Lecture 2

Hyperbolic AMROC solvers

Course *Block-structured Adaptive Mesh Refinement 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

- Conservation laws
  - Basics of finite volume methods
  - Splitting methods, second derivatives

Outline

Outline

Outline

Outline

Outline

Hyperbolic Conservation Laws

$$\frac{\partial}{\partial t}\mathbf{q}(\mathbf{x},t) + \sum_{n=1}^{d} \frac{\partial}{\partial x_n}\mathbf{f}_n(\mathbf{q}(\mathbf{x},t)) = 0\,, \quad D \subset \{(\mathbf{x},t) \in \mathbb{R}^d \times \mathbb{R}_0^+\}$$

Hyperbolic Conservation Laws

$$\frac{\partial}{\partial t}\mathbf{q}(\mathbf{x},t) + \sum_{n=1}^{d} \frac{\partial}{\partial x_n}\mathbf{f}_n(\mathbf{q}(\mathbf{x},t)) = 0\,, \quad D \subset \{(\mathbf{x},t) \in \mathbb{R}^d \times \mathbb{R}_0^+\}$$

$\mathbf{q} = \mathbf{q}(\mathbf{x},t) \in S \subset \mathbb{R}^M$ - vector of state, $\mathbf{f}_n(\mathbf{q}) \in \mathrm{C}^1(S,\mathbb{R}^M)$ - flux functions,

Hyperbolic Conservation Laws

$$\frac{\partial}{\partial t}\mathbf{q}(\mathbf{x},t) + \sum_{n=1}^{d} \frac{\partial}{\partial x_n}\mathbf{f}_n(\mathbf{q}(\mathbf{x},t)) = \mathbf{s}(\mathbf{q}(\mathbf{x},t))\,, \quad D \subset \{(\mathbf{x},t) \in \mathbb{R}^d \times \mathbb{R}_0^+\}$$

$\mathbf{q} = \mathbf{q}(\mathbf{x},t) \in S \subset \mathbb{R}^M$ - vector of state, $\mathbf{f}_n(\mathbf{q}) \in \mathrm{C}^1(S,\mathbb{R}^M)$ - flux functions, $\mathbf{s}(\mathbf{q}) \in \mathrm{C}^1(S,\mathbb{R}^M)$ - source term

Hyperbolic Conservation Laws

$$\frac{\partial}{\partial t}\mathbf{q}(\mathbf{x},t) + \sum_{n=1}^{d} \frac{\partial}{\partial x_n}\mathbf{f}_n(\mathbf{q}(\mathbf{x},t)) = \mathbf{s}(\mathbf{q}(\mathbf{x},t))\,, \quad D \subset \{(\mathbf{x},t) \in \mathbb{R}^d \times \mathbb{R}_0^+\}$$

$\mathbf{q} = \mathbf{q}(\mathbf{x},t) \in S \subset \mathbb{R}^M$ - vector of state, $\mathbf{f}_n(\mathbf{q}) \in \mathrm{C}^1(S,\mathbb{R}^M)$ - flux functions, $\mathbf{s}(\mathbf{q}) \in \mathrm{C}^1(S,\mathbb{R}^M)$ - source term

Definition (Hyperbolicity)

$\mathbf{A}(\mathbf{q},\nu) = \nu_1\mathbf{A}_1(\mathbf{q}) + \cdots + \nu_d\mathbf{A}_d(\mathbf{q})$ with $\mathbf{A}_n(\mathbf{q}) = \partial\mathbf{f}_n(\mathbf{q})/\partial\mathbf{q}$ has M real eigenvalues $\lambda_1(\mathbf{q},\nu) \le ... \le \lambda_M(\mathbf{q},\nu)$ and M linear independent right eigenvectors $\mathbf{r}_m(\mathbf{q},\nu)$.

Hyperbolic Conservation Laws

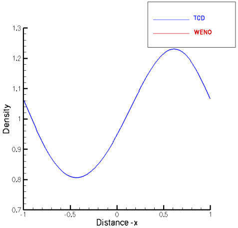
$$\frac{\partial}{\partial t}\mathbf{q}(\mathbf{x},t) + \sum_{n=1}^{d} \frac{\partial}{\partial x_n}\mathbf{f}_n(\mathbf{q}(\mathbf{x},t)) = \mathbf{s}(\mathbf{q}(\mathbf{x},t)), \quad D \subset \{(\mathbf{x},t) \in \mathbb{R}^d \times \mathbb{R}_0^+\}$$

$\mathbf{q} = \mathbf{q}(\mathbf{x},t) \in S \subset \mathbb{R}^M$ - vector of state, $\mathbf{f}_n(\mathbf{q}) \in \mathrm{C}^1(S,\mathbb{R}^M)$ - flux functions, $\mathbf{s}(\mathbf{q}) \in \mathrm{C}^1(S,\mathbb{R}^M)$ - source term

Definition (Hyperbolicity)

$\mathbf{A}(\mathbf{q},\nu) = \nu_1\mathbf{A}_1(\mathbf{q}) + \cdots + \nu_d\mathbf{A}_d(\mathbf{q})$ with $\mathbf{A}_n(\mathbf{q}) = \partial\mathbf{f}_n(\mathbf{q})/\partial\mathbf{q}$ has M real eigenvalues $\lambda_1(\mathbf{q},\nu) \le \ldots \le \lambda_M(\mathbf{q},\nu)$ and M linear independent right eigenvectors $\mathbf{r}_m(\mathbf{q},\nu)$.

If $\mathbf{f}_n(\mathbf{q})$ is nonlinear, classical solutions $\mathbf{q}(\mathbf{x},t) \in \mathrm{C}^1(D,S)$ do not generally exist, not even for $\mathbf{q}_0(\mathbf{x}) \in \mathrm{C}^1(\mathbb{R}^d,S)$ [Majda, 1984], [Godlewski and Raviart, 1996], [Kröner, 1997]



Example: Euler equations

Weak solutions

Integral form (Gauss's theorem):

$$\int_{\Omega} \mathbf{q}(\mathbf{x}, t+\Delta t)\, d\mathbf{x} - \int_{\Omega} \mathbf{q}(\mathbf{x}, t)\, d\mathbf{x}$$

$$+ \sum_{n=1}^{d} \int_{t}^{t+\Delta t} \int_{\partial\Omega} \mathbf{f}_n(\mathbf{q}(\mathbf{o}, t))\, \sigma_n(\mathbf{o})\, d\mathbf{o}\, dt = \int_{t}^{t+\Delta t} \int_{\Omega} \mathbf{s}(\mathbf{q}(\mathbf{x}, t))\, d\mathbf{x}$$

Weak solutions

Integral form (Gauss's theorem):

$$\int_\Omega \mathbf{q}(\mathbf{x}, t+\Delta t)\, d\mathbf{x} - \int_\Omega \mathbf{q}(\mathbf{x}, t)\, d\mathbf{x}$$
$$+ \sum_{n=1}^{d} \int_t^{t+\Delta t} \int_{\partial\Omega} \mathbf{f}_n(\mathbf{q}(\mathbf{o}, t))\, \sigma_n(\mathbf{o})\, d\mathbf{o}\, dt = \int_t^{t+\Delta t} \int_\Omega \mathbf{s}(\mathbf{q}(\mathbf{x}, t))\, d\mathbf{x}$$

Theorem (Weak solution)

$\mathbf{q}_0 \in \mathrm{L}^\infty_{loc}(\mathbb{R}^d, S)$. $\mathbf{q} \in \mathrm{L}^\infty_{loc}(D, S)$ *is weak solution if* $\mathbf{q}$ *satisfies*

Weak solutions

Integral form (Gauss's theorem):

$$\int_{\Omega} \mathbf{q}(\mathbf{x}, t+\Delta t)\, d\mathbf{x} - \int_{\Omega} \mathbf{q}(\mathbf{x}, t)\, d\mathbf{x} + \sum_{n=1}^{d} \int_{t}^{t+\Delta t} \int_{\partial\Omega} \mathbf{f}_n(\mathbf{q}(\mathbf{o}, t))\, \sigma_n(\mathbf{o})\, d\mathbf{o}\, dt = \int_{t}^{t+\Delta t} \int_{\Omega} \mathbf{s}(\mathbf{q}(\mathbf{x}, t))\, d\mathbf{x}$$

Theorem (Weak solution)

$\mathbf{q}_0 \in \mathrm{L}^{\infty}_{loc}(\mathbb{R}^d, S)$. $\mathbf{q} \in \mathrm{L}^{\infty}_{loc}(D, S)$ *is weak solution if* $\mathbf{q}$ *satisfies*

$$\int_0^{\infty} \int_{\mathbb{R}^d} \left[\frac{\partial \varphi}{\partial t} \cdot \mathbf{q} + \sum_{n=1}^{d} \frac{\partial \varphi}{\partial x_n} \cdot \mathbf{f}_n(\mathbf{q}) - \varphi \cdot \mathbf{s}(\mathbf{q}) \right] d\mathbf{x}\, dt + \int_{\mathbb{R}^d} \varphi(\mathbf{x}, 0) \cdot \mathbf{q}_0(\mathbf{x})\, d\mathbf{x} = 0$$

for any test function $\varphi \in \mathrm{C}^1_0(D, S)$

Examples

Euler equations

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_n}(\rho u_n) = 0$$

$$\frac{\partial}{\partial t}(\rho u_k) + \frac{\partial}{\partial x_n}(\rho u_k u_n + \delta_{kn} p) = 0 \; , \quad k = 1, \dots, d$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_n}(u_n(\rho E + p)) = 0$$

Examples

Euler equations

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_n}\big(\rho u_n\big) = 0$$

$$\frac{\partial}{\partial t}\big(\rho u_k\big) + \frac{\partial}{\partial x_n}\big(\rho u_k u_n + \delta_{kn} p\big) = 0 \;, \quad k = 1, \dots, d$$

$$\frac{\partial}{\partial t}\big(\rho E\big) + \frac{\partial}{\partial x_n}\big(u_n(\rho E + p)\big) = 0$$

with polytrope gas equation of state

$$p = (\gamma - 1)\big(\rho E - \frac{1}{2}\rho u_n u_n\big)$$

Examples

Euler equations

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_n}(\rho u_n) = 0$$

$$\frac{\partial}{\partial t}(\rho u_k) + \frac{\partial}{\partial x_n}(\rho u_k u_n + \delta_{kn} p) = 0 \;, \quad k = 1, \ldots, d$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_n}\big(u_n(\rho E + p)\big) = 0$$

with polytrope gas equation of state

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho u_n u_n\right)$$

have structure

$$\partial_t \mathbf{q}(\mathbf{x}, t) + \nabla \cdot \mathbf{f}(\mathbf{q}(\mathbf{x}, t)) = 0$$

Examples II

Navier-Stokes equations

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

$$\frac{\partial}{\partial t}\left(\rho u_k\right) + \frac{\partial}{\partial x_n}\left(\rho u_k u_n + \delta_{kn} p - \tau_{kn}\right) = 0 \;, \quad k = 1, \ldots, d$$

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

Examples II

Navier-Stokes equations

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_n}(\rho u_n) = 0$$

$$\frac{\partial}{\partial t}(\rho u_k) + \frac{\partial}{\partial x_n}(\rho u_k u_n + \delta_{kn} p - \tau_{kn}) = 0 \;, \quad k = 1, \dots, d$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_n}(u_n(\rho E + p) + q_n - \tau_{nj} u_j) = 0$$

with stress tensor

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

and heat conduction

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

Examples II

Navier-Stokes equations

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_n}(\rho u_n) = 0$$

$$\frac{\partial}{\partial t}(\rho u_k) + \frac{\partial}{\partial x_n}(\rho u_k u_n + \delta_{kn} p - \tau_{kn}) = 0 \;, \quad k = 1, \ldots, d$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_n}(u_n(\rho E + p) + q_n - \tau_{nj} u_j) = 0$$

with stress tensor

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

and heat conduction

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

have structure

$$\partial_t \mathbf{q}(\mathbf{x}, t) + \nabla \cdot \mathbf{f}(\mathbf{q}(\mathbf{x}, t)) + \nabla \cdot \mathbf{h}(\mathbf{q}(\mathbf{x}, t), \nabla \mathbf{q}(\mathbf{x}, t)) = 0$$

Examples II

Navier-Stokes equations

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_n}(\rho u_n) = 0$$

$$\frac{\partial}{\partial t}(\rho u_k) + \frac{\partial}{\partial x_n}(\rho u_k u_n + \delta_{kn} p - \tau_{kn}) = 0 \; , \quad k = 1, \dots, d$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_n}(u_n(\rho E + p) + q_n - \tau_{nj} u_j) = 0$$

with stress tensor

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

and heat conduction

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

have structure

$$\partial_t \mathbf{q}(\mathbf{x}, t) + \nabla \cdot \mathbf{f}(\mathbf{q}(\mathbf{x}, t)) + \nabla \cdot \mathbf{h}(\mathbf{q}(\mathbf{x}, t), \nabla \mathbf{q}(\mathbf{x}, t)) = 0$$

Type can be either hyperbolic or parabolic

Derivation

Assume $\partial_t \mathbf{q} + \partial_x \mathbf{f}(\mathbf{q}) + \partial_x \mathbf{h}(\mathbf{q}(\cdot, \partial_x \mathbf{q})) = \mathbf{s}(\mathbf{q})$

Derivation

Assume $\partial_t \mathbf{q} + \partial_x \mathbf{f}(\mathbf{q}) + \partial_x \mathbf{h}(\mathbf{q}(\cdot, \partial_x \mathbf{q})) = \mathbf{s}(\mathbf{q})$

Time discretization $t_n = n\Delta t$, discrete volumes
$I_j = [x_j - \frac{1}{2}\Delta x, x_j + \frac{1}{2}\Delta x[=: [x_{j-1/2}, x_{j+1/2}[$

Derivation

Assume $\partial_t \mathbf{q} + \partial_x \mathbf{f}(\mathbf{q}) + \partial_x \mathbf{h}(\mathbf{q}(\cdot, \partial_x \mathbf{q})) = \mathbf{s}(\mathbf{q})$

Time discretization $t_n = n\Delta t$, discrete volumes
$I_j = [x_j - \frac{1}{2}\Delta x, x_j + \frac{1}{2}\Delta x[=: [x_{j-1/2}, x_{j+1/2}[$

Using approximations $\mathbf{Q}_j(t) \approx \frac{1}{|I_j|} \int\limits_{I_j} \mathbf{q}(\mathbf{x}, t)\, dx, \quad \mathbf{s}(\mathbf{Q}_j(t)) \approx \frac{1}{|I_j|} \int\limits_{I_j} \mathbf{s}(\mathbf{q}(\mathbf{x}, t))\, dx$

and numerical fluxes

$\mathbf{F}\left(\mathbf{Q}_j(t), \mathbf{Q}_{j+1}(t)\right) \approx \mathbf{f}(\mathbf{q}(x_{j+1/2}, t)), \quad \mathbf{H}\left(\mathbf{Q}_j(t), \mathbf{Q}_{j+1}(t)\right) \approx \mathbf{h}(\mathbf{q}(x_{j+1/2}, t), \nabla \mathbf{q}(x_{j+1/2}, t))$

Derivation

Assume $\partial_t \mathbf{q} + \partial_x \mathbf{f}(\mathbf{q}) + \partial_x \mathbf{h}(\mathbf{q}(\cdot, \partial_x \mathbf{q})) = \mathbf{s}(\mathbf{q})$

Time discretization $t_n = n\Delta t$, discrete volumes
$I_j = [x_j - \frac{1}{2}\Delta x, x_j + \frac{1}{2}\Delta x[=: [x_{j-1/2}, x_{j+1/2}[$

Using approximations $\mathbf{Q}_j(t) \approx \frac{1}{|I_j|} \int\limits_{I_j} \mathbf{q}(\mathbf{x}, t)\, dx, \quad \mathbf{s}(\mathbf{Q}_j(t)) \approx \frac{1}{|I_j|} \int\limits_{I_j} \mathbf{s}(\mathbf{q}(\mathbf{x}, t))\, dx$

and numerical fluxes
$\mathbf{F}\left(\mathbf{Q}_j(t), \mathbf{Q}_{j+1}(t)\right) \approx \mathbf{f}(\mathbf{q}(x_{j+1/2}, t)), \quad \mathbf{H}\left(\mathbf{Q}_j(t), \mathbf{Q}_{j+1}(t)\right) \approx \mathbf{h}(\mathbf{q}(x_{j+1/2}, t), \nabla \mathbf{q}(x_{j+1/2}, t))$
yields after integration (Gauss theorem)

$$\mathbf{Q}_j(t_{n+1}) = \mathbf{Q}_j(t_n) - \frac{1}{\Delta x} \int\limits_{t_n}^{t_{n+1}} \left[\mathbf{F}\left(\mathbf{Q}_j(t), \mathbf{Q}_{j+1}(t)\right) - \mathbf{F}\left(\mathbf{Q}_{j-1}(t), \mathbf{Q}_j(t)\right)\right] dt -$$

$$\frac{1}{\Delta x} \int\limits_{t_n}^{t_{n+1}} \left[\mathbf{H}\left(\mathbf{Q}_j(t), \mathbf{Q}_{j+1}(t)\right) - \mathbf{H}\left(\mathbf{Q}_{j-1}(t), \mathbf{Q}_j(t)\right)\right] dt + \int\limits_{t_n}^{t_{n+1}} \mathbf{s}(\mathbf{Q}_j(t))\, dt$$

Derivation

Assume $\partial_t \mathbf{q} + \partial_x \mathbf{f}(\mathbf{q}) + \partial_x \mathbf{h}(\mathbf{q}(\cdot, \partial_x \mathbf{q})) = \mathbf{s}(\mathbf{q})$

Time discretization $t_n = n\Delta t$, discrete volumes
$I_j = [x_j - \frac{1}{2}\Delta x, x_j + \frac{1}{2}\Delta x[=: [x_{j-1/2}, x_{j+1/2}[$

Using approximations $\mathbf{Q}_j(t) \approx \frac{1}{|I_j|} \int\limits_{I_j} \mathbf{q}(\mathbf{x}, t)\, dx, \quad \mathbf{s}(\mathbf{Q}_j(t)) \approx \frac{1}{|I_j|} \int\limits_{I_j} \mathbf{s}(\mathbf{q}(\mathbf{x}, t))\, dx$

and numerical fluxes

$\mathbf{F}\left(\mathbf{Q}_j(t), \mathbf{Q}_{j+1}(t)\right) \approx \mathbf{f}(\mathbf{q}(x_{j+1/2}, t)), \quad \mathbf{H}\left(\mathbf{Q}_j(t), \mathbf{Q}_{j+1}(t)\right) \approx \mathbf{h}(\mathbf{q}(x_{j+1/2}, t), \nabla \mathbf{q}(x_{j+1/2}, t))$

yields after integration (Gauss theorem)

$$\mathbf{Q}_j(t_{n+1}) = \mathbf{Q}_j(t_n) - \frac{1}{\Delta x} \int\limits_{t_n}^{t_{n+1}} \left[\mathbf{F}\left(\mathbf{Q}_j(t), \mathbf{Q}_{j+1}(t)\right) - \mathbf{F}\left(\mathbf{Q}_{j-1}(t), \mathbf{Q}_j(t)\right)\right] dt -$$
$$\frac{1}{\Delta x} \int\limits_{t_n}^{t_{n+1}} \left[\mathbf{H}\left(\mathbf{Q}_j(t), \mathbf{Q}_{j+1}(t)\right) - \mathbf{H}\left(\mathbf{Q}_{j-1}(t), \mathbf{Q}_j(t)\right)\right] dt + \int\limits_{t_n}^{t_{n+1}} \mathbf{s}(\mathbf{Q}_j(t))\, dt$$

For instance:

$$\mathbf{Q}_j^{n+1} = \mathbf{Q}_j^n - \frac{\Delta t}{\Delta x}\left[\mathbf{F}\left(\mathbf{Q}_j^n, \mathbf{Q}_{j+1}^n\right) - \mathbf{F}\left(\mathbf{Q}_{j-1}^n, \mathbf{Q}_j^n\right)\right] -$$
$$\frac{\Delta t}{\Delta x}\left[\mathbf{H}\left(\mathbf{Q}_j^n, \mathbf{Q}_{j+1}^n\right) - \mathbf{H}\left(\mathbf{Q}_{j-1}^n, \mathbf{Q}_j^n\right)\right] + \Delta t \mathbf{s}(\mathbf{Q}_j^n)\, dt$$

Splitting methods

Solve homogeneous PDE and ODE successively!

$$\mathcal{H}^{(\Delta t)}: \quad \partial_t \mathbf{q} + \nabla \cdot \mathbf{f}(\mathbf{q}) = 0 \;, \quad \text{IC: } \mathbf{Q}(t_m) \stackrel{\Delta t}{\Longrightarrow} \tilde{\mathbf{Q}}$$

$$\mathcal{S}^{(\Delta t)}: \quad \partial_t \mathbf{q} = \mathbf{s}(\mathbf{q}) \;, \quad \text{IC: } \tilde{\mathbf{Q}} \stackrel{\Delta t}{\Longrightarrow} \mathbf{Q}(t_m + \Delta t)$$

Splitting methods

Solve homogeneous PDE and ODE successively!

$$\mathcal{H}^{(\Delta t)}: \quad \partial_t \mathbf{q} + \nabla \cdot \mathbf{f}(\mathbf{q}) = 0 \,, \qquad \text{IC: } \mathbf{Q}(t_m) \stackrel{\Delta t}{\Longrightarrow} \tilde{\mathbf{Q}}$$

$$\mathcal{S}^{(\Delta t)}: \quad \partial_t \mathbf{q} = \mathbf{s}(\mathbf{q}) \,, \qquad \text{IC: } \tilde{\mathbf{Q}} \stackrel{\Delta t}{\Longrightarrow} \mathbf{Q}(t_m + \Delta t)$$

1st-order Godunov splitting: $\mathbf{Q}(t_m + \Delta t) = \mathcal{S}^{(\Delta t)} \mathcal{H}^{(\Delta t)}(\mathbf{Q}(t_m))$,

Splitting methods

Solve homogeneous PDE and ODE successively!

$$\mathcal{H}^{(\Delta t)}: \quad \partial_t \mathbf{q} + \nabla \cdot \mathbf{f}(\mathbf{q}) = 0 \,, \quad \text{IC: } \mathbf{Q}(t_m) \stackrel{\Delta t}{\Longrightarrow} \tilde{\mathbf{Q}}$$

$$\mathcal{S}^{(\Delta t)}: \quad \partial_t \mathbf{q} = \mathbf{s}(\mathbf{q}) \,, \quad \text{IC: } \tilde{\mathbf{Q}} \stackrel{\Delta t}{\Longrightarrow} \mathbf{Q}(t_m + \Delta t)$$

1st-order Godunov splitting: $\mathbf{Q}(t_m + \Delta t) = \mathcal{S}^{(\Delta t)}\mathcal{H}^{(\Delta t)}(\mathbf{Q}(t_m))$,
2nd-order Strang splitting : $\mathbf{Q}(t_m + \Delta t) = \mathcal{S}^{(\frac{1}{2}\Delta t)}\mathcal{H}^{(\Delta t)}\mathcal{S}^{(\frac{1}{2}\Delta t)}(\mathbf{Q}(t_m))$

Splitting methods

Solve homogeneous PDE and ODE successively!

$$\mathcal{H}^{(\Delta t)}: \quad \partial_t \mathbf{q} + \nabla \cdot \mathbf{f}(\mathbf{q}) = 0 \,, \quad \text{IC: } \mathbf{Q}(t_m) \stackrel{\Delta t}{\Longrightarrow} \tilde{\mathbf{Q}}$$
$$\mathcal{S}^{(\Delta t)}: \quad \partial_t \mathbf{q} = \mathbf{s}(\mathbf{q}) \,, \quad \text{IC: } \tilde{\mathbf{Q}} \stackrel{\Delta t}{\Longrightarrow} \mathbf{Q}(t_m + \Delta t)$$

1st-order Godunov splitting: $\mathbf{Q}(t_m + \Delta t) = \mathcal{S}^{(\Delta t)}\mathcal{H}^{(\Delta t)}(\mathbf{Q}(t_m))$,
2nd-order Strang splitting : $\mathbf{Q}(t_m + \Delta t) = \mathcal{S}^{(\frac{1}{2}\Delta t)}\mathcal{H}^{(\Delta t)}\mathcal{S}^{(\frac{1}{2}\Delta t)}(\mathbf{Q}(t_m))$

1st-order dimensional splitting for $\mathcal{H}^{(\cdot)}$:

$$\mathcal{X}_1^{(\Delta t)}: \quad \partial_t \mathbf{q} + \partial_{x_1} \mathbf{f}_1(\mathbf{q}) = 0 \,, \quad \text{IC: } \mathbf{Q}(t_m) \stackrel{\Delta t}{\Longrightarrow} \tilde{\mathbf{Q}}^{1/2}$$
$$\mathcal{X}_2^{(\Delta t)}: \quad \partial_t \mathbf{q} + \partial_{x_2} \mathbf{f}_2(\mathbf{q}) = 0 \,, \quad \text{IC: } \tilde{\mathbf{Q}}^{1/2} \stackrel{\Delta t}{\Longrightarrow} \tilde{\mathbf{Q}}$$

[Toro, 1999]

Conservative scheme for diffusion equation

Consider $\partial_t q - c\Delta q = 0$ with $c \in \mathbb{R}^+$

Conservative scheme for diffusion equation

Consider $\partial_t q - c\Delta q = 0$ with $c \in \mathbb{R}^+$, which is readily discretized as

$$Q_{jk}^{n+1} = Q_{jk}^n + c\frac{\Delta t}{\Delta x_1^2}\left(Q_{j+1,k}^n - 2Q_{jk}^n + Q_{j-1,k}^n\right) + c\frac{\Delta t}{\Delta x_2^2}\left(Q_{j,k+1}^n - 2Q_{jk}^n + Q_{j,k-1}^n\right)$$

Conservative scheme for diffusion equation

Consider $\partial_t q - c\Delta q = 0$ with $c \in \mathbb{R}^+$, which is readily discretized as

$$Q_{jk}^{n+1} = Q_{jk}^n + c\frac{\Delta t}{\Delta x_1^2}\left(Q_{j+1,k}^n - 2Q_{jk}^n + Q_{j-1,k}^n\right) + c\frac{\Delta t}{\Delta x_2^2}\left(Q_{j,k+1}^n - 2Q_{jk}^n + Q_{j,k-1}^n\right)$$

or conservatively

$$Q_{jk}^{n+1} = Q_{jk}^n + c\frac{\Delta t}{\Delta x_1}\left(H^1_{j+\frac{1}{2},k} - H^1_{j-\frac{1}{2},k}\right) + c\frac{\Delta t}{\Delta x_2}\left(H^2_{j,k+\frac{1}{2}} - H^2_{j,k-\frac{1}{2}}\right)$$

Conservative scheme for diffusion equation

Consider $\partial_t q - c\Delta q = 0$ with $c \in \mathbb{R}^+$, which is readily discretized as

$$Q_{jk}^{n+1} = Q_{jk}^n + c\frac{\Delta t}{\Delta x_1^2}\left(Q_{j+1,k}^n - 2Q_{jk}^n + Q_{j-1,k}^n\right) + c\frac{\Delta t}{\Delta x_2^2}\left(Q_{j,k+1}^n - 2Q_{jk}^n + Q_{j,k-1}^n\right)$$

or conservatively

$$Q_{jk}^{n+1} = Q_{jk}^n + c\frac{\Delta t}{\Delta x_1}\left(H^1_{j+\frac{1}{2},k} - H^1_{j-\frac{1}{2},k}\right) + c\frac{\Delta t}{\Delta x_2}\left(H^2_{j,k+\frac{1}{2}} - H^2_{j,k-\frac{1}{2}}\right)$$

Von Neumann stability analysis: Insert single eigenmode $\hat{Q}(t)e^{ik_1x_1}e^{ik_2x_2}$ into discretization

$$\hat{Q}^{n+1} = \hat{Q}^n + C_1\left(\hat{Q}^n e^{ik_1\Delta x_1} - 2\hat{Q}^n + \hat{Q}^n e^{-ik_1\Delta x_1}\right) + C_2\left(\hat{Q}^n e^{ik_2\Delta x_2} - 2\hat{Q}^n + \hat{Q}^n e^{-ik_2\Delta x_2}\right)$$

with $C_\iota = c\frac{\Delta t}{\Delta x_\iota^2},\ \iota = 1, 2,$

Conservative scheme for diffusion equation

Consider $\partial_t q - c\Delta q = 0$ with $c \in \mathbb{R}^+$, which is readily discretized as

$$Q_{jk}^{n+1} = Q_{jk}^n + c\frac{\Delta t}{\Delta x_1^2}\left(Q_{j+1,k}^n - 2Q_{jk}^n + Q_{j-1,k}^n\right) + c\frac{\Delta t}{\Delta x_2^2}\left(Q_{j,k+1}^n - 2Q_{jk}^n + Q_{j,k-1}^n\right)$$

or conservatively

$$Q_{jk}^{n+1} = Q_{jk}^n + c\frac{\Delta t}{\Delta x_1}\left(H^1_{j+\frac{1}{2},k} - H^1_{j-\frac{1}{2},k}\right) + c\frac{\Delta t}{\Delta x_2}\left(H^2_{j,k+\frac{1}{2}} - H^2_{j,k-\frac{1}{2}}\right)$$

Von Neumann stability analysis: Insert single eigenmode $\hat{Q}(t)e^{ik_1x_1}e^{ik_2x_2}$ into discretization

$$\hat{Q}^{n+1} = \hat{Q}^n + C_1\left(\hat{Q}^n e^{ik_1\Delta x_1} - 2\hat{Q}^n + \hat{Q}^n e^{-ik_1\Delta x_1}\right) + C_2\left(\hat{Q}^n e^{ik_2\Delta x_2} - 2\hat{Q}^n + \hat{Q}^n e^{-ik_2\Delta x_2}\right)$$

with $C_\iota = c\frac{\Delta t}{\Delta x_\iota^2}$, $\iota = 1,2$, which gives after inserting $e^{ik_\iota x_\iota} = \cos(k_\iota x_\iota) + i\sin(k_\iota x_\iota)$

$$\hat{Q}^{n+1} = \hat{Q}^n\left(1 + 2C_1(\cos(k_1\Delta x_1) - 1) + 2C_2(\cos(k_2\Delta x_2) - 1)\right)$$

Conservative scheme for diffusion equation

Consider $\partial_t q - c\Delta q = 0$ with $c \in \mathbb{R}^+$, which is readily discretized as

$$Q_{jk}^{n+1} = Q_{jk}^n + c\frac{\Delta t}{\Delta x_1^2}\left(Q_{j+1,k}^n - 2Q_{jk}^n + Q_{j-1,k}^n\right) + c\frac{\Delta t}{\Delta x_2^2}\left(Q_{j,k+1}^n - 2Q_{jk}^n + Q_{j,k-1}^n\right)$$

or conservatively

$$Q_{jk}^{n+1} = Q_{jk}^n + c\frac{\Delta t}{\Delta x_1}\left(H^1_{j+\frac{1}{2},k} - H^1_{j-\frac{1}{2},k}\right) + c\frac{\Delta t}{\Delta x_2}\left(H^2_{j,k+\frac{1}{2}} - H^2_{j,k-\frac{1}{2}}\right)$$

Von Neumann stability analysis: Insert single eigenmode $\hat{Q}(t)e^{ik_1x_1}e^{ik_2x_2}$ into discretization

$$\hat{Q}^{n+1} = \hat{Q}^n + C_1\left(\hat{Q}^n e^{ik_1\Delta x_1} - 2\hat{Q}^n + \hat{Q}^n e^{-ik_1\Delta x_1}\right) + C_2\left(\hat{Q}^n e^{ik_2\Delta x_2} - 2\hat{Q}^n + \hat{Q}^n e^{-ik_2\Delta x_2}\right)$$

with $C_\iota = c\frac{\Delta t}{\Delta x_\iota^2}$, $\iota = 1,2$, which gives after inserting $e^{ik_\iota x_\iota} = \cos(k_\iota x_\iota) + i\sin(k_\iota x_\iota)$

$$\hat{Q}^{n+1} = \hat{Q}^n\left(1 + 2C_1(\cos(k_1\Delta x_1) - 1) + 2C_2(\cos(k_2\Delta x_2) - 1)\right)$$

Stability requires

$$|1 + 2C_1(\cos(k_1\Delta x_1) - 1) + 2C_2(\cos(k_2\Delta x_2) - 1)| \le 1$$

Conservative scheme for diffusion equation

Consider $\partial_t q - c\Delta q = 0$ with $c \in \mathbb{R}^+$, which is readily discretized as

$$Q_{jk}^{n+1} = Q_{jk}^n + c\frac{\Delta t}{\Delta x_1^2}\left(Q_{j+1,k}^n - 2Q_{jk}^n + Q_{j-1,k}^n\right) + c\frac{\Delta t}{\Delta x_2^2}\left(Q_{j,k+1}^n - 2Q_{jk}^n + Q_{j,k-1}^n\right)$$

or conservatively

$$Q_{jk}^{n+1} = Q_{jk}^n + c\frac{\Delta t}{\Delta x_1}\left(H^1_{j+\frac{1}{2},k} - H^1_{j-\frac{1}{2},k}\right) + c\frac{\Delta t}{\Delta x_2}\left(H^2_{j,k+\frac{1}{2}} - H^2_{j,k-\frac{1}{2}}\right)$$

Von Neumann stability analysis: Insert single eigenmode $\hat{Q}(t)e^{ik_1x_1}e^{ik_2x_2}$ into discretization

$$\hat{Q}^{n+1} = \hat{Q}^n + C_1\left(\hat{Q}^n e^{ik_1\Delta x_1} - 2\hat{Q}^n + \hat{Q}^n e^{-ik_1\Delta x_1}\right) + C_2\left(\hat{Q}^n e^{ik_2\Delta x_2} - 2\hat{Q}^n + \hat{Q}^n e^{-ik_2\Delta x_2}\right)$$

with $C_\iota = c\frac{\Delta t}{\Delta x_\iota^2}$, $\iota = 1, 2$, which gives after inserting $e^{ik_\iota x_\iota} = \cos(k_\iota x_\iota) + i\sin(k_\iota x_\iota)$

$$\hat{Q}^{n+1} = \hat{Q}^n\left(1 + 2C_1(\cos(k_1\Delta x_1) - 1) + 2C_2(\cos(k_2\Delta x_2) - 1)\right)$$

Stability requires

$$|1 + 2C_1(\cos(k_1\Delta x_1) - 1) + 2C_2(\cos(k_2\Delta x_2) - 1)| \le 1$$

i.e.

$$|1 - 4C_1 - 4C_2| \le 1$$

from which we derive the stability condition

$$0 \le c\left(\frac{\Delta t}{\Delta x_1^2} + \frac{\Delta t}{\Delta x_2^2}\right) \le \frac{1}{2}$$

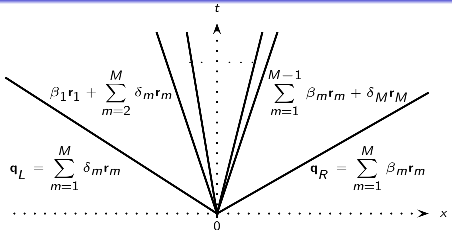
Outline

Linear upwind schemes

Consider Riemann problem

$$\frac{\partial}{\partial t}\mathbf{q}(x,t)+\mathbf{A}\frac{\partial}{\partial x}\mathbf{q}(x,t)=\mathbf{0}\,,\quad x\in\mathbb{R}\,,\quad t>0$$

Linear upwind schemes



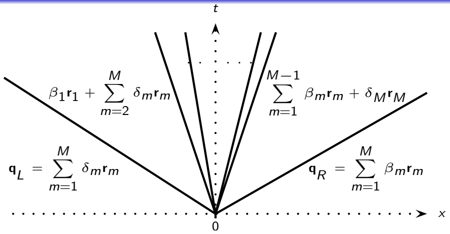
Consider Riemann problem

$$\frac{\partial}{\partial t}\mathbf{q}(x,t)+\mathbf{A}\frac{\partial}{\partial x}\mathbf{q}(x,t)=\mathbf{0}\,,\quad x\in\mathbb{R}\,,\quad t>0$$

Has exact solution

$$\mathbf{q}(x,t)=\mathbf{q}_L+\sum_{\lambda_m<x/t} a_m\mathbf{r}_m=\mathbf{q}_R-\sum_{\lambda_m\geq x/t} a_m\mathbf{r}_m=\sum_{\lambda_m\geq x/t}\delta_m\mathbf{r}_m+\sum_{\lambda_m<x/t}\beta_m\mathbf{r}_m$$

Linear upwind schemes



Consider Riemann problem

$$\frac{\partial}{\partial t}\mathbf{q}(x,t)+\mathbf{A}\frac{\partial}{\partial x}\mathbf{q}(x,t)=\mathbf{0}\,,\quad x\in\mathbb{R}\,,\quad t>0$$

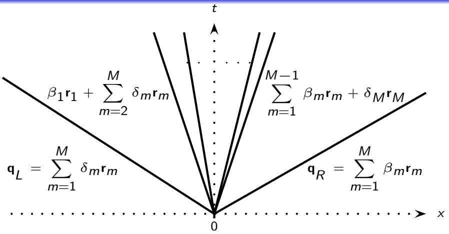
Has exact solution

$$\mathbf{q}(x,t)=\mathbf{q}_L+\sum_{\lambda_m<x/t} a_m\mathbf{r}_m=\mathbf{q}_R-\sum_{\lambda_m\ge x/t} a_m\mathbf{r}_m=\sum_{\lambda_m\ge x/t}\delta_m\mathbf{r}_m+\sum_{\lambda_m<x/t}\beta_m\mathbf{r}_m$$

Use Riemann problem to evaluate numerical flux $\mathbf{F}(\mathbf{q}_L,\mathbf{q}_R):=\mathbf{f}(\mathbf{q}(0,t))=\mathbf{A}\mathbf{q}(0,t)$ as

$$\mathbf{F}(\mathbf{q}_L,\mathbf{q}_R)=\mathbf{A}\mathbf{q}_L+\sum_{\lambda_m<0} a_m\lambda_m\mathbf{r}_m=\mathbf{A}\mathbf{q}_R-\sum_{\lambda_m\ge 0} a_m\lambda_m\mathbf{r}_m=\sum_{\lambda_m\ge 0}\delta_m\lambda_m\mathbf{r}_m+\sum_{\lambda_m<0}\beta_m\lambda_m\mathbf{r}_m$$

Linear upwind schemes



Consider Riemann problem

$$\frac{\partial}{\partial t}\mathbf{q}(x,t)+\mathbf{A}\frac{\partial}{\partial x}\mathbf{q}(x,t)=\mathbf{0}\,,\quad x\in\mathbb{R}\,,\quad t>0$$

Has exact solution

$$\mathbf{q}(x,t)=\mathbf{q}_L+\sum_{\lambda_m<x/t} a_m\mathbf{r}_m=\mathbf{q}_R-\sum_{\lambda_m\geq x/t} a_m\mathbf{r}_m=\sum_{\lambda_m\geq x/t}\delta_m\mathbf{r}_m+\sum_{\lambda_m<x/t}\beta_m\mathbf{r}_m$$

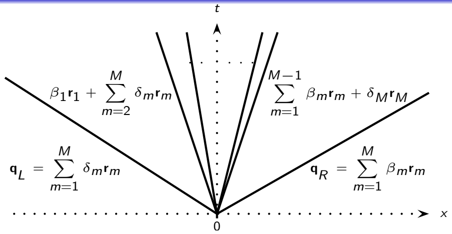
Use Riemann problem to evaluate numerical flux $\mathbf{F}(\mathbf{q}_L,\mathbf{q}_R):=\mathbf{f}(\mathbf{q}(0,t))=\mathbf{A}\mathbf{q}(0,t)$ as

$$\mathbf{F}(\mathbf{q}_L,\mathbf{q}_R)=\mathbf{A}\mathbf{q}_L+\sum_{\lambda_m<0} a_m\lambda_m\mathbf{r}_m=\mathbf{A}\mathbf{q}_R-\sum_{\lambda_m\geq 0} a_m\lambda_m\mathbf{r}_m=\sum_{\lambda_m\geq 0}\delta_m\lambda_m\mathbf{r}_m+\sum_{\lambda_m<0}\beta_m\lambda_m\mathbf{r}_m$$

Use $\quad \lambda_m^+=\max(\lambda_m,0)\,,\qquad \lambda_m^-=\min(\lambda_m,0)$

to define $\quad \mathbf{\Lambda}^+:=\mathrm{diag}(\lambda_1^+,\ldots,\lambda_M^+)\,,\qquad \mathbf{\Lambda}^-:=\mathrm{diag}(\lambda_1^-,\ldots,\lambda_M^-)$

Linear upwind schemes



Consider Riemann problem

$$\frac{\partial}{\partial t}\mathbf{q}(x,t)+\mathbf{A}\frac{\partial}{\partial x}\mathbf{q}(x,t)=\mathbf{0}\,,\quad x\in\mathbb{R}\,,\quad t>0$$

Has exact solution

$$\mathbf{q}(x,t)=\mathbf{q}_L+\sum_{\lambda_m<x/t} a_m\mathbf{r}_m=\mathbf{q}_R-\sum_{\lambda_m\geq x/t} a_m\mathbf{r}_m=\sum_{\lambda_m\geq x/t}\delta_m\mathbf{r}_m+\sum_{\lambda_m<x/t}\beta_m\mathbf{r}_m$$

Use Riemann problem to evaluate numerical flux $\mathbf{F}(\mathbf{q}_L,\mathbf{q}_R):=\mathbf{f}(\mathbf{q}(0,t))=\mathbf{A}\mathbf{q}(0,t)$ as

$$\mathbf{F}(\mathbf{q}_L,\mathbf{q}_R)=\mathbf{A}\mathbf{q}_L+\sum_{\lambda_m<0} a_m\lambda_m\mathbf{r}_m=\mathbf{A}\mathbf{q}_R-\sum_{\lambda_m\geq 0} a_m\lambda_m\mathbf{r}_m=\sum_{\lambda_m\geq 0}\delta_m\lambda_m\mathbf{r}_m+\sum_{\lambda_m<0}\beta_m\lambda_m\mathbf{r}_m$$

Use $\quad \lambda_m^+=\max(\lambda_m,0)\,,\qquad \lambda_m^-=\min(\lambda_m,0)$

to define $\quad \mathbf{\Lambda}^+:=\mathrm{diag}(\lambda_1^+,\ldots,\lambda_M^+)\,,\qquad \mathbf{\Lambda}^-:=\mathrm{diag}(\lambda_1^-,\ldots,\lambda_M^-)$

and $\quad \mathbf{A}^+:=\mathbf{R}\,\mathbf{\Lambda}^+\,\mathbf{R}^{-1}\,,\qquad \mathbf{A}^-:=\mathbf{R}\,\mathbf{\Lambda}^-\,\mathbf{R}^{-1}\qquad$ which gives

$$\mathbf{F}(\mathbf{q}_L,\mathbf{q}_R)=\mathbf{A}\mathbf{q}_L+\mathbf{A}^-\Delta\mathbf{q}=\mathbf{A}\mathbf{q}_R-\mathbf{A}^+\Delta\mathbf{q}=\mathbf{A}^+\mathbf{q}_L+\mathbf{A}^-\mathbf{q}_R$$

with $\Delta\mathbf{q}=\mathbf{q}_R-\mathbf{q}_L$

Flux difference splitting

Godunov-type scheme with $\Delta \mathbf{Q}^n_{j+1/2} = \mathbf{Q}^n_{j+1} - \mathbf{Q}^n_j$

$$\mathbf{Q}^{n+1}_j = \mathbf{Q}^n_j - \frac{\Delta t}{\Delta x}\left(\mathbf{A}^- \Delta \mathbf{Q}^n_{j+1/2} + \mathbf{A}^+ \Delta \mathbf{Q}^n_{j-1/2}\right)$$

Flux difference splitting

Godunov-type scheme with $\Delta\mathbf{Q}^n_{j+1/2} = \mathbf{Q}^n_{j+1} - \mathbf{Q}^n_j$

$$\mathbf{Q}^{n+1}_j = \mathbf{Q}^n_j - \frac{\Delta t}{\Delta x}\left(\mathbf{A}^-\Delta\mathbf{Q}^n_{j+1/2} + \mathbf{A}^+\Delta\mathbf{Q}^n_{j-1/2}\right)$$

Use linearization $\bar{\mathbf{f}}(\bar{\mathbf{q}}) = \hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)\bar{\mathbf{q}}$ and construct scheme for nonlinear problem as

$$\mathbf{Q}^{n+1}_j = \mathbf{Q}^n_j - \frac{\Delta t}{\Delta x}\left(\hat{\mathbf{A}}^-(\mathbf{Q}^n_j, \mathbf{Q}^n_{j+1})\Delta\mathbf{Q}^n_{j+\frac{1}{2}} + \hat{\mathbf{A}}^+(\mathbf{Q}^n_{j-1}, \mathbf{Q}^n_j)\Delta\mathbf{Q}^n_{j-\frac{1}{2}}\right)$$

Flux difference splitting

Godunov-type scheme with $\Delta \mathbf{Q}^n_{j+1/2} = \mathbf{Q}^n_{j+1} - \mathbf{Q}^n_j$

$$\mathbf{Q}^{n+1}_j = \mathbf{Q}^n_j - \frac{\Delta t}{\Delta x}\left(\mathbf{A}^- \Delta \mathbf{Q}^n_{j+1/2} + \mathbf{A}^+ \Delta \mathbf{Q}^n_{j-1/2}\right)$$

Use linearization $\bar{\mathbf{f}}(\bar{\mathbf{q}}) = \hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)\bar{\mathbf{q}}$ and construct scheme for nonlinear problem as

$$\mathbf{Q}^{n+1}_j = \mathbf{Q}^n_j - \frac{\Delta t}{\Delta x}\left(\hat{\mathbf{A}}^-(\mathbf{Q}^n_j, \mathbf{Q}^n_{j+1})\Delta \mathbf{Q}^n_{j+\frac{1}{2}} + \hat{\mathbf{A}}^+(\mathbf{Q}^n_{j-1}, \mathbf{Q}^n_j)\Delta \mathbf{Q}^n_{j-\frac{1}{2}}\right)$$

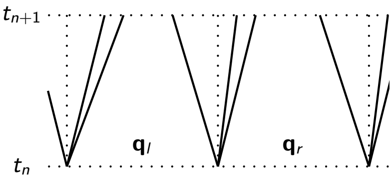
stability condition

$$\max_{j\in\mathbb{Z}} |\hat{\lambda}_{m,j+\frac{1}{2}}| \frac{\Delta t}{\Delta x} \le 1 \;, \quad \text{for all } m = 1, \ldots, M$$

[LeVeque, 1992]

Roe's approximate Riemann solver

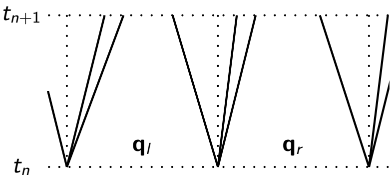
Choosing $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ [Roe, 1981]:



Roe's approximate Riemann solver

Choosing $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ [Roe, 1981]:

(i) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ has real eigenvalues

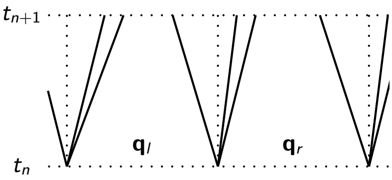


Roe's approximate Riemann solver

Choosing $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ [Roe, 1981]:

(i) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ has real eigenvalues

(ii) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R) \to \dfrac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}}$ as $\mathbf{q}_L, \mathbf{q}_R \to \mathbf{q}$



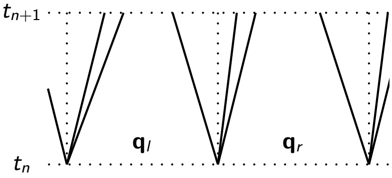
Roe's approximate Riemann solver

Choosing $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ [Roe, 1981]:

(i) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ has real eigenvalues

(ii) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R) \to \dfrac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}}$ as $\mathbf{q}_L, \mathbf{q}_R \to \mathbf{q}$

(iii) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)\Delta\mathbf{q} = \mathbf{f}(\mathbf{q}_R) - \mathbf{f}(\mathbf{q}_L)$



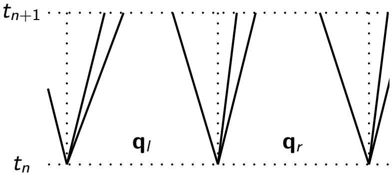
Roe's approximate Riemann solver

Choosing $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ [Roe, 1981]:

(i) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ has real eigenvalues

(ii) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R) \to \dfrac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}}$ as $\mathbf{q}_L, \mathbf{q}_R \to \mathbf{q}$

(iii) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)\Delta\mathbf{q} = \mathbf{f}(\mathbf{q}_R) - \mathbf{f}(\mathbf{q}_L)$



For Euler equations:

$$\hat{\rho} = \frac{\sqrt{\rho_L}\rho_R + \sqrt{\rho_R}\rho_L}{\sqrt{\rho_L} + \sqrt{\rho_R}} = \sqrt{\rho_L \rho_R} \quad \text{and} \quad \hat{v} = \frac{\sqrt{\rho_L} v_L + \sqrt{\rho_R} v_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad \text{for } v = u_n, H$$

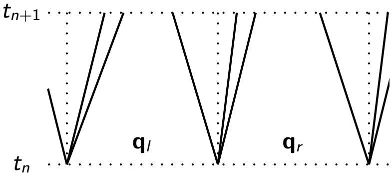
Roe's approximate Riemann solver

Choosing $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ [Roe, 1981]:

(i) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)$ has real eigenvalues

(ii) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R) \to \dfrac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}}$ as $\mathbf{q}_L, \mathbf{q}_R \to \mathbf{q}$

(iii) $\hat{\mathbf{A}}(\mathbf{q}_L, \mathbf{q}_R)\Delta\mathbf{q} = \mathbf{f}(\mathbf{q}_R) - \mathbf{f}(\mathbf{q}_L)$



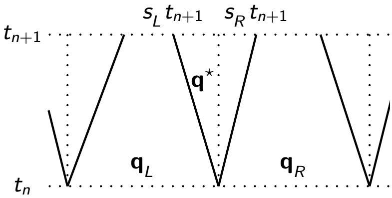
For Euler equations:

$$\hat{\rho} = \frac{\sqrt{\rho_L}\rho_R + \sqrt{\rho_R}\rho_L}{\sqrt{\rho_L} + \sqrt{\rho_R}} = \sqrt{\rho_L \rho_R} \quad \text{and} \quad \hat{v} = \frac{\sqrt{\rho_L} v_L + \sqrt{\rho_R} v_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad \text{for } v = u_n, H$$

Wave decomposition: $\Delta\mathbf{q} = \mathbf{q}_r - \mathbf{q}_l = \sum_m a_m \hat{\mathbf{r}}_m$

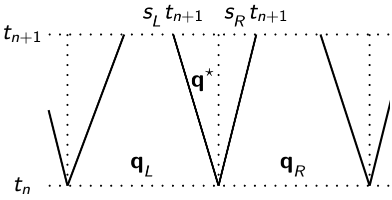
$$\begin{aligned} \mathbf{F}(\mathbf{q}_L, \mathbf{q}_R) &= \mathbf{f}(\mathbf{q}_L) + \sum_{\hat{\lambda}_m < 0} \hat{\lambda}_m \, a_m \, \hat{\mathbf{r}}_m = \mathbf{f}(\mathbf{q}_R) - \sum_{\hat{\lambda}_m \geq 0} \hat{\lambda}_m \, a_m \, \hat{\mathbf{r}}_m \\ &= \frac{1}{2}\left(\mathbf{f}(\mathbf{q}_L) + \mathbf{f}(\mathbf{q}_R) - \sum_m |\hat{\lambda}_m| \, a_m \, \hat{\mathbf{r}}_m \right) \end{aligned}$$

Harten-Lax-Van Leer (HLL) approximate Riemann solver



$$\bar{\mathbf{q}}(x,t) = \begin{cases} \mathbf{q}_L\,, & x < s_L\, t \\ \mathbf{q}^\star\,, & s_L\, t \le x \le s_R\, t \\ \mathbf{q}_R\,, & x > s_R\, t \end{cases}$$

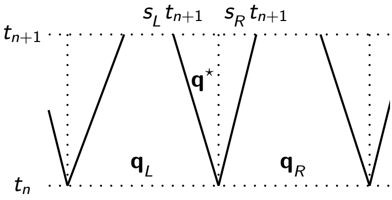
Harten-Lax-Van Leer (HLL) approximate Riemann solver



$$\bar{\mathbf{q}}(x,t) = \begin{cases} \mathbf{q}_L\,, & x < s_L\, t \\ \mathbf{q}^\star\,, & s_L\, t \le x \le s_R\, t \\ \mathbf{q}_R\,, & x > s_R\, t \end{cases}$$

$$\mathbf{F}_{HLL}(\mathbf{q}_L, \mathbf{q}_R) = \begin{cases} \mathbf{f}(\mathbf{q}_L)\,, & 0 < s_L\,, \\ \dfrac{s_R \mathbf{f}(\mathbf{q}_L) - s_L \mathbf{f}(\mathbf{q}_R) + s_L s_R (\mathbf{q}_R - \mathbf{q}_L)}{s_R - s_L}\,, & s_L \le 0 \le s_R\,, \\ \mathbf{f}(\mathbf{q}_R)\,, & 0 > s_R\,, \end{cases}$$

Harten-Lax-Van Leer (HLL) approximate Riemann solver



$$\bar{\mathbf{q}}(x,t) = \begin{cases} \mathbf{q}_L \,, & x < s_L\, t \\ \mathbf{q}^\star \,, & s_L\, t \le x \le s_R\, t \\ \mathbf{q}_R \,, & x > s_R\, t \end{cases}$$

$$\mathbf{F}_{HLL}(\mathbf{q}_L, \mathbf{q}_R) = \begin{cases} \mathbf{f}(\mathbf{q}_L) \,, & 0 < s_L \,, \\ \dfrac{s_R \mathbf{f}(\mathbf{q}_L) - s_L \mathbf{f}(\mathbf{q}_R) + s_L s_R (\mathbf{q}_R - \mathbf{q}_L)}{s_R - s_L} \,, & s_L \le 0 \le s_R \,, \\ \mathbf{f}(\mathbf{q}_R) \,, & 0 > s_R \,, \end{cases}$$

Euler equations:

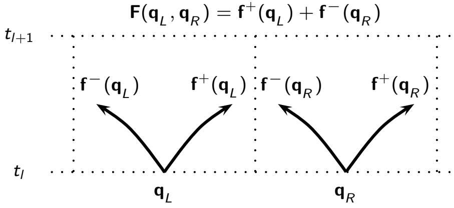
$$s_L = \min(u_{1,L} - c_L, u_{1,R} - c_R) \,, \quad s_R = \max(u_{1,L} + c_l, u_{1,R} + c_R)$$

[Toro, 1999], HLLC: [Toro et al., 1994]

Flux vector splitting

Splitting

$$\mathbf{f}(\mathbf{q}) = \mathbf{f}^{+}(\mathbf{q}) + \mathbf{f}^{-}(\mathbf{q})$$



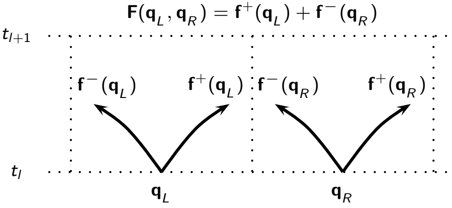
Flux vector splitting

Splitting

$$\mathbf{f}(\mathbf{q}) = \mathbf{f}^+(\mathbf{q}) + \mathbf{f}^-(\mathbf{q})$$

derived under restriction $\hat{\lambda}_m^+ \geq 0$ and $\hat{\lambda}_m^- \leq 0$ for all $m = 1, \ldots, M$ for

$$\hat{\mathbf{A}}^+(\mathbf{q}) = \frac{\partial \mathbf{f}^+(\mathbf{q})}{\partial \mathbf{q}}, \quad \hat{\mathbf{A}}^-(\mathbf{q}) = \frac{\partial \mathbf{f}^-(\mathbf{q})}{\partial \mathbf{q}}$$



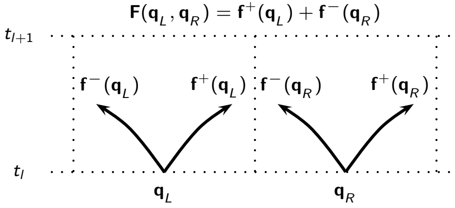
Flux vector splitting

Splitting

$$\mathbf{f}(\mathbf{q}) = \mathbf{f}^+(\mathbf{q}) + \mathbf{f}^-(\mathbf{q})$$

derived under restriction $\hat{\lambda}_m^+ \ge 0$ and $\hat{\lambda}_m^- \le 0$ for all $m = 1, \ldots, M$ for

$$\hat{\mathbf{A}}^+(\mathbf{q}) = \frac{\partial \mathbf{f}^+(\mathbf{q})}{\partial \mathbf{q}}, \quad \hat{\mathbf{A}}^-(\mathbf{q}) = \frac{\partial \mathbf{f}^-(\mathbf{q})}{\partial \mathbf{q}}$$



plus reproduction of regular upwinding

$$\begin{array}{lllllllll} \mathbf{f}^+(\mathbf{q}) & = & \mathbf{f}(\mathbf{q}), & \mathbf{f}^-(\mathbf{q}) & = & \mathbf{0} & \text{if} & \lambda_m \ge 0 & \text{for all} \quad m = 1, \ldots, M \\ \mathbf{f}^+(\mathbf{q}) & = & \mathbf{0}, & \mathbf{f}^-(\mathbf{q}) & = & \mathbf{f}(\mathbf{q}) & \text{if} & \lambda_m \le 0 & \text{for all} \quad m = 1, \ldots, M \end{array}$$

Then use

$$\mathbf{F}(\mathbf{q}_L, \mathbf{q}_R) = \mathbf{f}^+(\mathbf{q}_L) + \mathbf{f}^-(\mathbf{q}_R)$$

Steger-Warming

Required $\mathbf{f}(\mathbf{q}) = \mathbf{A}(\mathbf{q})\,\mathbf{q}$

Steger-Warming

Required $\mathbf{f}(\mathbf{q}) = \mathbf{A}(\mathbf{q})\,\mathbf{q}$

$$\lambda_m^+ = \frac{1}{2}\left(\lambda_m + |\lambda_m|\right) \qquad \lambda_m^- = \frac{1}{2}\left(\lambda_m - |\lambda_m|\right)$$

$$\mathbf{A}^+(\mathbf{q}) := \mathbf{R}(\mathbf{q})\,\mathbf{\Lambda}^+(\mathbf{q})\,\mathbf{R}^{-1}(\mathbf{q})\,, \qquad \mathbf{A}^-(\mathbf{q}) := \mathbf{R}(\mathbf{q})\,\mathbf{\Lambda}^-(\mathbf{q})\,\mathbf{R}^{-1}(\mathbf{q})$$

Steger-Warming

Required $\mathbf{f}(\mathbf{q}) = \mathbf{A}(\mathbf{q})\,\mathbf{q}$

$$\lambda_m^+ = \frac{1}{2}\left(\lambda_m + |\lambda_m|\right) \qquad \lambda_m^- = \frac{1}{2}\left(\lambda_m - |\lambda_m|\right)$$

$$\mathbf{A}^+(\mathbf{q}) := \mathbf{R}(\mathbf{q})\,\mathbf{\Lambda}^+(\mathbf{q})\,\mathbf{R}^{-1}(\mathbf{q})\,, \qquad \mathbf{A}^-(\mathbf{q}) := \mathbf{R}(\mathbf{q})\,\mathbf{\Lambda}^-(\mathbf{q})\,\mathbf{R}^{-1}(\mathbf{q})$$

Gives

$$\mathbf{f}(\mathbf{q}) = \mathbf{A}^+(\mathbf{q})\,\mathbf{q} + \mathbf{A}^-(\mathbf{q})\,\mathbf{q}$$

and the numerical flux

$$\mathbf{F}(\mathbf{q}_L, \mathbf{q}_R) = \mathbf{A}^+(\mathbf{q}_L)\,\mathbf{q}_L + \mathbf{A}^-(\mathbf{q}_R)\,\mathbf{q}_R$$

Steger-Warming

Required $\mathbf{f}(\mathbf{q}) = \mathbf{A}(\mathbf{q})\,\mathbf{q}$

$$\lambda_m^+ = \frac{1}{2}\left(\lambda_m + |\lambda_m|\right) \qquad \lambda_m^- = \frac{1}{2}\left(\lambda_m - |\lambda_m|\right)$$

$$\mathbf{A}^+(\mathbf{q}) := \mathbf{R}(\mathbf{q})\,\boldsymbol{\Lambda}^+(\mathbf{q})\,\mathbf{R}^{-1}(\mathbf{q})\,, \qquad \mathbf{A}^-(\mathbf{q}) := \mathbf{R}(\mathbf{q})\,\boldsymbol{\Lambda}^-(\mathbf{q})\,\mathbf{R}^{-1}(\mathbf{q})$$

Gives

$$\mathbf{f}(\mathbf{q}) = \mathbf{A}^+(\mathbf{q})\,\mathbf{q} + \mathbf{A}^-(\mathbf{q})\,\mathbf{q}$$

and the numerical flux

$$\mathbf{F}(\mathbf{q}_L, \mathbf{q}_R) = \mathbf{A}^+(\mathbf{q}_L)\,\mathbf{q}_L + \mathbf{A}^-(\mathbf{q}_R)\,\mathbf{q}_R$$

Jacobians of the split fluxes are identical to $\mathbf{A}^\pm(\mathbf{q})$ only in linear case

$$\frac{\partial \mathbf{f}^\pm(\mathbf{q})}{\partial \mathbf{q}} = \frac{\partial\left(\mathbf{A}^\pm(\mathbf{q})\,\mathbf{q}\right)}{\partial \mathbf{q}} = \mathbf{A}^\pm(\mathbf{q}) + \frac{\partial \mathbf{A}^\pm(\mathbf{q})}{\partial \mathbf{q}}\,\mathbf{q}$$

Further methods: Van Leer FVS [Toro, 1999], AUSM [Wada and Liou, 1997]

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]

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

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

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*

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

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.

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

Wave Propagation with flux limiting

Wave Propagation Method [LeVeque, 1997] is built on the flux differencing approach $\mathcal{A}^{\pm}\Delta := \hat{\mathbf{A}}^{\pm}(\mathbf{q}_L, \mathbf{q}_R)\Delta\mathbf{q}$ 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

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

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

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

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 $\tilde{\mathcal{A}}^{\pm}\Delta_{j\pm 1/2} := \mathcal{A}^{+}\Delta_{j\pm 1/2} + \tilde{\mathbf{F}}_{j\pm 1/2}$ one can develop a truly two-dimensional one-step method [Langseth and LeVeque, 2000]

$$
\begin{aligned}
\mathbf{Q}_{jk}^{n+1} = \mathbf{Q}_{jk}^{n} - \frac{\Delta t}{\Delta x_1}\Bigg(&\tilde{\mathcal{A}}^{-}\Delta_{j+\frac{1}{2},k} - \frac{1}{2}\frac{\Delta t}{\Delta x_2}\left[\mathcal{A}^{-}\tilde{\mathcal{B}}^{-}\Delta_{j+1,k+\frac{1}{2}} + \mathcal{A}^{-}\tilde{\mathcal{B}}^{+}\Delta_{j+1,k-\frac{1}{2}}\right] + \\
&\tilde{\mathcal{A}}^{+}\Delta_{j-\frac{1}{2},k} - \frac{1}{2}\frac{\Delta t}{\Delta x_2}\left[\mathcal{A}^{+}\tilde{\mathcal{B}}^{-}\Delta_{j-1,k+\frac{1}{2}} + \mathcal{A}^{+}\tilde{\mathcal{B}}^{+}\Delta_{j-1,k-\frac{1}{2}}\right]\Bigg) \\
- \frac{\Delta t}{\Delta x_2}\Bigg(&\tilde{\mathcal{B}}^{-}\Delta_{j,k+\frac{1}{2}} - \frac{1}{2}\frac{\Delta t}{\Delta x_1}\left[\mathcal{B}^{-}\tilde{\mathcal{A}}^{-}\Delta_{j+\frac{1}{2},k+1} + \mathcal{B}^{-}\tilde{\mathcal{A}}^{+}\Delta_{j-\frac{1}{2},k+1}\right] + \\
&\tilde{\mathcal{B}}^{+}\Delta_{j,k-\frac{1}{2}} - \frac{1}{2}\frac{\Delta t}{\Delta x_1}\left[\mathcal{B}^{+}\tilde{\mathcal{A}}^{-}\Delta_{j+\frac{1}{2},k-1} + \mathcal{B}^{+}\tilde{\mathcal{A}}^{+}\Delta_{j-\frac{1}{2},k-1}\right]\Bigg)
\end{aligned}
$$

that is stable for

$$
\left\{\max_{j\in\mathbb{Z}}|\hat{\lambda}_{m,j+\frac{1}{2}}|\frac{\Delta t}{\Delta x_1}, \max_{k\in\mathbb{Z}}|\hat{\lambda}_{m,k+\frac{1}{2}}|\frac{\Delta t}{\Delta x_2}\right\} \le 1\,, \quad \text{for all } m = 1,\ldots,M
$$

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

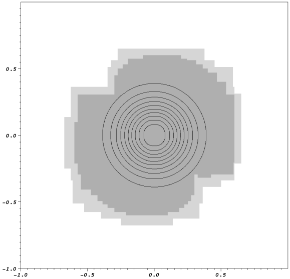
SAMR accuracy verification

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



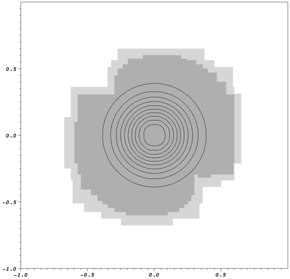
SAMR accuracy verification

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 $f_1 = \dfrac{x^v_{1,l+1} - x^i_{1,l}}{2\Delta x_{1,l}}$, $f_2 = \dfrac{x^w_{2,l+1} - x^j_{2,l}}{2\Delta x_{2,l}}$ 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

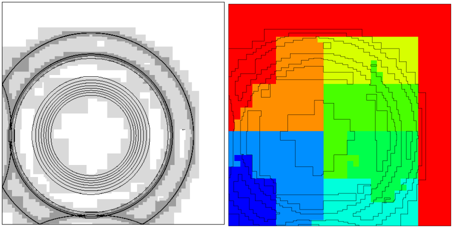
Fully two-dimensional Wave Propagation Method, Minmod limiter

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

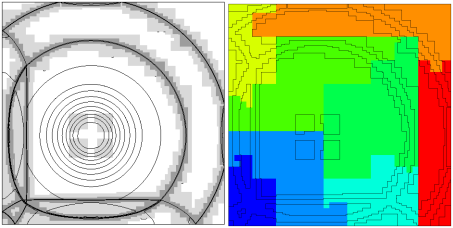
Benchmark run: blast wave in 2D

- 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

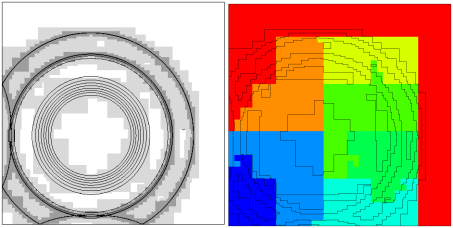


After 79 time steps

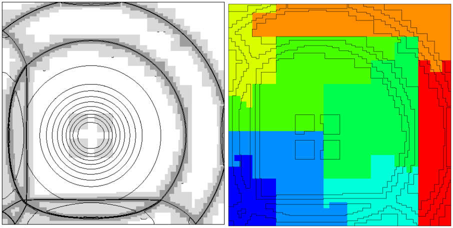
Benchmark run: blast wave in 2D

- 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`

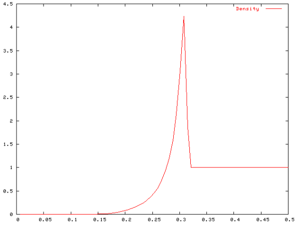
Benchmark run 2: point-explosion in 3D

- 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

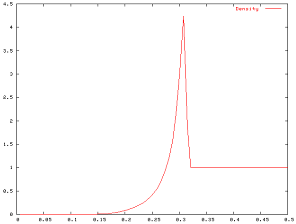
$l_{\max} = 4$ solution



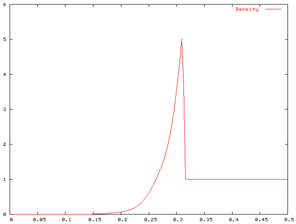
Benchmark run 2: point-explosion in 3D

- 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

$l_{\max} = 4$ solution

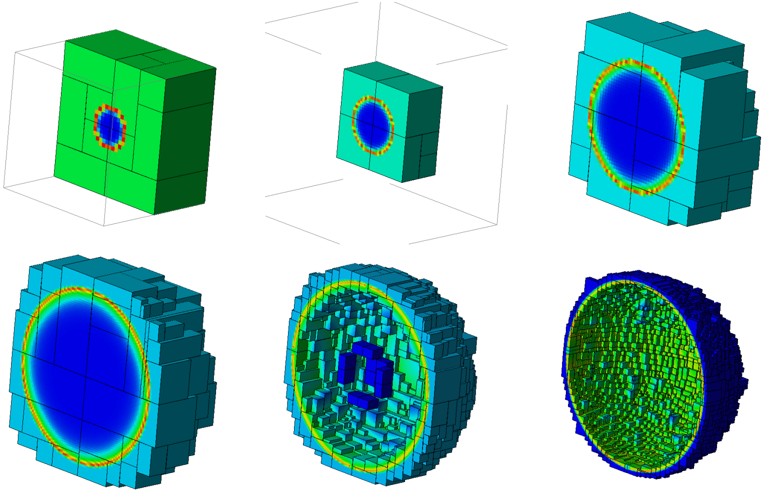


$l_{\max} = 5$ solution



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

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	$l_{max}=2$ Grids	$l_{max}=2$ Cells	$l_{max}=3$ Grids	$l_{max}=3$ Cells	$l_{max}=4$ Grids	$l_{max}=4$ Cells	$l_{max}=5$ Grids	$l_{max}=5$ 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	$l_{max}=2$ Grids	$l_{max}=2$ Cells	$l_{max}=3$ Grids	$l_{max}=3$ Cells	$l_{max}=4$ Grids	$l_{max}=4$ Cells	$l_{max}=5$ Grids	$l_{max}=5$ 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

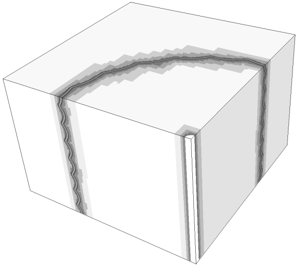
- Subgrid terms σ_{kn}, σ_n^e, σ_n^i are computed by Pullin's stretched-vortex model
- Cutoff Δ_c is set to local SAMR resolution Δx_l
- It remains to solve the Navier-Stokes equations in the hyperbolic regime
  - 3rd order WENO method (hybridized with a tuned centered difference stencil) for convection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

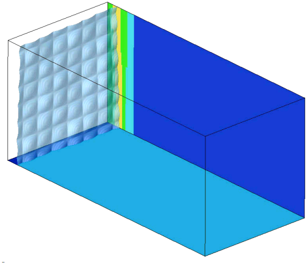
Example: Cylindrical Richtmyer-Meshkov instability

- 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)

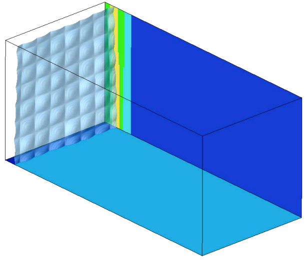


Task	2ms (%)	5ms (%)	10ms (%)
Integration	45.3	65.9	52.0
Boundary setting	44.3	28.6	41.9
Flux correction	7.2	3.4	4.1
Interpolation	0.9	0.4	0.3
Reorganization	1.6	1.2	1.2
Misc.	0.6	0.5	0.5
Max. imbalance	1.25	1.23	1.30

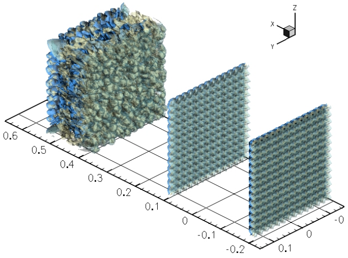
`code/amroc/doc/html/apps/weno_2applications_2euler_23d_2RM_`
`_AirSF6_2src_2Problem_8h_source.html`

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]:

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

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

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

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

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

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

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

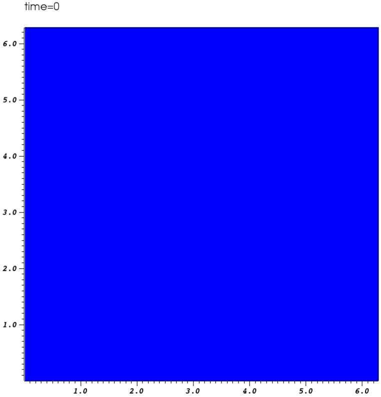
$$\frac{\partial B_z}{\partial t} + \frac{\partial}{\partial x}(u_x B_z - B_z u_x) + \frac{\partial}{\partial y}(u_y B_z - B_y u_z) = 0$$

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

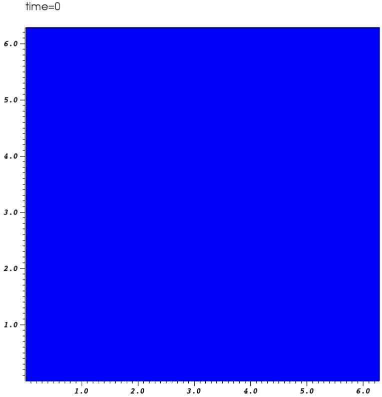
Orszag-Tang vortex

- 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



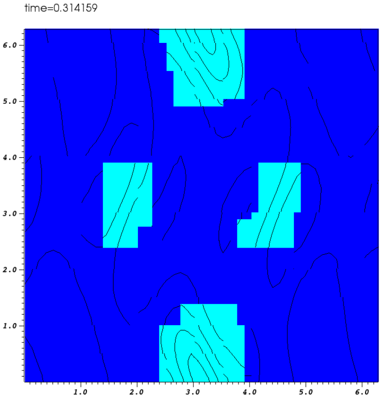
Multi-resolution criterion with hierarchical thresholding

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

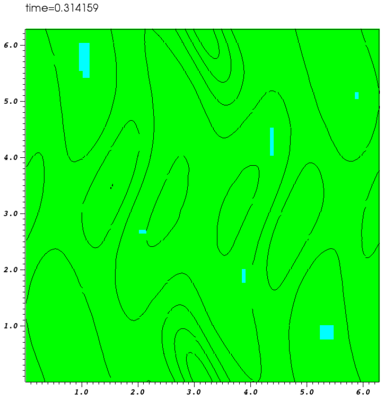
Orszag-Tang vortex

- 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

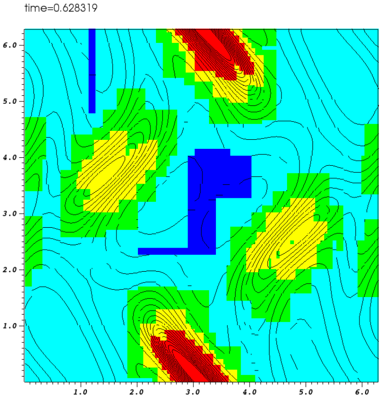
code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html

Orszag-Tang vortex

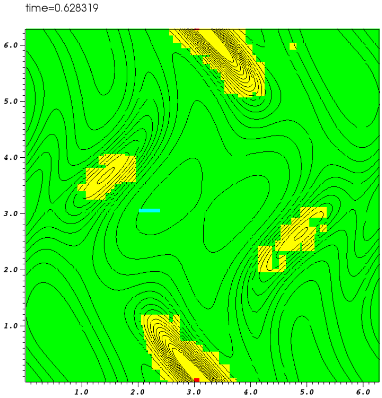
- Adaptive solution on 50 × 50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

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

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



Scaled gradient of ρ



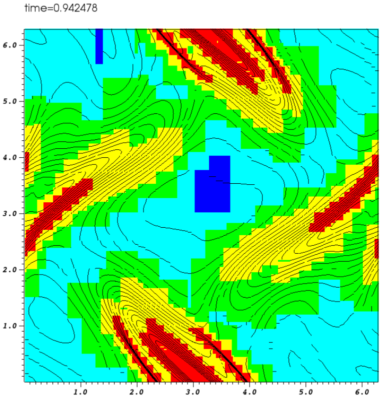
Multi-resolution criterion with hierarchical thresholding

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

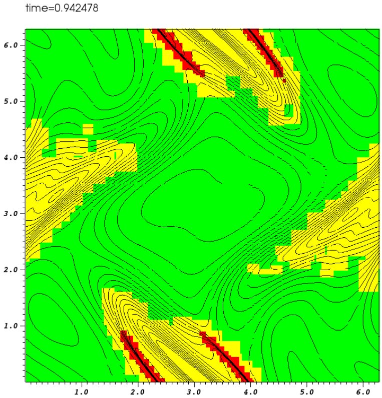
Orszag-Tang vortex

- 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



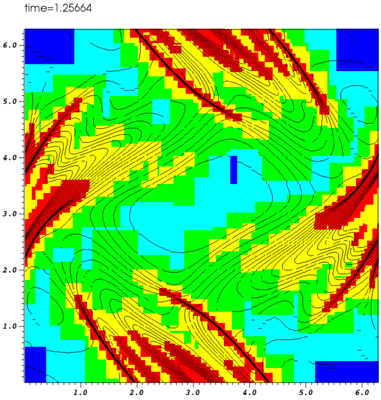
Multi-resolution criterion with hierarchical thresholding

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

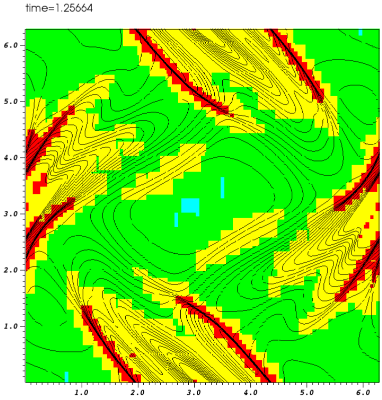
Orszag-Tang vortex

- 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

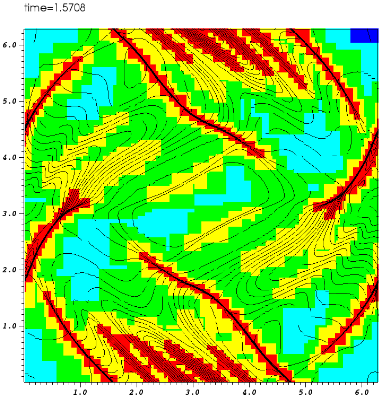
code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html

Orszag-Tang vortex

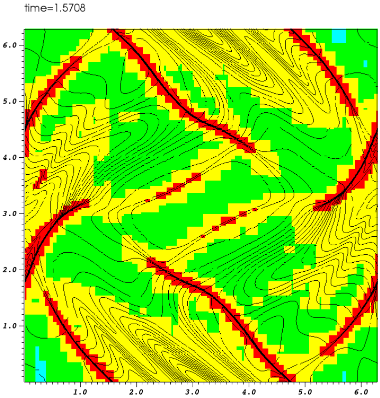
- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

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

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



Scaled gradient of ρ



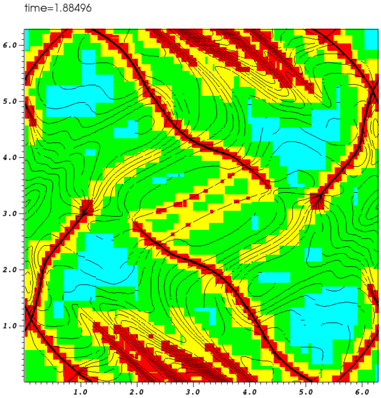
Multi-resolution criterion with hierarchical thresholding

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

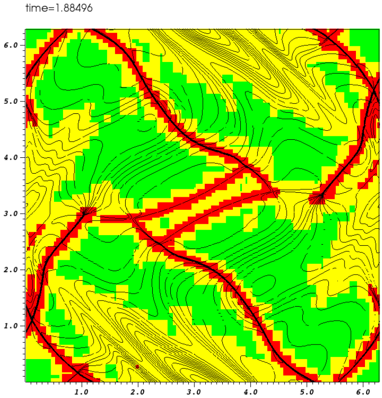
Orszag-Tang vortex

- 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



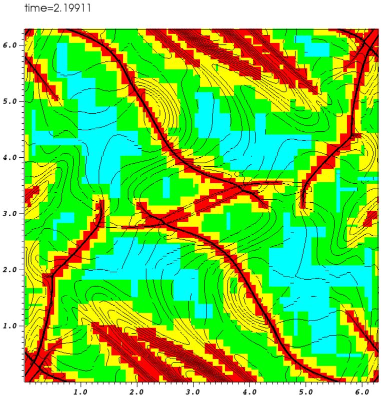
Multi-resolution criterion with hierarchical thresholding

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

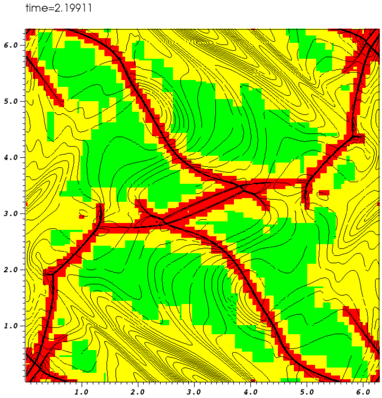
Orszag-Tang vortex

- 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



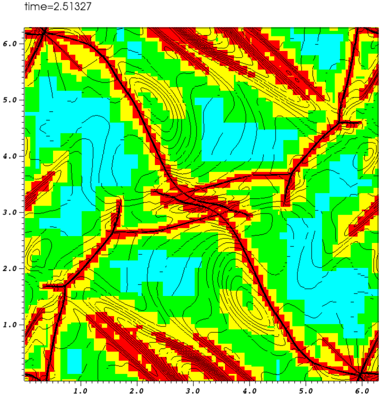
Multi-resolution criterion with hierarchical thresholding

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

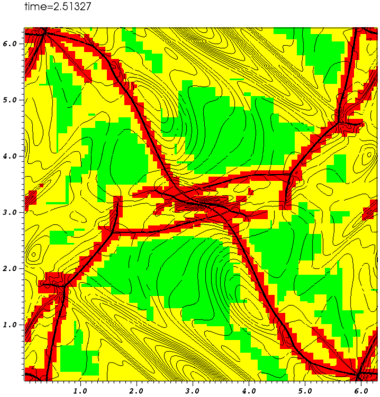
Orszag-Tang vortex

- 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



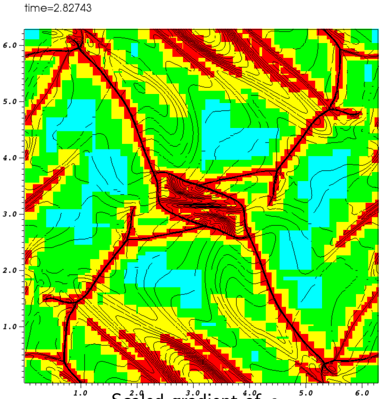
Multi-resolution criterion with hierarchical thresholding

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

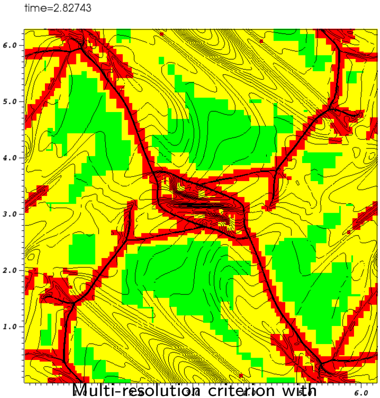
Orszag-Tang vortex

- 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

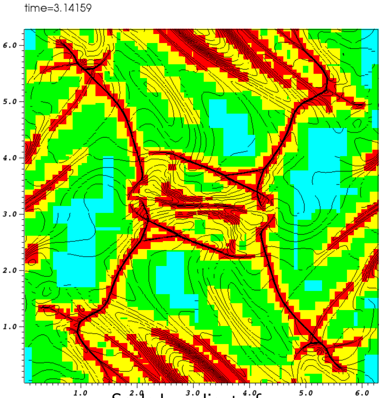
code/amroc/doc/html/apps/mhd_2applications_2eglm_22d_2OrszagTangVortex_2src_2Problem_8h_source.html

Orszag-Tang vortex

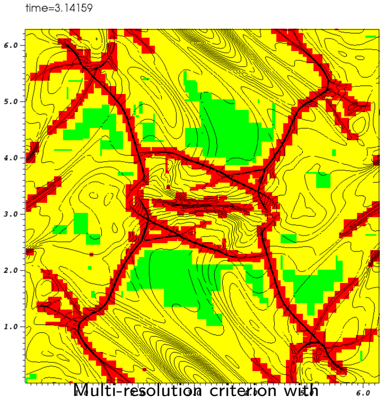
- Adaptive solution on 50×50 grid with 4 additional levels refined by $r_l = 2$
- Initial condition

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

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



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.

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

[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.

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

[Kröner, 1997] Kröner, D. (1997). *Numerical schemes for conservation laws*. John Wiley & Sons and B. G. Teubner, New York, Leipzig.

References II

[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.

[LeVeque, 1992] LeVeque, R. J. (1992). *Numerical methods for conservation laws*. Birkhäuser, Basel.

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

[Majda, 1984] Majda, A. (1984). *Compressible fluid flow and systems of conservation laws in several space variables*. Applied Mathematical Sciences Vol. 53. Springer-Verlag, New York.

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

References III

[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.

[Roe, 1981] Roe, P. L. (1981). Approximate Riemann solvers, parameter vectors and difference schemes. *J. Comput. Phys.*, 43:357–372.

[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.

[Toro, 1999] Toro, E. F. (1999). *Riemann solvers and numerical methods for fluid dynamics*. Springer-Verlag, Berlin, Heidelberg, 2nd edition.

[Toro et al., 1994] Toro, E. F., Spruce, M., and Speares, W. (1994). Restoration of the contact surface in the HLL-Riemann solver. *Shock Waves*, 4:25–34.

[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.

References IV

[Wada and Liou, 1997] Wada, Y. and Liou, M.-S. (1997). An accurate and robust flux splitting scheme for shock and contact discontinuities. *SIAM J. Sci. Comp.*, 18(3):633–657.