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

High-resolution methods
  MUSCL and wave propagation
  Further methods
Outline

High-resolution methods
- MUSCL and wave propagation
- Further methods

AMROC
- Overview and basic software design
- Classes

Hyperbolic AMROC solvers
Outline

High-resolution methods
   MUSCL and wave propagation
   Further methods

AMROC
   Overview and basic software design
   Classes

Clawpack solver
   AMR examples
   Software construction
Outline

**High-resolution methods**
- MUSCL and wave propagation
- Further methods

**AMROC**
- Overview and basic software design
- Classes

**Clawpack solver**
- AMR examples
- Software construction

**WENO solver**
- Large-eddy simulation
- Software construction
Outline

High-resolution methods
  MUSCL and wave propagation
  Further methods

AMROC
  Overview and basic software design
  Classes

Clawpack solver
  AMR examples
  Software construction

WENO solver
  Large-eddy simulation
  Software construction

MHD solver
  Ideal magneto-hydrodynamics simulation
  Software design
Outline

High-resolution methods
  MUSCL and wave propagation
  Further methods

AMROC
  Overview and basic software design
  Classes

Clawpack solver
  AMR examples
  Software construction

WENO solver
  Large-eddy simulation
  Software construction

MHD solver
  Ideal magneto-hydrodynamics simulation
  Software design
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 \( H^{(\Delta t)}(Q^n; j) \) TVD if \( TV(Q^l+1) \leq TV(Q^l) \) is satisfied for all discrete sequences \( Q^n \). Herein, \( TV(Q^l) := \sum_{j \in \mathbb{Z}} |Q_{j+1}^l - Q_j^l| \).

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

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]

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

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

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

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

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]

\[
\begin{align*}
\tilde{Q}_j^L &= Q_n^j + \frac{1}{4} \left[ (1 - \omega) \Phi^{+}_{j-\frac{1}{2}} \Delta_{j-\frac{1}{2}} + (1 + \omega) \Phi^{-}_{j+\frac{1}{2}} \Delta_{j+\frac{1}{2}} \right], \\
\tilde{Q}_j^R &= Q_n^j - \frac{1}{4} \left[ (1 - \omega) \Phi^{-}_{j+\frac{1}{2}} \Delta_{j+\frac{1}{2}} + (1 + \omega) \Phi^{+}_{j-\frac{1}{2}} \Delta_{j-\frac{1}{2}} \right]
\end{align*}
\]

with \( \Delta_{j-\frac{1}{2}} = Q_n^j - Q_{n-1}^j \), \( \Delta_{j+\frac{1}{2}} = Q_n^{j+1} - Q_{n}^j \).

\[
\Phi^{+}_{j-\frac{1}{2}} := \Phi \left( r^+_{j-\frac{1}{2}} \right), \quad \Phi^{-}_{j+\frac{1}{2}} := \Phi \left( r^-_{j+\frac{1}{2}} \right)
\]

with \( r^+_{j-\frac{1}{2}} := \frac{\Delta_{j+\frac{1}{2}}}{\Delta_{j-\frac{1}{2}}} \), \( r^-_{j+\frac{1}{2}} := \frac{\Delta_{j-\frac{1}{2}}}{\Delta_{j+\frac{1}{2}}} \)

and slope limiters, e.g., Minmod

\[
\Phi(r) = \max(0, \min(r, 1))
\]

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

\[
Q_j^* = Q_n^j - \frac{1}{2} \Delta t \Delta x \left( F(Q_{n,j+1}^n, Q_j^n) - F(Q_n^j, Q_{n,j-1}^n) \right)
\]

and constructing limited values from \( Q^* \) 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

\[ \mathcal{A}^{\pm} \Delta := \hat{A}^{\pm}(\mathbf{q}_L, \mathbf{q}_R) \Delta \mathbf{q} \]

and the waves \( \mathcal{W}_m := a_m \hat{\lambda}_m \), 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 with flux limiting

Wave Propagation Method [LeVeque, 1997] is built on the flux differencing approach

\[ \mathcal{A}^\pm \Delta := \hat{\mathcal{A}}^\pm (q_L, q_R) \Delta q \]  
and the waves \( \mathcal{W}_m := a_m \hat{r}_m \), i.e.

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

Wave Propagation 1D:

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

with

\[ \tilde{\mathcal{F}}_{j+\frac{1}{2}} = \frac{1}{2} \left| \mathcal{A} \right| \left( 1 - \frac{\Delta t}{\Delta x} \left| \mathcal{A} \right| \right) \Delta_{j+\frac{1}{2}} = \frac{1}{2} \sum_{m=1}^{M} \hat{\lambda}^m_{j+\frac{1}{2}} \left( 1 - \frac{\Delta t}{\Delta x} \left| \hat{\lambda}^m_{j+\frac{1}{2}} \right| \right) \mathcal{W}^m_{j+\frac{1}{2}} \]
Wave Propagation with flux limiting

Wave Propagation Method [LeVeque, 1997] is built on the flux differencing approach
\[ A^\pm \Delta := \hat{A}^\pm (q_L, q_R) \Delta q \]
and the waves \( \mathcal{W}_m := a_m \hat{r}_m \), i.e.
\[ A^- \Delta q = \sum_{\hat{\lambda}_m < 0} \hat{\lambda}_m \mathcal{W}_m, \quad A^+ \Delta q = \sum_{\hat{\lambda}_m \geq 0} \hat{\lambda}_m \mathcal{W}_m \]

Wave Propagation 1D:
\[ Q^{n+1} = Q^n - \frac{\Delta t}{\Delta x} (A^- \Delta_{j+\frac{1}{2}} + A^+ \Delta_{j-\frac{1}{2}}) - \frac{\Delta t}{\Delta x} (\tilde{F}_{j+\frac{1}{2}} - \tilde{F}_{j-\frac{1}{2}}) \]

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

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
\[ \Theta^m_{j+\frac{1}{2}} = \begin{cases} \frac{a_m^{j-\frac{1}{2}}}{a_m^{j+\frac{1}{2}}} & , \hat{\lambda}_{j+\frac{1}{2}}^m \geq 0, \\ \frac{a_m^{j+\frac{3}{2}}}{a_m^{j+\frac{1}{2}}} & , \hat{\lambda}_{j+\frac{1}{2}}^m < 0 \end{cases} \]
Wave Propagation Method in 2D

Writing \( \tilde{A}^\pm \Delta j_{\pm 1/2} := A^+ \Delta j_{\pm 1/2} + \tilde{F}_{j_{\pm 1/2}} \) one can develop a truly two-dimensional one-step method [Langseth and LeVeque, 2000]

\[
Q^{n+1}_{jk} = Q^n_{jk} - \frac{\Delta t}{\Delta x_1} \left( \tilde{A}^- \Delta j_{\pm 1/2},k - \frac{1}{2} \frac{\Delta t}{\Delta x_2} \left[ A^- \tilde{B}^- \Delta j_{1/2},k_{1/2} + A^- \tilde{B}^+ \Delta j_{1/2},k_{1/2} \right] + \tilde{A}^+ \Delta j_{-1/2},k - \frac{1}{2} \frac{\Delta t}{\Delta x_2} \left[ A^+ \tilde{B}^- \Delta j_{-1/2},k_{1/2} + A^+ \tilde{B}^+ \Delta j_{-1/2},k_{1/2} \right] \right)
\]

\[
- \frac{\Delta t}{\Delta x_2} \left( \tilde{B}^- \Delta j_{1/2},k_{1/2} - \frac{1}{2} \frac{\Delta t}{\Delta x_1} \left[ B^- \tilde{A}^- \Delta j_{1/2},k_{1/2} + B^- \tilde{A}^+ \Delta j_{1/2},k_{1/2} \right] + \tilde{B}^+ \Delta j_{-1/2},k_{1/2} - \frac{1}{2} \frac{\Delta t}{\Delta x_1} \left[ B^+ \tilde{A}^- \Delta j_{-1/2},k_{1/2} + B^+ \tilde{A}^+ \Delta j_{-1/2},k_{1/2} \right] \right)
\]

that is stable for

\[
\left\{ \max_{j \in \mathbb{Z}} |\hat{\lambda}_{m,j_{1/2}}| \frac{\Delta t}{\Delta x_1}, \max_{k \in \mathbb{Z}} |\hat{\lambda}_{m,k_{1/2}}| \frac{\Delta t}{\Delta x_2} \right\} \leq 1, \quad \text{for all } m = 1, \ldots, M
\]
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

\[
\tilde{Q}^v_j = \alpha_v Q^n_j + \beta_v \tilde{Q}^{v-1}_j + \gamma_v \frac{\Delta t}{\Delta x} \left( F_{j+\frac{1}{2}}(\tilde{Q}^{v-1}) - F_{j-\frac{1}{2}}(\tilde{Q}^{v-1}) \right)
\]

with \( \tilde{Q}^0 := Q^n \), \( \alpha_1 = 1 \), \( \beta_1 = 0 \); and \( Q^{n+1} := \tilde{Q}^\gamma \) after final stage \( \gamma \)
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

\[
\tilde{Q}_j^v = \alpha_v Q_j^n + \beta_v \tilde{Q}_j^{v-1} + \gamma_v \frac{\Delta t}{\Delta x} \left( F_{j+\frac{1}{2}}(\tilde{Q}^{v-1}) - F_{j-\frac{1}{2}}(\tilde{Q}^{v-1}) \right) 
\]

with \( \tilde{Q}^0 := Q^n \), \( \alpha_1 = 1 \), \( \beta_1 = 0 \); and \( Q^{n+1} := \tilde{Q}^{\gamma} \) after final stage \( \gamma \)

Typical storage-efficient SSPRK(3,3):

\[
\tilde{Q}^1 = Q^n + \Delta tF(Q^n), \quad \tilde{Q}^2 = \frac{3}{4} Q^n + \frac{1}{4} \tilde{Q}^1 + \frac{1}{4} \Delta tF(\tilde{Q}^1), \\
Q^{n+1} = \frac{1}{3} Q^n + \frac{2}{3} \tilde{Q}^2 + \frac{2}{3} \Delta tF(\tilde{Q}^2)
\]
Outline

High-resolution methods
  MUSCL and wave propagation
  Further methods

AMROC
  Overview and basic software design
  Classes

Clawpack solver
  AMR examples
  Software construction

WENO solver
  Large-eddy simulation
  Software construction

MHD solver
  Ideal magneto-hydrodynamics simulation
  Software design
Overview

▶ “Adaptive Mesh Refinement in Object-oriented C++”
▶ ~ 46,000 LOC for C++ SAMR kernel, ~ 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
▶ ~ 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

- 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

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

- 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

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

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 $\Delta x_{n,l} \sim l_{\text{max}} \prod_{\kappa=l+1}^{l_{\text{max}}} r_{\kappa}$ to uniquely identify a cell within the hierarchy.
Commonalities in software design

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

Hyperbolic AMROC solvers
Commonalities in software design

- Index coordinate system based on \( \Delta x_{n,l} \cong \prod_{\kappa=1}^{l_{\text{max}}} r_{\kappa} \) to uniquely identify a cell within 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 topological relations between lists of `Patch` objects to implement synchronization, restriction, prolongation, re-distribution.
Commonalities in software design

- Index coordinate system based on $\Delta x_{n,l} \approx l_{\text{max}} \prod_{\kappa=l+1}^\infty r_{\kappa}$ to uniquely identify a cell within 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 topological relations between lists of Patch objects to implement synchronization, 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**: 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**: 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**\(<\text{VectorType}, \text{FixupType}, \text{dim}>\) implements the conservative flux correction, holds lower dimensional GridFunctions for correction terms
  
  code/amroc/doc/html/amr/classAMRFixup.html

- **AMRFlagging**\(<\text{VectorType}, \text{FixupType}, \text{FlagType}, \text{dim}>\) 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**\(<\text{VectorType}, \text{dim}>\) 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**\(<\text{VectorType}, \text{dim}>\) is an interpolation at arbitrary point location, typically used for post-processing
  
  code/amroc/doc/html/amr/classAMRInterpolation.html
Outline

High-resolution methods
  MUSCL and wave propagation
  Further methods

AMROC
  Overview and basic software design
  Classes

Clawpack solver
  AMR examples
  Software construction

WENO solver
  Large-eddy simulation
  Software construction

MHD solver
  Ideal magneto-hydrodynamics simulation
  Software design

Hyperbolic AMROC solvers
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, \( t_{\text{end}} = 2 \)
- Two levels of adaptation with \( r_{1,2} = 2 \), finest level corresponds to \( N \times N \) uniform grid
SAMR accuracy verification

Gaussian density shape
\[ \rho(x_1, x_2) = 1 + e^{-\left(\sqrt{\frac{x_1^2 + x_2^2}{R}}\right)^2} \]

is advected with constant velocities \( u_1 = u_2 = 1 \), \( p_0 = 1 \), \( R = 1/4 \)

- 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

\[ \tilde{Q}^l_{v,w} := Q^l_{ij} + f_1(Q^l_{i+1,j} - Q^l_{i-1,j}) + f_2(Q^l_{i,j+1} - Q^l_{i,j-1}) \]

with factor \( f_1 = \frac{x^v_{1,l+1} - x^i_{1,l}}{2\Delta x_{1,l}} \), \( f_2 = \frac{x^w_{2,l+1} - x^j_{2,l}}{2\Delta x_{2,l}} \) 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**

<table>
<thead>
<tr>
<th>$N$</th>
<th>Unigrid</th>
<th>SAMR - fixup</th>
<th>SAMR - no fixup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error</td>
<td>Order</td>
<td>Error</td>
</tr>
<tr>
<td>20</td>
<td>0.10946400</td>
<td>1.369</td>
<td>0.01594820</td>
</tr>
<tr>
<td>40</td>
<td>0.04239430</td>
<td>1.590</td>
<td>0.00526693</td>
</tr>
<tr>
<td>80</td>
<td>0.01408160</td>
<td>1.514</td>
<td>0.00156516</td>
</tr>
<tr>
<td>160</td>
<td>0.00492945</td>
<td>1.754</td>
<td>0.00051513</td>
</tr>
<tr>
<td>320</td>
<td>0.00146132</td>
<td>1.805</td>
<td>0.00041809</td>
</tr>
<tr>
<td>640</td>
<td>0.00041809</td>
<td>1.805</td>
<td>0.00039904</td>
</tr>
</tbody>
</table>

**Fully two-dimensional Wave Propagation Method, Minmod limiter**

<table>
<thead>
<tr>
<th>$N$</th>
<th>Unigrid</th>
<th>SAMR - fixup</th>
<th>SAMR - no fixup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error</td>
<td>Order</td>
<td>Error</td>
</tr>
<tr>
<td>20</td>
<td>0.10620000</td>
<td>1.380</td>
<td>0.01536580</td>
</tr>
<tr>
<td>40</td>
<td>0.04079600</td>
<td>1.598</td>
<td>0.00505406</td>
</tr>
<tr>
<td>80</td>
<td>0.01348250</td>
<td>1.513</td>
<td>0.00147218</td>
</tr>
<tr>
<td>160</td>
<td>0.00472301</td>
<td>1.758</td>
<td>0.00044500</td>
</tr>
<tr>
<td>320</td>
<td>0.00139611</td>
<td>1.807</td>
<td>0.00039904</td>
</tr>
<tr>
<td>640</td>
<td>0.00039904</td>
<td>1.807</td>
<td>0.00039904</td>
</tr>
</tbody>
</table>
Benchmark run: blast wave in 2D

► 2D-Wave-Propagation Method with Roe’s approximate solver
► Base grid $150 \times 150$
► 2 levels: factor 2, 4

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

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 $150 \times 150$
- 2 levels: factor 2, 4

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

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
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: visualization of refinement

$l = 0$

$l = 1$

$l = 2$

$l = 3$

$l = 4$

$l = 5$
Benchmark run 2: performance results

### Number of grids and cells

<table>
<thead>
<tr>
<th>( l )</th>
<th>( l_{\text{max}} = 2 )</th>
<th>( l_{\text{max}} = 3 )</th>
<th>( l_{\text{max}} = 4 )</th>
<th>( l_{\text{max}} = 5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Grids</td>
<td>Cells</td>
<td>Grids</td>
<td>Cells</td>
</tr>
<tr>
<td>0</td>
<td>28</td>
<td>32,768</td>
<td>28</td>
<td>32,768</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>32,768</td>
<td>14</td>
<td>32,768</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>115,408</td>
<td>49</td>
<td>116,920</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>324</td>
<td>398,112</td>
<td>420</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>1405</td>
<td>1,487,312</td>
<td>1,498</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>5,266</td>
</tr>
<tr>
<td>( \sum )</td>
<td>180,944</td>
<td>580,568</td>
<td>2,234,272</td>
<td>8,429,624</td>
</tr>
</tbody>
</table>
Benchmark run 2: performance results

Number of grids and cells

<table>
<thead>
<tr>
<th>( l )</th>
<th>( l_{\text{max}} = 2 )</th>
<th>( l_{\text{max}} = 3 )</th>
<th>( l_{\text{max}} = 4 )</th>
<th>( l_{\text{max}} = 5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grids</td>
<td>Cells</td>
<td>Grids</td>
<td>Cells</td>
<td>Grids</td>
</tr>
<tr>
<td>0</td>
<td>28</td>
<td>32,768</td>
<td>28</td>
<td>32,768</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>32,768</td>
<td>14</td>
<td>32,768</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>115,408</td>
<td>49</td>
<td>116,920</td>
</tr>
<tr>
<td>3</td>
<td>324</td>
<td>398,112</td>
<td>420</td>
<td>555,744</td>
</tr>
<tr>
<td>4</td>
<td>1405</td>
<td>1,487,312</td>
<td>1,498</td>
<td>1,795,048</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \sum )</td>
<td>180,944</td>
<td>580,568</td>
<td>2,234,272</td>
<td>8,429,624</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Task [%]</th>
<th>( l_{\text{max}} = 2 )</th>
<th>( l_{\text{max}} = 3 )</th>
<th>( l_{\text{max}} = 4 )</th>
<th>( l_{\text{max}} = 5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration</td>
<td>73.7</td>
<td>77.2</td>
<td>72.9</td>
<td>37.8</td>
</tr>
<tr>
<td>Fixup</td>
<td>2.6</td>
<td>46</td>
<td>3.1</td>
<td>58</td>
</tr>
<tr>
<td>Boundary</td>
<td>10.1</td>
<td>79</td>
<td>6.3</td>
<td>78</td>
</tr>
<tr>
<td>Recomposition</td>
<td>7.4</td>
<td>8.0</td>
<td>15.1</td>
<td>50.4</td>
</tr>
<tr>
<td>Clustering</td>
<td>0.5</td>
<td>0.6</td>
<td>0.7</td>
<td>1.0</td>
</tr>
<tr>
<td>Output/Misc</td>
<td>5.7</td>
<td>4.0</td>
<td>3.6</td>
<td>1.7</td>
</tr>
<tr>
<td>Time [min]</td>
<td>0.5</td>
<td>5.1</td>
<td>73.0</td>
<td>2100.0</td>
</tr>
<tr>
<td>Uniform [min]</td>
<td>5.4</td>
<td>160</td>
<td>( \sim 5,000 )</td>
<td>( \sim 180,000 )</td>
</tr>
<tr>
<td>Factor of AMR savings</td>
<td>11</td>
<td>31</td>
<td>69</td>
<td>86</td>
</tr>
<tr>
<td>Time steps</td>
<td>15</td>
<td>27</td>
<td>52</td>
<td>115</td>
</tr>
</tbody>
</table>
Benchmark run 2: performance results

Number of grids and cells

<table>
<thead>
<tr>
<th>l</th>
<th>$l_{\text{max}} = 2$</th>
<th>$l_{\text{max}} = 3$</th>
<th>$l_{\text{max}} = 4$</th>
<th>$l_{\text{max}} = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Grids</td>
<td>Cells</td>
<td>Grids</td>
<td>Cells</td>
</tr>
<tr>
<td>0</td>
<td>28</td>
<td>32,768</td>
<td>28</td>
<td>32,768</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>32,768</td>
<td>14</td>
<td>32,768</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>115,408</td>
<td>49</td>
<td>116,920</td>
</tr>
<tr>
<td>3</td>
<td>324</td>
<td>398,112</td>
<td>420</td>
<td>555,744</td>
</tr>
<tr>
<td>4</td>
<td>1405</td>
<td>1,487,312</td>
<td>1,498</td>
<td>1,795,048</td>
</tr>
<tr>
<td>5</td>
<td>∑180,944</td>
<td>580,568</td>
<td>2,234,272</td>
<td>8,429,624</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Task [%]</th>
<th>$l_{\text{max}} = 2$</th>
<th>$l_{\text{max}} = 3$</th>
<th>$l_{\text{max}} = 4$</th>
<th>$l_{\text{max}} = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration</td>
<td>73.7</td>
<td>77.2</td>
<td>72.9</td>
<td>37.8</td>
</tr>
<tr>
<td>Fixup</td>
<td>2.6</td>
<td>46</td>
<td>3.1</td>
<td>58</td>
</tr>
<tr>
<td>Boundary</td>
<td>10.1</td>
<td>79</td>
<td>6.3</td>
<td>78</td>
</tr>
<tr>
<td>Recomposition</td>
<td>7.4</td>
<td>8.0</td>
<td>15.1</td>
<td>50.4</td>
</tr>
<tr>
<td>Clustering</td>
<td>0.5</td>
<td>0.6</td>
<td>0.7</td>
<td>1.0</td>
</tr>
<tr>
<td>Output/Misc</td>
<td>5.7</td>
<td>4.0</td>
<td>3.6</td>
<td>1.7</td>
</tr>
<tr>
<td>Time [min]</td>
<td>0.5</td>
<td>5.1</td>
<td>73.0</td>
<td>2100.0</td>
</tr>
<tr>
<td>Uniform [min]</td>
<td>5.4</td>
<td>160</td>
<td>(\sim 5,000)</td>
<td>(\sim 180,000)</td>
</tr>
<tr>
<td>Factor of AMR savings</td>
<td>11</td>
<td>31</td>
<td>69</td>
<td>86</td>
</tr>
<tr>
<td>Time steps</td>
<td>15</td>
<td>27</td>
<td>52</td>
<td>115</td>
</tr>
</tbody>
</table>

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

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

A generic main program \texttt{amroc/clawpack/mains/amr\_main.C} instantiates Integrator<\texttt{VectorType, dim}>, InitialCondition<\texttt{VectorType, dim}>, BoundaryConditions<\texttt{VectorType, dim}>

code/amroc/doc/html/clp/amr\_main\_8C.html

\textbf{Problem.h}: Allows simulation-specific alteration in class-library style by derivation from predefined classes specified in \texttt{ClpStdProblem.h}

code/amroc/doc/html/clp/ClpStdProblem\_8h.html

\textbf{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

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

High-resolution methods
  MUSCL and wave propagation
  Further methods

AMROC
  Overview and basic software design
  Classes

Clawpack solver
  AMR examples
  Software construction

WENO solver
  Large-eddy simulation
  Software construction

MHD solver
  Ideal magneto-hydrodynamics simulation
  Software design
Favre-averaged Navier-Stokes equations

\[
\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_n} (\bar{\rho} \bar{u}_n) = 0
\]

\[
\frac{\partial}{\partial t} (\bar{\rho} \bar{u}_k) + \frac{\partial}{\partial x_n} (\bar{\rho} \bar{u}_k \bar{u}_n + \delta_{kn} \bar{p} - \bar{\tau}_{kn} + \sigma_{kn}) = 0
\]

\[
\frac{\partial \bar{\rho} \bar{E}}{\partial t} + \frac{\partial}{\partial x_n} (\bar{u}_n (\bar{\rho} \bar{E} + \bar{p}) + \bar{q}_n - \bar{\tau}_{nj} \bar{u}_j + \sigma^{e}_{n}) = 0
\]

\[
\frac{\partial}{\partial t} (\bar{\rho} \bar{Y}_i) + \frac{\partial}{\partial x_n} (\bar{\rho} \bar{Y}_i \bar{u}_n + \bar{J}^i_n + \sigma^{i}_{n}) = 0
\]

with stress tensor

\[
\bar{\tau}_{kn} = \bar{\mu} \left( \frac{\partial \bar{u}_n}{\partial x_k} + \frac{\partial \bar{u}_k}{\partial x_n} \right) - \frac{2}{3} \bar{\mu} \frac{\partial \bar{u}_j}{\partial x_j} \delta_{in},
\]

heat conduction

\[
\bar{q}_n = -\bar{\lambda} \frac{\partial \bar{T}}{\partial x_n},
\]
Favre-averaged Navier-Stokes equations

\[
\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_n} (\bar{\rho} \bar{u}_n) = 0
\]
\[
\frac{\partial}{\partial t} (\bar{\rho} \bar{u}_k) + \frac{\partial}{\partial x_n} (\bar{\rho} \bar{u}_k \bar{u}_n + \delta_{kn} \bar{\rho} - \bar{\tau}_{kn} + \sigma_{kn}) = 0
\]
\[
\frac{\partial \bar{\rho} \bar{E}}{\partial t} + \frac{\partial}{\partial x_n} (\bar{u}_n (\bar{\rho} \bar{E} + \bar{p}) + \bar{q}_n - \bar{\tau}_{nj} \bar{u}_j + \sigma_{n}^e) = 0
\]
\[
\frac{\partial}{\partial t} (\bar{\rho} \bar{Y}_i) + \frac{\partial}{\partial x_n} (\bar{\rho} \bar{Y}_i \bar{u}_n + \bar{J}_n^i + \sigma_{n}^i) = 0
\]

with stress tensor

\[
\bar{\tau}_{kn} = \bar{\mu} \left( \frac{\partial \bar{u}_n}{\partial x_k} + \frac{\partial \bar{u}_k}{\partial x_n} \right) - \frac{2}{3} \bar{\mu} \frac{\partial \bar{u}_j}{\partial x_j} \delta_{in},
\]

heat conduction

\[
\bar{q}_n = -\tilde{\lambda} \frac{\partial \bar{T}}{\partial x_n},
\]

and inter-species diffusion

\[
\bar{J}_n^i = -\bar{\rho} \bar{D}_i \frac{\partial \bar{Y}_i}{\partial x_n}
\]
Favre-averaged Navier-Stokes equations

\[
\frac{\partial \bar{p}}{\partial t} + \frac{\partial}{\partial x_n} (\bar{p} \bar{u}_n) = 0
\]

\[
\frac{\partial}{\partial t} (\bar{p} \bar{u}_k) + \frac{\partial}{\partial x_n} (\bar{p} \bar{u}_k \bar{u}_n + \delta_{kn} \bar{p} - \tilde{\tau}_{kn} + \sigma_{kn}) = 0
\]

\[
\frac{\partial \bar{E}}{\partial t} + \frac{\partial}{\partial x_n} (\bar{u}_n (\bar{p} \bar{E} + \bar{p}) + \bar{q}_n - \tilde{\tau}_{nj} \bar{u}_j + \sigma_{n}^{e}) = 0
\]

\[
\frac{\partial}{\partial t} (\bar{\rho} \bar{Y}_i) + \frac{\partial}{\partial x_n} (\bar{\rho} \bar{Y}_i \bar{u}_n + \bar{J}_n + \sigma_{n}^{i}) = 0
\]

with stress tensor

\[
\tilde{\tau}_{kn} = \bar{\mu} \left( \frac{\partial \bar{u}_n}{\partial x_k} + \frac{\partial \bar{u}_k}{\partial x_n} \right) - \frac{2}{3} \bar{\mu} \frac{\partial \bar{u}_j}{\partial x_j} \delta_{in},
\]

heat conduction

\[
\tilde{q}_n = -\bar{\lambda} \frac{\partial \bar{T}}{\partial x_n},
\]

and inter-species diffusion

\[
\tilde{J}_n = -\bar{\rho} \bar{D}_i \frac{\partial \bar{Y}_i}{\partial x_n}
\]

Favre-filtering

\[
\tilde{\phi} = \frac{\rho \phi}{\bar{\rho}} \quad \text{with} \quad \bar{\phi}(x, t; \Delta_c) = \int_{\Omega} G(x - x'; \Delta_c) \phi(x', t) dx'
\]
Numerical solution approach

- Subgrid terms $\sigma_{kn}$, $\sigma^e_n$, $\sigma^i_n$ are computed by Pullin’s stretched-vortex model
Numerical solution approach

- Subgrid terms $\sigma_{kn}$, $\sigma_{en}$, $\sigma_{in}$ are computed by Pullin’s stretched-vortex model
- Cutoff $\Delta_c$ is set to local SAMR resolution $\Delta x_l$
Numerical solution approach

▶ Subgrid terms $\sigma_{kn}$, $\sigma^e_n$, $\sigma^i_n$ are computed by Pullin’s stretched-vortex model
▶ Cutoff $\Delta_c$ is set to local SAMR resolution $\Delta x_l$
▶ 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 $\sigma_{kn}$, $\sigma^e_n$, $\sigma^i_n$ are computed by Pullin’s stretched-vortex model
- Cutoff $\Delta_c$ is set to local SAMR resolution $\Delta x_l$
- 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, $r_{1,2,3} = 2$
- $\sim 70,000$ 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 M cells in average instead of 3 M (uniform)

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

code/amroc/doc/html/apps/weno_2applications_2euler_23d_2RM_2AirSF6_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 $172 \times 56 \times 56$, $r_{1,2} = 2$, 10M cells in average instead of 3M (uniform)

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

Hyperbolic AMROC solvers

code/amroc/doc/html/apps/weno_2applications_2euler_23d_2RM_2AirSF6_2src_2Problem_8h_source.html
Flux correction for Runge-Kutta method

Recall Runge-Kutta temporal update

$$\tilde{Q}_j^v = \alpha_v Q_j^n + \beta_v \tilde{Q}_j^{v-1} + \gamma_v \frac{\Delta t}{\Delta x_k} \Delta F_k (\tilde{Q}_j^{v-1})$$
Flux correction for Runge-Kutta method

Recall Runge-Kutta temporal update

\[
\tilde{Q}_j^v = \alpha_v Q_j^n + \beta_v \tilde{Q}_j^{v-1} + \gamma_v \frac{\Delta t}{\Delta x_k} \Delta F^k(\tilde{Q}_j^{v-1})
\]

rewrite scheme as

\[
Q^{n+1} = Q^n - \sum_{v=1}^{\tau} \varphi_v \frac{\Delta t}{\Delta x_k} \Delta F^k(\tilde{Q}^{v-1}) \quad \text{with} \quad \varphi_v = \gamma_v \prod_{\nu=v+1}^{\tau} \beta_\nu
\]
Flux correction for Runge-Kutta method

Recall Runge-Kutta temporal update

$$\tilde{Q}_j^\nu = \alpha_v Q_j^n + \beta_v \tilde{Q}_j^{\nu-1} + \gamma_v \frac{\Delta t}{\Delta x_k} \Delta F_k (\tilde{Q}^{\nu-1})$$

rewrite scheme as

$$Q_n^{n+1} = Q_n^n - \sum_{\nu=1}^\tau \varphi_v \frac{\Delta t}{\Delta x_k} \Delta F_k (\tilde{Q}^{\nu-1}) \quad \text{with} \quad \varphi_v = \gamma_v \prod_{\nu=v+1}^{\nu+\tau} \beta_v$$

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

1. \( \delta F_{i-\frac{1}{2},j}^{1,l+1} := -\varphi_1 F_{i-\frac{1}{2},j}^{1,l} (\tilde{Q}^0) \), \( \delta F_{i-\frac{1}{2},j}^{1,l+1} := \delta F_{i-\frac{1}{2},j}^{1,l+1} - \sum_{\nu=2}^\tau \varphi_v F_{i-\frac{1}{2},j}^{1,l} (\tilde{Q}^{\nu-1}) \)

2. \( \delta F_{i-\frac{1}{2},j}^{1,l+1} := \delta F_{i-\frac{1}{2},j}^{1,l+1} + \frac{1}{r_{l+1}^2} \sum_{m=0}^{r_{l+1}-1} \varphi_v F_{i-\frac{1}{2},j}^{1,l+1} + \sum_{\nu=1}^{\tau} \sum_{\nu+\frac{1}{2},w+m}^\varphi (\tilde{Q}^{\nu-1} (t + \kappa \Delta t_{l+1})) \)
Flux correction for Runge-Kutta method

Recall Runge-Kutta temporal update

\[ \tilde{Q}_j^\nu = \alpha_\nu Q_j^n + \beta_\nu \tilde{Q}_j^{\nu-1} + \gamma_\nu \frac{\Delta t}{\Delta x_k} \Delta F^k(\tilde{Q}^{\nu-1}) \]

rewrite scheme as

\[ Q^{n+1} = Q^n - \sum_{\nu=1}^\tau \varphi_\nu \frac{\Delta t}{\Delta x_k} \Delta F^k(\tilde{Q}^{\nu-1}) \quad \text{with} \quad \varphi_\nu = \gamma_\nu \prod_{\nu=v+1}^{\nu=v+1} \beta_\nu \]

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

1. \( \delta F_{i-\frac{1}{2},j}^{1/l+1} := -\varphi_1 F_{i-\frac{1}{2},j}^{1,l} (\tilde{Q}^0) \), \( \delta F_{i-\frac{1}{2},j}^{1/l+1} := \delta F_{i-\frac{1}{2},j}^{1,l+1} - \sum_{\nu=2}^\tau \varphi_\nu F_{i-\frac{1}{2},j}^{1,l} (\tilde{Q}^{\nu-1}) \)

2. \( \delta F_{i-\frac{1}{2},j}^{1,l+1} := \delta F_{i-\frac{1}{2},j}^{1,l+1} + \frac{1}{r_{l+1}^2} \sum_{m=0}^{r_{l+1}-1} \sum_{\nu=1}^\tau \varphi_\nu F_{i,\nu+\frac{1}{2},w+m}^{1,l+1} (\tilde{Q}^{\nu-1}(t + \kappa \Delta t_{l+1})) \)

Storage-efficient SSPRK(3,3):

\[
\begin{array}{c|cccc}
\nu & \alpha_\nu & \beta_\nu & \gamma_\nu & \varphi_\nu \\
1 & 1 & 0 & 1 & \frac{1}{6} \\
2 & \frac{3}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{6} \\
3 & \frac{1}{2} & \frac{3}{2} & \frac{1}{2} & \frac{1}{3} \\
\end{array}
\]
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, all Fortran function names defined in `WENOStdFunctions.h`


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

High-resolution methods
  MUSCL and wave propagation
  Further methods

AMROC
  Overview and basic software design
  Classes

Clawpack solver
  AMR examples
  Software construction

WENO solver
  Large-eddy simulation
  Software construction

MHD solver
  Ideal magneto-hydrodynamics simulation
  Software design
Governing equations

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

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

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

\[
\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{u}^t \mathbf{B} - \mathbf{B}^t \mathbf{u}) = 0
\]

with equation of state

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

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]:

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

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

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

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

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

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

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

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

\[
\frac{\partial \psi}{\partial t} + c_h^2 \left( \frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y} \right) = -\frac{c_h^2}{c_p^2} \psi
\]
Orszag-Tang vortex

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

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

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

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

\[
\begin{align*}
\rho(x, y, 0) &= \gamma^2, & u_x(x, y, 0) &= -\sin(y), & u_y(x, y, 0) &= \sin(x), & u_z(x, y, 0) &= 0 \\
p(x, y, 0) &= \gamma, & B_x(x, y, 0) &= -\sin(y), & B_y(x, y, 0) &= 2\sin(x), & B_z(x, y, 0) &= 0
\end{align*}
\]
Orszag-Tang vortex

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

\[
\begin{align*}
\rho(x, y, 0) &= \gamma^2, & u_x(x, y, 0) &= -\sin(y), & u_y(x, y, 0) &= \sin(x), & u_z(x, y, 0) &= 0 \\
p(x, y, 0) &= \gamma, & B_x(x, y, 0) &= -\sin(y), & B_y(x, y, 0) &= 2\sin(x), & B_z(x, y, 0) &= 0
\end{align*}
\]

Scaled gradient of $\rho$

Multi-resolution criterion with hierarchical thresholding
Orszag-Tang vortex

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

\[
\begin{align*}
\rho(x, y, 0) &= \gamma^2, \quad u_x(x, y, 0) = -\sin(y), \quad u_y(x, y, 0) = \sin(x), \quad u_z(x, y, 0) = 0 \\
p(x, y, 0) &= \gamma, \quad B_x(x, y, 0) = -\sin(y), \quad B_y(x, y, 0) = 2\sin(x), \quad B_z(x, y, 0) = 0
\end{align*}
\]

Scaled gradient of \( \rho \)

Multi-resolution criterion with hierarchical thresholding

code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_20rszagTangVortex_2src_2Problem_8h_source.html
Ideal magneto-hydrodynamics simulation

**Orszag-Tang vortex**

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

\[
\begin{align*}
\rho(x, y, 0) &= \gamma^2, & u_x(x, y, 0) &= -\sin(y), & u_y(x, y, 0) &= \sin(x), & u_z(x, y, 0) &= 0 \\
p(x, y, 0) &= \gamma, & B_x(x, y, 0) &= -\sin(y), & B_y(x, y, 0) &= 2\sin(x), & B_z(x, y, 0) &= 0
\end{align*}
\]

Scaled gradient of $\rho$

Multi-resolution criterion with hierarchical thresholding

Hyperbolic AMROC solvers

code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html
Orszag-Tang vortex

- Adaptive solution on $50 \times 50$ 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 $\rho$

Multi-resolution criterion with hierarchical thresholding

code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_20rszagTangVortex_2src_2Problem_8h_source.html
Orszag-Tang vortex

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

$$
\begin{align*}
\rho(x, y, 0) &= \gamma^2, & u_x(x, y, 0) &= -\sin(y), & u_y(x, y, 0) &= \sin(x), & u_z(x, y, 0) &= 0 \\
p(x, y, 0) &= \gamma, & B_x(x, y, 0) &= -\sin(y), & B_y(x, y, 0) &= 2\sin(x), & B_z(x, y, 0) &= 0
\end{align*}
$$

Scaled gradient of $\rho$

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 $50 \times 50$ grid with 4 additional levels refined by $r_l = 2$
- Initial condition

\[
\begin{align*}
\rho(x, y, 0) &= \gamma^2, \quad u_x(x, y, 0) = -\sin(y), \quad u_y(x, y, 0) = \sin(x), \quad u_z(x, y, 0) = 0 \\
p(x, y, 0) &= \gamma, \quad B_x(x, y, 0) = -\sin(y), \quad B_y(x, y, 0) = 2\sin(x), \quad B_z(x, y, 0) = 0
\end{align*}
\]
Orszag-Tang vortex

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

\[
\begin{align*}
\rho(x, y, 0) &= \gamma^2, & u_x(x, y, 0) &= -\sin(y), & u_y(x, y, 0) &= \sin(x), & u_z(x, y, 0) &= 0 \\
p(x, y, 0) &= \gamma, & B_x(x, y, 0) &= -\sin(y), & B_y(x, y, 0) &= 2\sin(x), & B_z(x, y, 0) &= 0
\end{align*}
\]

Scaled gradient of $\rho$

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 $50 \times 50$ 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 $\rho$

Multi-resolution criterion with hierarchical thresholding

code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_20rszagTangVortex_2src_2Problem_8h_source.html
High-resolution methods
AMROC
Clawpack solver
WENO solver
MHD solver
References

Ideal magneto-hydrodynamics simulation

Orszag-Tang vortex

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

\[
\begin{align*}
\rho(x, y, 0) &= \gamma^2, & u_x(x, y, 0) &= -\sin(y), & u_y(x, y, 0) &= \sin(x), & u_z(x, y, 0) &= 0 \\
p(x, y, 0) &= \gamma, & B_x(x, y, 0) &= -\sin(y), & B_y(x, y, 0) &= 2 \sin(x), & B_z(x, y, 0) &= 0
\end{align*}
\]

Scaled gradient of $\rho$

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 reconstruction, 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++.
  

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


References II


References III


