbf{U}^-, \mathbf{U}^+\right) \cdot \mathbf{n} ds$$
$$= -\frac{1}{\Delta x}\left(\langle \hat{\mathbf{F}} \rangle_{i+\frac{1}{2},j} - \langle \hat{\mathbf{F}} \rangle_{i-\frac{1}{2},j}\right) - \frac{1}{\Delta y}\left(\langle \hat{\mathbf{G}} \rangle_{i,j+\frac{1}{2}} - \langle \hat{\mathbf{G}} \rangle_{i,j-\frac{1}{2}}\right)$$

where half-indices indicate integration over faces.

Two barriers to high order in multiple dimensions

- Face integral must be done accurately
- Numerical flux is defined *pointwise*, thus need accurate *pointwise* values of $\mathbf{U}$ on faces

Accurate construction of Riemann states

Multidimensional concerns

Issues with polynomials

- Matching stencils to multivariate polynomial spaces is hard
- Forming valid substencils for WENO is even harder
- Dimension-by-dimension approaches do work, but get messy

[1] Omitting many technical details

Accurate construction of Riemann states
Multidimensional concerns

Issues with polynomials

- Matching stencils to multivariate polynomial spaces is hard
- Forming valid substencils for WENO is even harder
- Dimension-by-dimension approaches do work, but get messy

Kernel based interpolation/recovery

Each SPD *kernel* $K : \Omega \times \Omega \to \mathbb{R}$, induces a *reproducing kernel Hilbert space*[1], $\mathcal{H}$, consisting of

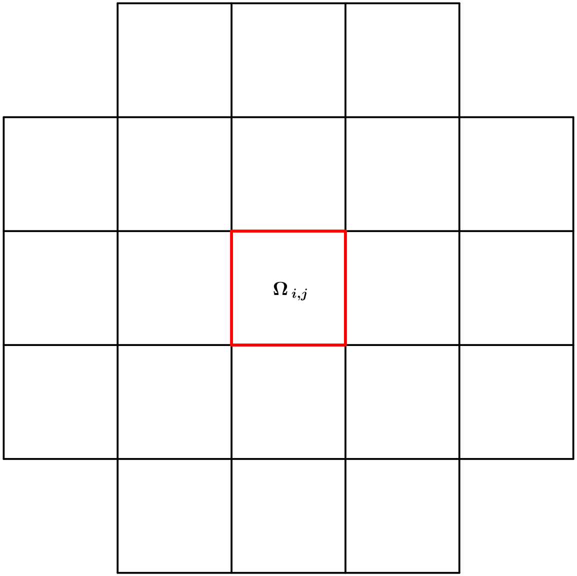
$$f(x) = \sum_i a_i K(x, x_i)$$

$$\sum_i \sum_j a_i a_j K(x_i, x_j) < \infty$$

For this talk: $K(x, y) = e^{-\frac{||x-y||^2}{2\ell^2}}$.

[1] Omitting many technical details

An exemplary stencil: $R = 2$



Kernel-based interpolation

Let $\{\boldsymbol{x}_i\} \subset \Omega$ be distinct, and $\mathbf{f}_i = f(\boldsymbol{x}_i)$ known.
Seek an interpolant of the form:

$$\widetilde{f}(\boldsymbol{x}) = \sum_{j=1}^{N} a_j K(\boldsymbol{x}, \boldsymbol{x}_j)$$

then enforcing that $\widetilde{f}(\boldsymbol{x}_i) = \mathbf{f}_i$, requires that $\mathbf{a}$ satisfy

$$\mathbf{K}\mathbf{a} = \mathbf{f}, \quad \mathrm{K}_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

Properties of kernel-based interpolation

- $K(\boldsymbol{x}, \boldsymbol{x}_j) \in \mathcal{H}$, hence $\widetilde{f} \in \mathcal{H}$
- $(f - \widetilde{f}) \perp_{\mathcal{H}} \operatorname{span}(K(\boldsymbol{x}, \boldsymbol{x}_j))$
- The matrix $\mathbf{K}$ is SPD if the kernel K is

Kernel-based generalized interpolation

Symmetric form

Let $\{\lambda_i\} \subset \mathcal{H}'$ be linearly independent, and $\mathbf{g}_i = \lambda_i f$ known.
Seek an interpolant of the form:

$$\widetilde{f}(x) = \sum_{j=1}^{N} a_j \lambda_j^{(y)} K(x, y)$$

then enforcing that $\lambda_i^{(x)} \widetilde{f}(x) = \mathbf{g}_i$, requires that $\mathbf{a}$ satisfy

$$\mathbf{Ca} = \mathbf{g}, \quad \mathrm{C}_{i,j} = \lambda_i^{(x)} \lambda_j^{(y)} K(x, y).$$

Kernel-based generalized interpolation

Symmetric form

Let $\{\lambda_i\} \subset \mathcal{H}'$ be linearly independent, and $\mathbf{g}_i = \lambda_i f$ known.
Seek an interpolant of the form:

$$\widetilde{f}(x) = \sum_{j=1}^{N} a_j \lambda_j^{(y)} K(x,y)$$

then enforcing that $\lambda_i^{(x)} \widetilde{f}(x) = \mathbf{g}_i$, requires that $\mathbf{a}$ satisfy

$$\mathbf{Ca} = \mathbf{g}, \quad \mathrm{C}_{i,j} = \lambda_i^{(x)} \lambda_j^{(y)} K(x,y).$$

Relationship to regular interpolation

- $\lambda_j^{(y)} K(x,y) \in \mathcal{H}_A$, hence $\widetilde{f} \in \mathcal{H}_A$
- $(f - \widetilde{f}) \perp_{\mathcal{H}_A} \operatorname{span}\left(\lambda_j^{(y)} K(x,y)\right)$
- Point evaluation functionals, $\lambda_j = \delta_{x_j}$, gives previous result
- The matrix $\mathbf{C}$ is still SPD so long as K is

Kernel-based generalized interpolation
Asymmetric form with polynomial constraints

Additionally let $\{\boldsymbol{\alpha}_k : |\boldsymbol{\alpha}_k| \leq D\}$ be a set of multi-indices.

Seek an interpolant of the form:

$$\widetilde{f}(x) = \sum_{j=1}^{N} a_j K(x, x_j) + \sum_{|\boldsymbol{\alpha}_k| \leq D} b_k \boldsymbol{x}^{\boldsymbol{\alpha}_k}$$

Kernel-based generalized interpolation
Asymmetric form with polynomial constraints

Additionally let $\{\boldsymbol{\alpha}_k : |\boldsymbol{\alpha}_k| \le D\}$ be a set of multi-indices.

Seek an interpolant of the form:

$$\widetilde{f}(x) = \sum_{j=1}^{N} a_j K(x, x_j) + \sum_{|\boldsymbol{\alpha}_k| \le D} b_k \boldsymbol{x}^{\boldsymbol{\alpha}_k}$$

then enforcing that $\lambda_i^{(x)} \widetilde{f}(x) = \mathbf{g}_i$, requires that $\mathbf{a}$ and $\mathbf{b}$ satisfy

$$\mathbf{Qa} + \mathbf{Pb} = \mathbf{g}, \quad \mathrm{Q}_{i,j} = \lambda_i^{(x)} K(x, x_j), \; \mathrm{P}_{i,j} = \lambda_i^{(x)} \boldsymbol{x}^{\boldsymbol{\alpha}_k}.$$

To be exact for all $\boldsymbol{x}^{\boldsymbol{\alpha}_k}$ we also need $\mathbf{P}^T \mathbf{a} = \mathbf{0}$, so ultimately solve

$$\begin{bmatrix} \mathbf{Q} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{0} \end{pmatrix}$$

Precomputing reconstruction vectors
Asymmetric form

We can solve

$$\begin{bmatrix} \mathbf{Q} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{0} \end{pmatrix},$$

then evaluate the resulting interpolant at $\boldsymbol{x}^*$

$$\widetilde{f}(\boldsymbol{x}^*) = \sum_{j=1}^{N} a_j K(\boldsymbol{x}^*, \boldsymbol{x}_j) + \sum_{|\boldsymbol{\alpha}_k| \leq D} b_k (\boldsymbol{x}^*)^{\boldsymbol{\alpha}_k} = \mathbf{T}^T \mathbf{a} + \mathbf{S}^T \mathbf{b}.$$

Precomputing reconstruction vectors

Asymmetric form

We can solve

$$\begin{bmatrix} \mathbf{Q} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{0} \end{pmatrix},$$

then evaluate the resulting interpolant at $\boldsymbol{x}^*$

$$\widetilde{f}(\boldsymbol{x}^*) = \sum_{j=1}^{N} a_j K(\boldsymbol{x}^*, \boldsymbol{x}_j) + \sum_{|\boldsymbol{\alpha}_k| \le D} b_k (\boldsymbol{x}^*)^{\boldsymbol{\alpha}_k} = \mathbf{T}^T \mathbf{a} + \mathbf{S}^T \mathbf{b}.$$

Hence the *reconstruction vector* can be precomputed from

$$\begin{bmatrix} \mathbf{Q}^T & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{r} \\ \mathbf{w} \end{pmatrix} = \begin{pmatrix} \mathbf{T} \\ \mathbf{S} \end{pmatrix},$$

giving simply $\widetilde{f}(\boldsymbol{x}^*) = \mathbf{r}^T \mathbf{g}$.

Accurate flux integrals
Transverse corrections

- We can now obtain accurate point estimates of the solution
- Call an (approximate) Riemann solver to find pointwise fluxes
- But *where* should we do this?

Accurate flux integrals
Transverse corrections

- We can now obtain accurate point estimates of the solution
- Call an (approximate) Riemann solver to find pointwise fluxes
- But *where* should we do this?

Buchmuller-Helzel correction
Generate pointwise fluxes at the center of each face, fit a polynomial in the transverse direction(s), integrate that polynomial exactly.

Accurate flux integrals
Transverse corrections

- We can now obtain accurate point estimates of the solution
- Call an (approximate) Riemann solver to find pointwise fluxes
- But *where* should we do this?

Buchmuller-Helzel correction
Generate pointwise fluxes at the center of each face, fit a polynomial in the transverse direction(s), integrate that polynomial exactly.

Kernel-based quadrature
Find kernel-based interpolant through the fluxes and integrate it exactly. (Potentially interesting in 3D)

Accurate flux integrals
Transverse corrections

- We can now obtain accurate point estimates of the solution
- Call an (approximate) Riemann solver to find pointwise fluxes
- But *where* should we do this?

Buchmuller-Helzel correction

Generate pointwise fluxes at the center of each face, fit a polynomial in the transverse direction(s), integrate that polynomial exactly.

Kernel-based quadrature

Find kernel-based interpolant through the fluxes and integrate it exactly. (Potentially interesting in 3D)

Gaussian quadrature

Solve multiple Riemann problems per face, then do Gaussian quadrature.
Ultimately, Riemann solvers are not that expensive so this is the easiest (and most stable) option.

Taylor-Green vortex problem
Compressible Euler equations

Triply periodic vortex on $[0, 2\pi]^3$ that quickly becomes turbulent.

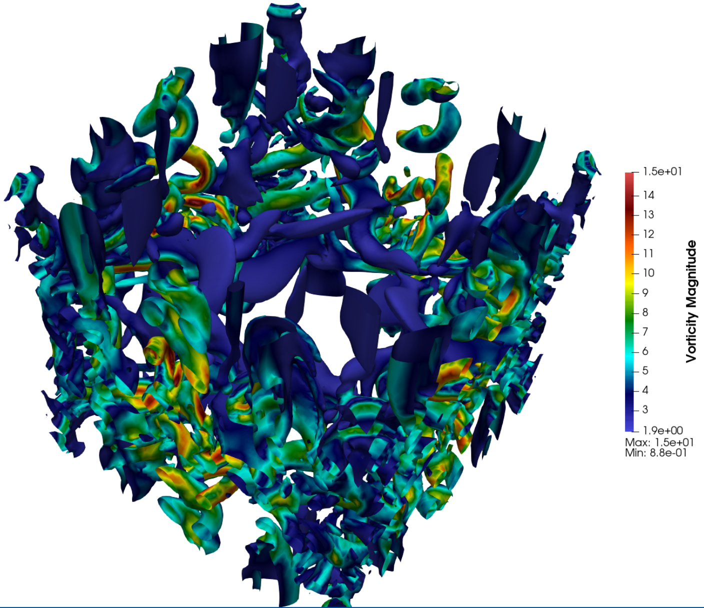
Initial conditions

$$\begin{cases} \rho &= 1 \\ u &= \sin(x)\cos(y)\cos(z) \\ v &= -\cos(x)\sin(y)\cos(z) \\ w &= 0 \\ p &= 100 + \frac{1}{16}\left(\cos(2x) + \cos(2y)\right)\left(2 + \cos(2z)\right) \end{cases}$$

Note: Without physical viscosity this problem is mostly of qualitative significance.

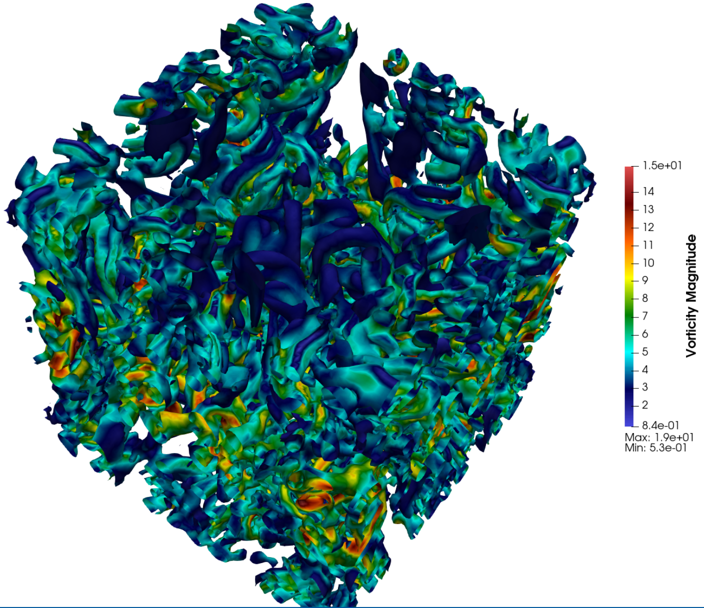
Taylor-Green vortex

$192 \times 192 \times 192$, Radius 2, $\ell = 24\Delta$



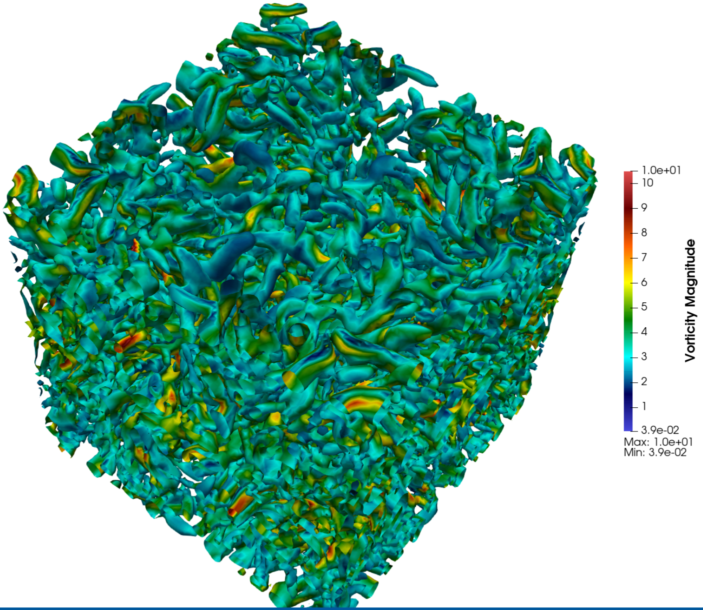
Taylor-Green vortex

$192 \times 192 \times 192$, Radius 2, $\ell = 24\Delta$



Taylor-Green vortex

$192 \times 192 \times 192$, Radius 2, $\ell = 24\Delta$



Dealing with shocks: WENO methods

Nonlinear reconstruction

The reconstruction presented is linear, i.e.

$$\widetilde{f}(\boldsymbol{x}^*) = \mathbf{r}^T \mathbf{g},$$

which is hopeless near discontinuities (Godunov)

Dealing with shocks: WENO methods

Nonlinear reconstruction

The reconstruction presented is linear, i.e.

$$\widetilde{f}(\boldsymbol{x}^*) = \mathbf{r}^T\mathbf{g},$$

which is hopeless near discontinuities (Godunov)

WENO (weighted essentially non-oscillatory) methods

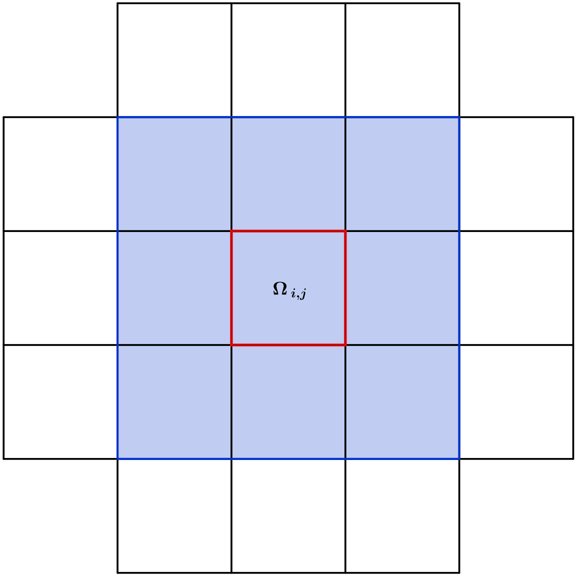
Break full stencil into substencils, use weighted combination of individual reconstructions

$$\widetilde{f}(\boldsymbol{x}^*) = \sum_{S_k \in \mathcal{S}_{i,j}} \omega_k \mathbf{r}_{(k)}^T \mathbf{g}_{(k)}$$

where $\mathcal{S}_{i,j}$ is set of substencils, and ω_k **depends** on the data in S_k.

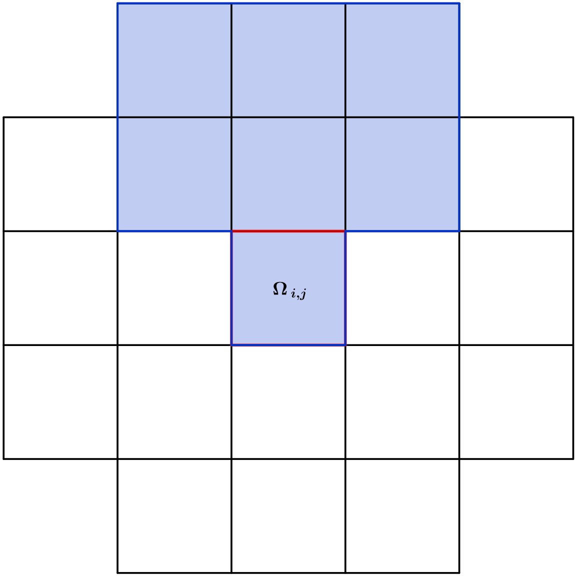
S_1: Central substencil

Substencils in the spirit of standard WENO



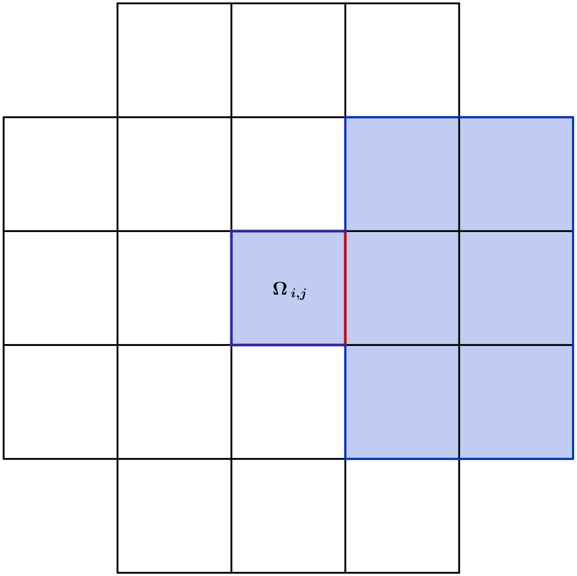
S_2: North substencil

Substencils in the spirit of standard WENO



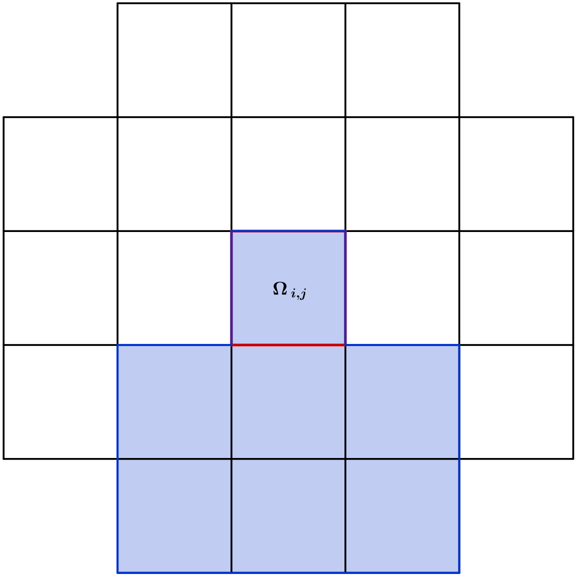
S_3: East substencil

Substencils in the spirit of standard WENO



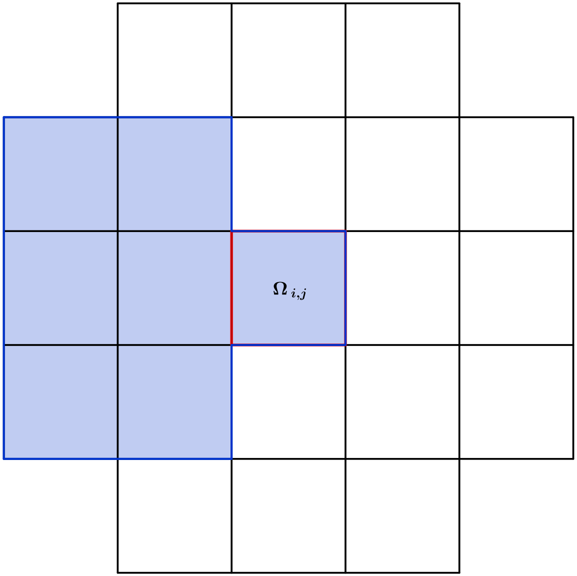
S_4: South substencil

Substencils in the spirit of standard WENO



S_5: West substencil

Substencils in the spirit of standard WENO



Optimal weights and standard WENO

The optimal linear weights γ_k minimize discrepancy in

$$\widetilde{f}(\boldsymbol{x}^*) \approx \sum_{k=1}^{5} \gamma_k \mathbf{r}_{(k)}^T \mathbf{g}_{(k)}$$

independent of the data.

Optimal weights and standard WENO

The optimal linear weights γ_k minimize discrepancy in

$$\widetilde{f}(\boldsymbol{x}^*) \approx \sum_{k=1}^{5} \gamma_k \mathbf{r}_{(k)}^T \mathbf{g}_{(k)}$$

independent of the data.

Desired behavior of ω_k

- For smooth data $\omega_k \approx \gamma_k$ on all substencils
- For rough data $\omega_k \approx 0$ on rough substencils

This is obtained by use of *smoothness indicators*.

Optimal weights and standard WENO

The optimal linear weights γ_k minimize discrepancy in

$$\widetilde{f}(\boldsymbol{x}^*) \approx \sum_{k=1}^{5} \gamma_k \mathbf{r}_{(k)}^T \mathbf{g}_{(k)}$$

independent of the data.

Desired behavior of ω_k

- For smooth data $\omega_k \approx \gamma_k$ on all substencils
- For rough data $\omega_k \approx 0$ on rough substencils

This is obtained by use of *smoothness indicators*.

Special cases: Polynomial reconstruction

For *some* polynomial degrees on *some* (sub)stencil choices, equality can be obtained (e.g. classical WENO5).

WENO-AO

Generally, no linear weights, γ_k, exist that can reproduce the accuracy of the full stencil.

WENO-AO

Generally, no linear weights, γ_k, exist that can reproduce the accuracy of the full stencil.

Adaptive order WENO

Let S_0 correspond to the full stencil, and include it explicitly:

$$\widetilde{f}(\boldsymbol{x}^*) = \frac{\omega_0}{\gamma_0}\mathbf{r}_{(0)}^T\mathbf{g}_{(0)} + \sum_{k=1}^{5}\left(\omega_k - \omega_0\frac{\gamma_k}{\gamma_0}\right)\mathbf{r}_{(k)}^T\mathbf{g}_{(k)}$$

WENO-AO

Generally, no linear weights, γ_k, exist that can reproduce the accuracy of the full stencil.

Adaptive order WENO

Let S_0 correspond to the full stencil, and include it explicitly:

$$\widetilde{f}(\boldsymbol{x}^*) = \frac{\omega_0}{\gamma_0}\mathbf{r}_{(0)}^T\mathbf{g}_{(0)} + \sum_{k=1}^{5}\left(\omega_k - \omega_0\frac{\gamma_k}{\gamma_0}\right)\mathbf{r}_{(k)}^T\mathbf{g}_{(k)}$$

Now we can choose γ_k solely to ensure stability, e.g.

$$\begin{aligned}\gamma_0 &= C_h,\\ \gamma_1 &= (1-C_h)C_l,\\ \gamma_2 = \gamma_3 = \gamma_4 = \gamma_5 &= \frac{(1-C_h)*(1-C_l)}{4},\end{aligned}$$

where $0 < C_h, C_l < 1$, e.g. $C_h = C_l = 0.8$.

Smoothness indicators
The last numerical ingredient

The smoothness of the solution on each substencil can be measured by

$$\beta_k = \sum_{|\alpha|=1}^{2} \sum_q w_q \left(\left. \frac{\partial^{|\alpha|} \widetilde{f}_k}{\partial \boldsymbol{x}^\alpha} \right|_{\boldsymbol{x}_q} \right)^2,$$

Then nonlinear weights are formed using a modified WENO-Z scheme

$$\tau = \left| \beta_0 - \frac{1}{4} \sum_{k=2}^{5} \beta_k \right|$$

$$\widetilde{\omega}_k = \gamma_k \left(1 + \left(\frac{\tau}{\beta_k + \epsilon} \right)^p \right)$$

$$\omega_k = \frac{\widetilde{\omega}_k}{\sum \widetilde{\omega}_k}$$

Smoothness indicators

The last numerical ingredient

The smoothness of the solution on each substencil can be measured by

$$\beta_k = \sum_{|\alpha|=1}^{2} \sum_{q} w_q \left(\left. \frac{\partial^{|\alpha|} \widetilde{f}_k}{\partial \boldsymbol{x}^\alpha} \right|_{\boldsymbol{x}_q} \right)^2 ,$$

Then nonlinear weights are formed using a modified WENO-Z scheme

$$\tau = \left| \beta_0 - \frac{1}{4} \sum_{k=2}^{5} \beta_k \right|$$

$$\widetilde{\omega}_k = \gamma_k \left(1 + \left(\frac{\tau}{\beta_k + \epsilon} \right)^p \right)$$

$$\omega_k = \frac{\widetilde{\omega}_k}{\sum \widetilde{\omega}_k}$$

Observation: Mixed derivatives can be safely ignored.

Reconstruction Variables
Linearized primitive variables

- The reconstruction scheme is defined for scalar data
- Must do reconstruction *componentwise*
- Reconstructing conservative variables directly is a bad idea

Note: Any linear combination of cell-averages is still a cell-average.

Reconstruction Variables
Linearized primitive variables

- The reconstruction scheme is defined for scalar data
- Must do reconstruction *componentwise*
- Reconstructing conservative variables directly is a bad idea

Note: Any linear combination of cell-averages is still a cell-average.

Generic transformation

Fix some matrix Φ and set

$$\mathbf{W}_{i,j} = \mathbf{\Phi}\mathbf{U}_{i,j}, \quad \forall (i,j) \in \mathcal{S}_0,$$

reconstruct over $\mathbf{W}$ componentwise and map back with $\mathbf{\Phi}^*$.

Reconstruction Variables
Linearized primitive variables

- The reconstruction scheme is defined for scalar data
- Must do reconstruction *componentwise*
- Reconstructing conservative variables directly is a bad idea

Note: Any linear combination of cell-averages is still a cell-average.

Generic transformation
Fix some matrix Φ and set

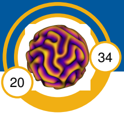
$$\mathbf{W}_{i,j} = \mathbf{\Phi}\mathbf{U}_{i,j}, \quad \forall(i,j) \in \mathcal{S}_0,$$

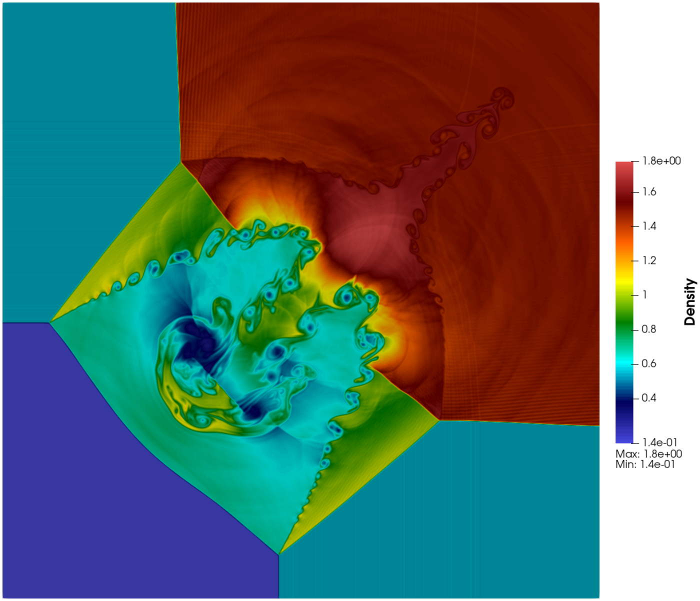
reconstruct over $\mathbf{W}$ componentwise and map back with $\mathbf{\Phi}^*$.

Specific transformations
- Decompose $\frac{\partial \mathbf{F}}{\partial \mathbf{U}}\Big|_{\widetilde{\mathbf{U}}} = \mathbf{R}\mathbf{\Lambda}\mathbf{L}$, set $\mathbf{\Phi} = \mathbf{L}$ and $\mathbf{\Phi}^{-1} = \mathbf{R}$
  - These are characteristic variables (direction dependent!)
- Set $\mathbf{\Phi} = \frac{\partial \mathbf{V}}{\partial \mathbf{U}}\Big|_{\widetilde{\mathbf{U}}}$ and $\mathbf{\Phi}^{-1} = \frac{\partial \mathbf{U}}{\partial \mathbf{V}}\Big|_{\mathbf{V}(\widetilde{\mathbf{U}})}$
  - We've dubbed these *linearized primitive* variables

2D Riemann problem configuration 3

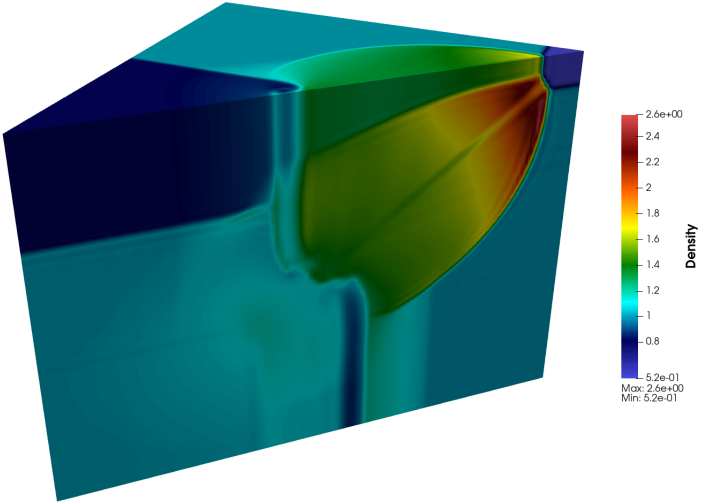
1024×1024, Radius 2, $\ell = 24\Delta$





3D Riemann problem configuration 1

$128 \times 128 \times 128$, Radius 2, $\ell = 24\Delta$



Positivity Preservation
When WENO fails

- Strong shocks can still generate negative densities/pressures
- Putting hard floors on density/pressure generally fails
- Only recourse is to mix in a first-order correction

Positivity Preservation
When WENO fails

- Strong shocks can still generate negative densities/pressures
- Putting hard floors on density/pressure generally fails
- Only recourse is to mix in a first-order correction

Applying corrections

Replace Riemann states as $\mathbf{U}^* \leftarrow \mathbf{U}_{ctr} + \theta\left(\mathbf{U}^* - \mathbf{U}_{ctr}\right)$

- $\theta \in [0, 1]$, $\theta = 1$ for no correction
- Need to find largest θ that gives a valid state
- Must use the same θ for *all* Riemann states on a cell
- Trivial to make density valid, trickier for pressure
  - Ultimately θ will be a root of a quadratic with messy coefficients
- Set bounds on density and pressure using Balsara's self-adjusting limiter

Overview of RHS evaluation

Method-of-lines integrator

Evaluate all spatial terms on current state

1. Fill ghost cells

Overview of RHS evaluation
Method-of-lines integrator

Evaluate all spatial terms on current state

1. Fill ghost cells
2. Reconstruct Riemann states
   - Convert values in stencil to reconstruction variables
   - Evaluate smoothness indicators
   - Form nonlinear weights
   - Form face values
   - Convert back from reconstruction variables
   - Enforce positivity

Overview of RHS evaluation

Method-of-lines integrator

Evaluate all spatial terms on current state

1. Fill ghost cells
2. Reconstruct Riemann states
    - Convert values in stencil to reconstruction variables
    - Evaluate smoothness indicators
    - Form nonlinear weights
    - Form face values
    - Convert back from reconstruction variables
    - Enforce positivity
3. Fill ghost Riemann states

Overview of RHS evaluation

Method-of-lines integrator

Evaluate all spatial terms on current state

1. Fill ghost cells
2. Reconstruct Riemann states
    - Convert values in stencil to reconstruction variables
    - Evaluate smoothness indicators
    - Form nonlinear weights
    - Form face values
    - Convert back from reconstruction variables
    - Enforce positivity
3. Fill ghost Riemann states
4. Call Riemann solver
    - Track largest signal velocity for time stepper

Overview of RHS evaluation

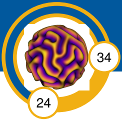
Method-of-lines integrator

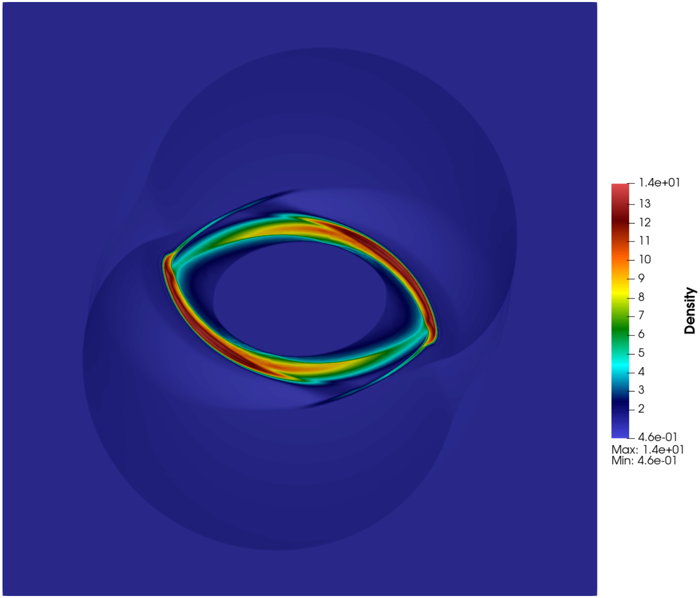
Evaluate all spatial terms on current state

1. Fill ghost cells
2. Reconstruct Riemann states
   - Convert values in stencil to reconstruction variables
   - Evaluate smoothness indicators
   - Form nonlinear weights
   - Form face values
   - Convert back from reconstruction variables
   - Enforce positivity
3. Fill ghost Riemann states
4. Call Riemann solver
   - Track largest signal velocity for time stepper
5. Integrate fluxes and accumulate into RHS
6. Accumulate source terms into RHS

MHD Rotor

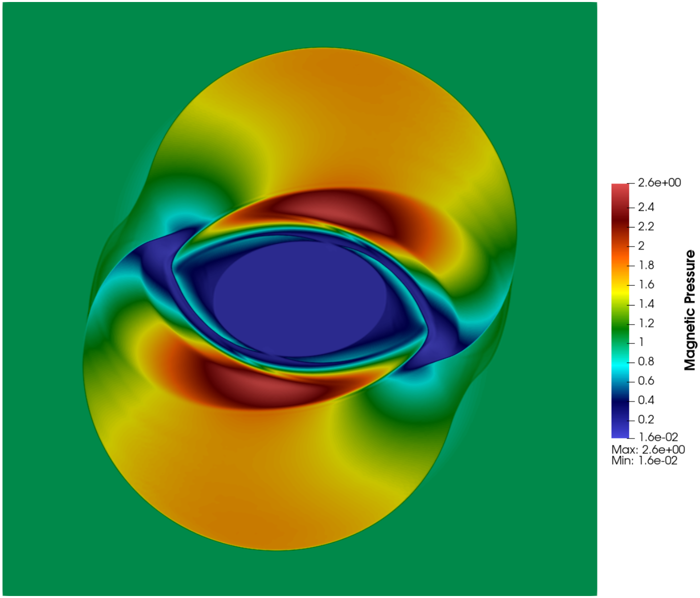
1024×1024, Radius 2, $\ell = 24\Delta$





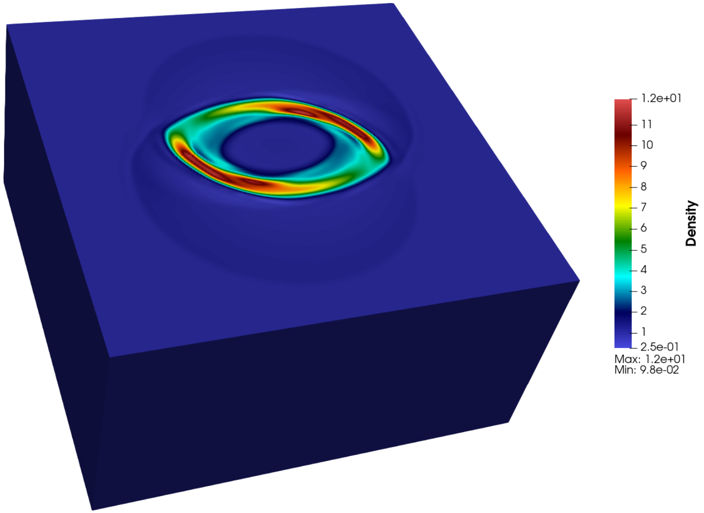
MHD Rotor

1024×1024, Radius 2, $\ell = 24\Delta$



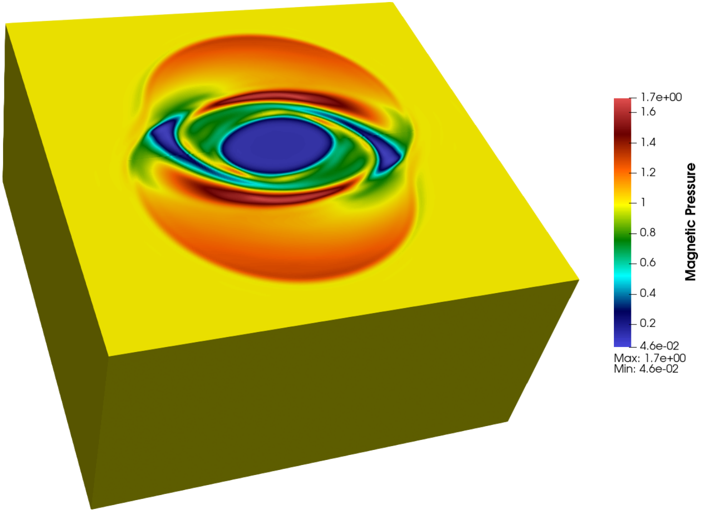
MHD Rotor

$192 \times 192 \times 192$, Radius 2, $\ell = 24\Delta$



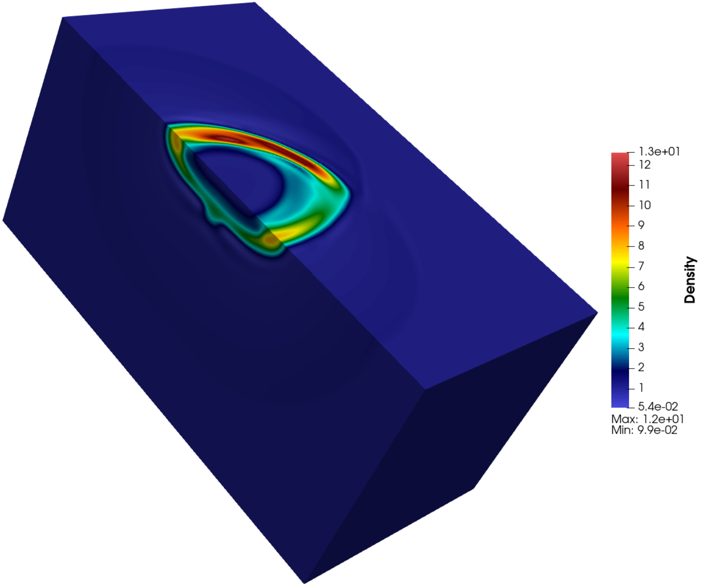
MHD Rotor

$192 \times 192 \times 192$, Radius 2, $\ell = 24\Delta$



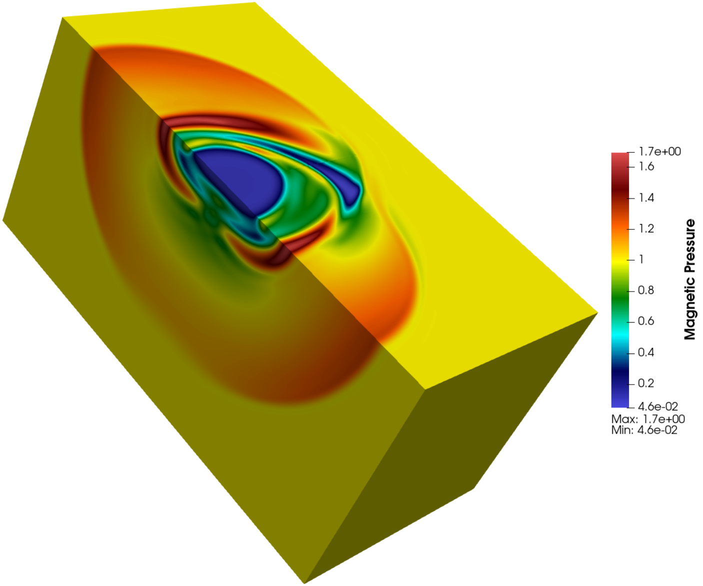
MHD Rotor

$192 \times 192 \times 192$, Radius 2, $\ell = 24\Delta$



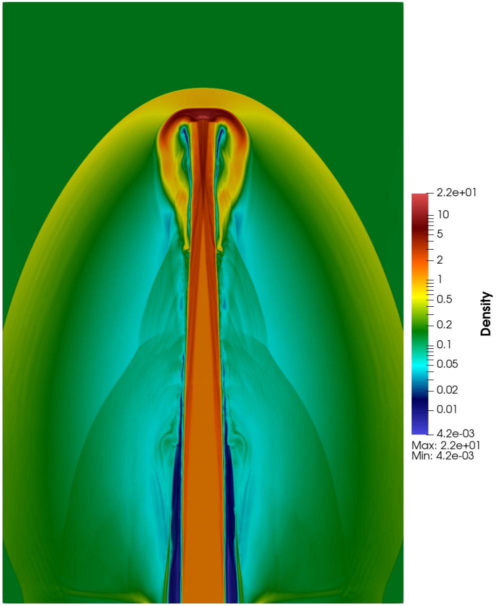
MHD Rotor

$192 \times 192 \times 192$, Radius 2, $\ell = 24\Delta$



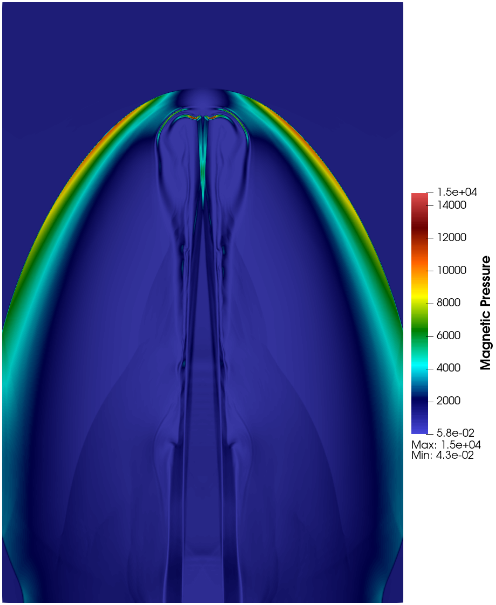
Magnetized Astrophysical Jet

1024×1536, Radius 2, $\ell = 24\Delta$



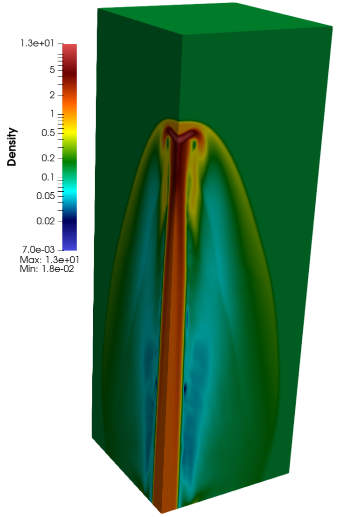
Magnetized Astrophysical Jet

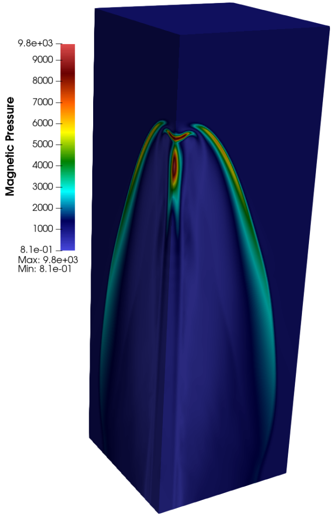
1024×1536, Radius 2, $\ell = 24\Delta$



Magnetized Astrophysical Jet

$128 \times 384 \times 128$, Radius 2, $\ell = 24\Delta$





Expensive parts of RHS evaluation

Evaluate all spatial terms on current state

1. Fill ghost cells
2. **Reconstruct Riemann states**
   - **Convert values in stencil to reconstruction variables**
   - **Evaluate smoothness indicators**
   - Form nonlinear weights
   - Form face values
   - **Convert back from reconstruction variables**
   - Enforce positivity
3. Fill ghost Riemann states
4. Call Riemann solver
   - Track largest signal velocity for time stepper
5. Integrate fluxes and accumulate into RHS
6. Accumulate source terms into RHS

A cheap local smoothness indicator

- Nonlinear reconstruction is only really needed near shocks
- Not necessary in smooth regions
- Seemingly unnecessary even for contact discontinuities

A cheap local smoothness indicator

- Nonlinear reconstruction is only really needed near shocks
- Not necessary in smooth regions
- Seemingly unnecessary even for contact discontinuities

WENO is mostly independent of the system we are solving.
Can the physics furnish a cheap smoothness indicator?

A cheap local smoothness indicator

- Nonlinear reconstruction is only really needed near shocks
- Not necessary in smooth regions
- Seemingly unnecessary even for contact discontinuities

WENO is mostly independent of the system we are solving.
Can the physics furnish a cheap smoothness indicator?

Large time derivatives of pressure indicate shocks

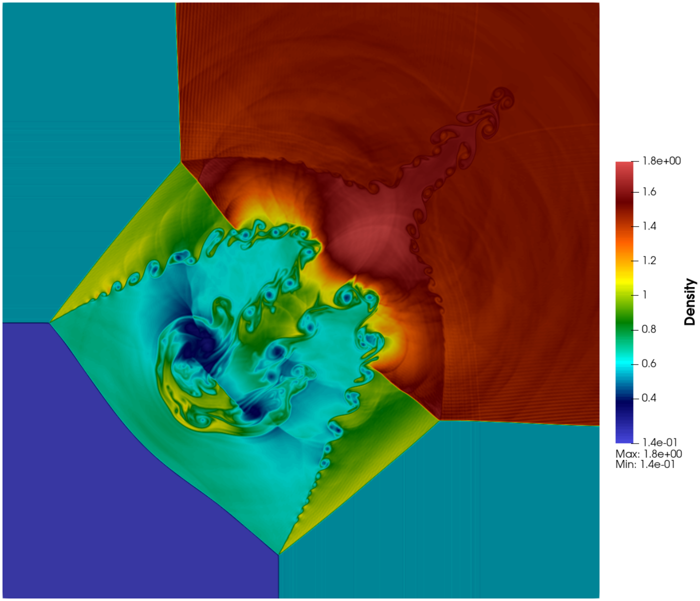
Only do WENO if:

$$\frac{\left|p^{(n)} - p^{(n-1)}\right|}{\min\left\{p^{(n)}, p^{(n-1)}\right\}} > C\Delta t,$$

for some C, typically $C \approx 10$. (Adapted from Alina Chertok's LSI)

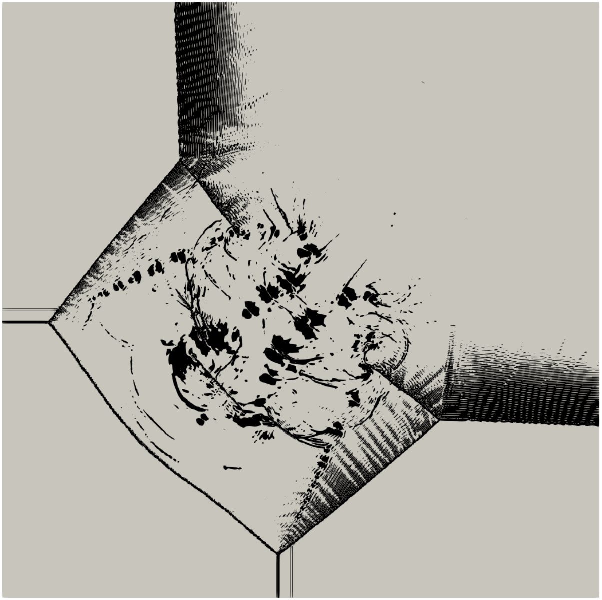
2D Riemann problem configuration 3

1024×1024, Radius 2, $\ell = 24\Delta$



2D Riemann problem configuration 3

WENO Map for $C = 10$



Accelerators and Heterogeneous Computing

The vast majority of the work to be done is the same for each cell.
$\Rightarrow$ Huge amounts of data parallelism.

Accelerators and Heterogeneous Computing

The vast majority of the work to be done is the same for each cell.
$\Rightarrow$ Huge amounts of data parallelism.

Difficulties with heterogeneous architectures

- Cache misses and memory latency are huge problem for all hardware choices
- Different hardware perform best with different layouts of data in memory
- Supporting a variety of hardware is headache

Accelerators and Heterogeneous Computing

The vast majority of the work to be done is the same for each cell.
$\Rightarrow$ Huge amounts of data parallelism.

Difficulties with heterogeneous architectures

- Cache misses and memory latency are huge problem for all hardware choices
- Different hardware perform best with different layouts of data in memory
- Supporting a variety of hardware is headache

```
for (int i=0; i<nX; i++) {
  for (int j=0; j<nY; j++) {
    uNew(i,j) = f(uOld(i,j),
                  uOld(i+1,j),uOld(i-1,j),
                  uOld(i,j+1),uOld(i,j-1));
  }
}
```

Kokkos

Performance portability

- Data layout and parallel hardware are tightly coupled
- Provide a simple abstraction for launching parallel work
- Provide data structures that conform to the chosen architecture
  - These also need to account for *where* data is stored

Kokkos

Performance portability

- Data layout and parallel hardware are tightly coupled
- Provide a simple abstraction for launching parallel work
- Provide data structures that conform to the chosen architecture
  - These also need to account for *where* data is stored

```
auto rng = MDRange<Cuda,Rank<2>>({0,0},{nX,nY});
parallel_for(``Kernel Name'',rng,
  KOKKOS_LAMBDA (const int i,const int j) {
    uNew(i,j) = f(uOld(i,j),
                   uOld(i+1,j),uOld(i-1,j),
                   uOld(i,j+1),uOld(i,j-1));
  });
```

Distributed memory parallelism

Communication points

Evaluate all spatial terms on current state

1. **Fill ghost cells**
2. Reconstruct Riemann states
   - Convert values in stencil to reconstruction variables
   - Evaluate smoothness indicators
   - Form nonlinear weights
   - Form face values
   - Convert back from reconstruction variables
   - Enforce positivity
3. **Fill ghost Riemann states**
4. Call Riemann solver
   - **Track largest signal velocity for time stepper**
5. Integrate fluxes and accumulate into RHS
6. Accumulate source terms into RHS

Final thoughts

Conclusion

- Kernel based reconstruction is very flexible
  - Easy to go straight from averages to point values
- There are many choices for reconstruction variables
- Effective positivity preservation is reasonably straightforward
- Kokkos provides an excellent way to parallelize code

Final thoughts

Conclusion

- Kernel based reconstruction is very flexible
  - Easy to go straight from averages to point values
- There are many choices for reconstruction variables
- Effective positivity preservation is reasonably straightforward
- Kokkos provides an excellent way to parallelize code

Next steps

- Evaluate MHD accuracy fully
- Continue on relativistic hydrodynamics
- Investigate HWENO methods
- Extend to AMR
- Incorporate physical viscosity

The isentropic vortex problem

A truly nonlinear benchmark problem

The Euler equations on $[-L,L]^2$ with periodic boundaries and initial condition

$$\begin{pmatrix}\rho\\ u\\ v\\ p\end{pmatrix} = \begin{pmatrix}T^{1/(\gamma-1)}\\ 1-y\omega\\ 1+x\omega\\ T^{\gamma/(\gamma-1)}\end{pmatrix}$$

$$T = 1 - \frac{\gamma-1}{8\gamma\pi^2}e^{1-x^2-y^2}$$

$$\omega = \frac{1}{2\pi}e^{(1-x^2-y^2)/2}$$

recover the initial condition at time $T_f = 2L$

The isentropic vortex problem

$\Omega = [-10, 10]^2$, $\ell = 2$

Grid	L_1 Error	L_1 Order	L_∞ Error	L_∞ Order
		$R = 2$		
50^2	$1.43e-1$	–	$2.29e-2$	–
100^2	$1.49e-2$	**3.27**	$4.49e-3$	**2.35**
200^2	$6.20e-4$	**4.58**	$9.52e-5$	**5.56**
400^2	$2.04e-5$	**4.93**	$3.24e-6$	**4.88**
		$R = 3$		
50^2	$8.37e-2$	–	$1.89e-2$	–
100^2	$2.36e-3$	**5.15**	$3.13e-4$	**5.91**
200^2	$3.18e-5$	**6.21**	$1.06e-5$	**4.89**
400^2	$2.72e-7$	**6.87**	$9.58e-8$	**6.78**
		$R = 4$		
50^2	$4.42e-2$	–	$9.31e-3$	–
100^2	$6.94e-4$	**5.99**	$2.34e-4$	**5.31**
200^2	$2.53e-6$	**8.10**	$1.11e-6$	**7.72**
400^2	$5.70e-9$	**8.80**	$2.30e-9$	**8.92**

Systems of interest

Special relativistic Euler equations

$$\mathbf{U} = \begin{pmatrix} D \\ S_x \\ S_y \\ S_z \\ \tau \end{pmatrix} = \begin{pmatrix} \rho W \\ \rho h W^2 u \\ \rho h W^2 v \\ \rho h W^2 w \\ \rho W (hW - 1) - p \end{pmatrix} \qquad \mathbf{F} = \begin{pmatrix} Du \\ S_x u + p \\ S_x v \\ S_x w \\ S_x - Du \end{pmatrix},$$

where,

$$\begin{aligned} W &= (1 - \mathbf{v} \cdot \mathbf{v})^{-1/2} \\ h &= 1 + \gamma e \\ e &= \frac{p}{(\gamma - 1)\rho} \end{aligned}$$

Stabilizing large ℓ

We need to compute

$$\mathbf{z}^T = \mathbf{w}^T \mathbf{C}^{(-1)},$$

where $\mathbf{C}$ and $\mathbf{w}$ both depend on ℓ.

- Large values of ℓ tend to give more accurate interpolants
- Large values of ℓ give horribly conditioned linear systems

Stable evaluation of prediction vectors

Consider $\epsilon = \ell^{-1}$, and allow complex ϵ. Then

- $z_i(\ell^{-1}) = \mathbf{w}^T \mathbf{C}^{(-1)} \mathbf{e}_i$ is holomorphic apart from isolated poles
- Evaluate $z_i(\ell^{-1})$ on a circle in $\mathbb{C}$ where computation is stable
- Back out an approximate Laurent expansion of $z_i(\ell^{-1})$
- Evaluate that Laurent expansion at the real $\epsilon = \ell^{-1}$ of interest