Lecture 3

Hyperbolic AMROC solvers

Course *Block-structured Adaptive Finite Volume Methods in C++*

Ralf Deiterding
University of Southampton
Engineering and the Environment
Highfield Campus, Southampton SO17 1BJ, UK

E-mail: r.deiterding@soton.ac.uk

Outline

Outline

Outline

Outline

Outline

Outline

High-resolution methods

Objective: Higher-order accuracy in smooth solution regions but no spurious oscillations near large gradients

Consistent monotone methods converge toward the entropy solution, but

High-resolution methods

Objective: Higher-order accuracy in smooth solution regions but no spurious oscillations near large gradients
Consistent monotone methods converge toward the entropy solution, but

Theorem

A monotone method is at most first order accurate.

Proof: [Harten et al., 1976]

High-resolution methods

Objective: Higher-order accuracy in smooth solution regions but no spurious oscillations near large gradients
Consistent monotone methods converge toward the entropy solution, but

Theorem

A monotone method is at most first order accurate.

Proof: [Harten et al., 1976]

Definition (TVD property)

Scheme $\mathcal{H}^{(\Delta t)}(\mathbf{Q}^n;j)$ TVD if $TV(\mathbf{Q}^{l+1}) \le TV(\mathbf{Q}^l)$ is satisfied for all discrete sequences $\mathbf{Q}^n$. Herein, $TV(\mathbf{Q}^l) := \sum_{j\in\mathbb{Z}} |\mathbf{Q}^l_{j+1} - \mathbf{Q}^l_j|$.

TVD schemes: no new extrema, local minima are non-decreasing, local maxima are non-increasing (termed *monotonicity-preserving*). *Monotonicity-preserving* higher-order schemes are at least 5-point methods. Proofs: [Harten, 1983]

High-resolution methods

Objective: Higher-order accuracy in smooth solution regions but no spurious oscillations near large gradients

Consistent monotone methods converge toward the entropy solution, but

Theorem

A monotone method is at most first order accurate.

Proof: [Harten et al., 1976]

Definition (TVD property)

Scheme $\mathcal{H}^{(\Delta t)}(\mathbf{Q}^n; j)$ TVD if $TV(\mathbf{Q}^{l+1}) \le TV(\mathbf{Q}^l)$ is satisfied for all discrete sequences $\mathbf{Q}^n$. Herein, $TV(\mathbf{Q}^l) := \sum_{j\in\mathbb{Z}} |\mathbf{Q}^l_{j+1} - \mathbf{Q}^l_j|$.

TVD schemes: no new extrema, local minima are non-decreasing, local maxima are non-increasing (termed *monotonicity-preserving*). *Monotonicity-preserving* higher-order schemes are at least 5-point methods. Proofs: [Harten, 1983]

TVD concept is proven [Godlewski and Raviart, 1996] for scalar schemes only but nevertheless used to construct *high resolution* schemes.

Monotonicity-preserving scheme can converge toward non-physical weak solutions.

MUSCL slope limiting

Monotone Upwind Schemes for Conservation Laws [van Leer, 1979]

$$\tilde{Q}^L_{j+\frac{1}{2}} = Q^n_j + \frac{1}{4}\left[(1-\omega)\,\Phi^+_{j-\frac{1}{2}}\Delta_{j-\frac{1}{2}} + (1+\omega)\,\Phi^-_{j+\frac{1}{2}}\Delta_{j+\frac{1}{2}}\right] \;,$$

$$\tilde{Q}^R_{j-\frac{1}{2}} = Q^n_j - \frac{1}{4}\left[(1-\omega)\,\Phi^-_{j+\frac{1}{2}}\Delta_{j+\frac{1}{2}} + (1+\omega)\,\Phi^+_{j-\frac{1}{2}}\Delta_{j-\frac{1}{2}}\right]$$

with $\Delta_{j-1/2} = Q^n_j - Q^n_{j-1}$, $\Delta_{j+1/2} = Q^n_{j+1} - Q^n_j$.

MUSCL slope limiting

Monotone Upwind Schemes for Conservation Laws [van Leer, 1979]

$$\tilde{Q}^L_{j+\frac{1}{2}} = Q_j^n + \frac{1}{4}\left[(1-\omega)\,\Phi^+_{j-\frac{1}{2}}\Delta_{j-\frac{1}{2}} + (1+\omega)\,\Phi^-_{j+\frac{1}{2}}\Delta_{j+\frac{1}{2}}\right] \;,$$
$$\tilde{Q}^R_{j-\frac{1}{2}} = Q_j^n - \frac{1}{4}\left[(1-\omega)\,\Phi^-_{j+\frac{1}{2}}\Delta_{j+\frac{1}{2}} + (1+\omega)\,\Phi^+_{j-\frac{1}{2}}\Delta_{j-\frac{1}{2}}\right]$$

with $\Delta_{j-1/2} = Q_j^n - Q_{j-1}^n$, $\Delta_{j+1/2} = Q_{j+1}^n - Q_j^n$.

$$\Phi^+_{j-\frac{1}{2}} := \Phi\left(r^+_{j-\frac{1}{2}}\right) \;, \quad \Phi^-_{j+\frac{1}{2}} := \Phi\left(r^-_{j+\frac{1}{2}}\right) \quad \text{with} \quad r^+_{j-\frac{1}{2}} := \frac{\Delta_{j+\frac{1}{2}}}{\Delta_{j-\frac{1}{2}}} \;, \quad r^-_{j+\frac{1}{2}} := \frac{\Delta_{j-\frac{1}{2}}}{\Delta_{j+\frac{1}{2}}}$$

and *slope limiters*, e.g., *Minmod*

$$\Phi(r) = \max(0, \min(r, 1))$$

MUSCL slope limiting

Monotone Upwind Schemes for Conservation Laws [van Leer, 1979]

$$\tilde{Q}^L_{j+\frac{1}{2}} = Q^n_j + \frac{1}{4}\left[(1-\omega)\,\Phi^+_{j-\frac{1}{2}}\Delta_{j-\frac{1}{2}} + (1+\omega)\,\Phi^-_{j+\frac{1}{2}}\Delta_{j+\frac{1}{2}}\right] \;,$$

$$\tilde{Q}^R_{j-\frac{1}{2}} = Q^n_j - \frac{1}{4}\left[(1-\omega)\,\Phi^-_{j+\frac{1}{2}}\Delta_{j+\frac{1}{2}} + (1+\omega)\,\Phi^+_{j-\frac{1}{2}}\Delta_{j-\frac{1}{2}}\right]$$

with $\Delta_{j-1/2} = Q^n_j - Q^n_{j-1}$, $\Delta_{j+1/2} = Q^n_{j+1} - Q^n_j$.

$$\Phi^+_{j-\frac{1}{2}} := \Phi\left(r^+_{j-\frac{1}{2}}\right) \;, \quad \Phi^-_{j+\frac{1}{2}} := \Phi\left(r^-_{j+\frac{1}{2}}\right) \quad \text{with} \quad r^+_{j-\frac{1}{2}} := \frac{\Delta_{j+\frac{1}{2}}}{\Delta_{j-\frac{1}{2}}} \;, \quad r^-_{j+\frac{1}{2}} := \frac{\Delta_{j-\frac{1}{2}}}{\Delta_{j+\frac{1}{2}}}$$

and *slope limiters*, e.g., *Minmod*

$$\Phi(r) = \max(0, \min(r, 1))$$

Using a midpoint rule for temporal integration, e.g.,

$$Q^\star_j = Q^n_j - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(F(Q^n_{j+1}, Q^n_j) - F(Q^n_j, Q^n_{j-1})\right)$$

and constructing limited values from $Q^\star$ to be used in FV scheme gives a TVD method if

$$\frac{1}{2}\left[(1-\omega)\Phi(r) + (1+\omega)\, r\,\Phi\left(\frac{1}{r}\right)\right] < \min(2, 2r)$$

is satisfied for $r > 0$. Proof: [Hirsch, 1988]

Wave Propagation with flux limiting

Wave Propagation Method [LeVeque, 1997] is built on the flux differencing approach $\mathcal{A}^{\pm}\Delta := \hat{\mathbf{A}}^{\pm}(\mathbf{q}_L, \mathbf{q}_R)\Delta\mathbf{q}$ and the *waves* $\mathcal{W}_m := a_m\hat{\mathbf{r}}_m$, i.e.

$$\mathcal{A}^{-}\Delta\mathbf{q} = \sum_{\hat{\lambda}_m<0} \hat{\lambda}_m \mathcal{W}_m \;, \quad \mathcal{A}^{+}\Delta\mathbf{q} = \sum_{\hat{\lambda}_m\ge 0} \hat{\lambda}_m \mathcal{W}_m$$

Wave Propagation with flux limiting

Wave Propagation Method [LeVeque, 1997] is built on the flux differencing approach $\mathcal{A}^{\pm}\Delta := \hat{\mathbf{A}}^{\pm}(\mathbf{q}_L, \mathbf{q}_R)\Delta\mathbf{q}$ and the *waves* $\mathcal{W}_m := a_m \hat{\mathbf{r}}_m$, i.e.

$$\mathcal{A}^{-}\Delta\mathbf{q} = \sum_{\hat{\lambda}_m < 0} \hat{\lambda}_m \mathcal{W}_m \;, \quad \mathcal{A}^{+}\Delta\mathbf{q} = \sum_{\hat{\lambda}_m \ge 0} \hat{\lambda}_m \mathcal{W}_m$$

Wave Propagation 1D:

$$\mathbf{Q}^{n+1} = \mathbf{Q}_j^n - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^{-}\Delta_{j+\frac{1}{2}} + \mathcal{A}^{+}\Delta_{j-\frac{1}{2}}\right) - \frac{\Delta t}{\Delta x}\left(\tilde{\mathbf{F}}_{j+\frac{1}{2}} - \tilde{\mathbf{F}}_{j-\frac{1}{2}}\right)$$

with

$$\tilde{\mathbf{F}}_{j+\frac{1}{2}} = \frac{1}{2}\,|\mathcal{A}|\left(1 - \frac{\Delta t}{\Delta x}\,|\mathcal{A}|\right)\Delta_{j+\frac{1}{2}} = \frac{1}{2}\sum_{m=1}^{M} |\hat{\lambda}^m_{j+\frac{1}{2}}| \left(1 - \frac{\Delta t}{\Delta x}|\hat{\lambda}^m_{j+\frac{1}{2}}|\right) \tilde{\mathcal{W}}^m_{j+\frac{1}{2}}$$

Wave Propagation with flux limiting

Wave Propagation Method [LeVeque, 1997] is built on the flux differencing approach $\mathcal{A}^{\pm}\Delta := \hat{\mathbf{A}}^{\pm}(\mathbf{q}_L, \mathbf{q}_R)\Delta\mathbf{q}$ and the *waves* $\mathcal{W}_m := a_m \hat{\mathbf{r}}_m$, i.e.

$$\mathcal{A}^{-}\Delta\mathbf{q} = \sum_{\hat{\lambda}_m<0} \hat{\lambda}_m \mathcal{W}_m\,, \quad \mathcal{A}^{+}\Delta\mathbf{q} = \sum_{\hat{\lambda}_m\geq 0} \hat{\lambda}_m \mathcal{W}_m$$

Wave Propagation 1D:

$$\mathbf{Q}^{n+1} = \mathbf{Q}_j^n - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^{-}\Delta_{j+\frac{1}{2}} + \mathcal{A}^{+}\Delta_{j-\frac{1}{2}}\right) - \frac{\Delta t}{\Delta x}\left(\tilde{\mathbf{F}}_{j+\frac{1}{2}} - \tilde{\mathbf{F}}_{j-\frac{1}{2}}\right)$$

with

$$\tilde{\mathbf{F}}_{j+\frac{1}{2}} = \frac{1}{2}\left|\mathcal{A}\right|\left(1 - \frac{\Delta t}{\Delta x}\left|\mathcal{A}\right|\right)\Delta_{j+\frac{1}{2}} = \frac{1}{2}\sum_{m=1}^{M}\left|\hat{\lambda}^m_{j+\frac{1}{2}}\right|\left(1 - \frac{\Delta t}{\Delta x}\left|\hat{\lambda}^m_{j+\frac{1}{2}}\right|\right)\tilde{\mathcal{W}}^m_{j+\frac{1}{2}}$$

and wave limiter

$$\tilde{\mathcal{W}}^m_{j+\frac{1}{2}} = \Phi(\Theta^m_{j+\frac{1}{2}})\,\mathcal{W}^m_{j+\frac{1}{2}}$$

with

$$\Theta^m_{j+\frac{1}{2}} = \begin{cases} a^m_{j-\frac{1}{2}}/a^m_{j+\frac{1}{2}}\,, & \hat{\lambda}^m_{j+\frac{1}{2}} \geq 0\,, \\ a^m_{j+\frac{3}{2}}/a^m_{j+\frac{1}{2}}\,, & \hat{\lambda}^m_{j+\frac{1}{2}} < 0 \end{cases}$$

Wave Propagation Method in 2D

Writing $\tilde{\mathcal{A}}^\pm\Delta_{j\pm1/2} := \mathcal{A}^+\Delta_{j\pm1/2} + \tilde{\mathbf{F}}_{j\pm1/2}$ one can develop a truly two-dimensional one-step method [Langseth and LeVeque, 2000]

$$
\begin{aligned}
\mathbf{Q}_{jk}^{n+1} = \mathbf{Q}_{jk}^n - \frac{\Delta t}{\Delta x_1}\Bigg(&\tilde{\mathcal{A}}^-\Delta_{j+\frac12,k} - \frac12\frac{\Delta t}{\Delta x_2}\left[\mathcal{A}^-\tilde{\mathcal{B}}^-\Delta_{j+1,k+\frac12} + \mathcal{A}^-\tilde{\mathcal{B}}^+\Delta_{j+1,k-\frac12}\right] + \\
&\tilde{\mathcal{A}}^+\Delta_{j-\frac12,k} - \frac12\frac{\Delta t}{\Delta x_2}\left[\mathcal{A}^+\tilde{\mathcal{B}}^-\Delta_{j-1,k+\frac12} + \mathcal{A}^+\tilde{\mathcal{B}}^+\Delta_{j-1,k-\frac12}\right]\Bigg) \\
- \frac{\Delta t}{\Delta x_2}\Bigg(&\tilde{\mathcal{B}}^-\Delta_{j,k+\frac12} - \frac12\frac{\Delta t}{\Delta x_1}\left[\mathcal{B}^-\tilde{\mathcal{A}}^-\Delta_{j+\frac12,k+1} + \mathcal{B}^-\tilde{\mathcal{A}}^+\Delta_{j-\frac12,k+1}\right] + \\
&\tilde{\mathcal{B}}^+\Delta_{j,k-\frac12} - \frac12\frac{\Delta t}{\Delta x_1}\left[\mathcal{B}^+\tilde{\mathcal{A}}^-\Delta_{j+\frac12,k-1} + \mathcal{B}^+\tilde{\mathcal{A}}^+\Delta_{j-\frac12,k-1}\right]\Bigg)
\end{aligned}
$$

that is stable for

$$
\left\{\max_{j\in\mathbb{Z}}|\hat{\lambda}_{m,j+\frac12}|\frac{\Delta t}{\Delta x_1}, \max_{k\in\mathbb{Z}}|\hat{\lambda}_{m,k+\frac12}|\frac{\Delta t}{\Delta x_2}\right\} \le 1\,, \quad \text{for all } m = 1,\ldots,M
$$

Further high-resolution methods

Some further high-resolution methods (good overview in [Laney, 1998]):

- FCT: 2nd order [Oran and Boris, 2001]

Further high-resolution methods

Some further high-resolution methods (good overview in [Laney, 1998]):

- FCT: 2nd order [Oran and Boris, 2001]
- ENO/WENO: 3rd order [Shu, 97]

Further high-resolution methods

Some further high-resolution methods (good overview in [Laney, 1998]):

- FCT: 2nd order [Oran and Boris, 2001]
- ENO/WENO: 3rd order [Shu, 97]
- PPM: 3rd order [Colella and Woodward, 1984]

Further high-resolution methods

Some further high-resolution methods (good overview in [Laney, 1998]):

- FCT: 2nd order [Oran and Boris, 2001]
- ENO/WENO: 3rd order [Shu, 97]
- PPM: 3rd order [Colella and Woodward, 1984]

Further high-resolution methods

Some further high-resolution methods (good overview in [Laney, 1998]):

- FCT: 2nd order [Oran and Boris, 2001]
- ENO/WENO: 3rd order [Shu, 97]
- PPM: 3rd order [Colella and Woodward, 1984]

3rd order methods must make use of strong-stability preserving Runge-Kutta methods [Gottlieb et al., 2001] for time integration that use a multi-step update

$$\tilde{\mathbf{Q}}_j^{\upsilon} = \alpha_{\upsilon}\mathbf{Q}_j^n + \beta_{\upsilon}\tilde{\mathbf{Q}}_j^{\upsilon-1} + \gamma_{\upsilon}\frac{\Delta t}{\Delta x}\left(\mathbf{F}_{j+\frac{1}{2}}(\tilde{\mathbf{Q}}^{\upsilon-1}) - \mathbf{F}_{j-\frac{1}{2}}(\tilde{\mathbf{Q}}^{\upsilon-1})\right)$$

with $\tilde{\mathbf{Q}}^0 := \mathbf{Q}^n$, $\alpha_1 = 1$, $\beta_1 = 0$; and $\mathbf{Q}^{n+1} := \tilde{\mathbf{Q}}^{\Upsilon}$ after final stage Υ

Further high-resolution methods

Some further high-resolution methods (good overview in [Laney, 1998]):

- FCT: 2nd order [Oran and Boris, 2001]
- ENO/WENO: 3rd order [Shu, 97]
- PPM: 3rd order [Colella and Woodward, 1984]

3rd order methods must make use of strong-stability preserving Runge-Kutta methods [Gottlieb et al., 2001] for time integration that use a multi-step update

$$\tilde{\mathbf{Q}}_j^{\upsilon} = \alpha_\upsilon \mathbf{Q}_j^n + \beta_\upsilon \tilde{\mathbf{Q}}_j^{\upsilon-1} + \gamma_\upsilon \frac{\Delta t}{\Delta x} \left(\mathbf{F}_{j+\frac{1}{2}}(\tilde{\mathbf{Q}}^{\upsilon-1}) - \mathbf{F}_{j-\frac{1}{2}}(\tilde{\mathbf{Q}}^{\upsilon-1}) \right)$$

with $\tilde{\mathbf{Q}}^0 := \mathbf{Q}^n$, $\alpha_1 = 1$, $\beta_1 = 0$; and $\mathbf{Q}^{n+1} := \tilde{\mathbf{Q}}^{\Upsilon}$ after final stage Υ

Typical storage-efficient SSPRK(3,3):

$$\tilde{\mathbf{Q}}^1 = \mathbf{Q}^n + \Delta t \mathcal{F}(\mathbf{Q}^n), \quad \tilde{\mathbf{Q}}^2 = \frac{3}{4}\mathbf{Q}^n + \frac{1}{4}\tilde{\mathbf{Q}}^1 + \frac{1}{4}\Delta t \mathcal{F}(\tilde{\mathbf{Q}}^1),$$
$$\mathbf{Q}^{n+1} = \frac{1}{3}\mathbf{Q}^n + \frac{2}{3}\tilde{\mathbf{Q}}^2 + \frac{2}{3}\Delta t \mathcal{F}(\tilde{\mathbf{Q}}^2)$$

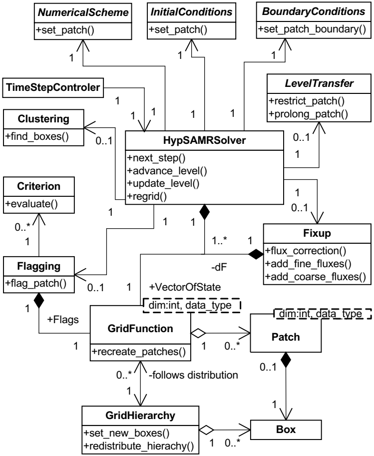
Outline

Overview

- "Adaptive Mesh Refinement in Object-oriented C++"
- $\sim 46,000$ LOC for C++ SAMR kernel, $\sim 140,000$ total C++, C, Fortran-77
- uses parallel hierarchical data structures that have evolved from DAGH
- Implements explicit SAMR with different finite volume solvers
- Embedded boundary method, FSI coupling
- The Virtual Test Facility: AMROC V2.0 plus solid mechanics solvers
- $\sim 430,000$ lines of code total in C++, C, Fortran-77, Fortran-90
- autoconf / automake environment with support for typical parallel high-performance system
- http://www.vtf.website [Deiterding et al., 2006][Deiterding et al., 2007]

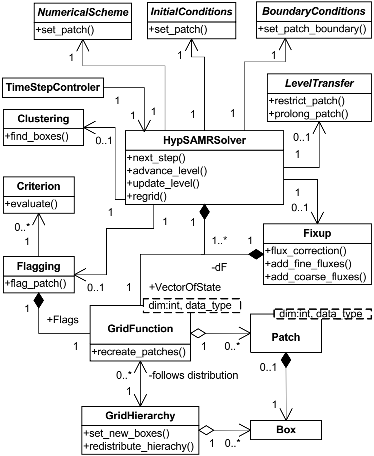
UML design of AMROC

- Classical framework approach with generic main program in C++



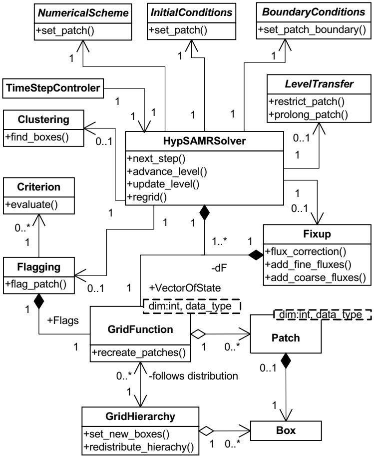
UML design of AMROC

- Classical framework approach with generic main program in C++
- Customization / modification in Problem.h include file by derivation from base classes and redefining virtual interface functions



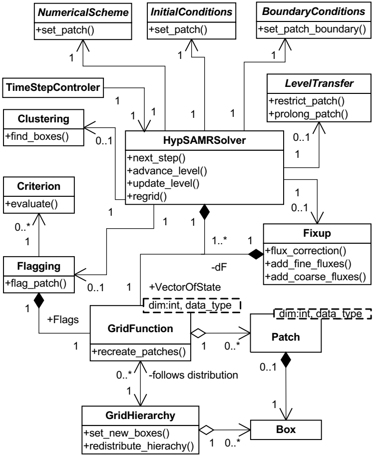
UML design of AMROC

- Classical framework approach with generic main program in C++
- Customization / modification in Problem.h include file by derivation from base classes and redefining virtual interface functions
- Predefined, scheme-specific classes provided for standard simulations



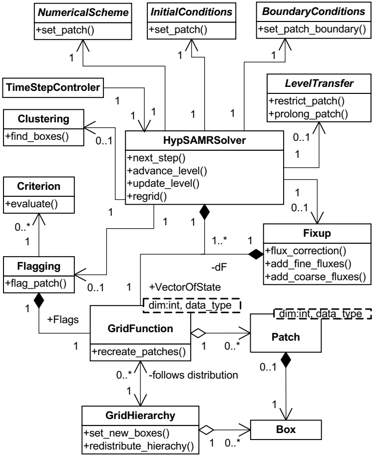
UML design of AMROC

- Classical framework approach with generic main program in C++
- Customization / modification in Problem.h include file by derivation from base classes and redefining virtual interface functions
- Predefined, scheme-specific classes provided for standard simulations
- Clawpack, WENO: Standard simulations require only linking to F77 functions for initial and boundary conditions, source terms. No C++ knowledge required



UML design of AMROC

- Classical framework approach with generic main program in C++
- Customization / modification in Problem.h include file by derivation from base classes and redefining virtual interface functions
- Predefined, scheme-specific classes provided for standard simulations
- Clawpack, WENO: Standard simulations require only linking to F77 functions for initial and boundary conditions, source terms. No C++ knowledge required
- Expert usage (algorithm modification, advanced output, etc.) in C++



Commonalities in software design

- Index coordinate system based on $\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{max}} r_\kappa$ to uniquely identify a cell witin the hierarchy

Commonalities in software design

- Index coordinate system based on $\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$ to uniquely identify a cell witin the hierarchy
- `Box<dim>`, `BoxList<dim>` class that define rectangular regions $G_{m,l}$ by `lowerleft`, `upperright`, `stepsize` and specify topological operations $\cap$, $\cup$, $\setminus$

Commonalities in software design

- Index coordinate system based on $\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$ to uniquely identify a cell witin the hierarchy
- `Box<dim>`, `BoxList<dim>` class that define rectangular regions $G_{m,l}$ by `lowerleft`, `upperright`, `stepsize` and specify topological operations $\cap$, $\cup$, $\setminus$
- `Patch<dim,type>` class that assigns data to a rectangular grid $G_{m,l}$

Commonalities in software design

- Index coordinate system based on $\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$ to uniquely identify a cell witin the hierarchy
- `Box<dim>`, `BoxList<dim>` class that define rectangular regions $G_{m,l}$ by `lowerleft`, `upperright`, `stepsize` and specify topological operations $\cap$, $\cup$, $\setminus$
- `Patch<dim,type>` class that assigns data to a rectangular grid $G_{m,l}$
- A class, here `GridFunction<dim,type>`, that defines topogical relations between lists of Patch objects to implement sychronization, restriction, prolongation, re-distribution

Commonalities in software design

- Index coordinate system based on $\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$ to uniquely identify a cell witin the hierarchy
- `Box<dim>`, `BoxList<dim>` class that define rectangular regions $G_{m,l}$ by `lowerleft`, `upperright`, `stepsize` and specify topological operations $\cap$, $\cup$, $\setminus$
- `Patch<dim,type>` class that assigns data to a rectangular grid $G_{m,l}$
- A class, here `GridFunction<dim,type>`, that defines topogical relations between lists of Patch objects to implement sychronization, restriction, prolongation, re-distribution
- Hierarchical parallel data structures are typically C++, routines on patches often Fortran

Hierarchical data structures

Directory `amroc/hds`. Key classes:

- **Coords**: Point in index coordinator system
  `code/amroc/doc/html/hds/classCoords.html`
- **BBox**: Rectangular region
  `code/amroc/doc/html/hds/classBBox.html`
- **BBoxList**: Set of BBox elements
  `code/amroc/doc/html/hds/classBBoxList.html`
- **GridBox**: Has a BBox member, but adds level and partitioning information
  `code/amroc/doc/html/hds/classGridBox.html`
- **GridBoxList**: Set of GridBox elements
  `code/amroc/doc/html/hds/classBBoxList.html`
- **GridData**<**Type, dim**>: Creates array data of Type of same dimension as BBox, has extensive math operators
  `code/amroc/doc/html/hds/classGridData_3_01Type_00_012_01_4.html`
- **Vector**<**Scalar, length**>: Vector of state is usually Vector<double, N>
  `code/amroc/doc/html/hds/classVector.html`

Hierarchical data structures - II

- **GridDataBlock<Type, dim>**: The Patch-class. Has a GridData<Type, dim>member, knows about relations of current patch within AMR hierarchy

  `code/amroc/doc/html/hds/classGridDataBlock.html`

- **GridFunction<Type, dim>**: Uses GridDataBlock<Type, dim>objects to organize hierarchical data of Type after receiving GridBoxLists. Has extensive math operators for whole levels. Recreates GridDataBlock<Type, dim>lists automatically when GridBoxList changes. Calls interlevel operations are automatically when required.

  `code/amroc/doc/html/hds/classGridFunction.html`

- **GridHierarchy<Type, dim>**: Uses sets of GridBoxList to organize topology of the hierarchy. All GridFunction<Type, dim>are members and receive updated GridBoxList after regridding and repartitioning. Calls DAGHDistribution of partitioning. Implements parallel Recompose().

  `code/amroc/doc/html/hds/classGridHierarchy.html`

AMR level

Directory amroc/amr. Central class is **AMRSolver<VectorType, FixupType, FlagType, dim>**:

code/amroc/doc/html/amr/classAMRSolver.html

- Uses **Integrator<VectorType, dim>** to interface and call the patch-wise numerical update

  code/amroc/doc/html/amr/classIntegrator.html

- Uses **InitialCondition<VectorType, dim >** to call initial conditions patch-wise

  code/amroc/doc/html/amr/classInitialCondition.html

- Uses **BoundaryConditions<VectorType, dim >** to call boundary conditions per side and patch

  code/amroc/doc/html/amr/classBoundaryConditions.html

- Fortran interfaces to above classes are in amroc/amr/F77Interfaces, convenient C++ interfaces in amroc/amr/Interfaces.
- Implements parallel AdvanceLevel(), RegridLevel().

AMR level - II

- **AMRFixup<VectorType, FixupType, dim>** implements the conservative flux correction, holds lower dimensional GridFunctions for correction terms

  `code/amroc/doc/html/amr/classAMRFixup.html`

- **AMRFlagging<VectorType, FixupType, FlagType, dim>** calls a list of refinement criteria and stores results in scalar GridFunction for flags. All criteria are in amroc/amr/Criteria

  `code/amroc/doc/html/amr/classAMRFlagging.html`

- **LevelTransfer<VectorType, dim>** provides patch-wise interpolation and restriction routines that are passed as parameters to GridFunction

  `code/amroc/doc/html/amr/classLevelTransfer.html`

- **AMRTimeStep** implements time step control for a Solver

  `code/amroc/doc/html/amr/classAMRTimeStep.html`

- **AMRInterpolation<VectorType, dim>** is an interpolation at arbitrary point location, typically used for post-processing

  `code/amroc/doc/html/amr/classAMRInterpolation.html`

Outline

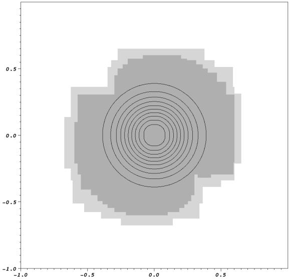
SAMR accuracy verification

Gaussian density shape

$$\rho(x_1, x_2) = 1 + e^{-\left(\frac{\sqrt{x_1^2+x_2^2}}{R}\right)^2}$$

is advected with constant velocities $u_1 = u_2 \equiv 1$, $p_0 \equiv 1$, $R = 1/4$

- Domain $[-1, 1] \times [-1, 1]$, periodic boundary conditions, $t_{end} = 2$
- Two levels of adaptation with $r_{1,2} = 2$, finest level corresponds to $N \times N$ uniform grid



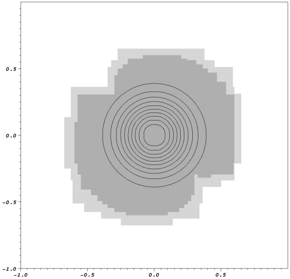
SAMR accuracy verification

Gaussian density shape

$$\rho(x_1,x_2) = 1 + e^{-\left(\frac{\sqrt{x_1^2+x_2^2}}{R}\right)^2}$$

is advected with constant velocities $u_1 = u_2 \equiv 1$, $p_0 \equiv 1$, $R = 1/4$

- Domain $[-1,1]\times[-1,1]$, periodic boundary conditions, $t_{end} = 2$
- Two levels of adaptation with $r_{1,2} = 2$, finest level corresponds to $N \times N$ uniform grid



Use *locally* conservative interpolation

$$\check{\mathbf{Q}}^l_{v,w} := \mathbf{Q}^l_{ij} + f_1(\mathbf{Q}^l_{i+1,j} - \mathbf{Q}^l_{i-1,j}) + f_2(\mathbf{Q}^l_{i,j+1} - \mathbf{Q}^l_{i,j-1})$$

with factor $f_1 = \dfrac{x^v_{1,l+1} - x^i_{1,l}}{2\Delta x_{1,l}}$, $\quad f_2 = \dfrac{x^w_{2,l+1} - x^j_{2,l}}{2\Delta x_{2,l}}$ to also test flux correction

This prolongation operator is not monotonicity preserving! Only applicable to smooth problems.

code/amroc/doc/html/apps/clawpack_2applications_2euler_22d_2GaussianPulseAdvection_2src_2Problem_8h_source.html

SAMR accuracy verification: results

VanLeer flux vector splitting with dimensional splitting, Minmod limiter

N	Unigrid		SAMR - fixup			SAMR - no fixup		
	Error	Order	Error	Order	$\Delta\rho$	Error	Order	$\Delta\rho$
20	0.10946400							
40	0.04239430	1.369						
80	0.01408160	1.590	0.01594820		0	0.01595980		2e-5
160	0.00492945	1.514	0.00526693	1.598	0	0.00530538	1.589	2e-5
320	0.00146132	1.754	0.00156516	1.751	0	0.00163837	1.695	-1e-5
640	0.00041809	1.805	0.00051513	1.603	0	0.00060021	1.449	-6e-5

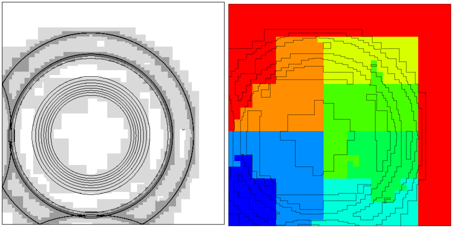
Fully two-dimensional Wave Propagation Method, Minmod limiter

N	Unigrid		SAMR - fixup			SAMR - no fixup		
	Error	Order	Error	Order	$\Delta\rho$	Error	Order	$\Delta\rho$
20	0.10620000							
40	0.04079600	1.380						
80	0.01348250	1.598	0.01536580		0	0.01538820		2e-5
160	0.00472301	1.513	0.00505406	1.604	0	0.00510499	1.592	5e-5
320	0.00139611	1.758	0.00147218	1.779	0	0.00152387	1.744	7e-5
640	0.00039904	1.807	0.00044500	1.726	0	0.00046587	1.710	6e-5

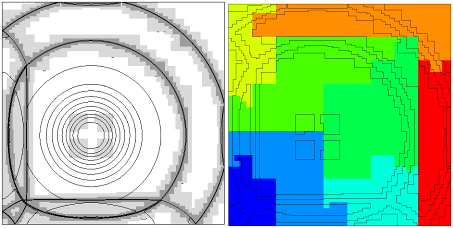
Benchmark run: blast wave in 2D

- 2D-Wave-Propagation Method with Roe's approximate solver
- Base grid 150×150
- 2 levels: factor 2, 4

Task [%]	$P=1$	$P=2$	$P=4$	$P=8$	$P=16$
Update by $\mathcal{H}^{(\cdot)}$	86.6	83.4	76.7	64.1	51.9
Flux correction	1.2	1.6	3.0	7.9	10.7
Boundary setting	3.5	5.7	10.1	15.6	18.3
Recomposition	5.5	6.1	7.4	9.9	14.0
Misc.	4.9	3.2	2.8	2.5	5.1
Time [min]	**151.9**	**79.2**	**43.4**	**23.3**	**13.9**
Efficiency [%]	**100.0**	**95.9**	**87.5**	**81.5**	**68.3**



After 38 time steps

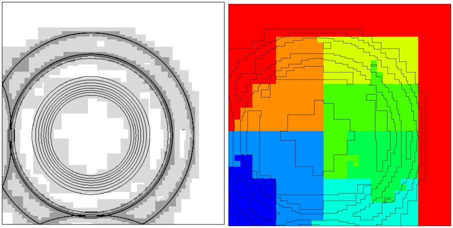


After 79 time steps

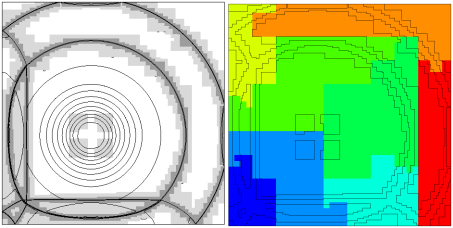
Benchmark run: blast wave in 2D

- 2D-Wave-Propagation Method with Roe's approximate solver
- Base grid 150 × 150
- 2 levels: factor 2, 4

Task [%]	$P\!=\!1$	$P\!=\!2$	$P\!=\!4$	$P\!=\!8$	$P\!=\!16$
Update by $\mathcal{H}^{(\cdot)}$	86.6	83.4	76.7	64.1	51.9
Flux correction	1.2	1.6	3.0	7.9	10.7
Boundary setting	3.5	5.7	10.1	15.6	18.3
Recomposition	5.5	6.1	7.4	9.9	14.0
Misc.	4.9	3.2	2.8	2.5	5.1
Time [min]	**151.9**	**79.2**	**43.4**	**23.3**	**13.9**
Efficiency [%]	**100.0**	**95.9**	**87.5**	**81.5**	**68.3**



After 38 time steps



After 79 time steps

`code/amroc/doc/html/apps/clawpack_2applications_2euler_22d_2Box_2src_2Problem_8h_source.html`

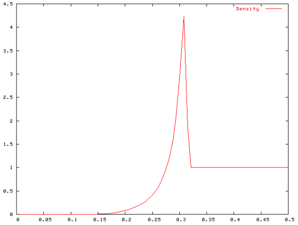
Benchmark run 2: point-explosion in 3D

- Benchmark from the Chicago workshop on AMR methods, September 2003
- Sedov explosion - energy deposition in sphere of radius 4 finest cells
- 3D-Wave-Prop. Method with hybrid Roe-HLL scheme
- Base grid 32^3
- Refinement factor $r_l = 2$
- Effective resolutions: 128^3, 256^3, 512^3, 1024^3
- Grid generation efficiency $\eta_{tol} = 85\%$
- Proper nesting enforced
- Buffer of 1 cell

Benchmark run 2: point-explosion in 3D

- Benchmark from the Chicago workshop on AMR methods, September 2003
- Sedov explosion - energy deposition in sphere of radius 4 finest cells
- 3D-Wave-Prop. Method with hybrid Roe-HLL scheme
- Base grid 32^3
- Refinement factor $r_l = 2$
- Effective resolutions: 128^3, 256^3, 512^3, 1024^3
- Grid generation efficiency $\eta_{tol} = 85\%$
- Proper nesting enforced
- Buffer of 1 cell

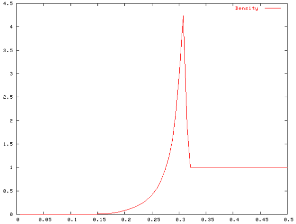
$l_{\text{max}} = 4$ solution



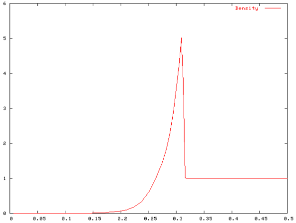
Benchmark run 2: point-explosion in 3D

- Benchmark from the Chicago workshop on AMR methods, September 2003
- Sedov explosion - energy deposition in sphere of radius 4 finest cells
- 3D-Wave-Prop. Method with hybrid Roe-HLL scheme
- Base grid 32^3
- Refinement factor $r_l = 2$
- Effective resolutions: 128^3, 256^3, 512^3, 1024^3
- Grid generation efficiency $\eta_{tol} = 85\%$
- Proper nesting enforced
- Buffer of 1 cell

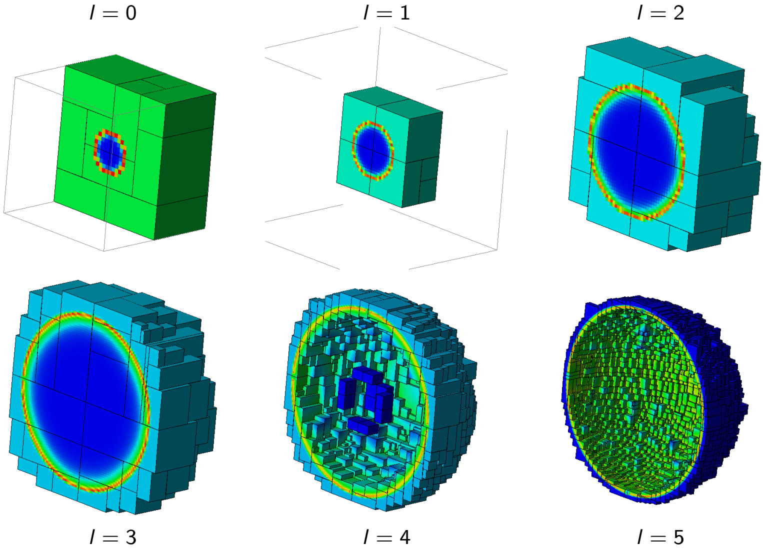
$l_{\text{max}} = 4$ solution



$l_{\text{max}} = 5$ solution



Benchmark run 2: visualization of refinement



Benchmark run 2: performance results

Number of grids and cells

l	$l_{\max} = 2$		$l_{\max} = 3$		$l_{\max} = 4$		$l_{\max} = 5$	
	Grids	Cells	Grids	Cells	Grids	Cells	Grids	Cells
0	28	32,768	28	32,768	33	32,768	34	32,768
1	8	32,768	14	32,768	20	32,768	20	32,768
2	63	115,408	49	116,920	43	125,680	50	125,144
3			324	398,112	420	555,744	193	572,768
4					1405	1,487,312	1,498	1,795,048
5							5,266	5,871,128
$\sum$		180,944		580,568		2,234,272		8,429,624

Benchmark run 2: performance results

Number of grids and cells

l	$l_{\max}=2$		$l_{\max}=3$		$l_{\max}=4$		$l_{\max}=5$	
	Grids	Cells	Grids	Cells	Grids	Cells	Grids	Cells
0	28	32,768	28	32,768	33	32,768	34	32,768
1	8	32,768	14	32,768	20	32,768	20	32,768
2	63	115,408	49	116,920	43	125,680	50	125,144
3			324	398,112	420	555,744	193	572,768
4					1405	1,487,312	1,498	1,795,048
5							5,266	5,871,128
$\sum$		180,944		580,568		2,234,272		8,429,624

Breakdown of CPU time on 8 nodes SGI Altix 3000 (Linux-based shared memory system)

Task [%]	$l_{\max}=2$		$l_{\max}=3$		$l_{\max}=4$		$l_{\max}=5$	
Integration	73.7		77.2		72.9		37.8	
Fixup	2.6	*46*	3.1	*58*	2.6	*42*	2.2	*45*
Boundary	10.1	*79*	6.3	*78*	5.1	*56*	6.9	*78*
Recomposition	7.4		8.0		15.1		50.4	
Clustering	0.5		0.6		0.7		1.0	
Output/Misc	5.7		4.0		3.6		1.7	
Time [min]	0.5		5.1		73.0		2100.0	
Uniform [min]	5.4		160		~5,000		~180,000	
Factor of AMR savings	11		31		69		86	
Time steps	15		27		52		115	

Benchmark run 2: performance results

Number of grids and cells

l	$l_{max}=2$		$l_{max}=3$		$l_{max}=4$		$l_{max}=5$	
	Grids	Cells	Grids	Cells	Grids	Cells	Grids	Cells
0	28	32,768	28	32,768	33	32,768	34	32,768
1	8	32,768	14	32,768	20	32,768	20	32,768
2	63	115,408	49	116,920	43	125,680	50	125,144
3			324	398,112	420	555,744	193	572,768
4					1405	1,487,312	1,498	1,795,048
5							5,266	5,871,128
$\sum$		180,944		580,568		2,234,272		8,429,624

Breakdown of CPU time on 8 nodes SGI Altix 3000 (Linux-based shared memory system)

Task [%]	$l_{max}=2$		$l_{max}=3$		$l_{max}=4$		$l_{max}=5$	
Integration	73.7		77.2		72.9		37.8	
Fixup	2.6	*46*	3.1	*58*	2.6	*42*	2.2	*45*
Boundary	10.1	*79*	6.3	*78*	5.1	*56*	6.9	*78*
Recomposition	7.4		8.0		15.1		50.4	
Clustering	0.5		0.6		0.7		1.0	
Output/Misc	5.7		4.0		3.6		1.7	
Time [min]	0.5		5.1		73.0		2100.0	
Uniform [min]	5.4		160		∼5,000		∼180,000	
Factor of AMR savings	11		31		69		86	
Time steps	15		27		52		115	

`code/amroc/doc/html/apps/clawpack_2applications_2euler_23d_2Sedov_2src_2Problem_8h_source.html`

Components

Directory `amroc/clawpack/src` contains generic Fortran functions:

- **?d/integrator_extended**: Contains an extended version of Clawpack 3.0 by R. LeVeque. The MUSCL approach was added, 3d fully implemented, interfaces have been adjusted for AMROC. These codes are equation independent.

  `code/amroc/doc/html/clp/files.html`

- **?d/equations**: Contains equation-specific Riemann solvers, flux functions as F77 routines.
- **?d/interpolation**: Contains patch-wise interpolation and restriction operators in F77.

Directory `amroc/clawpack` contains the generic C++ classes to interface the F77 library from ?d/integrator_extended with AMROC:

- **ClpIntegrator<VectorType, AuxVectorType, dim >**: Interfaces the F77 library from ?d/integrator_extended to Integrator<VectorType, dim>. Key function to fill is CalculateGrid().

  `code/amroc/doc/html/clp/classClpIntegrator_3_01VectorType_00_01AuxVectorType_00_012_01_4.html`

Components - II

- **ClpFixup<VectorType, FixupType, AuxVectorType, dim >**: The conservative flux correction is more complex in the waves of the flux difference splitting schemes. This specialization of AMRFixup<VectorType, FixupType, dim>considers this.

  `code/amroc/doc/html/clp/classClpFixup.html`

- A generic main program **amroc/clawpack/mains/amr_main.C** instantiates Integrator<VectorType, dim>, InitialCondition<VectorType, dim >, BoundaryConditions<VectorType, dim >

  `code/amroc/doc/html/clp/amr__main_8C.html`

- **Problem.h**: Allows simulation-specific alteration in class-libary style by derivation from predefined classes specified in **ClpStdProblem.h**

  `code/amroc/doc/html/clp/ClpStdProblem_8h.html`

- **ClpProblem.h**: General include before equation-specific C++ definition file is read that defines VectorType and provides Fortran function names required by amroc/amr/F77Interfaces classes

  `code/amroc/doc/html/clp/ClpProblem_8h_source.html` `code/amroc/doc/html/clp/euler2_8h.html`

Functions to link in Makefile.am

Interface objects from amroc/amr/F77Interfaces are used to mimic the interface of standard Clawpack, which constructs specific simulations by linking F77 functions. Required functions are:

- **init.f**: Initial conditions.
- **physbd.f**: Boundary conditions.
- **combl.f**: Initialize application specific common blocks.
- **$(EQUATION)/rp/rpn.f and rpt.f**: Equation-specific Riemann solvers in normal and transverse direction.
- **$(EQUATION)/rp/flx.f, $(EQUATION)/rp/rec.f**: Flux and reconstruction for MUSCL slope limiting (if used), otherwise dummy-routines/flx.f and dummy-routines/rec.f may be used.
- **$(EQUATION)/rp/chk.f**: Physical consistency check for debugging.
- **src.f**: Source term for a splitting method., otherwise dummy-routines/src.f can be linked.
- **setaux.f**: Set data in an additional patch-wise auxiliary array, otherwise dummy-routines/saux.f can be linked.

Outline

Favre-averaged Navier-Stokes equations

$$\frac{\partial\bar\rho}{\partial t}+\frac{\partial}{\partial x_n}\left(\bar\rho\tilde u_n\right)=0$$
$$\frac{\partial}{\partial t}\left(\bar\rho\tilde u_k\right)+\frac{\partial}{\partial x_n}\left(\bar\rho\tilde u_k\tilde u_n+\delta_{kn}\bar p-\tilde\tau_{kn}+\sigma_{kn}\right)=0$$
$$\frac{\partial\bar\rho\bar E}{\partial t}+\frac{\partial}{\partial x_n}\left(\tilde u_n(\bar\rho\bar E+\bar p)+\tilde q_n-\tilde\tau_{nj}\tilde u_j+\sigma_n^e\right)=0$$
$$\frac{\partial}{\partial t}\left(\bar\rho\tilde Y_i\right)+\frac{\partial}{\partial x_n}\left(\bar\rho\tilde Y_i\tilde u_n+\tilde J_n^i+\sigma_n^i\right)=0$$

with stress tensor

$$\tilde\tau_{kn}=\tilde\mu\left(\frac{\partial\tilde u_n}{\partial x_k}+\frac{\partial\tilde u_k}{\partial x_n}\right)-\frac{2}{3}\tilde\mu\frac{\partial\tilde u_j}{\partial x_j}\delta_{in}\ ,$$

heat conduction

$$\tilde q_n=-\tilde\lambda\frac{\partial\tilde T}{\partial x_n}\ ,$$

Favre-averaged Navier-Stokes equations

$$\frac{\partial \bar\rho}{\partial t} + \frac{\partial}{\partial x_n}\left(\bar\rho \tilde u_n\right) = 0$$

$$\frac{\partial}{\partial t}\left(\bar\rho \tilde u_k\right) + \frac{\partial}{\partial x_n}\left(\bar\rho \tilde u_k \tilde u_n + \delta_{kn}\bar p - \tilde\tau_{kn} + \sigma_{kn}\right) = 0$$

$$\frac{\partial \bar\rho \bar E}{\partial t} + \frac{\partial}{\partial x_n}\left(\tilde u_n(\bar\rho \bar E + \bar p) + \tilde q_n - \tilde\tau_{nj}\tilde u_j + \sigma_n^e\right) = 0$$

$$\frac{\partial}{\partial t}\left(\bar\rho \tilde Y_i\right) + \frac{\partial}{\partial x_n}\left(\bar\rho \tilde Y_i \tilde u_n + \tilde J_n^i + \sigma_n^i\right) = 0$$

with stress tensor

$$\tilde\tau_{kn} = \tilde\mu\left(\frac{\partial \tilde u_n}{\partial x_k} + \frac{\partial \tilde u_k}{\partial x_n}\right) - \frac{2}{3}\tilde\mu\frac{\partial \tilde u_j}{\partial x_j}\delta_{in}\ ,$$

heat conduction

$$\tilde q_n = -\tilde\lambda\frac{\partial \tilde T}{\partial x_n}\ ,$$

and inter-species diffusion

$$\tilde J_n^i = -\bar\rho \tilde D_i \frac{\partial \tilde Y_i}{\partial x_n}$$

Favre-averaged Navier-Stokes equations

$$\frac{\partial\bar\rho}{\partial t}+\frac{\partial}{\partial x_n}\left(\bar\rho\tilde u_n\right)=0$$

$$\frac{\partial}{\partial t}\left(\bar\rho\tilde u_k\right)+\frac{\partial}{\partial x_n}\left(\bar\rho\tilde u_k\tilde u_n+\delta_{kn}\bar p-\tilde\tau_{kn}+\sigma_{kn}\right)=0$$

$$\frac{\partial\bar\rho\bar E}{\partial t}+\frac{\partial}{\partial x_n}\left(\tilde u_n(\bar\rho\bar E+\bar p)+\tilde q_n-\tilde\tau_{nj}\tilde u_j+\sigma_n^e\right)=0$$

$$\frac{\partial}{\partial t}\left(\bar\rho\tilde Y_i\right)+\frac{\partial}{\partial x_n}\left(\bar\rho\tilde Y_i\tilde u_n+\tilde J_n^i+\sigma_n^i\right)=0$$

with stress tensor

$$\tilde\tau_{kn}=\tilde\mu\left(\frac{\partial\tilde u_n}{\partial x_k}+\frac{\partial\tilde u_k}{\partial x_n}\right)-\frac{2}{3}\tilde\mu\frac{\partial\tilde u_j}{\partial x_j}\delta_{in}\;,$$

heat conduction

$$\tilde q_n=-\tilde\lambda\frac{\partial\tilde T}{\partial x_n}\;,$$

and inter-species diffusion

$$\tilde J_n^i=-\bar\rho\tilde D_i\frac{\partial\tilde Y_i}{\partial x_n}$$

Favre-filtering

$$\tilde\phi=\frac{\overline{\rho\phi}}{\bar\rho}\quad\text{with}\quad\bar\phi(\mathbf{x},t;\Delta_c)=\int_\Omega G(\mathbf{x}-\mathbf{x}';\Delta_c)\phi(\mathbf{x}',t)d\mathbf{x}'$$

Numerical solution approach

- Subgrid terms σ_{kn}, σ_n^e, σ_n^i are computed by Pullin's stretched-vortex model

Numerical solution approach

- Subgrid terms σ_{kn}, σ_n^e, σ_n^i are computed by Pullin's stretched-vortex model
- Cutoff Δ_c is set to local SAMR resolution Δx_l

Numerical solution approach

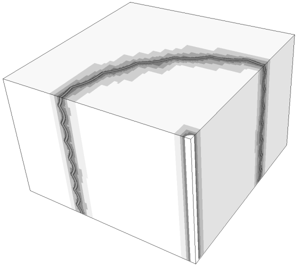
- Subgrid terms σ_{kn}, σ_n^e, σ_n^i are computed by Pullin's stretched-vortex model
- Cutoff Δ_c is set to local SAMR resolution Δx_l
- It remains to solve the Navier-Stokes equations in the hyperbolic regime
  - 3rd order WENO method (hybridized with a tuned centered difference stencil) for convection
  - 2nd order conservative centered differences for diffusion

Numerical solution approach

- Subgrid terms σ_{kn}, σ_n^e, σ_n^i are computed by Pullin's stretched-vortex model
- Cutoff Δ_c is set to local SAMR resolution Δx_l
- It remains to solve the Navier-Stokes equations in the hyperbolic regime
  - 3rd order WENO method (hybridized with a tuned centered difference stencil) for convection
  - 2nd order conservative centered differences for diffusion

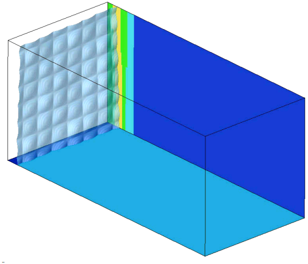
Example: Cylindrical Richtmyer-Meshkov instability

- Sinusoidal interface between two gases hit by shock wave
- Objective is correctly predict turbulent mixing
- Embedded boundary method used to regularize apex
- AMR base grid $95 \times 95 \times 64$ cells, $r_{1,2,3} = 2$
- $\sim 70,000\,\mathrm{h}$ CPU on 32 AMD 2.5GHZ-quad-core nodes



Planar Richtmyer-Meshkov instability

- Perturbed Air-SF6 interface shocked and re-shocked by Mach 1.5 shock
- Containment of turbulence in refined zones
- 96 CPUs IBM SP2-Power3
- WENO-TCD scheme with LES model
- AMR base grid $172 \times 56 \times 56$, $r_{1,2} = 2$, $10\,\mathrm{M}$ cells in average instead of $3\,\mathrm{M}$ (uniform)

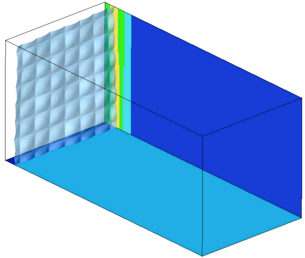


Task	2ms (%)	5ms (%)	10ms (%)
Integration	45.3	65.9	52.0
Boundary setting	44.3	28.6	41.9
Flux correction	7.2	3.4	4.1
Interpolation	0.9	0.4	0.3
Reorganization	1.6	1.2	1.2
Misc.	0.6	0.5	0.5
Max. imbalance	1.25	1.23	1.30

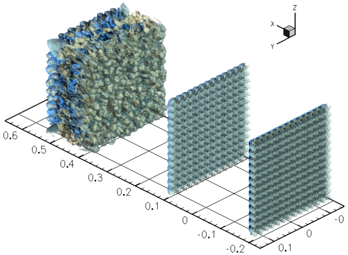
`code/amroc/doc/html/apps/weno_2applications_2euler_23d_2RM_`
`_AirSF6_2src_2Problem_8h_source.html`

Planar Richtmyer-Meshkov instability

- Perturbed Air-SF6 interface shocked and re-shocked by Mach 1.5 shock
- Containment of turbulence in refined zones
- 96 CPUs IBM SP2-Power3
- WENO-TCD scheme with LES model
- AMR base grid $172 \times 56 \times 56$, $r_{1,2} = 2$, $10\,\mathrm{M}$ cells in average instead of $3\,\mathrm{M}$ (uniform)



Task	2ms (%)	5ms (%)	10ms (%)
Integration	45.3	65.9	52.0
Boundary setting	44.3	28.6	41.9
Flux correction	7.2	3.4	4.1
Interpolation	0.9	0.4	0.3
Reorganization	1.6	1.2	1.2
Misc.	0.6	0.5	0.5
Max. imbalance	1.25	1.23	1.30



code/amroc/doc/html/apps/weno_2applications_2euler_23d_2RM_
_AirSF6_2src_2Problem_8h_source.html

Flux correction for Runge-Kutta method

Recall Runge-Kutta temporal update

$$\tilde{\mathbf{Q}}_j^{\upsilon} = \alpha_\upsilon \mathbf{Q}_j^n + \beta_\upsilon \tilde{\mathbf{Q}}_j^{\upsilon-1} + \gamma_\upsilon \frac{\Delta t}{\Delta x_k} \Delta \mathbf{F}^k(\tilde{\mathbf{Q}}^{\upsilon-1})$$

Flux correction for Runge-Kutta method

Recall Runge-Kutta temporal update

$$\tilde{\mathbf{Q}}_j^{\upsilon} = \alpha_\upsilon \mathbf{Q}_j^n + \beta_\upsilon \tilde{\mathbf{Q}}_j^{\upsilon-1} + \gamma_\upsilon \frac{\Delta t}{\Delta x_k} \Delta \mathbf{F}^k(\tilde{\mathbf{Q}}^{\upsilon-1})$$

rewrite scheme as

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n - \sum_{\upsilon=1}^{\Upsilon} \varphi_\upsilon \frac{\Delta t}{\Delta x_k} \Delta \mathbf{F}^k(\tilde{\mathbf{Q}}^{\upsilon-1}) \quad \text{with} \quad \varphi_\upsilon = \gamma_\upsilon \prod_{\nu=\upsilon+1}^{\Upsilon} \beta_\nu$$

Flux correction for Runge-Kutta method

Recall Runge-Kutta temporal update

$$\tilde{\mathbf{Q}}_j^{\upsilon} = \alpha_\upsilon \mathbf{Q}_j^n + \beta_\upsilon \tilde{\mathbf{Q}}_j^{\upsilon-1} + \gamma_\upsilon \frac{\Delta t}{\Delta x_k} \Delta \mathbf{F}^k(\tilde{\mathbf{Q}}^{\upsilon-1})$$

rewrite scheme as

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n - \sum_{\upsilon=1}^{\Upsilon} \varphi_\upsilon \frac{\Delta t}{\Delta x_k} \Delta \mathbf{F}^k(\tilde{\mathbf{Q}}^{\upsilon-1}) \quad \text{with} \quad \varphi_\upsilon = \gamma_\upsilon \prod_{\nu=\upsilon+1}^{\Upsilon} \beta_\nu$$

Flux correction to be used [Pantano et al., 2007]

1. $\delta \mathbf{F}^{1,l+1}_{i-\frac{1}{2},j} := -\varphi_1 \mathbf{F}^{1,l}_{i-\frac{1}{2},j}(\tilde{\mathbf{Q}}^0)\,, \qquad \delta \mathbf{F}^{1,l+1}_{i-\frac{1}{2},j} := \delta \mathbf{F}^{1,l+1}_{i-\frac{1}{2},j} - \sum_{\upsilon=2}^{\Upsilon} \varphi_\upsilon \mathbf{F}^{1,l}_{i-\frac{1}{2},j}(\tilde{\mathbf{Q}}^{\upsilon-1})$
2. $\delta \mathbf{F}^{1,l+1}_{i-\frac{1}{2},j} := \delta \mathbf{F}^{1,l+1}_{i-\frac{1}{2},j} + \frac{1}{r_{l+1}^2} \sum_{m=0}^{r_{l+1}-1} \sum_{\upsilon=1}^{\Upsilon} \varphi_\upsilon \mathbf{F}^{1,l+1}_{v+\frac{1}{2},w+m} \left(\tilde{\mathbf{Q}}^{\upsilon-1}(t+\kappa \Delta t_{l+1})\right)$

Flux correction for Runge-Kutta method

Recall Runge-Kutta temporal update

$$\tilde{\mathbf{Q}}_j^{\upsilon} = \alpha_\upsilon \mathbf{Q}_j^n + \beta_\upsilon \tilde{\mathbf{Q}}_j^{\upsilon-1} + \gamma_\upsilon \frac{\Delta t}{\Delta x_k} \Delta \mathbf{F}^k(\tilde{\mathbf{Q}}^{\upsilon-1})$$

rewrite scheme as

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n - \sum_{\upsilon=1}^{\Upsilon} \varphi_\upsilon \frac{\Delta t}{\Delta x_k} \Delta \mathbf{F}^k(\tilde{\mathbf{Q}}^{\upsilon-1}) \quad \text{with} \quad \varphi_\upsilon = \gamma_\upsilon \prod_{\nu=\upsilon+1}^{\Upsilon} \beta_\nu$$

Flux correction to be used [Pantano et al., 2007]

1. $\delta \mathbf{F}^{1,l+1}_{i-\frac{1}{2},j} := -\varphi_1 \mathbf{F}^{1,l}_{i-\frac{1}{2},j}(\tilde{\mathbf{Q}}^0)\,, \qquad \delta \mathbf{F}^{1,l+1}_{i-\frac{1}{2},j} := \delta \mathbf{F}^{1,l+1}_{i-\frac{1}{2},j} - \sum_{\upsilon=2}^{\Upsilon} \varphi_\upsilon \mathbf{F}^{1,l}_{i-\frac{1}{2},j}(\tilde{\mathbf{Q}}^{\upsilon-1})$
2. $\delta \mathbf{F}^{1,l+1}_{i-\frac{1}{2},j} := \delta \mathbf{F}^{1,l+1}_{i-\frac{1}{2},j} + \frac{1}{r_{l+1}^2} \sum_{m=0}^{r_{l+1}-1} \sum_{\upsilon=1}^{\Upsilon} \varphi_\upsilon \mathbf{F}^{1,l+1}_{v+\frac{1}{2},w+m} \left(\tilde{\mathbf{Q}}^{\upsilon-1}(t+\kappa\Delta t_{l+1})\right)$

Storage-efficient SSPRK(3,3):

υ	α_υ	β_υ	γ_υ	φ_υ
1	1	0	1	$\frac{1}{6}$
2	$\frac{3}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{6}$
3	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{2}{3}$

Components

Directory `amroc/weno/src` contains the Fortran-90 solver library:

- **generic**: Implements the hybrid WENO-TCD method for Euler and Navier-Stokes equations, characteristic boundary conditions. Code uses F90 modules.

  `code/amroc/doc/html/weno/files.html`

- **equations**: Contains routines that specify between LES and laminar flow, the criterion for scheme hybridization, source terms handled by splitting.

Directory `amroc/weno` contains the generic C++ class to interface the F90 library with AMROC:

- **WENOIntegrator**$<$**VectorType, dim** $>$: Interfaces the F90 solver to Integrator$<$VectorType, dim$>$. CalculateGrid() is called separately for each stage of the Runge-Kutta time integrator.

  `code/amroc/doc/html/weno/classWENOIntegrator.html`

- **WENOFixup**$<$**VectorType, FixupType, dim** $>$: A specialized conservative flux correction that accumulates the correction terms throughout the stages of the Runge-Kutta time integrator.

  `code/amroc/doc/html/weno/classWENOFixup.html`

Components - II

- **WENOInterpolation$<$VectorType, InterpolationType, OutputType, dim $>$**: Is a quite elaborate data collection class based on AMRInterpolation$<$VectorType, dim$>$geared toward statistics processing typical for turbulent simulations. Has run-time function parser.

  `code/amroc/doc/html/weno/classWENOStatistics.html`

The interface otherwise follows the Clawpack integration closely:

- Generic main program **amroc/clawpack/mains/amr_main.C** is re-used.
- **Problem.h**: simulation-specific alteration of the C++ predefined classes specified in **WENOStdProblem.h**

  `code/amroc/doc/html/weno/WENOStdProblem_8h.html`

- **WENOProblem.h**: Include required C++ class definitions definitions, all Fortran function names defined in **WENOStdFunctions.h**

  `code/amroc/doc/html/weno/WENOProblem_8h_source.html code/amroc/doc/html/weno/WENOStdFunctions_8h.html`

- Interface objects from **amroc/amr/F77Interfaces** re-used and functions linked in Makefile.am as with Clawpack integrator

Outline

Governing equations

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot \left[\rho \mathbf{u}^{\mathbf{t}} \mathbf{u} + \left(p + \frac{\mathbf{B} \cdot \mathbf{B}}{2} \right) \mathbf{I} - \mathbf{B}^{\mathbf{t}} \mathbf{B} \right] = \mathbf{0}$$

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot \left[\left(\rho E + p + \frac{\mathbf{B} \cdot \mathbf{B}}{2} \right) \mathbf{u} - (\mathbf{u} \cdot \mathbf{B})\, \mathbf{B} \right] = 0$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot \left(\mathbf{u}^{\mathbf{t}} \mathbf{B} - \mathbf{B}^{\mathbf{t}} \mathbf{u} \right) = \mathbf{0}$$

with equation of state

$$p = (\gamma - 1) \left(\rho E - \rho \frac{\mathbf{u}^2}{2} - \frac{\mathbf{B}^2}{2} \right)$$

The ideal MDH model is still hyperbolic, yet by re-writing the induction equation, one finds that the magnetic field has to satisfy at all times t the elliptic constraint

$$\nabla \cdot \mathbf{B} = 0.$$

Generalized Lagrangian multipliers for divergence control

Hyperbolic-parabolic correction of 2d ideal MHD model [Dedner et al., 2002]:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_x}{\partial x} + \frac{\partial \rho u_y}{\partial y} = 0$$

$$\frac{\partial (\rho u_x)}{\partial t} + \frac{\partial}{\partial x}\left[\rho u_x^2 + p\left(p + \frac{\mathbf{B}\cdot\mathbf{B}}{2}\right) - B_x^2\right] + \frac{\partial}{\partial y}(\rho u_x u_y - B_x B_y) = 0$$

$$\frac{\partial (\rho u_y)}{\partial t} + \frac{\partial}{\partial x}(\rho u_x u_y - B_x B_y) + \frac{\partial}{\partial y}\left[\rho u_y^2 + p\left(p + \frac{\mathbf{B}\cdot\mathbf{B}}{2}\right) - B_y^2\right] = 0$$

$$\frac{\partial (\rho u_z)}{\partial t} + \frac{\partial}{\partial x}(\rho u_z u_x - B_z B_x) + \frac{\partial}{\partial y}(\rho u_z u_y - B_z B_y) = 0$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x}\left[\left(\rho E + p + \frac{\mathbf{B}\cdot\mathbf{B}}{2}\right)\mathbf{u_x} - (\mathbf{u}\cdot\mathbf{B})B_x\right] + \frac{\partial}{\partial y}\left[\left(\rho E + p + \frac{\mathbf{B}\cdot\mathbf{B}}{2}\right)\mathbf{u_y} - (\mathbf{u}\cdot\mathbf{B})B_y\right] = 0$$

$$\frac{\partial B_x}{\partial t} + \frac{\partial \psi}{\partial x} + \frac{\partial}{\partial y}(u_y B_x - B_y u_x) = 0$$

$$\frac{\partial B_y}{\partial t} + \frac{\partial}{\partial x}(u_x B_y - B_x u_y) + \frac{\partial \psi}{\partial y} = 0$$

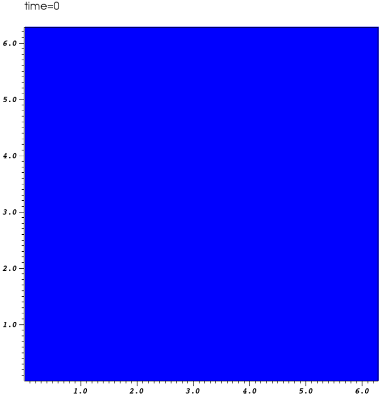
$$\frac{\partial B_z}{\partial t} + \frac{\partial}{\partial x}(u_x B_z - B_z u_x) + \frac{\partial}{\partial y}(u_y B_z - B_y u_z) = 0$$

$$\frac{\partial \psi}{\partial t} + c_h^2\left(\frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y}\right) = -\frac{c_h^2}{c_p^2}\psi$$

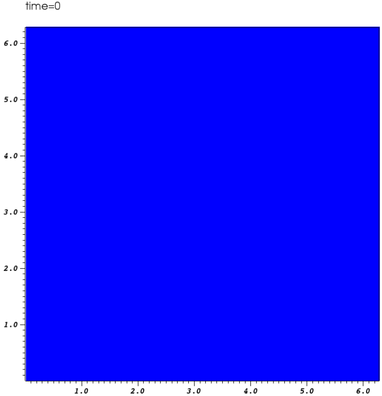
Orszag-Tang vortex

- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

$$\rho(x,y,0) = \gamma^2, \quad u_x(x,y,0) = -\sin(y), \quad u_y(x,y,0) = \sin(x), \quad u_z(x,y,0) = 0$$
$$p(x,y,0) = \gamma, \quad B_x(x,y,0) = -\sin(y), \quad B_y(x,y,0) = 2\sin(x), \quad B_z(x,y,0) = 0$$



Scaled gradient of ρ



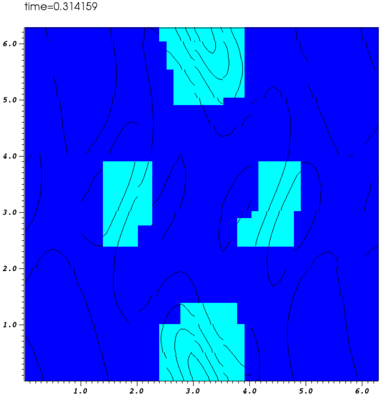
Multi-resolution criterion with hierarchical thresholding

code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html

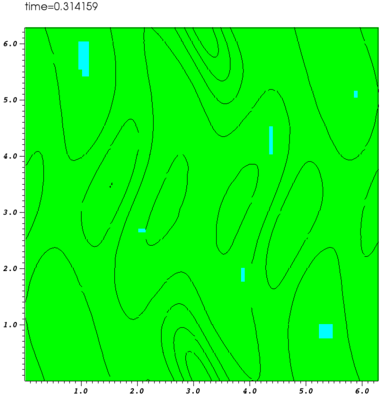
Orszag-Tang vortex

- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

$$\rho(x,y,0) = \gamma^2, \quad u_x(x,y,0) = -\sin(y), \quad u_y(x,y,0) = \sin(x), \quad u_z(x,y,0) = 0$$
$$p(x,y,0) = \gamma, \quad B_x(x,y,0) = -\sin(y), \quad B_y(x,y,0) = 2\sin(x), \quad B_z(x,y,0) = 0$$



Scaled gradient of ρ



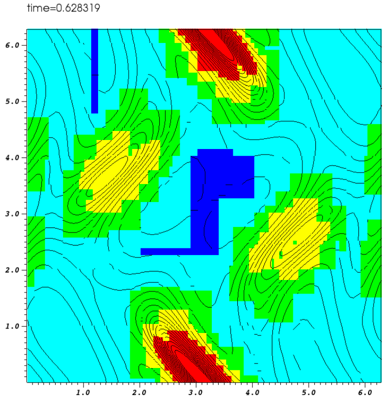
Multi-resolution criterion with hierarchical thresholding

code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html

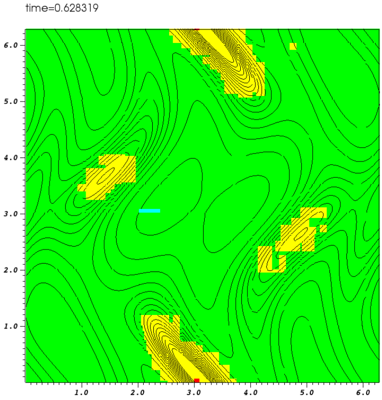
Orszag-Tang vortex

- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

$$\rho(x,y,0) = \gamma^2, \quad u_x(x,y,0) = -\sin(y), \quad u_y(x,y,0) = \sin(x), \quad u_z(x,y,0) = 0$$
$$p(x,y,0) = \gamma, \quad B_x(x,y,0) = -\sin(y), \quad B_y(x,y,0) = 2\sin(x), \quad B_z(x,y,0) = 0$$



Scaled gradient of ρ



Multi-resolution criterion with hierarchical thresholding

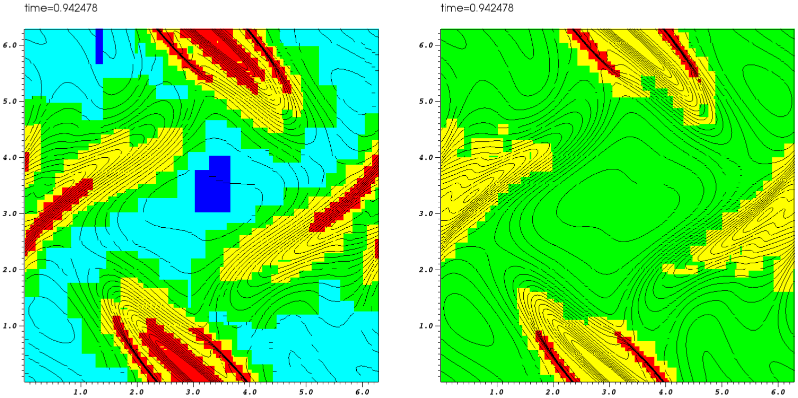
code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html

Orszag-Tang vortex

- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

$$\rho(x,y,0) = \gamma^2, \quad u_x(x,y,0) = -\sin(y), \quad u_y(x,y,0) = \sin(x), \quad u_z(x,y,0) = 0$$

$$p(x,y,0) = \gamma, \quad B_x(x,y,0) = -\sin(y), \quad B_y(x,y,0) = 2\sin(x), \quad B_z(x,y,0) = 0$$



Scaled gradient of ρ

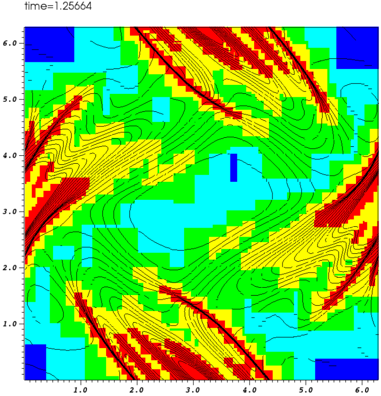
Multi-resolution criterion with hierarchical thresholding

code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html

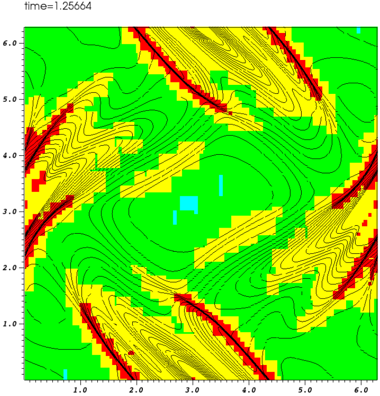
Orszag-Tang vortex

- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

$$\rho(x,y,0) = \gamma^2, \quad u_x(x,y,0) = -\sin(y), \quad u_y(x,y,0) = \sin(x), \quad u_z(x,y,0) = 0$$
$$p(x,y,0) = \gamma, \quad B_x(x,y,0) = -\sin(y), \quad B_y(x,y,0) = 2\sin(x), \quad B_z(x,y,0) = 0$$



Scaled gradient of ρ



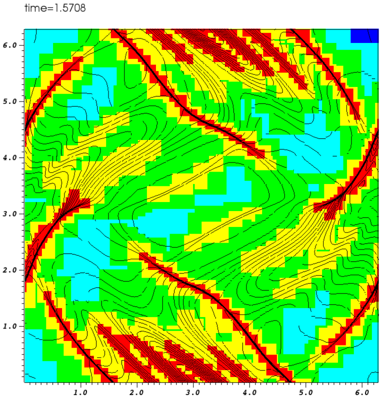
Multi-resolution criterion with hierarchical thresholding

code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html

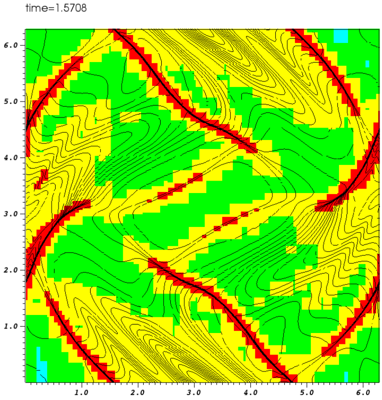
Orszag-Tang vortex

- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

$$\rho(x,y,0) = \gamma^2, \quad u_x(x,y,0) = -\sin(y), \quad u_y(x,y,0) = \sin(x), \quad u_z(x,y,0) = 0$$
$$p(x,y,0) = \gamma, \quad B_x(x,y,0) = -\sin(y), \quad B_y(x,y,0) = 2\sin(x), \quad B_z(x,y,0) = 0$$



Scaled gradient of ρ



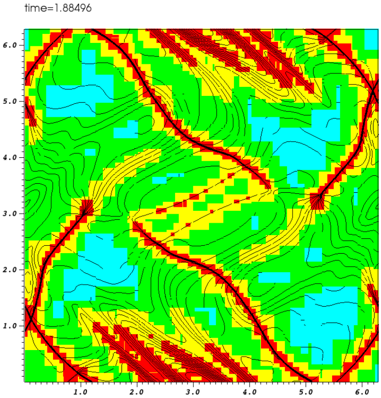
Multi-resolution criterion with hierarchical thresholding

code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html

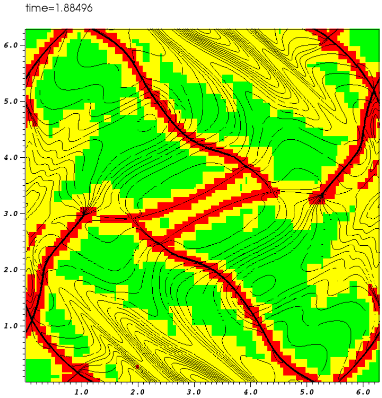
Orszag-Tang vortex

- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

$$\rho(x,y,0) = \gamma^2, \quad u_x(x,y,0) = -\sin(y), \quad u_y(x,y,0) = \sin(x), \quad u_z(x,y,0) = 0$$
$$p(x,y,0) = \gamma, \quad B_x(x,y,0) = -\sin(y), \quad B_y(x,y,0) = 2\sin(x), \quad B_z(x,y,0) = 0$$



Scaled gradient of ρ



Multi-resolution criterion with hierarchical thresholding

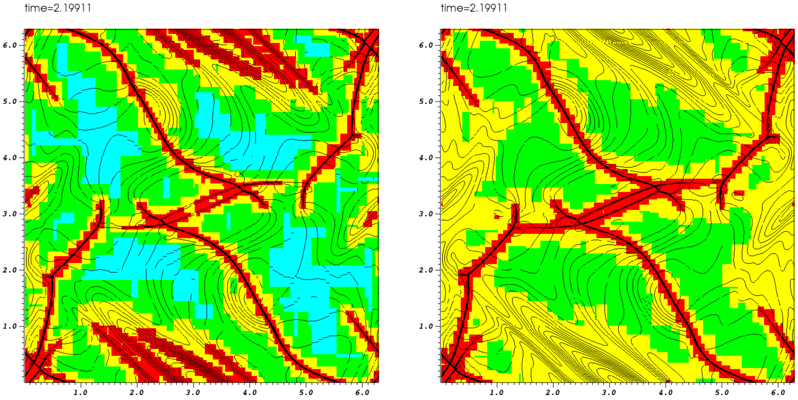
code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html

Orszag-Tang vortex

- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

$$\rho(x,y,0) = \gamma^2, \quad u_x(x,y,0) = -\sin(y), \quad u_y(x,y,0) = \sin(x), \quad u_z(x,y,0) = 0$$

$$p(x,y,0) = \gamma, \quad B_x(x,y,0) = -\sin(y), \quad B_y(x,y,0) = 2\sin(x), \quad B_z(x,y,0) = 0$$



Scaled gradient of ρ

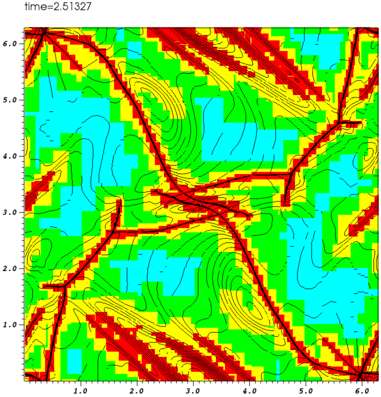
Multi-resolution criterion with hierarchical thresholding

code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html

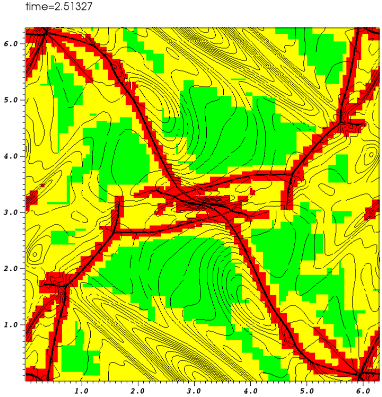
Orszag-Tang vortex

- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

$$\rho(x,y,0) = \gamma^2, \quad u_x(x,y,0) = -\sin(y), \quad u_y(x,y,0) = \sin(x), \quad u_z(x,y,0) = 0$$
$$p(x,y,0) = \gamma, \quad B_x(x,y,0) = -\sin(y), \quad B_y(x,y,0) = 2\sin(x), \quad B_z(x,y,0) = 0$$



Scaled gradient of ρ



Multi-resolution criterion with hierarchical thresholding

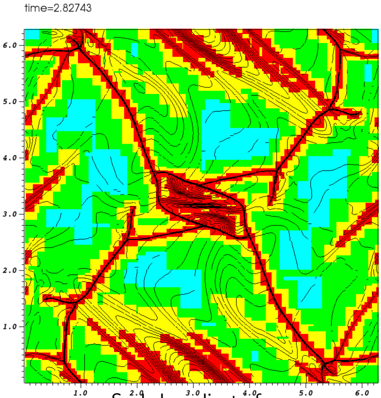
code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html

Orszag-Tang vortex

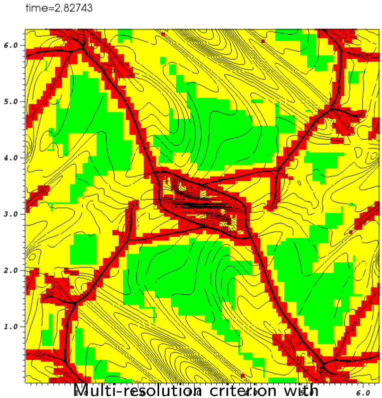
- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

$$\rho(x,y,0) = \gamma^2, \quad u_x(x,y,0) = -\sin(y), \quad u_y(x,y,0) = \sin(x), \quad u_z(x,y,0) = 0$$

$$p(x,y,0) = \gamma, \quad B_x(x,y,0) = -\sin(y), \quad B_y(x,y,0) = 2\sin(x), \quad B_z(x,y,0) = 0$$



Scaled gradient of ρ



Multi-resolution criterion with hierarchical thresholding

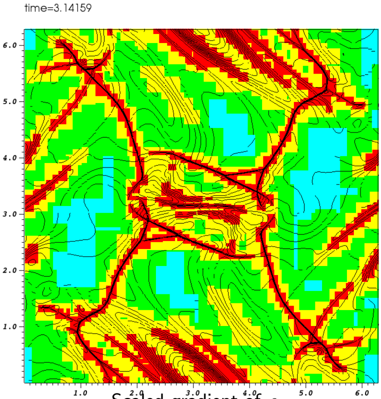
`code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html`

Orszag-Tang vortex

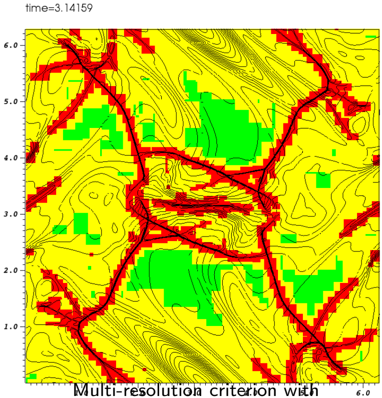
- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

$$\rho(x,y,0) = \gamma^2, \quad u_x(x,y,0) = -\sin(y), \quad u_y(x,y,0) = \sin(x), \quad u_z(x,y,0) = 0$$

$$p(x,y,0) = \gamma, \quad B_x(x,y,0) = -\sin(y), \quad B_y(x,y,0) = 2\sin(x), \quad B_z(x,y,0) = 0$$



Scaled gradient of ρ



Multi-resolution criterion with hierarchical thresholding

`code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html`

Classes

Directory `amroc/mhd` contains the integrator that is in C++ throughout:

- **EGLM2D<DataType >**: GLM method in 2d plus standard initial and boundary conditions. Internal functions for flux evaluation, MUSCL resconstruction, etc. are member functions.

  `code/amroc/doc/html/mhd/classEGLM2D.html`

- Is derived from **SchemeBase<vector_type, dim >**, which is designed for the C++ interface classes in amroc/amr/Interfaces.

  `code/amroc/doc/html/amr/classSchemeBase.html`

- **amroc/amr/Interfaces**: Provides **SchemeIntegrator<SchemeType, dim >**, **SchemeInitialCondition<SchemeType, dim >**, **SchemeBoundaryCondition<SchemeType, dim >** and further interfaces that use classes derived from **SchemeBase<vector_type, dim >** as template parameter. This provides a single-class location for new schemes in C++.

  `code/amroc/doc/html/amr/classSchemeIntegrator.html` `code/amroc/doc/html/amr/classSchemeInitialCondition.html`

- **Problem.h**: Specific simulation is defined in Problem.h only. Predefined classes specified in **MHDStdProblem.h** and **MHDProblem.h** similar but simpler as before.

  `code/amroc/doc/html/mhd/MHDStdProblem_8h.html` `code/amroc/doc/html/mhd/MHDProblem_8h_source.html`

Further hyperbolic solvers

- **amroc/rim**: Riemann invariant manifold method. 2d implementation in F77 with straightforward integration into AMROC.

  `code/amroc/doc/html/rim/files.html`

- **amroc/balans**: 2nd order accurate central difference scheme. 2d implementation in F77 and excellent template for Fortran scheme incorporation into AMROC.

  `code/amroc/doc/html/balans/files.html`

References I

[Colella and Woodward, 1984] Colella, P. and Woodward, P. (1984). The piecewise parabolic method (PPM) for gas-dynamical simulations. *J. Comput. Phys.*, 54:174–201.

[Dedner et al., 2002] Dedner, A., Kemm, F., Kröner, D., Munz, C.-D., Schnitzer, T., and Wesenberg, M. (2002). Hyperbolic divergence cleaning for the MHD equations. *J. Comput. Phys.*, 175:645–673.

[Deiterding et al., 2007] Deiterding, R., Cirak, F., Mauch, S. P., and Meiron, D. I. (2007). A virtual test facility for simulating detonation- and shock-induced deformation and fracture of thin flexible shells. *Int. J. Multiscale Computational Engineering*, 5(1):47–63.

[Deiterding et al., 2006] Deiterding, R., Radovitzky, R., Mauch, S. P., Noels, L., Cummings, J. C., and Meiron, D. I. (2006). A virtual test facility for the efficient simulation of solid materials under high energy shock-wave loading. *Engineering with Computers*, 22(3-4):325–347.

[Godlewski and Raviart, 1996] Godlewski, E. and Raviart, P.-A. (1996). *Numerical approximation of hyperbolic systems of conservation laws*. Springer Verlag, New York.

References II

[Gottlieb et al., 2001] Gottlieb, S., Shu, C.-W., and Tadmor, E. (2001). Strong stability-preserving high-order time discretization methods. *SIAM Review*, 43(1):89–112.

[Harten, 1983] Harten, A. (1983). High resolution schemes for hyperbolic conservation laws. *J. Comput. Phys.*, 49:357–393.

[Harten et al., 1976] Harten, A., Hyman, J. M., and Lax, P. D. (1976). On finite-difference approximations and entropy conditions for shocks. *Comm. Pure Appl. Math.*, 29:297–322.

[Hirsch, 1988] Hirsch, C. (1988). *Numerical computation of internal and external flows*. John Wiley & Sons, Chichester.

[Laney, 1998] Laney, C. B. (1998). *Computational gasdynamics*. Cambridge University Press, Cambridge.

[Langseth and LeVeque, 2000] Langseth, J. and LeVeque, R. (2000). A wave propagation method for three dimensional conservation laws. *J. Comput. Phys.*, 165:126–166.

References III

[LeVeque, 1997] LeVeque, R. J. (1997). Wave propagation algorithms for multidimensional hyperbolic systems. *J. Comput. Phys.*, 131(2):327–353.

[Oran and Boris, 2001] Oran, E. S. and Boris, J. P. (2001). *Numerical simulation of reactive flow*. Cambridge Univ. Press, Cambridge, 2nd edition.

[Pantano et al., 2007] Pantano, C., Deiterding, R., Hill, D. J., and Pullin, D. I. (2007). A low-numerical dissipation patch-based adaptive mesh refinement method for large-eddy simulation of compressible flows. *J. Comput. Phys.*, 221(1):63–87.

[Shu, 97] Shu, C.-W. (97). Essentially non-oscillatory and weigted essentially non-oscillatory schemes for hyperbolic conservation laws. Technical Report CR-97-206253, NASA.

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