Block-structured Adaptive Mesh Refinement in C++

The AMROC Framework for Parallel AMR

Short course at INPE, 30th June to 1st July 2016

Supported by Fapesp grants 2015/50403–0 and 2015/25624–2

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

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

Structure of the lectures

1. Structured adaptive mesh refinement
   - Background and available SAMR software
   - The recursive SAMR algorithm
   - Overview of the AMROC software system
   - Distributed memory parallelization
2. Hyperbolic AMROC solvers
   - Shock-capturing schemes for gas dynamics
   - Higher-order discretizations
   - Magneto-hydroynamics
   - Code snippets for interfacing with AMROC
3. Discussion session
   - Demo of AMROC
   - Installation on student computers
   - Running examples, etc.

Structure of the lectures - II

4. Complex hyperbolic SAMR applications
   - Consideration of non-Cartesian geometries
   - Shock-induced combustion simulation with AMROC
   - Fluid-structure interaction with the Virtual Test Facility
   - Combressible turbulence simulation
5. Advanced topics
   - Adaptive lattice Boltzmann methods with AMROC
   - Large eddy simulation of subsonic problems
   - Using SAMR for geometric multigrid methods

Useful references I

Finite volume methods for hyperbolic problems

- LeVeque, R. J. (2002). *Finite volume methods for hyperbolic problems*. Cambridge University Press, Cambridge, New York.
- Godlewski, E. and Raviart, P.-A. (1996). *Numerical approximation of hyperbolic systems of conservation laws*. Springer Verlag, New York.
- Toro, E. F. (1999). *Riemann solvers and numerical methods for fluid dynamics*. Springer-Verlag, Berlin, Heidelberg, 2nd edition.
- Laney, C. B. (1998). *Computational gasdynamics*. Cambridge University Press, Cambridge.

Structured Adaptive Mesh Refinement

- Berger, M. and Colella, P. (1988). Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84.
- Bell, J., Berger, M., Saltzman, J., and Welcome, M. (1994). Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comp.*, 15(1):127–138.
- Berger, M. and LeVeque, R. (1998). Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Numer. Anal.*, 35(6):2298–2316.

Useful references II

- Deiterding, R. (2011). Block-structured adaptive mesh refinement - theory, implementation and application, *Series in Applied and Industrial Mathematics: Proceedings*, 34: 97–150.

Lattice-Boltzmann methods

- Succi, S. (2001). *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford Science Publications.
- Guo, Z., Shu, C. (2013). *Lattice Boltzmann Method and Its Applications in Engineering*, World Scientific.
- Hähnel, D. (2004). *Molekulare Gasdynamik*, Springer.
- Aidun, C. K., Clausen, J. A. (2010). Lattice-Boltzmann method for complex flows. *Annu. Rev. Fluid Mech.*, 42: 439–472.

Adaptive multigrid (finite difference and finite element based in textbooks)

- Hackbusch, W. (1985). *Multi-Grid Methods and Applications*. Springer Verlag, Berlin, Heidelberg.
- Briggs, W. L., Henson, V. E., and McCormick, S. F. (2001). *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, 2nd edition.
- Trottenberg, U., Oosterlee, C., and Schüller, A. (2001). *Multigrid*. Academic Press, San Antonio.

Useful references III

- Martin, D. F. (1998). *A cell-centered adaptive projection method for the incompressible Euler equations*. PhD thesis, University of California at Berkeley.

Implementation, parallelization

- Hornung, R. D., Wissink, A. M., and Kohn, S. H. (2006). Managing complex data and geometry in parallel structured AMR applications. *Engineering with Computers*, 22:181–195.
- Rendleman, C. A., Beckner, V. E., Lijewski, M., Crutchfield, W., and Bell, J. B. (2000). Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, 3:147–157.
- Deiterding, R. (2005). Construction and application of an AMR algorithm for distributed memory computers. In Plewa, T., Linde, T., and Weirs, V. G., editors, *Adaptive Mesh Refinement - Theory and Applications*, volume 41 of *Lecture Notes in Computational Science and Engineering*, pages 361–372. Springer.

Applications (from my own work only)

- Deiterding, R. and Wood, S (2013). Parallel adaptive fluid-structure interaction simulation of explosions impacting on building structures. *Computers & Fluids*, 88: 719–729.

Useful references IV

- Deiterding, R., Radovitzky, R., Mauch, S. P., Noels, L., Cummings, J. C., and Meiron, D. I. (2006). A virtual test facility for the efficient simulation of solid materials under high energy shock-wave loading. *Engineering with Computers*, 22(3-4):325–347.
- 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.
- Barton, P. T., Deiterding, R. and Meiron, D. I. and Pullin, D. I. (2013). Eulerian adaptive finite-difference method for high-velocity impact and penetration problems, *J. Comput. Phys.*, 240: 76–99.
- Perotti, L. E., Deiterding, R., Inaba, D, K., Shepherd, J. E. and Ortiz, M. (2013). Elastic response of water-filled fiber composite tubes under shock wave loading, *Int. J. Solids and Structures*, 50: 473–486.
- Deiterding, R. (2009). A parallel adaptive method for simulating shock-induced combustion with detailed chemical kinetics in complex domains. *Computers & Structures*, 87:769–783.
- Ziegler, J. L., Deiterding, R. Shepherd, J. E. and Pullin, D. I. (2011). An adaptive high-order hybrid scheme for compressive, viscous flows with detailed chemistry. *J. Comput. Phys.*, 230(20): 7598–7630.

Useful references V

- Gomes, A. K. F., Domingues, M. O., Schneider, K., Mendes, O., Deiterding, R. (2015). An adaptive multiresolution method for ideal magnetohydrodynamics using divergence cleaning with parabolic-hyperbolic correction. *Applied Numerical Mathematics* 95: 199–213.
- Deiterding, R. and Wood, S. L. (2015). A dynamically adaptive lattice Boltzmann method for predicting wake phenomena in fully coupled wind engineering problems. *IV Int. Conf. on Coupled Problems in Science and Engineering* 489–500.

Lecture 1
Structured adaptive mesh refinement

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

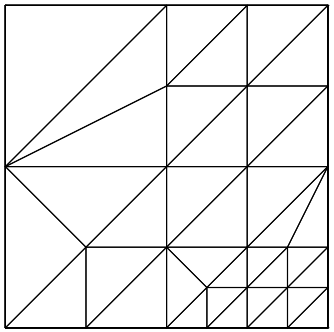
Outline

Elements of adaptive algorithms

- Base grid
- Solver
- Error indicators
- Grid manipulation
- Interpolation (restriction and prolongation)
- Load-balancing

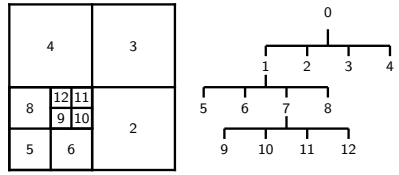
Adaptivity on unstructured meshes

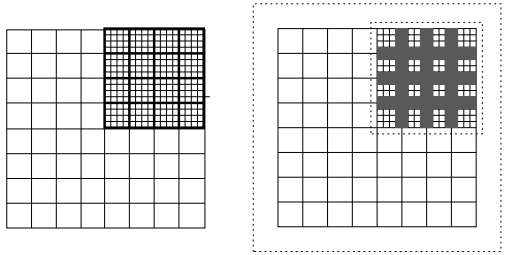
- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
- Neighborhoods have to stored
- \+ Geometric flexible
- \+ No hanging nodes
- \+ Easy to implement
- \- Higher order difficult to achieve
- \- Cell aspect ratio must be considered
- \- Fragmented data
- \- Cache-reuse / vectorizaton nearly impossible
- \- Complex load-balancing
- \- Complex synchronization



Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
- Time-step refinement
- Global index coordinate system
- Neighborhoods need not be stored
- \+ Numerical scheme only for single regular block necessary
- \+ Easy to implement
- \+ Simple load-balancing
- \+ Parent/Child relations according to tree
- +/- Cache-reuse / vectorization only in data block

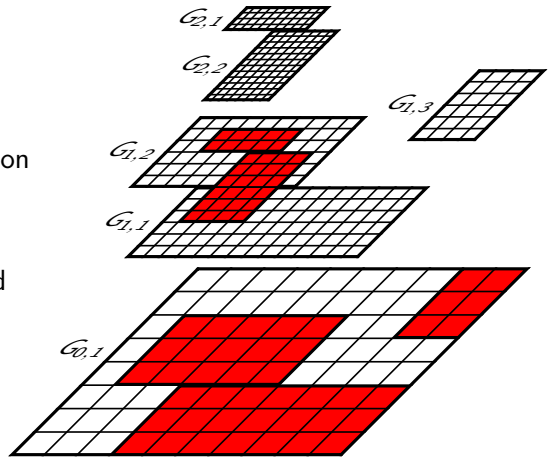




Wasted boundary space in a quad-tree

Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
- Block (aka patch) based data structures
- Global index coordinate system
- \+ Numerical scheme only for single patch necessary
- \+ Efficient cache-reuse / vectorization possible
- \+ Simple load-balancing
- \+ Minimal synchronization overhead
- \- Cells without mark are refined
- \- Hanging nodes unavoidable
- \- Cluster-algorithm necessary
- \- Difficult to implement



Simplified structured designs

Distributed memory parallelization fully supported if not otherwise states.

- PARAMESH (Parallel Adaptive Mesh Refinement)
  - Library based on uniform refinement blocks [MacNeice et al., 2000]
  - Both multigrid and explicit algorithms considered
  - http://sourceforge.net/projects/paramesh
- Flash code (AMR code for astrophysical thermonuclear flashes)
  - Built on PARAMESH
  - Solves the magneto-hydrodynamic equations with self-gravitation
  - http://www.flash.uchicago.edu/site/flashcode
- Uintah (AMR code for simulation of accidental fires and explosions)
  - Only explicit algorithms considered
  - FSI coupling Material Point Method and ICE Method (Implicit, Continuous fluid, Eulerian)
  - http://www.uintah.utah.edu
- DAGH/Grace [Parashar and Browne, 1997]
  - Just C++ data structures but no methods
  - All grids are aligned to bases mesh coarsened by factor 2
  - http://userweb.cs.utexas.edu/users/dagh

Systems that support general SAMR

- SAMRAI - Structured Adaptive Mesh Refinement Application Infrastructure
  - Very mature SAMR system [Hornung et al., 2006]
  - Explicit algorithms directly supported, implicit methods through interface to Hypre package
  - Mapped geometry and some embedded boundary support
  - https://computation-rnd.llnl.gov/SAMRAI/software.php
- BoxLib, AmrLib, MGLib, HGProj
  - Berkley-Lab-AMR collection of C++ classes by J. Bell et al., 50,000 LOC [Rendleman et al., 2000]
  - Both multigrid and explicit algorithms supported
  - https://ccse.lbl.gov/Downloads/index.html
- Chombo
  - Redesign and extension of BoxLib by P. Colella et al.
  - Both multigrid and explicit algorithms demonstrated
  - Some embedded boundary support
  - https://commons.lbl.gov/display/chombo

Further SAMR software

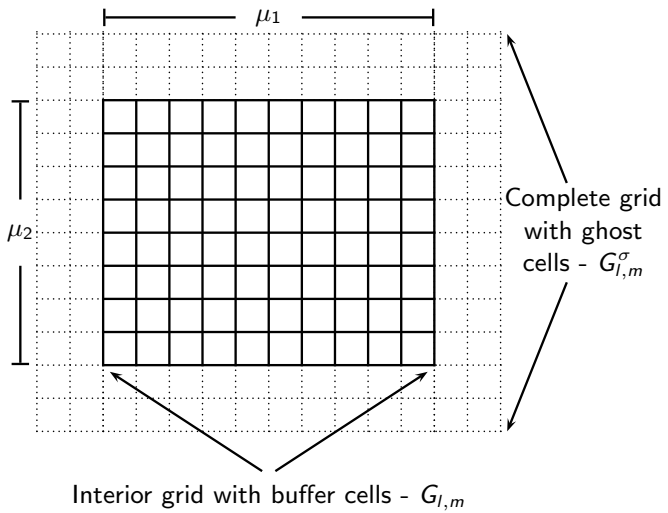
- Overture (Object-oriented tools for solving PDEs in complex geometries)
  - Overlapping meshes for complex geometries by W. Henshaw et al. [Brown et al., 1997]
  - Explicit and implicit algorithms supported
  - http://www.overtureframework.org
- AMRClaw within Clawpack [Berger and LeVeque, 1998]
  - Serial 2D Fortran 77 code for the explicit Wave Propagation method with own memory management
  - http://depts.washington.edu/clawpack
- Amrita by J. Quirk
  - Only 2D explicit finite volume methods supported
  - Embedded boundary algorithm
  - http://www.amrita-cfd.org
- Cell-based Cartesian AMR: RAGE
  - Embedded boundary method
  - Explicit and implicit algorithms
  - [Gittings et al., 2008]

Outline

The mth refinement grid on level l

Notations:

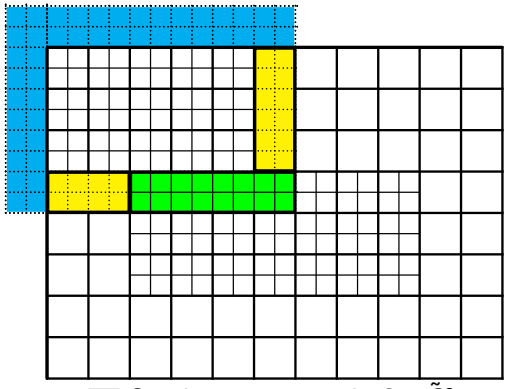
- Boundary: $\partial G_{l,m}$
- Hull: $\bar{G}_{l,m} = G_{l,m} \cup \partial G_{l,m}$
- Ghost cell region: $\tilde{G}^{\sigma}_{l,m} = G^{\sigma}_{l,m} \backslash \bar{G}_{l,m}$



Refinement data

- Resolution: $\Delta t_l := \dfrac{\Delta t_{l-1}}{r_l}$ and $\Delta x_{n,l} := \dfrac{\Delta x_{n,l-1}}{r_l}$
- Refinement factor: $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$
- Integer coordinate system for internal organization [Bell et al., 1994]:
  $\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$
- Computational Domain: $G_0 = \bigcup_{m=1}^{M_0} G_{0,m}$
- Domain of level l: $G_l := \bigcup_{m=1}^{M_l} G_{l,m}$ with $G_{l,m} \cap G_{l,n} = \emptyset$ for $m \neq n$
- Refinements are properly nested: $G_l^1 \subset G_{l-1}$
- Assume a FD scheme with stencil radius s. Necessary data:
  - Vector of state: $\mathbf{Q}^l := \bigcup_m \mathbf{Q}(G_{l,m}^s)$
  - Numerical fluxes: $\mathbf{F}^{n,l} := \bigcup_m \mathbf{F}^n(\bar{G}_{l,m})$
  - Flux corrections: $\delta\mathbf{F}^{n,l} := \bigcup_m \delta\mathbf{F}^n(\partial G_{l,m})$

Setting of ghost cells



Synchronization with G_l - $\tilde{S}_{l,m}^s = \tilde{G}_{l,m}^s \cap G_l$
Physical boundary conditions - $\tilde{P}_{l,m}^s = \tilde{G}_{l,m}^s \backslash G_0$
Interpolation from G_{l-1} - $\tilde{I}_{l,m}^s = \tilde{G}_{l,m}^s \backslash (\tilde{S}_{l,m}^s \cup \tilde{P}_{l,m}^s)$

Numerical update

Time-explicit conservative finite volume scheme

$$\mathcal{H}^{(\Delta t)}: \ \mathbf{Q}_{jk}(t+\Delta t) = \mathbf{Q}_{jk}(t) - \frac{\Delta t}{\Delta x_1}\left(\mathbf{F}^1_{j+\frac{1}{2},k} - \mathbf{F}^1_{j-\frac{1}{2},k}\right) - \frac{\Delta t}{\Delta x_2}\left(\mathbf{F}^2_{j,k+\frac{1}{2}} - \mathbf{F}^2_{j,k-\frac{1}{2}}\right)$$

`UpdateLevel(`l`)`

```
For all m = 1 To M_l Do
    Q(G^s_{l,m}, t) --H^(Δt_l)--> Q(G_{l,m}, t + Δt_l) , F^n(Ḡ_{l,m}, t)
    If level l > 0
        Add F^n(∂G_{l,m}, t) to δF^{n,l}
    If level l + 1 exists
        Init δF^{n,l+1} with F^n(Ḡ_{l,m} ∩ ∂G_{l+1}, t)
```

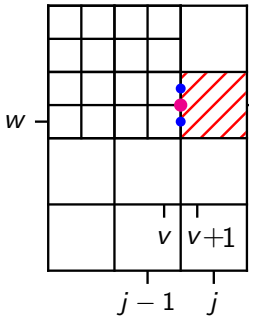

Conservative flux correction

Example: Cell j, k

$$\check{\mathbf{Q}}^l_{jk}(t+\Delta t_l) = \mathbf{Q}^l_{jk}(t) - \frac{\Delta t_l}{\Delta x_{1,l}}\left(\mathbf{F}^{1,l}_{j+\frac{1}{2},k} - \frac{1}{r_{l+1}^2}\sum_{\kappa=0}^{r_{l+1}-1}\sum_{\iota=0}^{r_{l+1}-1}\mathbf{F}^{1,l+1}_{v+\frac{1}{2},w+\iota}(t+\kappa\Delta t_{l+1})\right) - \frac{\Delta t_l}{\Delta x_{2,l}}\left(\mathbf{F}^{2,l}_{j,k+\frac{1}{2}} - \mathbf{F}^{2,l}_{j,k-\frac{1}{2}}\right)$$

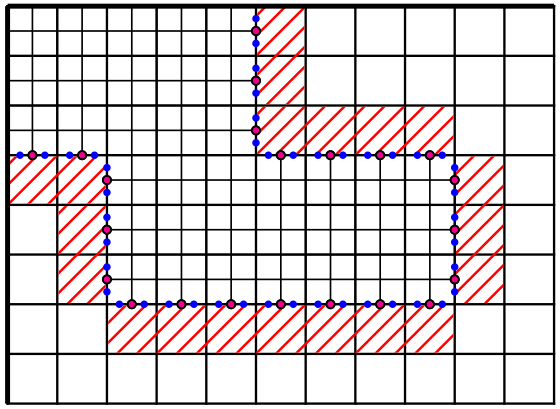
Correction pass:

1. $\delta\mathbf{F}^{1,l+1}_{j-\frac{1}{2},k} := -\mathbf{F}^{1,l}_{j-\frac{1}{2},k}$
2. $\delta\mathbf{F}^{1,l+1}_{j-\frac{1}{2},k} := \delta\mathbf{F}^{1,l+1}_{j-\frac{1}{2},k} + \dfrac{1}{r_{l+1}^2}\sum_{\iota=0}^{r_{l+1}-1}\mathbf{F}^{1,l+1}_{v+\frac{1}{2},w+\iota}(t+\kappa\Delta t_{l+1})$
3. $\check{\mathbf{Q}}^l_{jk}(t+\Delta t_l) := \mathbf{Q}^l_{jk}(t+\Delta t_l) + \dfrac{\Delta t_l}{\Delta x_{1,l}}\delta\mathbf{F}^{1,l+1}_{j-\frac{1}{2},k}$



Conservative flux correction II

- Level l cells needing correction $(G_{l+1}^{r_{l+1}} \setminus G_{l+1}) \cap G_l$
- Corrections $\delta\mathbf{F}^{n,l+1}$ stored on level $l+1$ along ∂G_{l+1} (lower-dimensional data coarsened by r_{l+1})
- Init $\delta\mathbf{F}^{n,l+1}$ with level l fluxes $\mathbf{F}^{n,l}(\bar{G}_l \cap \partial G_{l+1})$
- Add level $l+1$ fluxes $\mathbf{F}^{n,l+1}(\partial G_{l+1})$ to $\delta\mathbf{F}^{n,l}$



Cells to correct • $\mathbf{F}^{n,l}$ • $\mathbf{F}^{n,l+1}$ ∘ $\delta\mathbf{F}^{n,l+1}$

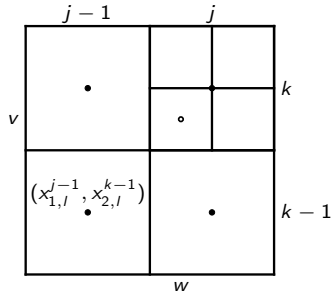
Level transfer operators

Conservative averaging (restriction):
Replace cells on level l covered by level $l+1$, i.e. $G_l \cap G_{l+1}$, by

$$\hat{\mathbf{Q}}^l_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$

Bilinear interpolation (prolongation):

$$\check{\mathbf{Q}}^{l+1}_{vw} := (1-f_1)(1-f_2)\,\mathbf{Q}^l_{j-1,k-1} + f_1(1-f_2)\,\mathbf{Q}^l_{j,k-1} + (1-f_1)f_2\,\mathbf{Q}^l_{j-1,k} + f_1 f_2\,\mathbf{Q}^l_{jk}$$



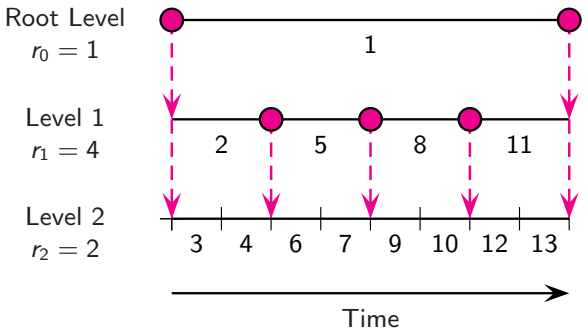
with factors $f_1 := \dfrac{x^v_{1,l+1} - x^{j-1}_{1,l}}{\Delta x_{1,l}}$, $f_2 := \dfrac{x^w_{2,l+1} - x^{k-1}_{2,l}}{\Delta x_{2,l}}$ derived from the spatial coordinates of the cell centers $(x^{j-1}_{1,l}, x^{k-1}_{2,l})$ and $(x^v_{1,l+1}, x^w_{2,l+1})$.

For boundary conditions on $\tilde{I}^s_l$: linear time interpolation

$$\tilde{\mathbf{Q}}^{l+1}(t+\kappa\Delta t_{l+1}) := \left(1 - \frac{\kappa}{r_{l+1}}\right) \check{\mathbf{Q}}^{l+1}(t) + \frac{\kappa}{r_{l+1}} \check{\mathbf{Q}}^{l+1}(t+\Delta t_l) \quad \text{for } \kappa = 0, \ldots r_{l+1}$$

Recursive integration order

- Space-time interpolation of coarse data to set $I^s_l, l > 0$
- Regridding:
  - Creation of new grids, copy existing cells on level $l > 0$
  - Spatial interpolation to initialize new cells on level $l > 0$



Regridding of finer levels.
Base level (●) stays fixed.

The basic recursive algorithm

```
AdvanceLevel(l)

  Repeat r_l times
      Set ghost cells of Q^l(t)
      If time to regrid?
          Regrid(l)
      UpdateLevel(l)
      If level l+1 exists?
          Set ghost cells of Q^l(t + Δt_l)
          AdvanceLevel(l + 1)
          Average Q^{l+1}(t + Δt_l) onto Q^l(t + Δt_l)
          Correct Q^l(t + Δt_l) with δF^{l+1}
      t := t + Δt_l

Start - Start integration on level 0

    l = 0,  r_0 = 1
    AdvanceLevel(l)
```

- Recursion
- Restriction and flux correction
- Re-organization of hierarchical data

[Berger and Colella, 1988][Berger and Oliger, 1984]

Regridding algorithm

`Regrid(`l`)` - Regrid all levels $\iota > l$

```
For ι = l_f Downto l Do
    Flag N^ι according to Q^ι(t)
    If level ι+1 exists?
        Flag N^ι below Ğ^{ι+2}
    Flag buffer zone on N^ι
    Generate Ğ^{ι+1} from N^ι
Ğ_l := G_l
For ι = l To l_f Do
    CĞ_ι := G_0 \ Ğ_ι
    Ğ_{ι+1} := Ğ_{ι+1} \ C Ğ_ι^1
Recompose(l)
```

- Refinement flags: $N^l := \bigcup_m N(\partial G_{l,m})$
- Activate flags below higher levels
- Flag buffer cells of $b > \kappa_r$ cells, κ_r steps between calls of `Regrid(`l`)`
- Special cluster algorithm
- Use complement operation to ensure proper nesting condition

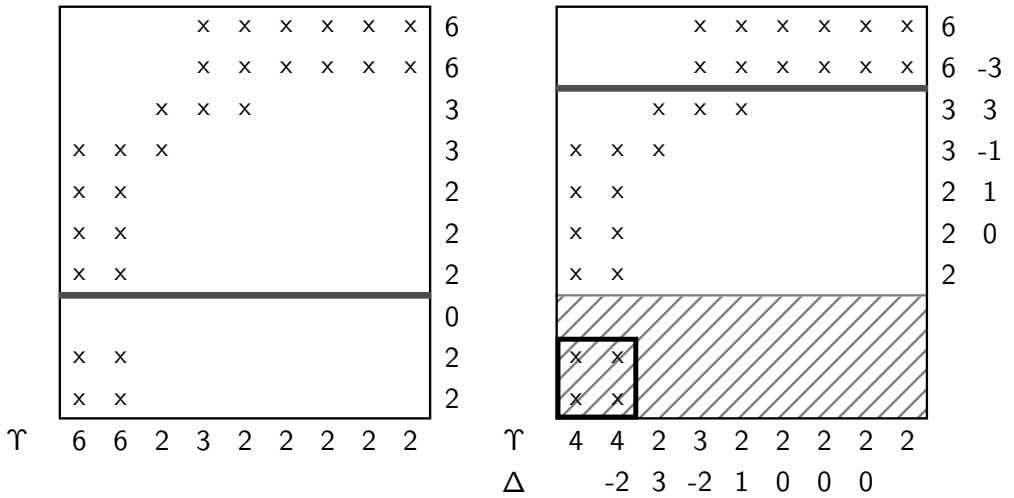
Recomposition of data

`Recompose(`l`)` - Reorganize all levels $\iota > l$

```
For ι = l+1 To l_f+1 Do
    Interpolate Q^{ι-1}(t) onto Q̆^ι(t)
    Copy Q^ι(t) onto Q̆^ι(t)
    Set ghost cells of Q̆^ι(t)
    Q^ι(t) := Q̆^ι(t),  G_ι := Ğ_ι
```

- Creates max. 1 level above l_f, but can remove multiple level if $\breve{G}_\iota$ empty (no coarsening!)
- Use spatial interpolation on entire data $\breve{\mathbf{Q}}^\iota(t)$
- Overwrite where old data exists
- Synchronization and physical boundary conditions

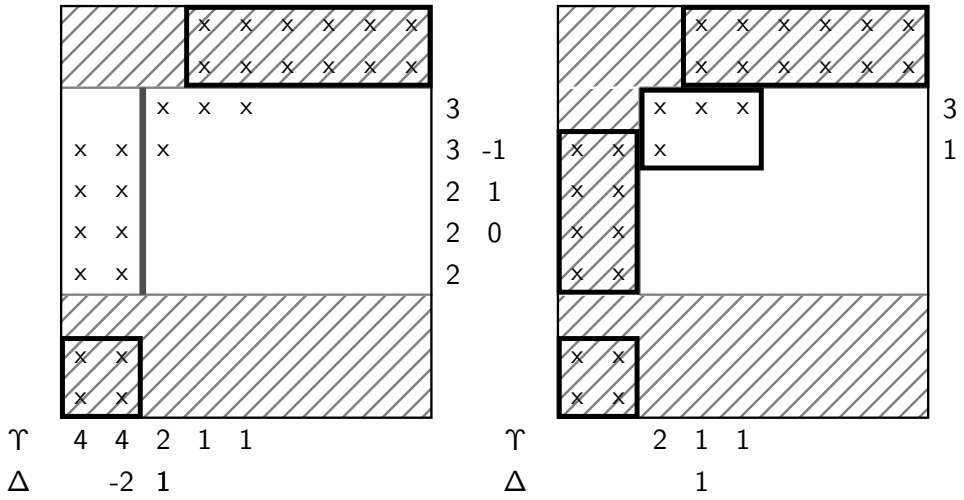
Clustering by signatures



Υ Flagged cells per row/column

Δ Second derivative of Υ, $\Delta = \Upsilon_{\nu+1} - 2\,\Upsilon_\nu + \Upsilon_{\nu-1}$

Technique from image detection: [Bell et al., 1994], see also [Berger and Rigoutsos, 1991], [Berger, 1986]



Recursive generation of $\breve{G}_{l,m}$

1. 0 in Υ
2. Largest difference in Δ
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$

Refinement criteria

Scaled gradient of scalar quantity w

$$|w(\mathbf{Q}_{j+1,k}) - w(\mathbf{Q}_{jk})| > \epsilon_w \,,\ |w(\mathbf{Q}_{j,k+1}) - w(\mathbf{Q}_{jk})| > \epsilon_w \,,\ |w(\mathbf{Q}_{j+1,k+1}) - w(\mathbf{Q}_{jk})| > \epsilon_w$$

Heuristic error estimation [Berger, 1982]:
Local truncation error of scheme of order o

$$\mathbf{q}(\mathbf{x}, t + \Delta t) - \mathcal{H}^{(\Delta t)}(\mathbf{q}(\cdot, t)) = \mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

For $\mathbf{q}$ smooth after 2 steps Δt

$$\mathbf{q}(\mathbf{x}, t + \Delta t) - \mathcal{H}_2^{(\Delta t)}(\mathbf{q}(\cdot, t - \Delta t)) = 2\,\mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

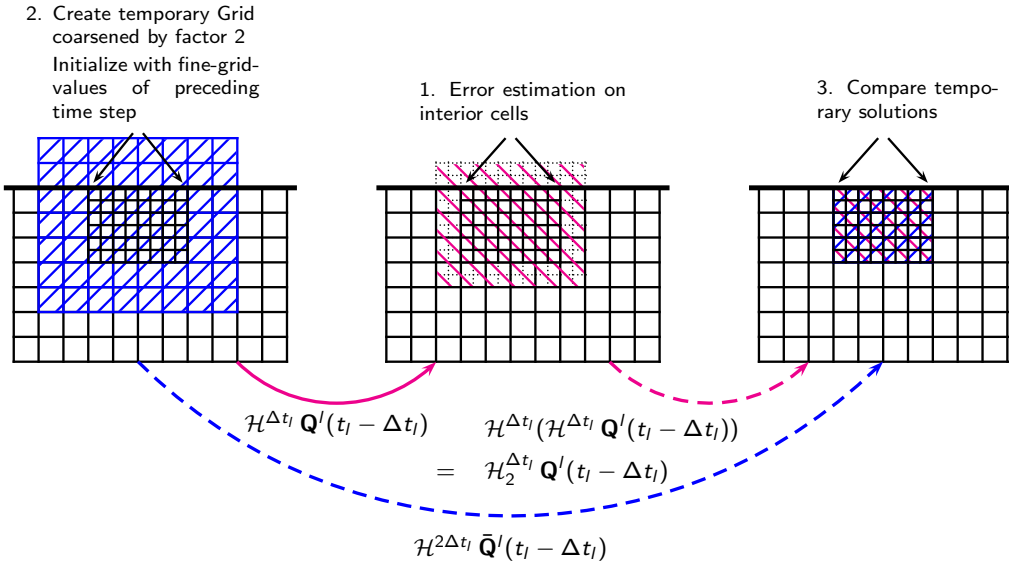
and after 1 step with $2\Delta t$

$$\mathbf{q}(\mathbf{x}, t + \Delta t) - \mathcal{H}^{(2\Delta t)}(\mathbf{q}(\cdot, t - \Delta t)) = 2^{o+1}\mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

Gives

$$\mathcal{H}_2^{(\Delta t)}(\mathbf{q}(\cdot, t - \Delta t)) - \mathcal{H}^{(2\Delta t)}(\mathbf{q}(\cdot, t - \Delta t)) = (2^{o+1} - 2)\mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

Heuristic error estimation for FV methods



Usage of heuristic error estimation

Current solution integrated tentatively 1 step with Δt_l and coarsened

$$\bar{\mathcal{Q}}(t_l + \Delta t_l) := \text{Restrict}\left(\mathcal{H}_2^{\Delta t_l}\,\mathbf{Q}^l(t_l - \Delta t_l)\right)$$

Previous solution coarsened and integrated 1 step with $2\Delta t_l$

$$\mathcal{Q}(t_l + \Delta t_l) := \mathcal{H}^{2\Delta t_l}\,\text{Restrict}\left(\mathbf{Q}^l(t_l - \Delta t_l)\right)$$

Local error estimation of scalar quantity w

$$\tau_{jk}^w := \frac{|w(\bar{\mathcal{Q}}_{jk}(t + \Delta t)) - w(\mathcal{Q}_{jk}(t + \Delta t))|}{2^{o+1} - 2}$$

In practice [Deiterding, 2003] use

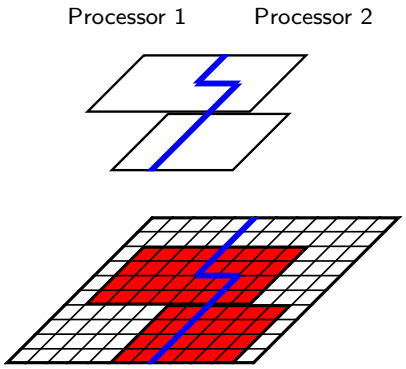
$$\frac{\tau_{jk}^w}{\max(|w(\mathcal{Q}_{jk}(t + \Delta t))|, S_w)} > \eta_w^r$$

Outline

Parallelization strategies

Decomposition of the hierarchical data

- Distribution of each grid
- Separate distribution of each level, cf. [Rendleman et al., 2000]
- Rigorous domain decomposition
  - Data of all levels resides on same node
  - Grid hierarchy defines unique "floor-plan"
  - Redistribution of data blocks during reorganization of hierarchical data
  - Synchronization when setting ghost cells



Rigorous domain decomposition formalized

Parallel machine with P identical nodes. P non-overlapping portions G_0^p, $p = 1, \ldots, P$ as

$$G_0 = \bigcup_{p=1}^{P} G_0^p \quad \text{with} \quad G_0^p \cap G_0^q = \emptyset \ \text{ for } p \neq q$$

Higher level domains G_l follow decomposition of root level

$$G_l^p := G_l \cap G_0^p$$

With $\mathcal{N}_l(\cdot)$ denoting number of cells, we estimate the workload as

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right]$$

Equal work distribution necessitates

$$\mathcal{L}^p := \frac{P \cdot \mathcal{W}(G_0^p)}{\mathcal{W}(G_0)} \approx 1 \quad \text{for all } p = 1, \ldots, P$$

[Deiterding, 2005]

Ghost cell setting

Local synchronization

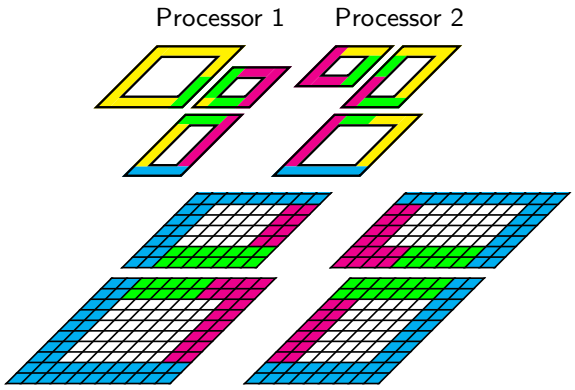
$$\tilde{S}_{l,m}^{s,p} = \tilde{G}_{l,m}^{s,p} \cap G_l^p$$

Parallel synchronization

$$\tilde{S}_{l,m}^{s,q} = \tilde{G}_{l,m}^{s,p} \cap G_l^q, q \neq p$$

Interpolation and physical boundary conditions remain strictly local

- Scheme $\mathcal{H}^{(\Delta t_l)}$ evaluated locally
- Restriction and propolongation local

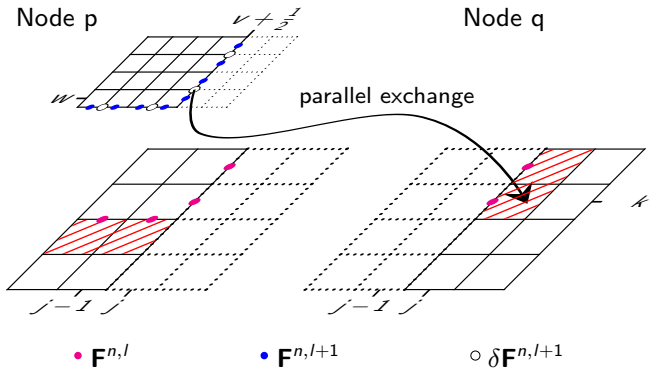


Ghost cell values:

Interpolation | Parallel synchronization
Local synchronization | Physical boundary

Parallel flux correction

1. Strictly local: Init $\delta\mathbf{F}^{n,l+1}$ with $\mathbf{F}^n(\bar{G}_{l,m} \cap \partial G_{l+1}, t)$
2. Strictly local: Add $\mathbf{F}^n(\partial G_{l,m}, t)$ to $\delta\mathbf{F}^{n,l}$
3. Parallel communication: Correct $\mathbf{Q}^l(t + \Delta t_l)$ with $\delta\mathbf{F}^{l+1}$



• $\mathbf{F}^{n,l}$ • $\mathbf{F}^{n,l+1}$ ○ $\delta\mathbf{F}^{n,l+1}$

The recursive algorithm in parallel

```
AdvanceLevel(l)

  Repeat r_l times
      Set ghost cells of Q^l(t)
      If time to regrid?
          Regrid(l)
      UpdateLevel(l)
      If level l+1 exists?
          Set ghost cells of Q^l(t + Δt_l)
          AdvanceLevel(l+1)
          Average Q^{l+1}(t + Δt_l) onto Q^l(t + Δt_l)
          Correct Q^l(t + Δt_l) with δF^{l+1}
      t := t + Δt_l

UpdateLevel(l)

  For all m = 1 To M_l Do
      Q(G^s_{l,m}, t) --H^{(Δt_l)}--> Q(G_{l,m}, t + Δt_l), F^n(Ḡ_{l,m}, t)
      If level l > 0
          Add F^n(∂G_{l,m}, t) to δF^{n,l}
      If level l+1 exists
          Init δF^{n,l+1} with F^n(Ḡ_{l,m} ∩ ∂G_{l+1}, t)
```

- Numerical update strictly local
- Inter-level transfer local
- Parallel synchronization
- Application of $\delta\mathbf{F}^{l+1}$ on ∂G_l^q

Regridding algorithm in parallel

```
Regrid(l) - Regrid all levels ι > l

  For ι = l_f Downto l Do
      Flag N^ι according to Q^ι(t)
      If level ι+1 exists?
          Flag N^ι below Ğ^{ι+2}
      Flag buffer zone on N^ι
      Generate Ğ^{ι+1} from N^ι
  Ğ_l := G_l
  For ι = l To l_f Do
      CĞ_ι := G_0 \ Ğ_ι
      Ğ_{ι+1} := Ğ_{ι+1} \ CĞ^1_ι
  Recompose(l)
```

- Need a ghost cell overlap of b cells to ensure correct setting of refinement flags in parallel
- Two options exist (we choose the latter):
  - Global clustering algorithm
  - Local clustering algorithm and concatenation of new lists $\check{G}^{\iota+1}$

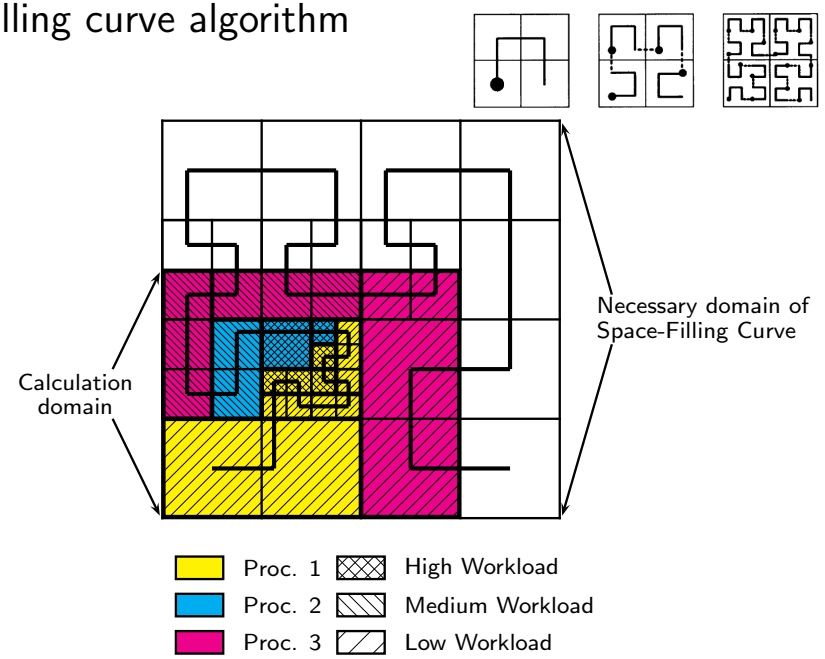
Recomposition algorithm in parallel

```
Recompose(l) - Reorganize all levels

  Generate G_0^p from {G_0, ..., G_l, Ğ_{l+1}, ..., Ğ_{l_f+1}}
  For ι = l+1 To l_f + 1 Do
      If ι > l
          Ğ_ι^p := Ğ_ι ∩ G_0^p
          Interpolate Q^{ι-1}(t) onto Q̆^ι(t)
      else
          Ğ_ι^p := G_ι ∩ G_0^p
          If ι > 0
              Copy δF^{n,ι} onto δF̆^{n,ι}
              δF^{n,ι} := δF̆^{n,ι}
      If ι ≥ l then κ_ι = 0 else κ_ι = 1
      For κ = 0 To κ_ι Do
          Copy Q^ι(t) onto Q̆^ι(t)
          Set ghost cells of Q̆^ι(t)
          Q^ι(t) := Q̆^ι(t)
      G_ι := Ğ_ι
```

- Global redistribution can also be required when regridding higher levels and $G_0, \ldots, G_l$ do not change (drawback of domain decomposition)
- When $\iota > l$ do nothing special
- For $\iota \leq l$, redistribute additionally
  - Flux corrections $\delta\mathbf{F}^{n,\iota}$
  - Already updated time level $\mathbf{Q}^\iota(t + \kappa\Delta t_\iota)$

Space-filling curve algorithm

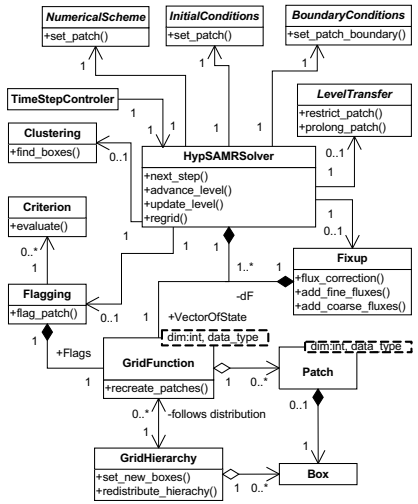


Overview

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

UML design of AMROC

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



Commonalities in software design

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

Hierarchical data structures

Directory `amroc/hds`. Key classes:

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

Hierarchical data structures - II

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

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

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

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

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

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

AMR level

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

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

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

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

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

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

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

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

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

AMR level - II

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

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

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

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

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

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

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

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

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

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

References I

[Bell et al., 1994] Bell, J., Berger, M., Saltzman, J., and Welcome, M. (1994). Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comp.*, 15(1):127–138.

[Berger, 1982] Berger, M. (1982). *Adaptive mesh refinement for hyperbolic differential equations*. PhD thesis, Stanford University. Report No. STAN-CS-82-924.

[Berger, 1986] Berger, M. (1986). Data structures for adaptive grid generation. *SIAM J. Sci. Stat. Comput.*, 7(3):904–916.

[Berger and Colella, 1988] Berger, M. and Colella, P. (1988). Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84.

[Berger and LeVeque, 1998] Berger, M. and LeVeque, R. (1998). Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Numer. Anal.*, 35(6):2298–2316.

[Berger and Oliger, 1984] Berger, M. and Oliger, J. (1984). Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512.

References II

[Berger and Rigoutsos, 1991] Berger, M. and Rigoutsos, I. (1991). An algorithm for point clustering and grid generation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1278–1286.

[Brown et al., 1997] Brown, D. L., Henshaw, W. D., and Quinlan, D. J. (1997). Overture: An object oriented framework for solving partial differential equations. In *Proc. ISCOPE 1997, appeared in Scientific Computing in Object-Oriented Parallel Environments*, number 1343 in Springer Lecture Notes in Computer Science.

[Deiterding, 2003] Deiterding, R. (2003). *Parallel adaptive simulation of multi-dimensional detonation structures*. PhD thesis, Brandenburgische Technische Universität Cottbus.

[Deiterding, 2005] Deiterding, R. (2005). Construction and application of an AMR algorithm for distributed memory computers. In Plewa, T., Linde, T., and Weirs, V. G., editors, *Adaptive Mesh Refinement - Theory and Applications*, volume 41 of *Lecture Notes in Computational Science and Engineering*, pages 361–372. Springer.

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

References III

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

[Gittings et al., 2008] Gittings, M., Weaver, R., Clover, M., Betlach, T., Byrne, N., Coker, R., Dendy, E., Hueckstaedt, R., New, K., Oakes, R., Rantal, D., and Stefan, R. (2008). The RAGE radiation-hydrodynamics code. *Comput. Sci. Disc.*, 1. doi:10.1088/1749-4699/1/1/015005.

[Hornung et al., 2006] Hornung, R. D., Wissink, A. M., and Kohn, S. H. (2006). Managing complex data and geometry in parallel structured AMR applications. *Engineering with Computers*, 22:181–195.

[MacNeice et al., 2000] MacNeice, P., Olson, K. M., Mobarry, C., deFainchtein, R., and Packer, C. (2000). PARAMESH: A parallel adaptive mesh refinement community toolkit. *Computer Physics Communications*, 126:330–354.

References IV

[Parashar and Browne, 1997] Parashar, M. and Browne, J. C. (1997). System engineering for high performance computing software: The HDDA/DAGH infrastructure for implementation of parallel structured adaptive mesh refinement. In *Structured Adaptive Mesh Refinement Grid Methods*, IMA Volumes in Mathematics and its Applications. Springer.

[Rendleman et al., 2000] Rendleman, C. A., Beckner, V. E., Lijewski, M., Crutchfield, W., and Bell, J. B. (2000). Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, 3:147–157.

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

Outline

Hyperbolic Conservation Laws

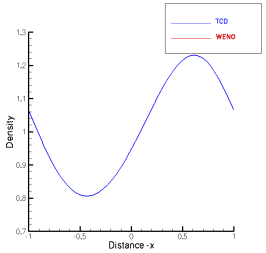
$$\frac{\partial}{\partial t}\mathbf{q}(\mathbf{x},t)+\sum_{n=1}^{d}\frac{\partial}{\partial x_n}\mathbf{f}_n(\mathbf{q}(\mathbf{x},t))=\mathbf{s}(\mathbf{q}(\mathbf{x},t)),\quad D\subset\{(\mathbf{x},t)\in\mathbb{R}^d\times\mathbb{R}_0^+\}$$

$\mathbf{q}=\mathbf{q}(\mathbf{x},t)\in S\subset\mathbb{R}^M$ - 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 $\lambda_1(\mathbf{q},\nu)\leq\ldots\leq\lambda_M(\mathbf{q},\nu)$ 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):

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

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

Examples

Euler equations

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

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

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

with polytrope gas equation of state

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

have structure

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

Examples II

Navier-Stokes equations

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

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

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

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

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

Type can be either hyperbolic or parabolic

Derivation

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

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

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)

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

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

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!

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

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

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 $\mathcal{H}^{(\cdot)}$:

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

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

[Toro, 1999]

Conservative scheme for diffusion equation

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

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

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

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

i.e.

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

from which we derive the stability condition

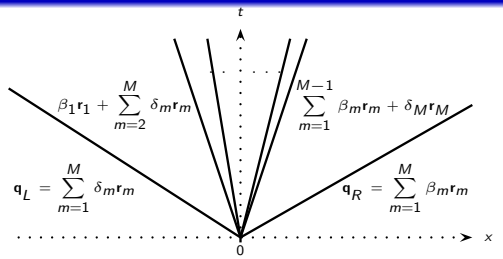
$$0 \leq c\left(\frac{\Delta t}{\Delta x_1^2} + \frac{\Delta t}{\Delta x_2^2}\right) \leq \frac{1}{2}$$

Outline

Linear upwind schemes

Consider Riemann problem

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



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

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

and $\mathbf{A}^+ := \mathbf{R}\mathbf{\Lambda}^+\mathbf{R}^{-1}\,, \quad \mathbf{A}^- := \mathbf{R}\mathbf{\Lambda}^-\mathbf{R}^{-1}$ 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 $\Delta\mathbf{Q}_{j+1/2}^n = \mathbf{Q}_{j+1}^n - \mathbf{Q}_j^n$

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

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

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

stability condition

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

[LeVeque, 1992]

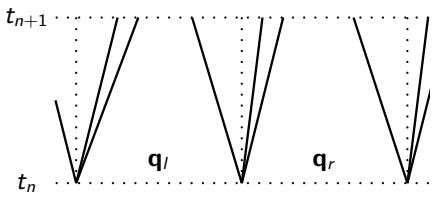
Roe's approximate Riemann solver

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

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



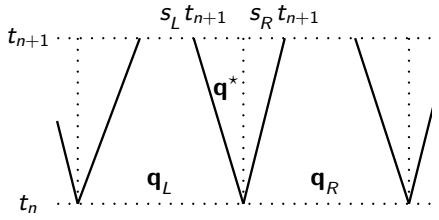
For Euler equations:

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

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



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

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

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]

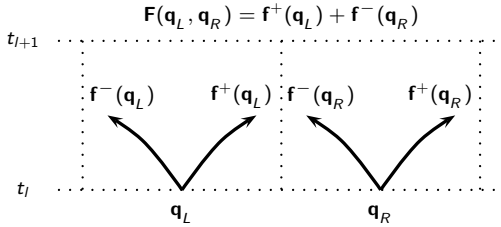
Flux vector splitting

Splitting

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

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

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



plus reproduction of regular upwinding

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

Then use

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

Steger-Warming

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

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

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

Gives

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

and the numerical flux

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

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

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

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

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

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 1D:

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

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

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

that is stable for

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

Further high-resolution methods

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

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

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

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

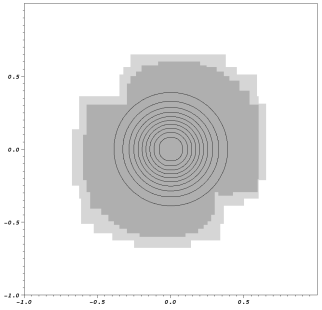
SAMR accuracy verification

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



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

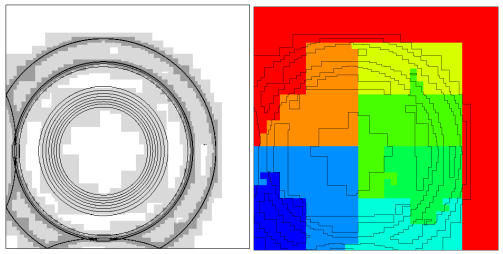
Fully two-dimensional Wave Propagation Method, Minmod limiter

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

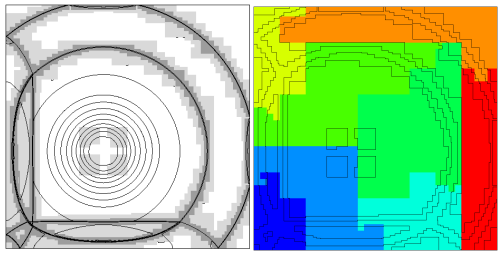
Benchmark run: blast wave in 2D

- 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



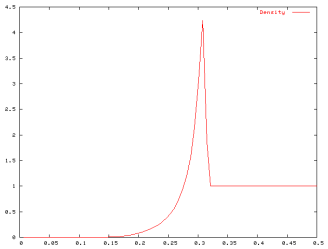
After 79 time steps

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

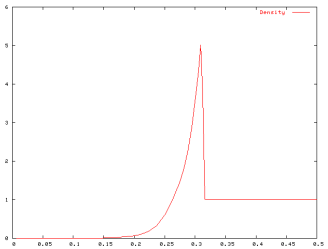
Benchmark run 2: point-explosion in 3D

- 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

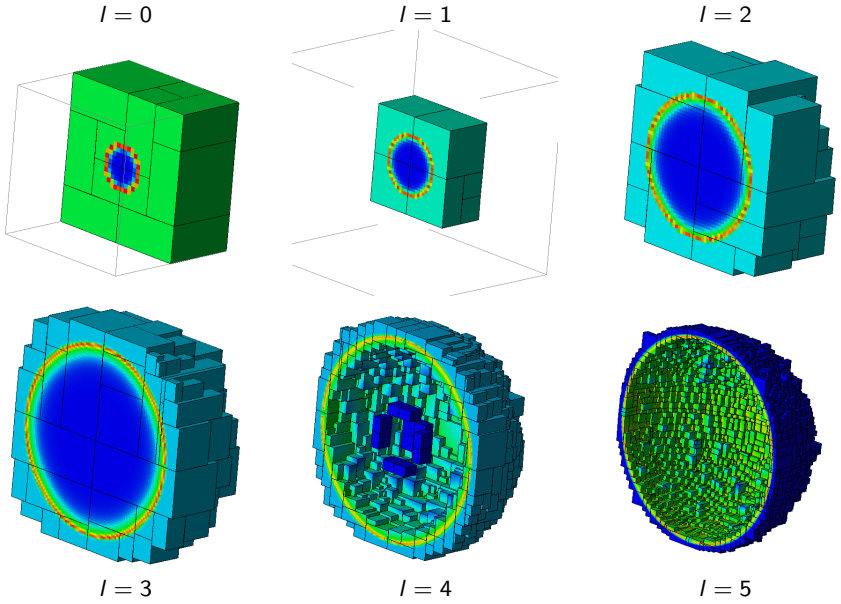
$l_{\max} = 4$ solution



$l_{\max} = 5$ solution



Benchmark run 2: visualization of refinement



Benchmark run 2: performance results

Number of grids and cells

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

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

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

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

$$\frac{\partial \bar{\rho}\bar{E}}{\partial t} + \frac{\partial}{\partial x_n}(\tilde{u}_n(\bar{\rho}\bar{E} + \bar{p}) + \tilde{q}_n - \tilde{\tau}_{nj}\tilde{u}_j + \sigma^e_n) = 0$$

$$\frac{\partial}{\partial t}(\bar{\rho}\tilde{Y}_i) + \frac{\partial}{\partial x_n}(\bar{\rho}\tilde{Y}_i\tilde{u}_n + \tilde{J}^i_n + \sigma^i_n) = 0$$

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

$$\tilde{J}^i_n = -\bar{\rho}\tilde{D}_i\frac{\partial \tilde{Y}_i}{\partial x_n}$$

Favre-filtering

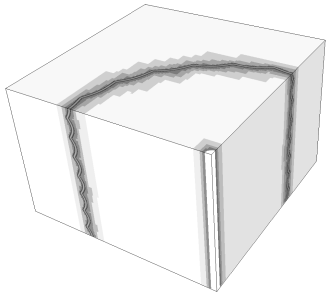
$$\tilde{\phi} = \frac{\overline{\rho\phi}}{\bar{\rho}} \quad \text{with} \quad \bar{\phi}(\mathbf{x}, t; \Delta_c) = \int_\Omega G(\mathbf{x} - \mathbf{x}'; \Delta_c)\phi(\mathbf{x}', t)d\mathbf{x}'$$

Numerical solution approach

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

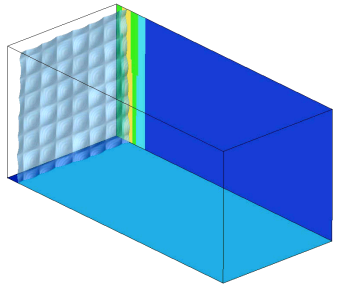
Example: Cylindrical Richtmyer-Meshkov instability

- 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



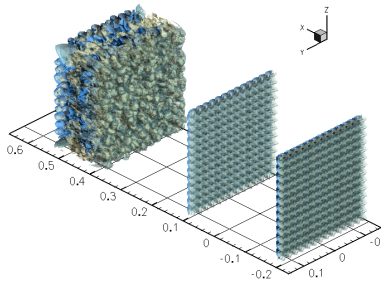
Planar Richtmyer-Meshkov instability

- 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`



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

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

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

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

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

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]:

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

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

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

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

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

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

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

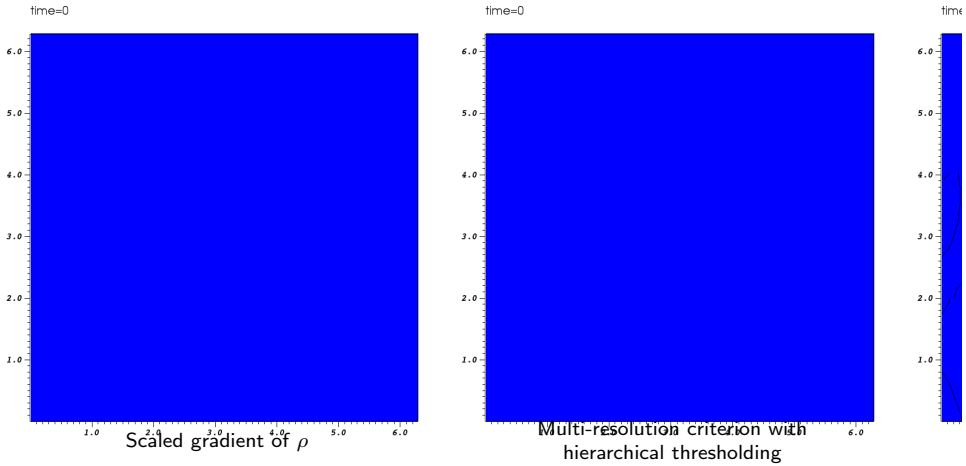
$$\frac{\partial B_z}{\partial t} + \frac{\partial}{\partial x} (u_x B_z - B_z u_x) + \frac{\partial}{\partial y} (u_y B_z - B_y u_z) = 0$$

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

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

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.

Lecture 3

Complex hyperbolic applications

Course *Block-structured Adaptive Mesh Refinement in C++*

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

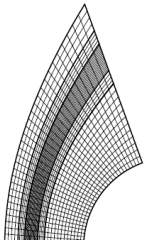
E-mail: r.deiterding@soton.ac.uk

Outline

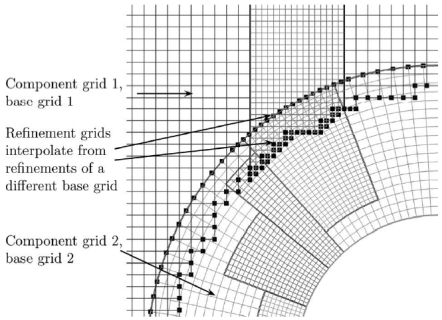
Outline

SAMR on boundary aligned meshes

Analytic or stored geometric mapping of the coordinates (graphic from [Yamaleev and Carpenter, 2002])



- Topology remains unchanged and thereby entire SAMR algorithm
- Patch solver and interpolation need to consider geometry transformation
- Handles boundary layers well



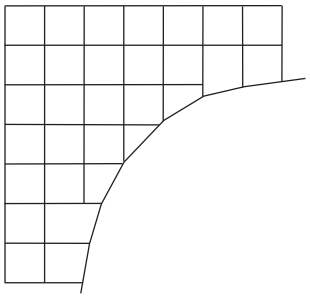
Overlapping adaptive meshes [Henshaw and Schwendeman, 2003], [Meakin, 1995]

- Idea is to use a non-Cartesian structured grids only near boundary
- Very suitable for moving objects with boundary layers
- Interpolation between meshes is usually non-conservative

Cut-cell techniques

Accurate embedded boundary method

$$V_j^{n+1}\mathbf{Q}_j^{n+1} = V_j^n\mathbf{Q}_j^n - \Delta t\left(A_{j+1/2}^{n+1/2}\,\mathbf{F}(\mathbf{Q},j) - A_{j-1/2}^{n+1/2}\,\mathbf{F}(\mathbf{Q},j-1)\right)$$



Methods that represent the boundary sharply:

- Cut-cell approach constructs appropriate finite volumes
- Conservative by construction. Correct boundary flux
- Key question: How to avoid small-cell time step restriction in explicit metyhods?
  - Cell merging: [Quirk, 1994a]
- Usually explicit geometry representation used [Aftosmis, 1997], but can also be implicit, cf. [Nourgaliev et al., 2003], [Murman et al., 2003]

Embedded boundary techniques

Volume of fluid methods that resemble a cut-cell technique on purely Cartesian mesh

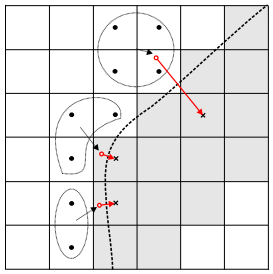
- Redistribution of boundary flux achieves conservation and bypasses time step restriction: [Pember et al., 1999], [Berger and Helzel, 2002]

Methods that diffuse the boundary in one cell (good overview in [Mittal and Iaccarino, 2005]):

- Related to the immersed boundary method by Peskin, cf. [Roma et al., 1999]
- Boundary prescription often by internal ghost cell values, cf. [Tseng and Ferziger, 2003]
- Not conservative by construction but conservative correction possible
- Usually combined with implicit geometry representation

K. J. Richards et al., On the use of the immersed boundary method for engine modeling

Level-set method for boundary embedding



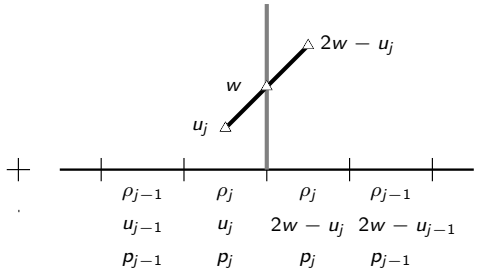
- Implicit boundary representation via distance function φ, normal $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$
- Complex boundary moving with local velocity $\mathbf{w}$, treat interface as moving rigid wall
- Construction of values in embedded boundary cells by interpolation / extrapolation

Interpolate / constant value extrapolate values at

$$\tilde{\mathbf{x}} = \mathbf{x} + 2\varphi\mathbf{n}$$

Velocity in ghost cells

$$\begin{aligned}\mathbf{u}' &= (2\mathbf{w}\cdot\mathbf{n} - \mathbf{u}\cdot\mathbf{n})\mathbf{n} + (\mathbf{u}\cdot\mathbf{t})\mathbf{t} \\ &= 2\left((\mathbf{w}-\mathbf{u})\cdot\mathbf{n}\right)\mathbf{n} + \mathbf{u}\end{aligned}$$



Closest point transform algorithm

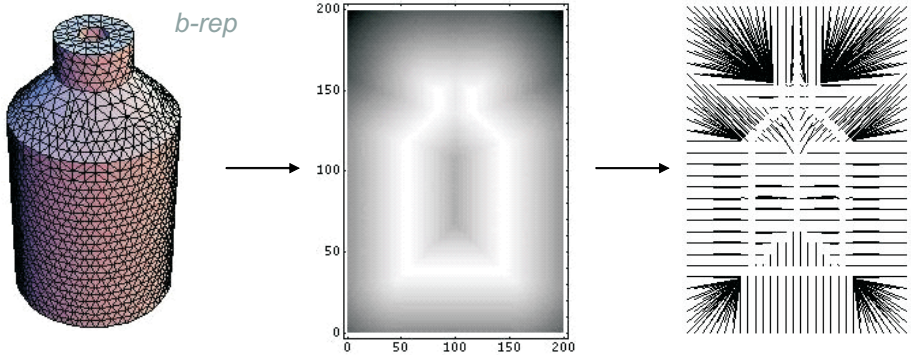
The signed distance φ to a surface $\mathcal{I}$ satisfies the eikonal equation [Sethian, 1999]

$$|\nabla\varphi| = 1 \quad \text{with} \quad \varphi\big|_{\mathcal{I}} = 0$$

Solution smooth but non-diferentiable across characteristics.

Distance computation trivial for non-overlapping elementary shapes but difficult to do efficiently for triangulated surface meshes:

- Geometric solution approach with plosest-point-transform algorithm [Mauch, 2003]



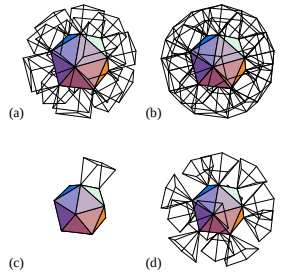
Surface mesh $\mathcal{I}$ Distance φ Normal to closest point

The characteristic / scan conversion algorithm

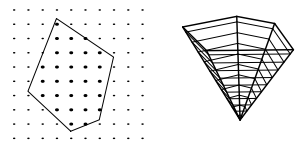
1. Build the characteristic polyhedrons for the surface mesh
2. For each face/edge/vertex
   - 2.1 Scan convert the polyhedron.
   - 2.2 Compute distance to that primitive for the scan converted points
3. Computational complexity.
   - $O(m)$ to build the b-rep and the polyhedra.
   - $O(n)$ to scan convert the polyhedra and compute the distance, etc.
4. Problem reduction by evaluation only within specified max. distance

[Mauch, 2003], see also [Deiterding et al., 2006]

Characteristic polyhedra for faces, edges, and vertices



Slicing and scan conversion of apolygon



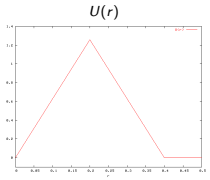
Accuracy test: stationary vortex

Construct non-trivial *radially symmetric* and *stationary* solution by balancing hydrodynamic pressure and centripetal force per volume element, i.e.

$$\frac{d}{dr}p(r) = \rho(r)\frac{U(r)^2}{r}$$

For $\rho_0 \equiv 1$ and the velocity field

$$U(r) = \alpha \cdot \begin{cases} 2r/R & \text{if } 0 < r < R/2, \\ 2(1 - r/R) & \text{if } R/2 \le r \le R, \\ 0 & \text{if } r > R, \end{cases}$$



one gets with boundary condition $p(R) = p_0 \equiv 2$ the pressure distribution

$$p(r) = p_0 + 2\rho_0\alpha^2 \cdot \begin{cases} r^2/R^2 + 1 - 2\log 2 & \text{if } 0 < r < R/2, \\ r^2/R^2 + 3 - 4r/R + 2\log(r/R) & \text{if } R/2 \le r \le R, \\ 0 & \text{if } r > R. \end{cases}$$

Entire solution for Euler equations reads

$\rho(x_1, x_2, t) = \rho_0,\ u_1(x_1, x_2, t) = -U(r)\ \sin\phi,\ u_2(x_1, x_2, t) = U(r)\ \cos\phi,\ p(x_1, x_2, t) = p(r)$

for all $t \ge 0$ with $r = \sqrt{(x_1 - x_{1,c})^2 + (x_2 - x_{2,c})^2}$ and $\phi = \arctan\dfrac{x_2 - x_{2,c}}{x_1 - x_{1,c}}$

Stationary vortex: results

Compute one full rotation, Roe solver, embedded slip wall boundary conditions
$x_{1,c} = 0.5,\ x_{2,c} = 0.5,\ R = 0.4,\ t_{end} = 1,\ \Delta h = \Delta x_1 = \Delta x_2 = 1/N,\ \alpha = R\pi$

No embedded boundary

N	Wave Propagation		Godunov Splitting	
	Error	Order	Error	Order
20	0.0111235		0.0182218	
40	0.0037996	1.55	0.0090662	1.01
80	0.0013388	1.50	0.0046392	0.97
160	0.0005005	1.42	0.0023142	1.00

Marginal shear flow along embedded boundary, $\alpha = R\pi,\ R_G = R,\ U_W = 0$

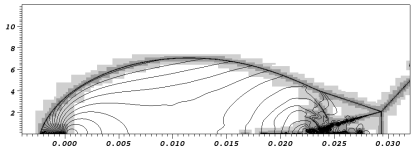
N	Wave Propagation			Godunov Splitting		
	Error	Order	Mass loss	Error	Order	Mass loss
20	0.0120056		0.0079236	0.0144203		0.0020241
40	0.0035074	1.78	0.0011898	0.0073070	0.98	0.0001300
80	0.0014193	1.31	0.0001588	0.0038401	0.93	-0.0001036
160	0.0005032	1.50	5.046e-05	0.0018988	1.02	-2.783e-06

Major shear flow along embedded boundary, $\alpha = R\pi,\ R_G = R/2,\ U_W = 0$

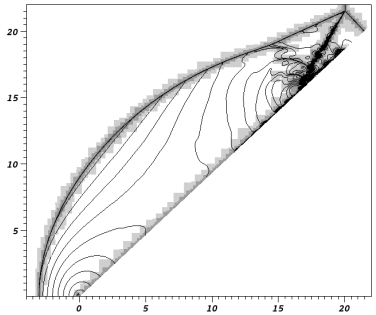
N	Wave Propagation			Godunov Splitting		
	Error	Order	Mass loss	Error	Order	Mass loss
20	0.0423925		0.0423925	0.0271446		0.0271446
40	0.0358735	0.24	0.0358735	0.0242260	0.16	0.0242260
80	0.0212340	0.76	0.0212340	0.0128638	0.91	0.0128638
160	0.0121089	0.81	0.0121089	0.0070906	0.86	0.0070906

Verification: shock reflection

- Reflection of a Mach 2.38 shock in nitrogen at 43^o wedge
- 2nd order MUSCL scheme with Roe solver, 2nd order multidimensional wave propagation method

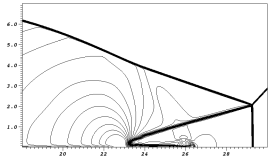


Cartesian base grid 360 × 160 cells on domain of 36 mm × 16 mm with up to 3 refinement levels with $r_l = 2, 4, 4$ and $\Delta x_{1,2} = 3.125\mu m$, 38 h CPU

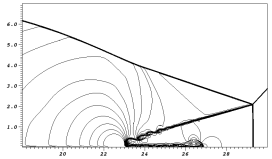


GFM base grid 390 × 330 cells on domain of 26 mm × 22 mm with up to 3 refinement levels with $r_l = 2, 4, 4$ and $\Delta x_{e,1,2} = 2.849\mu m$, 200 h CPU

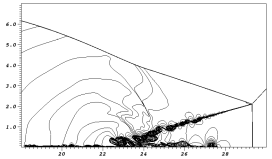
Shock reflection: SAMR solution for Euler equations



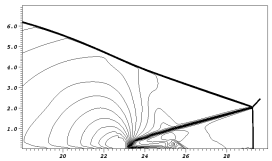
$\Delta x = 25$ mm



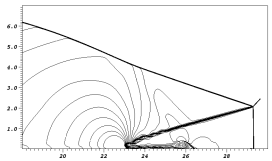
$\Delta x = 12.5$ mm



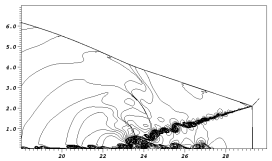
$\Delta x = 3.125$ mm



$\Delta x_e = 22.8$ mm

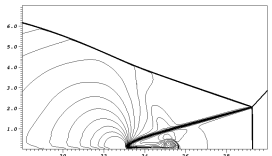


$\Delta x_e = 11.4$ mm

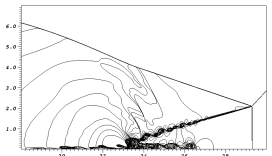


$\Delta x_e = 2.849$ mm

2nd order MUSCL scheme with Van Leer FVS, dimensional splitting



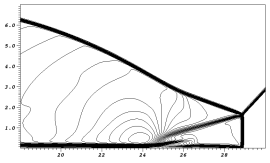
$\Delta x = 12.5$ mm



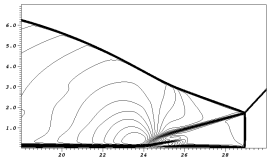
$\Delta x = 3.125$ mm

Shock reflection: solution for Navier-Stokes equations

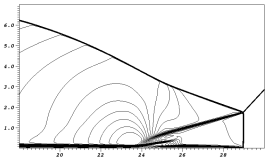
- No-slip boundary conditions enforced
- Conservative 2nd order centered differences to approximate stress tensor and heat flow



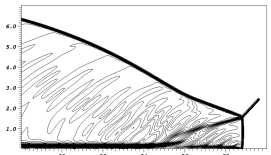
$\Delta x = 50\,\mathrm{mm}$



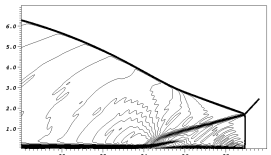
$\Delta x = 25\,\mathrm{mm}$



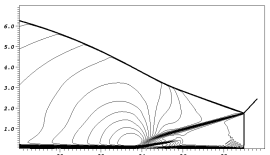
$\Delta x = 12.5\,\mathrm{mm}$, SAMR



$\Delta x_e = 45.6\,\mathrm{mm}$

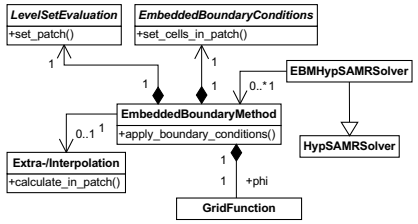


$\Delta x_e = 22.8\,\mathrm{mm}$



$\Delta x_e = 11.4\,\mathrm{mm}$, SAMR

Embedded boundary method



- Multiple independent EmbeddedBoundaryMethod objects possible
- Specialization of GFM boundary conditions
- The generic embedded boundary method is implemented in **GhostFluidMethod<VectorType, dim >** and has a **GFMLevelSet<DataType, dim >** and **GFMBoundary<VectorType, dim >** object.

  `code/amroc/doc/html/amr/classGhostFluidMethod.html` `code/amroc/doc/html/amr/classGFMLevelSet.html`

  `code/amroc/doc/html/amr/classGFMBoundary_3_01VectorType_00_012_01_4.html`
- Multiple **GhostFluidMethod<VectorType, dim >** can be registered with **AMRGFMSolver<VectorType, FixupType, FlagType, dim >** and are called as part of the boundary condition setting routine.

  `code/amroc/doc/html/amr/classAMRGFMSolver.html`
- Interface classes to **GFMLevelSet<DataType, dim >** and **GFMBoundary<VectorType, dim >** are available amroc/amr/F77Interfaces and in amroc/amr/Interfaces

  `code/amroc/doc/html/amr/classF77GFMBoundary.html` `code/amroc/doc/html/amr/classSchemeGFMBoundary.html` make the approach available to all current solvers
- For instance `code/amroc/doc/html/clp/ClpStdGFMProblem_8h.html` or `code/amroc/doc/html/weno/WENOStdGFMProblem_8h.html` in Problem.h uses AMRGFMSolver<>

Outline

Governing equations for premixed combustion

Euler equations with reaction terms

$$\frac{\partial \rho_i}{\partial t} + \frac{\partial}{\partial x_n}(\rho_i u_n) = \dot{\omega}_i\,, \quad i = 1, \ldots, K$$

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

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

Ideal gas law and Dalton's law for gas-mixtures

$$p(\rho_1, \ldots, \rho_K, T) = \sum_{i=1}^{K} p_i = \sum_{i=1}^{K} \rho_i \frac{\mathcal{R}}{W_i} T = \rho \frac{\mathcal{R}}{W} T \quad \text{with} \quad \sum_{i=1}^{K} \rho_i = \rho\,, Y_i = \frac{\rho_i}{\rho}$$

Caloric equation

$$h(Y_1, \ldots, Y_K, T) = \sum_{i=1}^{K} Y_i h_i(T) \quad \text{with} \quad h_i(T) = h_i^0 + \int_0^T c_{pi}(s) ds$$

Computation of $T = T(\rho_1, \ldots, \rho_K, e)$ from implicit equation

$$\sum_{i=1}^{K} \rho_i\, h_i(T) - \mathcal{R} T \sum_{i=1}^{K} \frac{\rho_i}{W_i} - \rho e = 0$$

for *thermally perfect* gases with $\gamma_i(T) = c_{pi}(T)/c_{vi}(T)$

Chemistry

Arrhenius-Kinetics:

$$\dot{\omega}_i = \sum_{j=1}^{M} (\nu^r_{ji} - \nu^f_{ji}) \left[k^f_j \prod_{n=1}^{K} \left(\frac{\rho_n}{W_n} \right)^{\nu^f_{jn}} - k^r_j \prod_{n=1}^{K} \left(\frac{\rho_n}{W_n} \right)^{\nu^r_{jn}} \right] \quad i = 1, \ldots, K$$

- Parsing of mechanisms with Chemkin-II
- Evalutation of $\dot{\omega}_i$ with automatically generated optimized Fortran-77 functions in the line of Chemkin-II

Integration of reaction rates: ODE integration in $\mathcal{S}^{(\cdot)}$ for Euler equations with chemical reaction

- Standard implicit or semi-implicit ODE-solver subcycles within each cell
- ρ, e, u_k remain unchanged!

$$\partial_t \rho_i = W_i \dot{\omega}_i(\rho_1, \ldots, \rho_K, T) \qquad i = 1, \ldots, K$$

Use Newton or bisection method to compute T iteratively.

Non-equilibrium mechanism for hydrogen-oxgen combustion

				A [cm, mol, s]	β	E_{act} [cal mol^{-1}]
1.	$H + O_2$	$\longrightarrow$	$O + OH$	1.86×10^{14}	0.00	16790.
2.	$O + OH$	$\longrightarrow$	$H + O_2$	1.48×10^{13}	0.00	680.
3.	$H_2 + O$	$\longrightarrow$	$H + OH$	1.82×10^{10}	1.00	8900.
4.	$H + OH$	$\longrightarrow$	$H_2 + O$	8.32×10^{09}	1.00	6950.
5.	$H_2O + O$	$\longrightarrow$	$OH + OH$	3.39×10^{13}	0.00	18350.
6.	$OH + OH$	$\longrightarrow$	$H_2O + O$	3.16×10^{12}	0.00	1100.
7.	$H_2O + H$	$\longrightarrow$	$H_2 + OH$	9.55×10^{13}	0.00	20300.
8.	$H_2 + OH$	$\longrightarrow$	$H_2O + H$	2.19×10^{13}	0.00	5150.
9.	$H_2O_2 + OH$	$\longrightarrow$	$H_2O + HO_2$	1.00×10^{13}	0.00	1800.
10.	$H_2O + HO_2$	$\longrightarrow$	$H_2O_2 + OH$	2.82×10^{13}	0.00	32790.
...	...		...	...	...	...
30.	$OH + M$	$\longrightarrow$	$O + H + M$	7.94×10^{19}	-1.00	103720.
31.	$O_2 + M$	$\longrightarrow$	$O + O + M$	5.13×10^{15}	0.00	115000.
32.	$O + O + M$	$\longrightarrow$	$O_2 + M$	4.68×10^{15}	-0.28	0.
33.	$H_2 + M$	$\longrightarrow$	$H + H + M$	2.19×10^{14}	0.00	96000.
34.	$H + H + M$	$\longrightarrow$	$H_2 + M$	3.02×10^{15}	0.00	0.

Third body efficiencies: $f(O_2) = 0.40$, $f(H_2O) = 6.50$

C. K. Westbrook. Chemical kinetics of hydrocarbon oxidation in gaseous detonations. *J. Combustion and Flame*, 46:191–210, 1982.

Riemann solver for combustion

(S1) Calculate standard Roe-averages $\hat{\rho}$, $\hat{u}_n$, $\hat{H}$, $\hat{Y}_i$, $\hat{T}$.

(S2) Compute $\hat{\gamma} := \hat{c}_p / \hat{c}_v$ with $\hat{c}_{\{p/v\}i} = \dfrac{1}{T_R - T_L} \displaystyle\int_{T_L}^{T_R} c_{\{p,v\}i}(\tau)\, d\tau$.

(S3) Calculate $\hat{\phi}_i := (\hat{\gamma} - 1) \left(\frac{\hat{\mathbf{u}}^2}{2} - \hat{h}_i \right) + \hat{\gamma} R_i \hat{T}$ with standard Roe-averages $\hat{e}_i$ or $\hat{h}_i$.

(S4) Calculate $\hat{c} := \left(\sum_{i=1}^{K} \hat{Y}_i \hat{\phi}_i - (\hat{\gamma} - 1)\hat{\mathbf{u}}^2 + (\hat{\gamma} - 1)\hat{H} \right)^{1/2}$.

(S5) Use $\Delta \mathbf{q} = \mathbf{q}_R - \mathbf{q}_L$ and Δp to compute the wave strengths a_m.

(S6) Calculate $\mathcal{W}_1 = a_1 \hat{\mathbf{r}}_1$, $\mathcal{W}_2 = \displaystyle\sum_{\iota=2}^{K+d} a_\iota \hat{\mathbf{r}}_\iota$, $\mathcal{W}_3 = a_{K+d+1} \hat{\mathbf{r}}_{K+d+1}$.

(S7) Evaluate $s_1 = \hat{u}_1 - \hat{c}$, $s_2 = \hat{u}_1$, $s_3 = \hat{u}_1 + \hat{c}$.

(S8) Evaluate $\rho^\star_{L/R}$, $u^\star_{1,L/R}$, $e^\star_{L/R}$, $c^\star_{1,L/R}$ from $\mathbf{q}^\star_L = \mathbf{q}_L + \mathcal{W}_1$ and $\mathbf{q}^\star_R = \mathbf{q}_R - \mathcal{W}_3$.

(S9) If $\rho^\star_{L/R} \le 0$ or $e^\star_{L/R} \le 0$ use $\mathbf{F}_{HLL}(\mathbf{q}_L, \mathbf{q}_R)$ and go to (S12).

(S10) Entropy correction: Evaluate $|\tilde{s}_\iota|$.
$\mathbf{F}_{Roe}(\mathbf{q}_L, \mathbf{q}_R) = \frac{1}{2} \left(\mathbf{f}(\mathbf{q}_L) + \mathbf{f}(\mathbf{q}_R) - \sum_{\iota=1}^{3} |\tilde{s}_\iota| \mathcal{W}_\iota \right)$

(S11) Positivity correction: Replace $\mathbf{F}_i$ by
$\mathbf{F}^\star_i = \mathbf{F}_\rho \cdot \begin{cases} Y^l_i, & \mathbf{F}_\rho \ge 0, \\ Y^r_i, & \mathbf{F}_\rho < 0. \end{cases}$

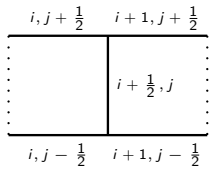
(S12) Evaluate maximal signal speed by $S = \max(|s_1|, |s_3|)$.

Riemann solver for combustion: carbuncle fix

Entropy corrections [Harten, 1983] [Harten and Hyman, 1983]

1. $|\tilde{s}_\iota| = \begin{cases} |s_\iota| & \text{if} |s_\iota| \ge 2\eta \\ \frac{|s^2_\iota|}{4\eta} + \eta & \text{otherwise} \end{cases}$
   $\eta = \frac{1}{2} \max_\iota \left\{ |s_\iota(\mathbf{q}_R) - s_\iota(\mathbf{q}_L)| \right\}$
2. Replace $|s_\iota|$ by $|\tilde{s}_\iota|$ only if $s_\iota(\mathbf{q}_L) < 0 < s_\iota(\mathbf{q}_R)$

2D modification of entropy correction [Sanders et al., 1998]:

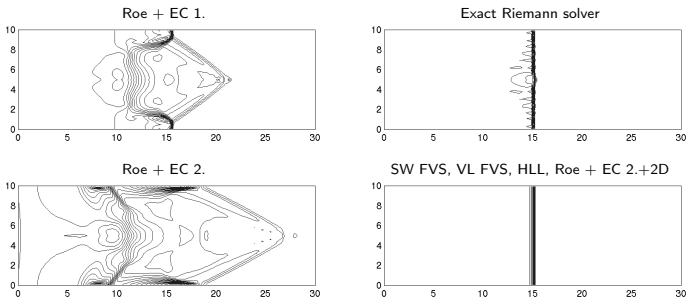


$$\tilde{\eta}_{i+1/2,j} = \max \left\{ \eta_{i+1/2,j}, \eta_{i,j-1/2}, \eta_{i,j+1/2}, \eta_{i+1,j-1/2}, \eta_{i+1,j+1/2} \right\}$$

Carbuncle phenomenon

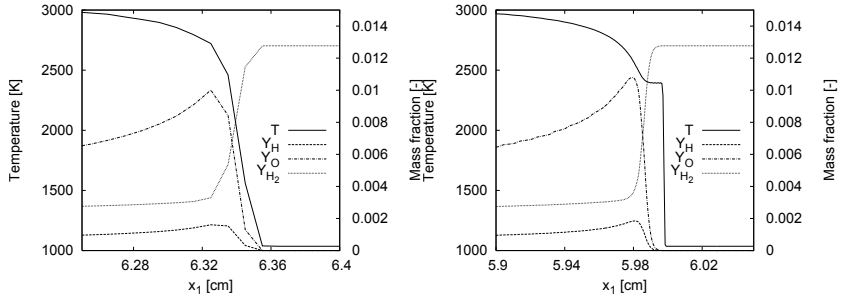
- [Quirk, 1994b]
- Test from [Deiterding, 2003]

`code/amroc/doc/html/apps/clawpack_2applications_2euler_2d_2Carbuncle_2src_2Problem_8h_source.html`



Detonations - motivation for SAMR

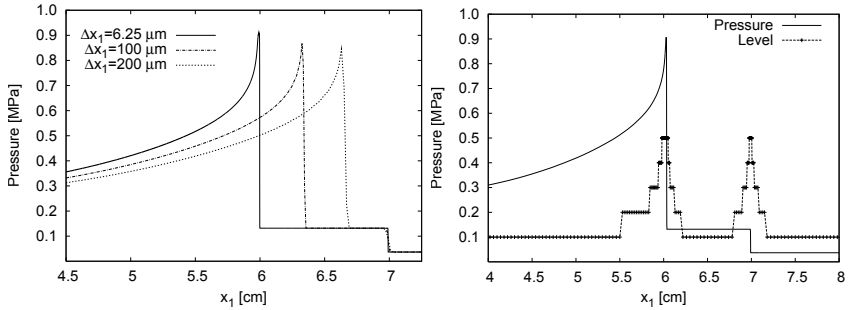
- Extremly high spatial resolution in reaction zone necessary.
- Minimal spatial resolution: $7 - 8\,\mathrm{Pts}/l_{ig} \longrightarrow \Delta x_1 \approx 0.2 - 0.175\,\mathrm{mm}$
- Uniform grids for typical geometries: $> 10^7\,\mathrm{Pts}$ in 2D, $> 10^9\,\mathrm{Pts}$ in 3D $\longrightarrow$ Self-adaptive finite volume method (AMR)



Approximation of $H_2 : O_2$ detonation at $\sim 1.5\,\mathrm{Pts}/l_{ig}$ (left) and $\sim 24\,\mathrm{Pts}/l_{ig}$ (right)

Detonation ignition in a shock tube

- Shock-induced detonation ignition of $H_2 : O_2 : Ar$ mixture at molar ratios 2:1:7 in closed 1d shock tube
- Insufficient resolution leads to inaccurate results
- Reflected shock is captured correctly by FV scheme, detonation is resolution dependent
- Fine mesh necessary in the induction zone at the head of the detonation



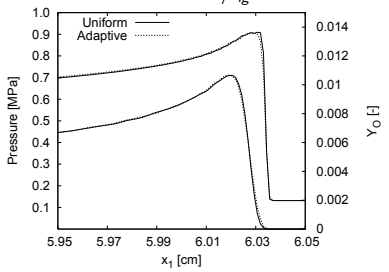
Left: Comparison of pressure distribution $t = 170\,\mu\mathrm{s}$ after shock reflection.Right: Domains of refinement levels

Detonation ignition in 1d - adaptive vs. uniform

Uniformly refined vs. dynamic adaptive simulations (Intel Xeon 3.4 GHz CPU)

	Uniform			Adaptive			
$\Delta x_1[\mu\mathrm{m}]$	Cells	$t_m[\mu\mathrm{s}]$	Time [s]	l_{max}	r_l	$t_m[\mu\mathrm{s}]$	Time [s]
400	300	166.1	31				
200	600	172.6	90	2	2	172.6	99
100	1200	175.5	277	3	2,2	175.8	167
50	2400	176.9	858	4	2,2,2	177.3	287
25	4800	177.8	2713	4	2,2,4	177.9	393
12.5	9600	178.3	9472	5	2,2,2,4	178.3	696
6.25	19200	178.6	35712	5	2,2,4,4	178.6	1370

$\sim 12\,\mathrm{Pts}/l_{ig}$



Refinement criteria:

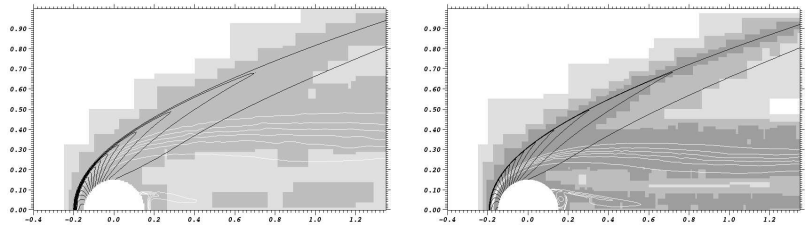
Y_i	$S_{Y_i} \cdot 10^{-4}$	$\eta^r_{Y_i} \cdot 10^{-3}$
O_2	10.0	2.0
H_2O	7.8	8.0
H	0.16	5.0
O	1.0	5.0
OH	1.8	5.0
H_2	1.3	2.0

$\epsilon_\rho = 0.07\,\mathrm{kg\,m^{-3}}$, $\epsilon_p = 50\,\mathrm{kPa}$

`code/amroc/doc/html/apps/clawpack_2applications_2euler__chem_21d_2ReflecDet_2src_2Problem_8h_source.html`

Shock-induced combustion around a sphere

- Spherical projectile of radius 1.5 mm travels with constant velocity $v_l = 2170.6\,\mathrm{m/s}$ through $H_2 : O_2 : Ar$ mixture (molar ratios 2:1:7) at 6.67 kPa and $T = 298\,\mathrm{K}$
- Cylindrical symmetric simulation on AMR base mesh of 70×40 cells
- Comparison of 3-level computation with refinement factors 2,2 ($\sim 5\,\mathrm{Pts}/l_{ig}$) and a 4-level computation with refinement factors 2,2,4 ($\sim 19\,\mathrm{Pts}/l_{ig}$) at $t = 350\,\mu\mathrm{s}$
- Higher resolved computation captures combustion zone visibly better and at slightly different position (see below)

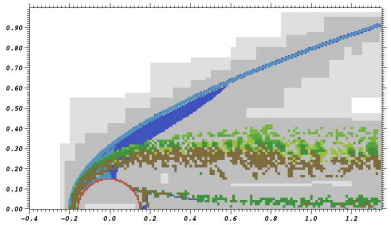


Iso-contours of p (black) and Y_{H_2} (white) on refinement domains for 3-level (left) and 4-level computation (right)

`code/amroc/doc/html/apps/clawpack_2applications_2euler__chem_22d_2Sphere_2src_2Problem_8h_source.html`

Combustion around a sphere - adaptation

Refinement indicators on $l = 2$ at $t = 350\,\mu$s. Blue: ϵ_ρ, light blue: ϵ_p, green shades: $\eta^r_{Y_i}$, red: embedded boundary

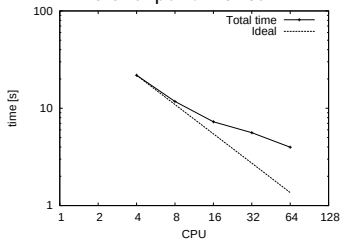


Refinement criteria:

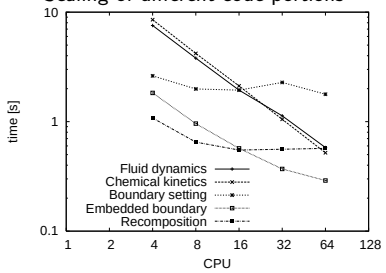
Y_i	$S_{Y_i} \cdot 10^{-4}$	$\eta^r_{Y_i} \cdot 10^{-4}$
O_2	10.0	4.0
H_2O	5.8	3.0
H	0.2	10.0
O	1.4	10.0
OH	2.3	10.0
H_2	1.3	4.0

$\epsilon_\rho = 0.02\,\mathrm{kg\,m^{-3}}$, $\epsilon_p = 16\,\mathrm{kPa}$

Parallel performance

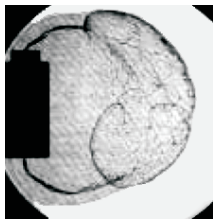


Scaling of different code portions

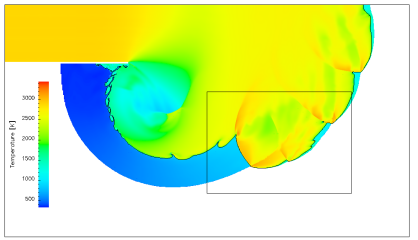


Detonation diffraction

- CJ detonation for $H_2 : O_2 : Ar/2 : 1 : 7$ at $T_0 = 298\,\mathrm{K}$ and $p_0 = 10\,\mathrm{kPa}$. Cell width $\lambda_c = 1.6\,\mathrm{cm}$
- Adaption criteria (similar as before):
  1. Scaled gradients of ρ and p
  2. Error estimation in Y_i by Richardson extrapolation
- $25\,\mathrm{Pts}/l_{ig}$. 5 refinement levels (2,2,2,4).
- Adaptive computations use up to $\sim 2.2\,\mathrm{M}$ instead of $\sim 150\,\mathrm{M}$ cells (uniform grid)
- $\sim 3850\,\mathrm{h}$ CPU ($\sim 80\,\mathrm{h}$ real time) on 48 nodes Athlon 1.4GHz

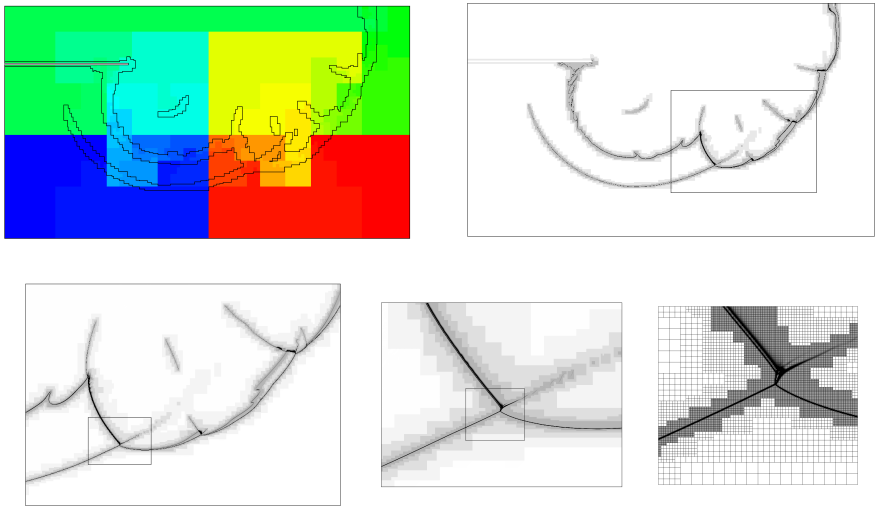


E. Schultz. *Detonation diffraction through an abrupt area expansion*. PhD thesis, California Institute of Technology, Pasadena, California, April 2000.



Detonation diffraction - adaptation

Final distribution to 48 nodes and density distribution on four refinement levels

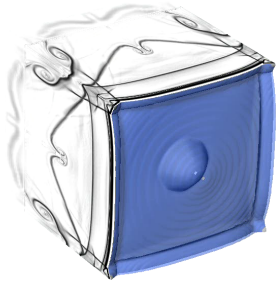


code/amroc/doc/html/apps/clawpack_2applications_2euler__chem_22d_2Diffraction_2src_2Problem_8h_source.html

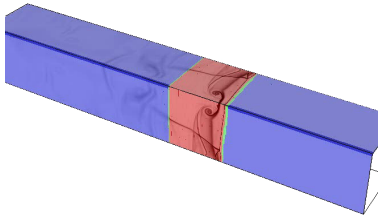
Detonation cell structure in 3D

- Simulation of only one quadrant
- $44.8\,\mathrm{Pts}/l_{ig}$ for $H_2 : O_2 : Ar$ CJ detonation
- SAMR base grid 400x24x24, 2 additional refinement levels (2, 4)
- Simulation uses $\sim 18\,\mathrm{M}$ cells instead of $\sim 118\,\mathrm{M}$ (unigrid)
- $\sim 51,000\,\mathrm{h}$ CPU on 128 CPU Compaq Alpha. $\mathcal{H}$: 37.6 %, $\mathcal{S}$: 25.1 %

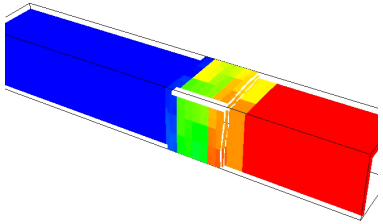
Schlieren and isosurface of Y_{OH}



Schlieren on refinement levels



Distribution to 128 processors



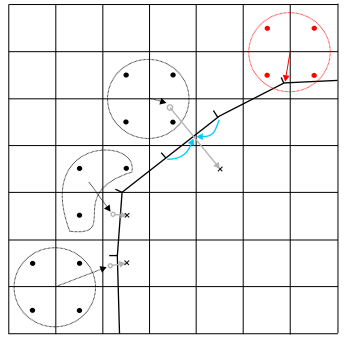
code/amroc/doc/html/apps/clawpack_2applications_2euler__chem_23d_2StrehlowH2O2_2StatDet_2src_2Problem_8h_source.html

Outline

Construction of coupling data

- Moving boundary/interface is treated as a moving contact discontinuity and represented by level set [Fedkiw, 2002][Arienti et al., 2003]
- One-sided construction of mirrored ghost cell and new FEM nodal point values
- FEM ansatz-function interpolation to obtain intermediate surface values
- Explicit coupling possible if geometry and velocities are prescribed for the more compressible medium [Specht, 2000]

$u_n^F := u_n^S(t)|_{\mathcal{I}}$
`UpdateFluid(`Δt`)`
$\sigma_{nn}^S := p^F(t+\Delta t)|_{\mathcal{I}}$
`UpdateSolid(`Δt`)`
$t := t + \Delta t$



Coupling conditions on interface

$$\left.\begin{array}{rcl} u_n^S &=& u_n^F \\ \sigma_{nn}^S &=& p^F \\ \sigma_{nm}^S &=& 0 \end{array}\right|_{\mathcal{I}}$$

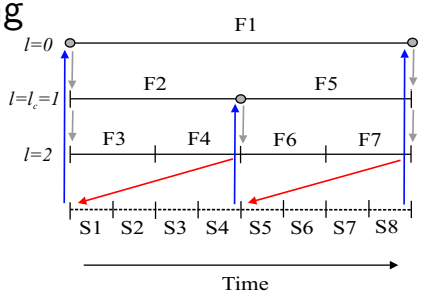
Usage of SAMR

- Eulerian SAMR + non-adaptive Lagrangian FEM scheme
- Exploit SAMR time step refinement for effective coupling to solid solver
  - Lagrangian simulation is called only at level $l_c \leq l_{\max}$
  - SAMR refines solid boundary at least at level l_c
  - Additional levels can be used resolve geometric ambiguities
- Nevertheless: Inserting sub-steps accommodates for time step reduction from the solid solver within an SAMR cycle
- Communication strategy:
  - Updated boundary info from solid solver must be received before regridding operation
  - Boundary data is sent to solid when highest level available
- Inter-solver communication (point-to-point or globally) managed on the fly special coupling module

SAMR algorithm for FSI coupling

`AdvanceLevel(`l`)`

`Repeat` r_l `times`
  `Set ghost cells of` $\mathbf{Q}^l(t)$
  `CPT(`φ^l`,` C^l`,` $\mathcal{I}$`,` δ_l`)`
  `If time to regrid?`
    `Regrid(`l`)`
  `UpdateLevel(`l`)`
  `If level` $l+1$ `exists?`
    `Set ghost cells of` $\mathbf{Q}^l(t+\Delta t_l)$
    `AdvanceLevel(`$l+1$`)`
    `Average` $\mathbf{Q}^{l+1}(t+\Delta t_l)$ `onto` $\mathbf{Q}^l(t+\Delta t_l)$
  `If` $l = l_c$`?`
    `SendInterfaceData(`$p^F(t+\Delta t_l)|_{\mathcal{I}}$`)`
    `If` $(t+\Delta t_l) < (t_0+\Delta t_0)$`?`
      `ReceiveInterfaceData(`$\mathcal{I}$`,` $\mathbf{u}^S|_{\mathcal{I}}$`)`
  $t := t + \Delta t_l$



- Call CPT algorithm before `Regrid(l)`
- Include also call to `CPT(·)` into `Recompose(l)` to ensure consistent level set data on levels that have changed
- Communicate boundary data on coupling level l_c

Fluid and solid update / exchange of time steps

`FluidStep( )`

$\Delta\tau_F := \min_{l=0,\cdots,l_{\max}} (R_l \cdot$ `StableFluidTimeStep(`l`)`, $\Delta\tau_S)$
$\Delta t_l := \Delta\tau_F / R_l$ `for` $l = 0, \cdots, L$
`ReceiveInterfaceData(`$\mathcal{I}$, $\mathbf{u}^S|_{\mathcal{I}}$`)`
`AdvanceLevel(0)`

`SolidStep( )`

$\Delta\tau_S := \min(K \cdot R_{l_c} \cdot$ `StableSolidTimeStep()`, $\Delta\tau_F)$
`Repeat` R_{l_c} `times`
  $t_{\text{end}} := t + \Delta\tau_S / R_{l_c}$, $\Delta t := \Delta\tau_S / (K R_{l_c})$
  `While` $t < t_{\text{end}}$
    `SendInterfaceData(`$\mathcal{I}(t)$, $\vec{u}^S|_{\mathcal{I}}(t)$`)`
    `ReceiveInterfaceData(`$p^F|_{\mathcal{I}}$`)`
    `UpdateSolid(`$p^F|_{\mathcal{I}}$, Δt`)`
    $t := t + \Delta t$
    $\Delta t := \min($`StableSolidTimeStep()`, $t_{\text{end}} - t)$

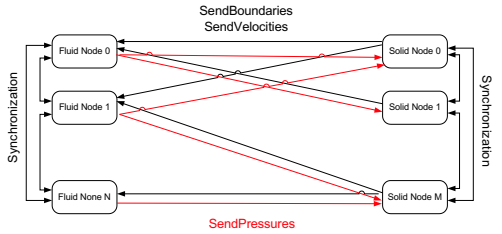
with $R_l = \prod_{\iota=0}^{l} r_\iota$

- Time step stays constant for R_{l_c} steps, which correponds to one fluid step at level 0

Parallelization strategy for coupled simulations

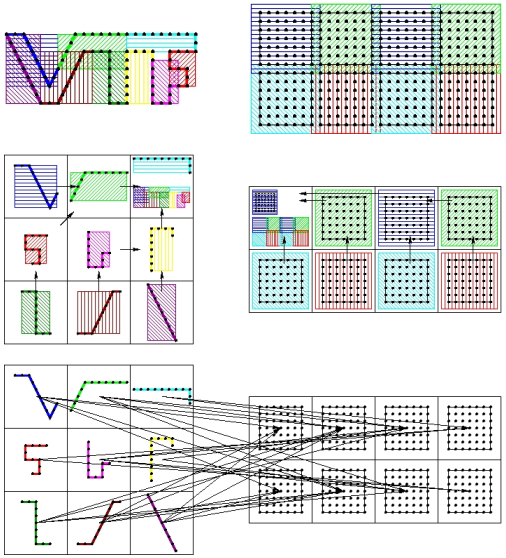
Coupling of an Eulerian FV fluid Solver and a Lagrangian FEM Solver:

- Distribute both meshes seperately and copy necessary nodal values and geometry data to fluid nodes
- Setting of ghost cell values becomes strictly local operation
- Construct new nodal values strictly local on fluid nodes and transfer them back to solid nodes
- Only surface data is transfered
- Asynchronous communication ensures scalability
- Generic encapsulated implementation guarantees reusability

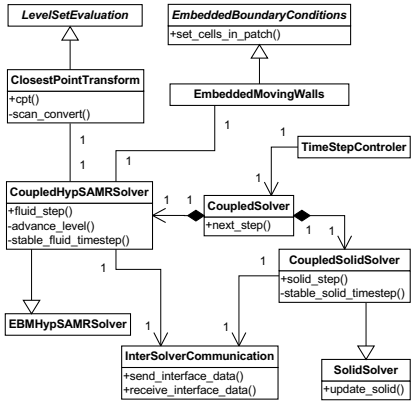


Eulerian/Lagrangian communication module

1. Put bounding boxes around each solid processors piece of the boundary and around each fluid processors grid
2. Gather, exchange and broadcast of bounding box information
3. Optimal point-to-point communication pattern, non-blocking



FSI coupling



- Coupling algorithm implemented in further derived HypSAMRSolver class
- Level set evaluation always with CPT algorithm
- Parallel communication through efficient non-blocking communication module ELC
- Time step selection for both solvers through CoupledSolver class

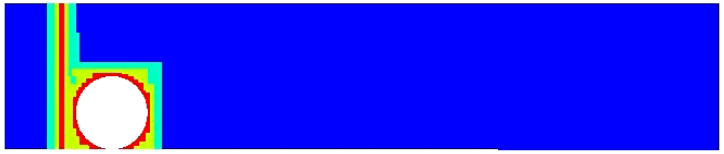
- **AMRELCGFMSolver<VectorType, FixupType, FlagType, dim >** is the derived AMRSolver<>class. `code/amroc/doc/html/amr/classAMRELCGFMSolver.html`
- Uses the Eulerian interface of the Lagrangian communication routines `code/stlib/doc/html/elc/elc__page.html`
- and the closest point transform algorithm `code/stlib/doc/html/cpt/cpt__page.html` through the **CPTLevelSet<DataType, dim >** `code/amroc/doc/html/amr/classCPTLevelSet.html`

Lift-up of a spherical body

Cylindrical body hit by Mach 3 shockwave, 2D test case by [Falcovitz et al., 1997]

Schlieren plot of density

Refinement levels



code/amroc/doc/html/apps/clawpack_2applications_2euler_22d_2SphereLiftOff_2src_2Problem_8h_source.html

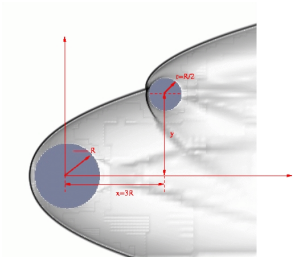
Proximal bodies in hypersonic flow

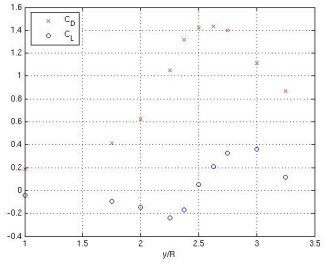
Flow modeled by Euler equations for a single polytropic gas with $p = (\gamma - 1)\,\rho e$

$$\partial_t \rho + \partial_{x_n}(\rho u_n) = 0\,, \quad \partial_t(\rho u_k) + \partial_{x_n}(\rho u_k u_n + \delta_{kn} p) = 0\,, \quad \partial_t(\rho E) + \partial_{x_n}(u_n(\rho E + p)) = 0$$

Numerical approximation with

- Finite volume flux-vector splitting scheme with MUSCL reconstruction, dimensional splitting
- Spherical bodies, force computation with overlaid lattitude-longitude mesh to obtain drag and lift coefficients $C_{D,L} = \dfrac{2F_{D,L}}{\rho v^2 \pi r^2}$
- inflow $M = 10$, C_D and C_L on secondary sphere, lateral position varied, no motion





Verification and validation

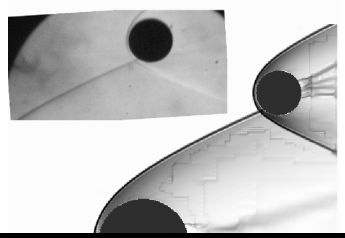
Static force measurements, $M = 10$: [Laurence et al., 2007]

- Refinement study: $40 \times 40 \times 32$ base grid , up to without AMR up to $\sim 209.7 \cdot 10^6$ cells, largest run $\sim 35,000\,\mathrm{h}$ CPU

$l_{\max}$	C_D	ΔC_D	C_L	ΔC_L
1	1.264		-0.176	
2	1.442	0.178	-0.019	0.157
3	1.423	-0.019	0.052	0.071
4	1.408	-0.015	0.087	0.035

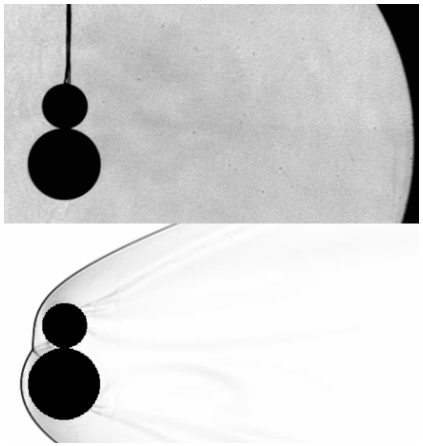
- Comparison with experimental results: 3 additional levels, $\sim 2000\,\mathrm{h}$ CPU

	Experimental	Computational
C_D	1.11 ± 0.08	1.01
C_L	0.29 ± 0.05	0.28



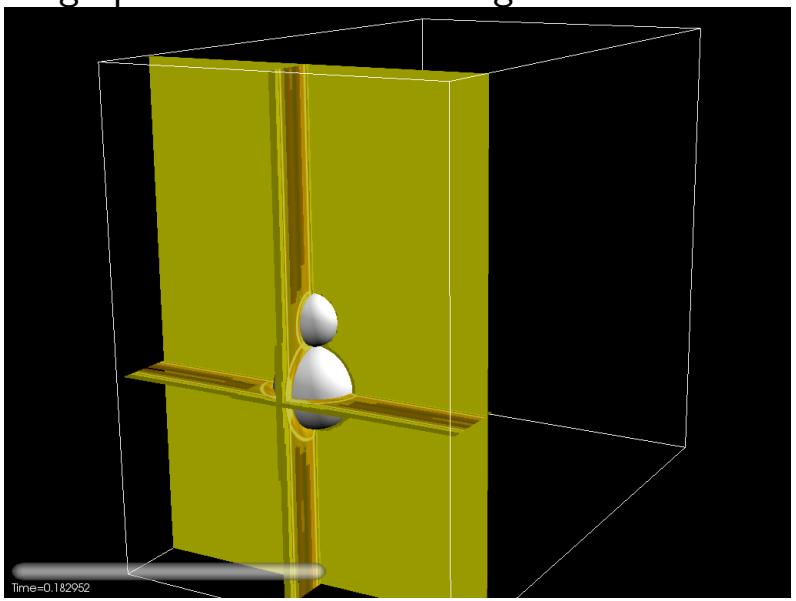
Dynamic motion, $M = 4$:

- Base grid $150 \times 125 \times 90$, two additional levels with $r_{1,2} = 2$
- 24,704 time steps, $36,808\,\mathrm{h}$ CPU on 256 cores IBM BG/P



[Laurence and Deiterding, 2011]

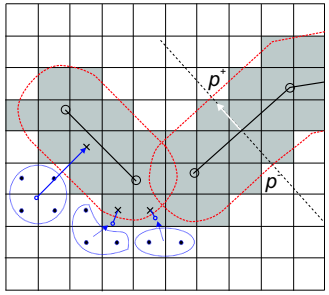
Schlieren graphics on refinement regions



code/amroc/doc/html/apps/clawpack_2applications_2euler_23d_2Spheres_2src_2Problem_8h_source.html

Treatment of thin structures

- Thin boundary structures or lower-dimensional shells require "thickening" to apply embedded boundary method
- Unsigned distance level set function φ
- Treat cells with $0 < \varphi < d$ as ghost fluid cells



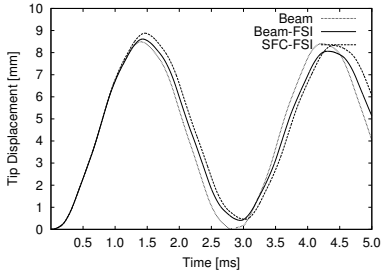
- Leaving φ unmodified ensures correctness of $\nabla\varphi$
- Use face normal in shell element to evaluate in $\Delta p = p^+ - p^-$
- Utilize finite difference solver using the beam equation

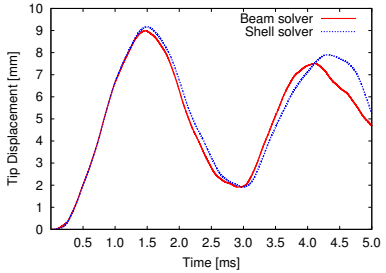
$$\rho_s h \frac{\partial^2 w}{\partial t^2} + EI \frac{\partial^4 w}{\partial \bar{x}^4} = p^F$$

to verify FSI algorithms

FSI verification by elastic vibration

- Thin steel plate (thickness $h = 1\,\mathrm{mm}$, length $50\,\mathrm{mm}$), clamped at lower end
- $\rho_s = 7600\,\mathrm{kg/m^3}$, $E = 220\,\mathrm{GPa}$, $I = h^3/12$, $\nu = 0.3$
- Modeled with beam solver (101 points) and thin-shell FEM solver (325 triangles) by F. Cirak
- Left: Coupling verification with constant instantenous loading by $\Delta p = 100\,\mathrm{kPa}$
- Right: FSI verification with Mach 1.21 shockwave in air ($\gamma = 1.4$)

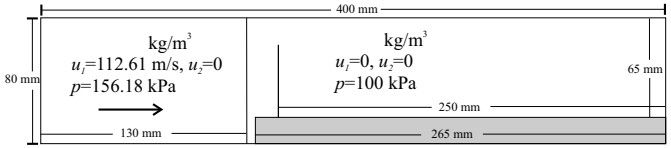




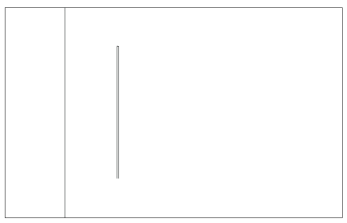
Shock-driven elastic panel motion

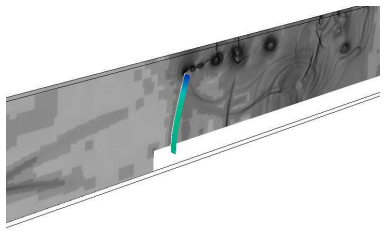
Test case suggested by [Giordano et al., 2005]

- Forward facing step geometry, fixed walls everywhere except at inflow



- SAMR base mesh $320 \times 64 (\times 2)$, $r_{1,2} = 2$
- Intel 3.4GHz Xeon dual processors, GB Ethernet interconnect
  - Beam-FSI: 12.25 h CPU on 3 fluid CPU + 1 solid CPU
    code/doc/html/capps/beam-amroc_2VibratingBeam_2src_2FluidProblem_8h_source.html,
    code/doc/html/capps/beam-amroc_2VibratingBeam_2src_2SolidProblem_8h_source.html
  - FEM-FSI: 322 h CPU on 14 fluid CPU + 2 solid CPU
    code/doc/html/capps/sfc-amroc_2VibratingPanel_2src_2FluidProblem_8h_source.html,
    code/doc/html/capps/VibratingPanel_2src_2ShellManagerSpecific_8h_source.html





$t = 1.56\,\mathrm{ms}$ after impact

Detonation-driven plastic deformation

Chapman-Jouguet detonation in a tube filled with a stoichiometric ethylene and oxygen ($C_2H_4 + 3\,O_2$, $295\,\mathrm{K}$) mixture. Euler equations with single exothermic reaction $A \longrightarrow B$

$$\partial_t \rho + \partial_{x_n}(\rho u_n) = 0\,, \quad \partial_t(\rho u_k) + \partial_{x_n}(\rho u_k u_n + \delta_{kn} p) = 0\,, k = 1, \ldots, d$$

$$\partial_t(\rho E) + \partial_{x_n}(u_n(\rho E + p)) = 0\,, \quad \partial_t(Y\rho) + \partial_{x_n}(Y \rho u_n) = \psi$$

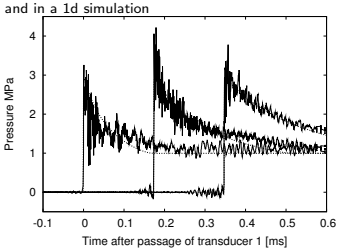
with

$$p = (\gamma - 1)(\rho E - \frac{1}{2}\rho u_n u_n - \rho Y q_0) \quad \text{and} \quad \psi = -kY\rho \exp\left(\frac{-E_A \rho}{p}\right)$$

modeled with heuristic detonation model by [Mader, 1979]

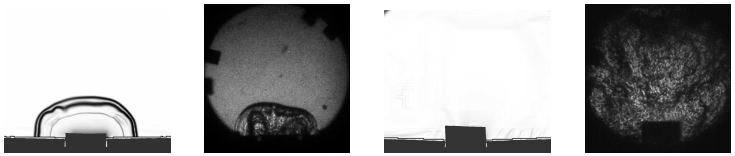
$V := \rho^{-1}$, $V_0 := \rho_0^{-1}$, $V_{CJ} := \rho_{CJ}$
$Y' := 1 - (V - V_0)/(V_{CJ} - V_0)$
If $0 \le Y' \le 1$ and $Y > 10^{-8}$ then
  If $Y < Y'$ and $Y' < 0.9$ then $Y' := 0$
  If $Y' < 0.99$ then $p' := (1 - Y')p_{CJ}$
    else $p' := p$
  $\rho_A := Y'\rho$
  $E := p'/(\rho(\gamma - 1)) + Y' q_0 + \frac{1}{2} u_n u_n$

Comparison of the pressure traces in the experiment and in a 1d simulation



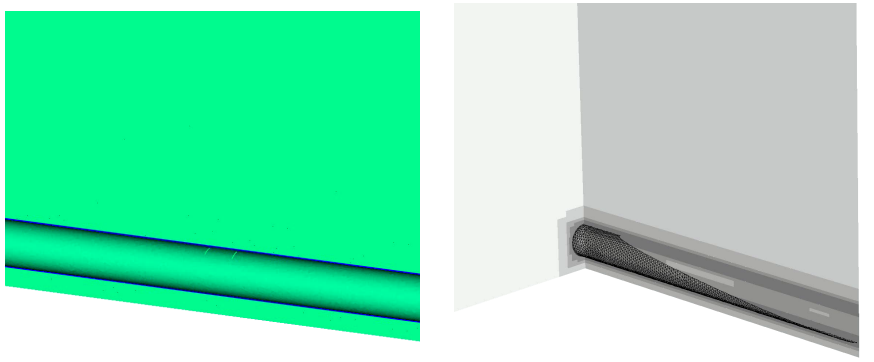
Tube with flaps

- Fluid: VanLeer FVS
  - Detonation model with $\gamma = 1.24$, $p_{\mathrm{CJ}} = 3.3\,\mathrm{MPa}$, $D_{\mathrm{CJ}} = 2376\,\mathrm{m/s}$
  - AMR base level: $104 \times 80 \times 242$, $r_{1,2} = 2$, $r_3 = 4$
  - $\sim 4 \cdot 10^7$ cells instead of $7.9 \cdot 10^9$ cells (uniform)
  - Tube and detonation fully refined
  - Thickening of 2D mesh: $0.81\,\mathrm{mm}$ on both sides (real $0.445\,\mathrm{mm}$)
- Solid: thin-shell solver by F. Cirak
  - Aluminum, J2 plasticity with hardening, rate sensitivity, and thermal softening
  - Mesh: 8577 nodes, 17056 elements
- 16+2 nodes 2.2 GHz AMD Opteron quad processor, PCI-X 4x Infiniband network, $\sim 4320\,\mathrm{h}$ CPU to $t_{end} = 450\,\mu\mathrm{s}$



0.032 ms 0.030 ms 0.212 ms 0.210 ms

Tube with flaps: results



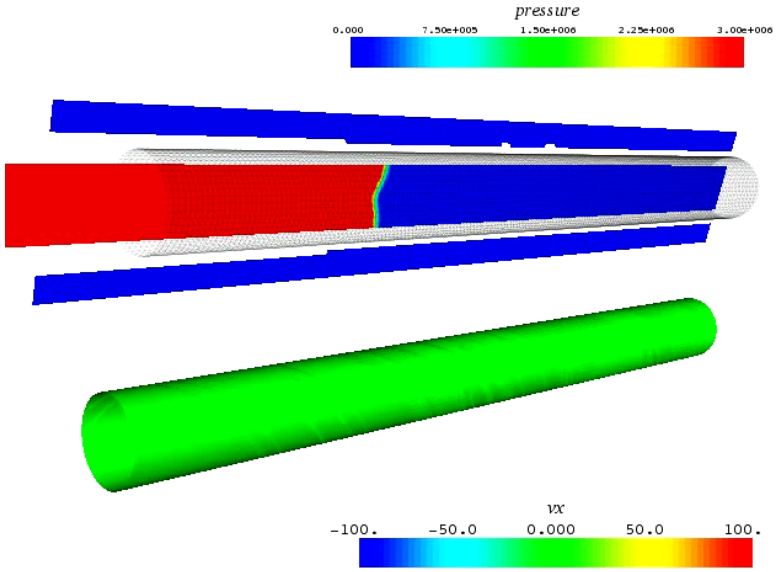
Fluid density and diplacement in y-direction in solid

Schlieren plot of fluid density on refinement levels

[Cirak et al., 2007]

code/doc/html/capps/sfc-amroc_2TubeCJBurnFlaps_2src_2FluidProblem_8h_source.html,
code/doc/html/capps/TubeCJBurnFlaps_2src_2ShellManagerSpecific_8h_source.html

Coupled fracture simulation



code/doc/html/capps/sfc-amroc_2TubeCJBurnFrac_2src_2FluidProblem_8h_source.html,
code/doc/html/capps/TubeCJBurnFrac_2src_2ShellManagerSpecific_8h_source.html

Underwater explosion modeling

Volume fraction based two-component model with $\sum_{i=1}^{m} \alpha^i = 1$, that defines mixture quantities as

$$\rho = \sum_{i=1}^{m} \alpha^i \rho^i \,, \quad \rho u_n = \sum_{i=1}^{m} \alpha^i \rho^i u_n^i \,, \quad \rho e = \sum_{i=1}^{m} \alpha^i \rho^i e^i$$

Assuming total pressure $p = (\gamma - 1)\,\rho e - \gamma p_\infty$ and speed of sound $c = (\gamma\,(p + p_\infty)/\rho)^{1/2}$ yields

$$\frac{p}{\gamma - 1} = \sum_{i=1}^{m} \frac{\alpha^i p^i}{\gamma^i - 1} \,, \quad \frac{\gamma p_\infty}{\gamma - 1} = \sum_{i=1}^{m} \frac{\alpha^i \gamma^i p_\infty^i}{\gamma^i - 1}$$

and the overall set of equations [Shyue, 1998]

$$\partial_t \rho + \partial_{x_n}(\rho u_n) = 0 \,, \quad \partial_t(\rho u_k) + \partial_{x_n}(\rho u_k u_n + \delta_{kn} p) = 0 \,, \quad \partial_t(\rho E) + \partial_{x_n}(u_n(\rho E + p)) = 0$$

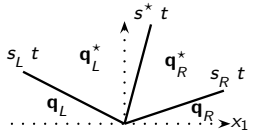
$$\frac{\partial}{\partial t}\left(\frac{1}{\gamma - 1}\right) + u_n \frac{\partial}{\partial x_n}\left(\frac{1}{\gamma - 1}\right) = 0 \,, \quad \frac{\partial}{\partial t}\left(\frac{\gamma p_\infty}{\gamma - 1}\right) + u_n \frac{\partial}{\partial x_n}\left(\frac{\gamma p_\infty}{\gamma - 1}\right) = 0$$

Oscillation free at contacts: [Abgrall and Karni, 2001][Shyue, 2006]

Approximate Riemann solver

Use HLLC approach because of robustness and positivity preservation

$$\mathbf{q}^{HLLC}(x_1,t) = \begin{cases} \mathbf{q}_L\,, & x_1 < s_L\,t, \\ \mathbf{q}_L^\star\,, & s_L\,t \le x_1 < s^\star\,t, \\ \mathbf{q}_R^\star\,, & s^\star\,t \le x_1 \le s_R\,t, \\ \mathbf{q}_R\,, & x_1 > s_R\,t, \end{cases}$$



Wave speed estimates [Davis, 1988] $s_L = \min\{u_{1,L} - c_L, u_{1,R} - c_R\}$, $s_R = \max\{u_{1,L} + c_L, u_{1,R} + c_R\}$
Unkown state [Toro et al., 1994]

$$s^\star = \frac{p_R - p_L + s_L u_{1,L}(s_L - u_{1,L}) - \rho_R u_{1,R}(s_R - u_{1,R})}{\rho_L(s_L - u_{1,L}) - \rho_R(s_R - u_{1,R})}$$

$$\mathbf{q}_\tau^\star = \left[\eta, \eta s^\star, \eta u_2, \eta\left[\frac{(\rho E)_\tau}{\rho_\tau} + (s^\star - u_{1,\tau})\left(s_\tau + \frac{p_\tau}{\rho_\tau(s_\tau - u_{1,\tau})}\right)\right], \frac{1}{\gamma_\tau - 1}, \frac{\gamma_\tau p_{\infty,\tau}}{\gamma_\tau - 1}\right]^T$$

$$\eta = \rho_\tau \frac{s_\tau - u_{1,\tau}}{s_\tau - s^\star}\,, \quad \tau = \{L, R\}$$

Evaluate waves as $\mathcal{W}_1 = \mathbf{q}_L^\star - \mathbf{q}_L$, $\mathcal{W}_2 = \mathbf{q}_R^\star - \mathbf{q}_L^\star$, $\mathcal{W}_3 = \mathbf{q}_R - \mathbf{q}_R^\star$ and $\lambda_1 = s_L$, $\lambda_2 = s^\star$, $\lambda_3 = s_R$ to compute the fluctuations $\mathcal{A}^-\Delta = \sum_{\lambda_\nu<0}\lambda_\nu\,\mathcal{W}_\nu$, $\mathcal{A}^+\Delta = \sum_{\lambda_\nu\ge 0}\lambda_\nu\,\mathcal{W}_\nu$ for $\nu = \{1,2,3\}$

Overall scheme: Wave Propagation method [Shyue, 2006]

Underwater explosion FSI simulations

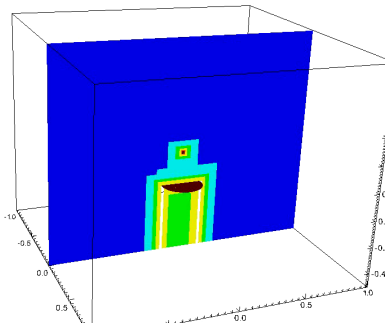
- Air: $\gamma^A = 1.4$, $p_\infty^A = 0$, $\rho^A = 1.29\,\mathrm{kg/m^3}$
- Water: $\gamma^W = 7.415$, $p_\infty^W = 296.2\,\mathrm{MPa}$, $\rho^W = 1027\,\mathrm{kg/m^3}$
- Cavitation modeling with pressure cut-off model at $p = -1\,\mathrm{MPa}$
- 3D simulation of deformation of air backed aluminum plate with $r = 85\,\mathrm{mm}$, $h = 3\,\mathrm{mm}$ from underwater explosion
  - Water basin [Ashani and Ghamsari, 2008] $2\,\mathrm{m} \times 1.6\,\mathrm{m} \times 2\,\mathrm{m}$
  - Explosion modeled as energy increase ($m_{C4} \cdot 6.06\,\mathrm{MJ/kg}$) in sphere with r=5mm
  - $\rho_s = 2719\,\mathrm{kg/m3}$, $E = 69\,\mathrm{GPa}$, $\nu = 0.33$, J2 plasticity model, yield stress $\sigma_y = 217.6\,\mathrm{MPa}$
- 3D simulation of copper plate $r = 32\,\mathrm{mm}$, $h = 0.25\,\mathrm{mm}$ rupturing due to water hammer
  - Water-filled shocktube $1.3\,\mathrm{m}$ with driver piston [Deshpande et al., 2006]
  - Piston simulated with separate level set, see [Deiterding et al., 2009] for pressure wave
  - $\rho_s = 8920\,\mathrm{kg/m3}$, $E = 130\,\mathrm{GPa}$, $\nu = 0.31$, J2 plasticity model, $\sigma_y = 38.5\,\mathrm{MPa}$, cohesive interface model, max. tensile stress $\sigma_c = 525\,\mathrm{MPa}$

Underwater explosion simulation

- AMR base grid $50 \times 40 \times 50$, $r_{1,2,3} = 2$, $r_4 = 4$, $l_c = 3$, highest level restricted to initial explosion center, 3rd and 4th level to plate vicinity
- Triangular mesh with 8148 elements
- Computations of 1296 coupled time steps to $t_{end} = 1\,\mathrm{ms}$
- 10+2 nodes 3.4 GHz Intel Xeon dual processor, $\sim 130\,\mathrm{h}$ CPU

Maximal deflection [mm]

	Exp.	Sim.
$20\,\mathrm{g}, d = 25\,\mathrm{cm}$	28.83	25.88
$30\,\mathrm{g}, d = 30\,\mathrm{cm}$	30.09	27.31



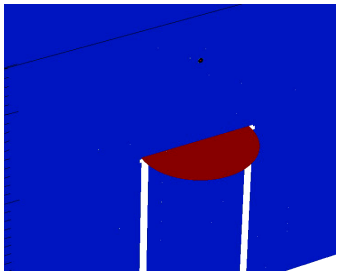
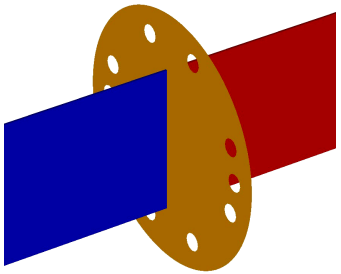
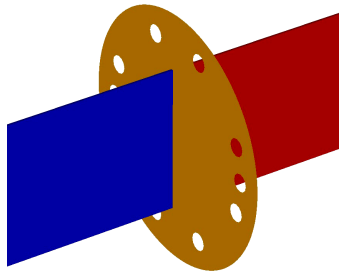


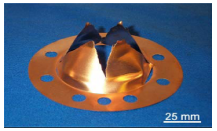
Plate in underwater shocktube

- AMR base mesh $374 \times 20 \times 20$, $r_{1,2} = 2$, $l_c = 2$, solid mesh: 8896 triangles
- ~ 1250 coupled time steps to $t_{end} = 1\,\mathrm{ms}$
- 6+6 nodes 3.4 GHz Intel Xeon dual processor, $\sim 800\,\mathrm{h}$ CPU

`code/doc/html/capps/sfc-amroc_2WaterBlastFracture_2src_2FluidProblem_8h_source.html`, `code/doc/html/capps/WaterBlastFracture_2src_2ShellManagerSpecific_8h_source.html`





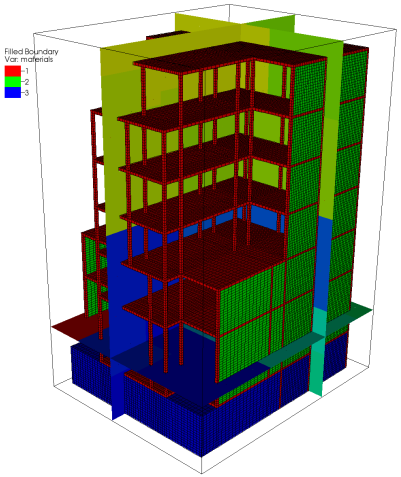


$p_0 = 64\,\mathrm{MPa}$

$p_0 = 173\,\mathrm{MPa}$

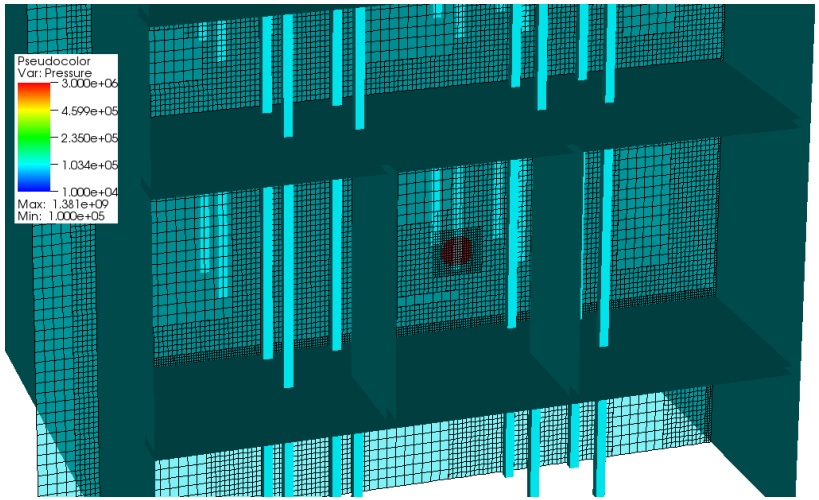
Blast explosion in a multistory building

- $20\,\mathrm{m} \times 40\,\mathrm{m} \times 25\,\mathrm{m}$ seven-story building similar to [Luccioni et al., 2004]
- Spherical energy deposition $\equiv 400\,\mathrm{kg}$ TNT, $r = 0.5\,\mathrm{m}$ in lobby of building
- SAMR: $80 \times 120 \times 90$ base level, three additional levels $r_{1,2} = 2$, $l_{\mathrm{fsi}} = 1$, $k = 1$
- Simulation with ground: 1,070 coupled time steps, $830\,\mathrm{h}$ CPU ($\sim 25.9\,\mathrm{h}$ wall time) on 31+1 cores
- $\sim 8,000,000$ cells instead of 55,296,000 (uniform)
- 69,709 hexahedral elements and with material parameters. [Deiterding and Wood, 2013]

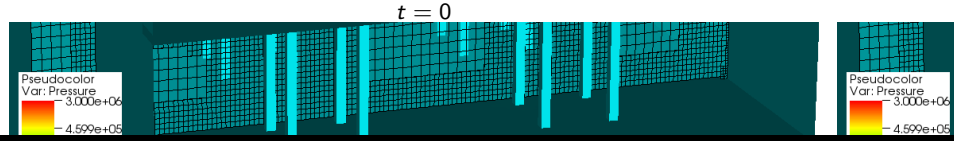


	ρ_s [kg/m^3]	σ_0 [MPa]	E_T [GPa]	β	K [GPa]	G [GPa]	$\bar{\epsilon}^p$	p_f [MPa]
Columns	2010	50	11.2	1.0	21.72	4.67	0.02	-30
Walls	2010	25	11.2	1.0	6.22	4.67	0.01	-15

Blast explosion in a multistory building – II



$t = 0$



References I

[Abgrall and Karni, 2001] Abgrall, R. and Karni, S. (2001). Computations of compressible multifluids. *J. Comput. Phys.*, 169:594–523.

[Aftosmis, 1997] Aftosmis, M. J. (1997). Solution adaptive Ccartesian grid methods for aerodynamic flows with complex geometries. Technical Report Lecture Series 1997-2, von Karman Institute for Fluid Dynamics.

[Arienti et al., 2003] Arienti, M., Hung, P., Morano, E., and Shepherd, J. E. (2003). A level set approach to Eulerian-Lagrangian coupling. *J. Comput. Phys.*, 185:213–251.

[Ashani and Ghamsari, 2008] Ashani, J. Z. and Ghamsari, A. K. (2008). Theoretical and experimental analysis of plastic response of isotropic circular plates subjected to underwater explosion loading. *Mat.-wiss. u. Werkstofftechn.*, 39(2):171–175.

[Berger and Helzel, 2002] Berger, M. J. and Helzel, C. (2002). Grid aligned h-box methods for conservation laws in complex geometries. In *Proc. 3rd Intl. Symp. Finite Volumes for Complex Applications*, Porquerolles.

[Cirak et al., 2007] Cirak, F., Deiterding, R., and Mauch, S. P. (2007). Large-scale fluid-structure interaction simulation of viscoplastic and fracturing thin shells subjected to shocks and detonations. *Computers & Structures*, 85(11-14):1049–1065.

References II

[Davis, 1988] Davis, S. F. (1988). Simplified second-order Godunov-type methods. *SIAM J. Sci. Stat. Comp.*, 9:445–473.

[Deiterding, 2003] Deiterding, R. (2003). *Parallel adaptive simulation of multi-dimensional detonation structures*. PhD thesis, Brandenburgische Technische Universität Cottbus.

[Deiterding et al., 2009] Deiterding, R., Cirak, F., and Mauch, S. P. (2009). Efficient fluid-structure interaction simulation of viscoplastic and fracturing thin-shells subjected to underwater shock loading. In Hartmann, S., Meister, A., Schäfer, M., and Turek, S., editors, *Int. Workshop on Fluid-Structure Interaction. Theory, Numerics and Applications, Herrsching am Ammersee 2008*, pages 65–80. kassel university press GmbH.

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

[Deiterding and Wood, 2013] Deiterding, R. and Wood, S. L. (2013). Parallel adaptive fluid-structure interaction simulations of explosions impacting on building structures. *Computers & Fluids*, 88:719–729.

References III

[Deshpande et al., 2006] Deshpande, V. S., Heaver, A., and Fleck, N. A. (2006). An underwater shock simulator. *Royal Society of London Proceedings Series A*, 462(2067):1021–1041.

[Falcovitz et al., 1997] Falcovitz, J., Alfandary, G., and Hanoch, G. (1997). A two-dimensional conservation laws scheme for compressible flows with moving boundaries. *J. Comput. Phys.*, 138:83–102.

[Fedkiw, 2002] Fedkiw, R. P. (2002). Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method. *J. Comput. Phys.*, 175:200–224.

[Giordano et al., 2005] Giordano, J., Jourdan, G., Burtschell, Y., Medale, M., Zeitoun, D. E., and Houas, L. (2005). Shock wave impacts on deforming panel, an application of fluid-structure interaction. *Shock Waves*, 14(1-2):103–110.

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

[Harten and Hyman, 1983] Harten, A. and Hyman, J. M. (1983). Self-adjusting grid methods for one-dimensional hyperbolic conservation laws. *J. Comput. Phys.*, 50:235–269.

References IV

[Henshaw and Schwendeman, 2003] Henshaw, W. D. and Schwendeman, D. W. (2003). An adaptive numerical scheme for high-speed reactive flow on overlapping grids. *J. Comput. Phys.*, 191:420–447.

[Laurence and Deiterding, 2011] Laurence, S. J. and Deiterding, R. (2011). Shock-wave surfing. *J. Fluid Mech.*, 676:369–431.

[Laurence et al., 2007] Laurence, S. J., Deiterding, R., and Hornung, H. G. (2007). Proximal bodies in hypersonic flows. *J. Fluid Mech.*, 590:209–237.

[Luccioni et al., 2004] Luccioni, B. M., Ambrosini, R. D., and Danesi, R. F. (2004). Analysis of building collapse under blast loads. *Engineering & Structures*, 26:63–71.

[Mader, 1979] Mader, C. L. (1979). *Numerical modeling of detonations*. University of California Press, Berkeley and Los Angeles, California.

[Mauch, 2003] Mauch, S. P. (2003). *Efficient Algorithms for Solving Static Hamilton-Jacobi Equations*. PhD thesis, California Institute of Technology.

[Meakin, 1995] Meakin, R. L. (1995). An efficient means of adaptive refinement within systems of overset grids. In *12th AIAA Computational Fluid Dynamics Conference, San Diego*, AIAA-95-1722-CP.

References V

[Mittal and Iaccarino, 2005] Mittal, R. and Iaccarino, G. (2005). Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261.

[Murman et al., 2003] Murman, S. M., Aftosmis, M. J., and Berger, M. J. (2003). Implicit approaches for moving boundaries in a 3-d Cartesian method. In *41st AIAA Aerospace Science Meeting*, AIAA 2003-1119.

[Nourgaliev et al., 2003] Nourgaliev, R. R., Dinh, T. N., and Theofanus, T. G. (2003). On capturing of interfaces in multimaterial compressible flows using a level-set-based Cartesian grid method. Technical Report 05/03-1, Center for Risk Studies and Safety, UC Santa Barbara.

[Pember et al., 1999] Pember, R. B., Bell, J. B., Colella, P., Crutchfield, W. Y., and Welcome, M. L. (1999). An adaptive Cartesian grid method for unsteady compressible flows in irregular regions. *J. Comput. Phys.*, 120:287–304.

[Quirk, 1994a] Quirk, J. J. (1994a). An alternative to unstructured grids for computing gas dynamics flows around arbitrarily complex two-dimensional bodies. *Computers Fluids*, 23:125–142.

[Quirk, 1994b] Quirk, J. J. (1994b). A contribution to the great Riemann solver debate. *Int. J. Numer. Meth. Fluids*, 18:555–574.

References VI

[Roma et al., 1999] Roma, A. M., Perskin, C. S., and Berger, M. J. (1999). An adaptive version of the immersed boundary method. *J. Comput. Phys.*, 153:509–534.

[Sanders et al., 1998] Sanders, R., Morano, E., and Druguett, M.-C. (1998). Multidimensional dissipation for upwind schemes: Stability and applications to gas dynamics. *J. Comput. Phys.*, 145:511–537.

[Sethian, 1999] Sethian, J. A. (1999). *Level set methods and fast marching methods*. Cambridge University Press, Cambridge, New York.

[Shyue, 1998] Shyue, K.-M. (1998). An efficient shock-capturing algorithm for compressible multicomponent problems. *J. Comput. Phys.*, 142:208–242.

[Shyue, 2006] Shyue, K.-M. (2006). A volume-fraction based algorithm for hybrid barotropic and non-barotropic two-fluid flow problems. *Shock Waves*, 15:407–423.

[Specht, 2000] Specht, U. (2000). *Numerische Simulation mechanischer Wellen an Fluid-Festkörper-Mediengrenzen*. Number 398 in VDI Reihe 7. VDU Verlag, Düsseldorf.

References VII

[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.

[Tseng and Ferziger, 2003] Tseng, Y.-H. and Ferziger, J. H. (2003). A ghost-cell immersed boundary method for flow in complex geometry. *J. Comput. Phys.*, 192:593–623.

[Yamaleev and Carpenter, 2002] Yamaleev, N. K. and Carpenter, M. H. (2002). On accuracy of adaptive grid methods for captured shocks. *J. Comput. Phys.*, 181:280–316.

Lecture 4
Advanced topics

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

Outline

Approximation of Boltzmann equation

Is based on solving the Boltzmann equation with a simplified collision operator

$$\partial_t f + \mathbf{u} \cdot \nabla f = \omega(f^{eq} - f)$$

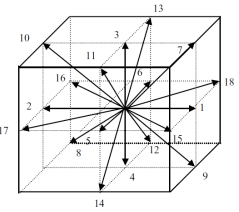
- $\mathrm{Kn} = l_f / L \ll 1$, where l_f is replaced with Δx
- Weak compressibilty and small Mach number assumed
- Assume a simplified phase space

Equation is approximated with a splitting approach.

1.) Transport step solves $\partial_t f_\alpha + \mathbf{e}_\alpha \cdot \nabla f_\alpha = 0$

Operator: $\mathcal{T}$: $\tilde{f}_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}, t)$

$$\rho(\mathbf{x}, t) = \sum_{\alpha=0}^{18} f_\alpha(\mathbf{x}, t), \quad \rho(\mathbf{x}, t) u_i(\mathbf{x}, t) = \sum_{\alpha=0}^{18} \mathbf{e}_{\alpha i} f_\alpha(\mathbf{x}, t)$$



Discrete velocities:

$$\mathbf{e}_\alpha = \begin{cases} 0, & \alpha = 0, \\ (\pm 1, 0, 0)c, (0, \pm 1, 0)c, (0, 0, \pm 1)c, & \alpha = 1, \ldots, 6, \\ (\pm 1, \pm 1, 0)c, (\pm 1, 0, \pm 1)c, (0, \pm 1, \pm 1)c, & \alpha = 7, \ldots, 18, \end{cases}$$

Approximation of thermal equilibrium

2.) Collision step solves $\partial_t f_\alpha = \omega(f_\alpha^{eq} - f_\alpha)$

Operator $\mathcal{C}$:

$$f_\alpha(\cdot, t + \Delta t) = \tilde{f}_\alpha(\cdot, t + \Delta t) + \omega_L \Delta t \left(\tilde{f}_\alpha^{eq}(\cdot, t + \Delta t) - \tilde{f}_\alpha(\cdot, t + \Delta t) \right)$$

with equilibrium function

$$f_\alpha^{eq}(\rho, \mathbf{u}) = \rho t_\alpha \left[1 + \frac{3 \mathbf{e}_\alpha \mathbf{u}}{c^2} + \frac{9(\mathbf{e}_\alpha \mathbf{u})^2}{2c^4} - \frac{3\mathbf{u}^2}{2c^2} \right]$$

with $t_\alpha = \frac{1}{9} \{3, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \}$

Pressure $\delta p = \sum_\alpha f_\alpha^{eq} c_s^2 = \rho c_s^2$.

Dev. stress $\Sigma_{ij} = \left(1 - \frac{\omega_L \Delta t}{2}\right) \sum_\alpha \mathbf{e}_{\alpha i} \mathbf{e}_{\alpha j} (f_\alpha^{eq} - f_\alpha)$

Is derived by assuming a Maxwell-Boltzmann distribution of f_α^{eq} and approximating the involved exp() function with a Taylor series to second-order accuracy.

Using the third-order equilibrium function

$$f_\alpha^{eq}(\rho, \mathbf{u}) = \rho t_\alpha \left[1 + \frac{3 \mathbf{e}_\alpha \mathbf{u}}{c^2} + \frac{9(\mathbf{e}_\alpha \mathbf{u})^2}{2c^4} - \frac{3\mathbf{u}^2}{2c^2} + \frac{\mathbf{e}_\alpha \mathbf{u}}{3c^2} \left(\frac{9(\mathbf{e}_\alpha \mathbf{u})^2}{2c^4} - \frac{3\mathbf{u}^2}{2c^2} \right) \right]$$

allows higher flow velocities.

Relation to Navier-Stokes equations

Inserting a Chapman-Enskog expansion, that is,

$$f_\alpha = f_\alpha(0) + \epsilon f_\alpha(1) + \epsilon^2 f_\alpha(2) + \ldots$$

and using

$$\frac{\partial}{\partial t} = \epsilon \frac{\partial}{\partial t_1} + \epsilon^2 \frac{\partial}{\partial t_2} + \ldots, \qquad \nabla = \epsilon \nabla_1 + \epsilon^2 \nabla_2 + \ldots$$

into the LBM and summing over α one can show that the continuity and moment equations are recoverd to $O(\epsilon^2)$ [Hou et al., 1996]

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0$$

$$\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u}$$

Kinematic viscosity and collision time are connected by

$$\nu = \frac{1}{3} \left(\frac{\tau_L}{\Delta t} - \frac{1}{2} \right) c \Delta x$$

from which one gets with $\sqrt{3} c_s = \frac{\Delta x}{\Delta t}$ [Hähnel, 2004]

$$\omega_L = \tau_L^{-1} = \frac{c_s^2}{\nu + \Delta t c_s^2 / 2}$$

Turbulence modeling

Pursue a large-eddy simulation approach with $\bar{f}_\alpha$ and $\bar{f}_\alpha^{eq}$, i.e.

1.) $\bar{\tilde{f}}_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = \bar{f}_\alpha(\mathbf{x}, t)$

2.) $\bar{f}_\alpha(\cdot, t + \Delta t) = \bar{\tilde{f}}_\alpha(\cdot, t + \Delta t) + \frac{1}{\tau^\star} \Delta t \left(\bar{\tilde{f}}_\alpha^{eq}(\cdot, t + \Delta t) - \bar{\tilde{f}}_\alpha(\cdot, t + \Delta t) \right)$

Effective viscosity: $\nu^\star = \nu + \nu_t = \frac{1}{3} \left(\frac{\tau_L^\star}{\Delta t} - \frac{1}{2} \right) c \Delta x$ with $\tau_L^\star = \tau_L + \tau_t$

Use Smagorinsky model to evaluate ν_t, e.g., $\nu_t = (C_{sm} \Delta x)^2 \bar{S}$, where

$$\bar{S} = \sqrt{2 \sum_{i,j} \bar{\mathbf{S}}_{ij} \bar{\mathbf{S}}_{ij}}$$

The filtered strain rate tensor $\bar{\mathbf{S}}_{ij} = (\partial_j \bar{u}_i + \partial_i \bar{u}_j)/2$ can be computed as a second moment as

$$\bar{\mathbf{S}}_{ij} = \frac{\Sigma_{ij}}{2 \rho c_s^2 \tau_L^\star \left(1 - \frac{\omega_L \Delta t}{2}\right)} = \frac{1}{2 \rho c_s^2 \tau_L^\star} \sum_\alpha \mathbf{e}_{\alpha i} \mathbf{e}_{\alpha j} (\bar{f}_\alpha^{eq} - \bar{f}_\alpha)$$

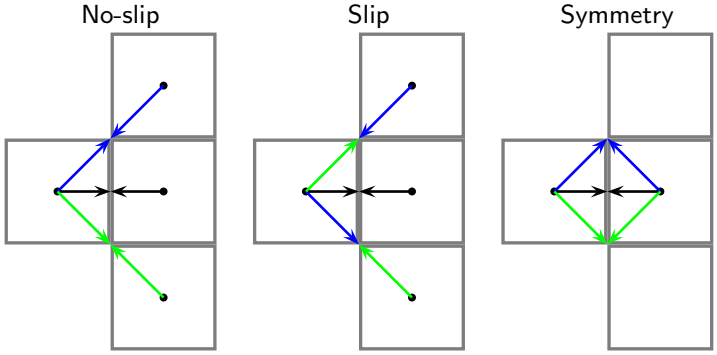
τ_t can be obtained as [Yu, 2004, Hou et al., 1996]

$$\tau_t = \frac{1}{2} \left(\sqrt{\tau_L^2 + 18 \sqrt{2} (\rho_0 c^2)^{-1} C_{sm}^2 \Delta x \bar{S}} - \tau_L \right)$$

Initial and boundary conditions

- Initial conditions are constructed as $f_\alpha^{eq}(\rho(t=0), \mathbf{u}(t=0))$

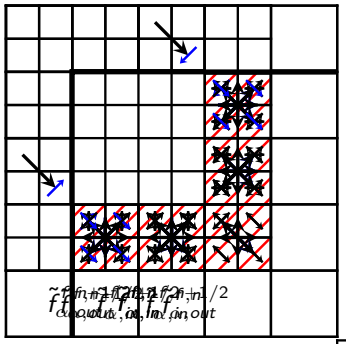
Simple boundary conditions:



- Outlet basically copies all distributions (zero gradient)
- Inlet and pressure boundary conditions use f_α^{eq}
- Embedded boundary conditions use ghost cell construction as before, then use $f_\alpha^{eq}(\rho', \mathbf{u}')$ to construct distributions in embedded ghost cells

Adaptive LBM

1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2} := \mathcal{C}(\tilde{f}_\alpha^{f,n})$ in interior.
4. $\tilde{f}_\alpha^{f,n+1/2} := \mathcal{T}(f_\alpha^{f,n+1/2})$ on whole fine mesh. $f_\alpha^{f,n+1} := \mathcal{C}(\tilde{f}_\alpha^{f,n+1/2})$ in interior.



5. Average $\tilde{f}_{\alpha,out}^{f,n+1/2}$ (inner halo layer), $\tilde{f}_{\alpha,out}^{f,n}$ (outer halo layer) to obtain $\tilde{f}_{\alpha,out}^{C,n}$.
6. Revert transport into halos: $\bar{f}_{\alpha,out}^{C,n} := \mathcal{T}^{-1}(\tilde{f}_{\alpha,out}^{C,n})$
7. Parallel synchronization of $f_\alpha^{C,n}, \bar{f}_{\alpha,out}^{C,n}$
8. Cell-wise update where correction is needed: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n}, \bar{f}_{\alpha,out}^{C,n})$

Algorithm equivalent to [Chen et al., 2006].

Classes

Directory `amroc/lbm` contains the lattice Boltzmann integrator that is in C++ throughout and also is built on the classes in amroc/amr/Interfaces.

- Several SchemeType-classes are already provided: **LBMD1Q3<DataType >**, **LBMD2Q9<DataType >**, **LBMD3Q19<DataType >**, **LBMD2Q9Thermal<DataType >**, **LBMD3Q19Thermal<DataType >** included a large number of boundary conditions.

  `code/amroc/doc/html/lbm/classLBMD1Q3.html code/amroc/doc/html/lbm/classLBMD2Q9.html`

  `code/amroc/doc/html/lbm/classLBMD3Q19Thermal.html`
- Using function within LBMD?D?, the special coarse-fine correction is implemented in **LBMFixup<LBMType, FixupType, dim>**

  `code/amroc/doc/html/lbm/classLBMFixup.html`
- **LBMIntegrator<LBMType, dim >**, **LBMGFMBoundary<LBMType, dim >**, etc. interface to the generic classes in amroc/amr/Interfaces

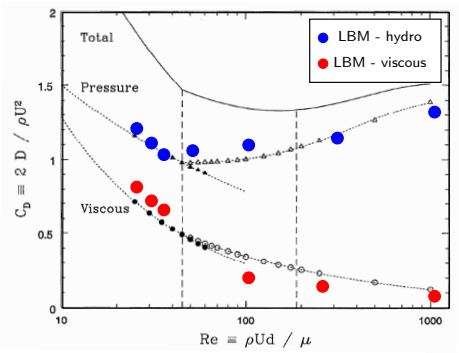
  `code/amroc/doc/html/amr/classSchemeGFMBoundary.html`
- **Problem.h**: Specific simulation is defined in Problem.h only. Predefined classes specified in **LBMStdProblem.h**, **LBMStdGFMProblem.h** and **LBMProblem.h**.

  `code/amroc/doc/html/lbm/LBMProblem_8h_source.html code/amroc/doc/html/lbm/LBMStdProblem_8h.html`

  `code/amroc/doc/html/lbm/LBMStdGFMProblem_8h.html`

Flow over 2D cylinder, $d = 2\,\mathrm{cm}$

- Air with $\nu = 1.61 \cdot 10^{-5}\,\mathrm{m}^2/\mathrm{s}$, $\rho = 1.205\,\mathrm{kg}/\mathrm{m}^3$
- Domain size $[-8d, 24d] \times [-8d, 8d]$
- Dynamic refinement based on velocity. Last level to refine structure further.
- Inflow from left. Characteristic boundary conditions [Schlaffer, 2013] elsewhere.



- Base lattice 320×160, 3 additional levels with factors $r_l = 2, 4, 4$.
- Resolution: ~ 320 points in diameter d
- Computation of C_D on 400 equidistant points along circle and averaged over time. Comparison above with [Henderson, 1995].

Flow over cylinder in 2d - $\mathrm{Re} = 300$, $u = 0.2415\,\mathrm{m/s}$

Isolines on refinement and distribution to processors

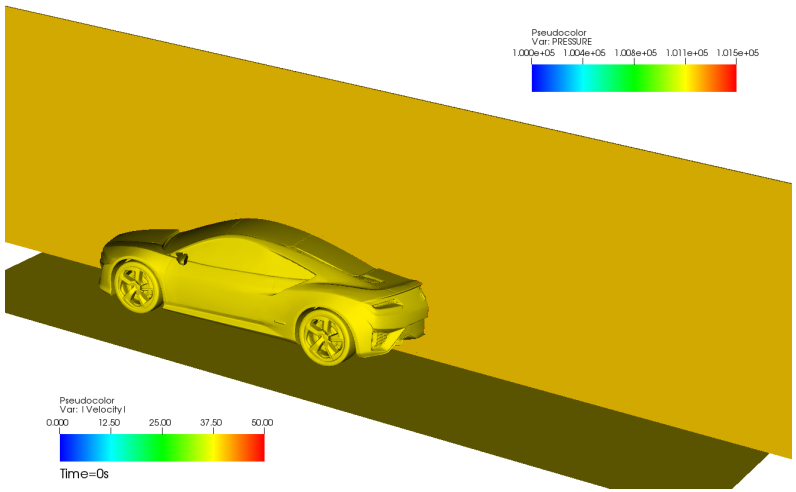
Mesh adaptation with LBM:

1. Level-wise evaluation of $\omega_L^l = \frac{c_s^2}{\nu + \Delta t_l c_s^2/2}$
2. Exchange of distributions streaming across refinement interfaces

`code/amroc/doc/html/apps/lbm_2applications_2Navier-Stokes_22d_2CylinderDrag_2src_2Problem_8h_source.html`

Wind tunnel simulation of a prototype car

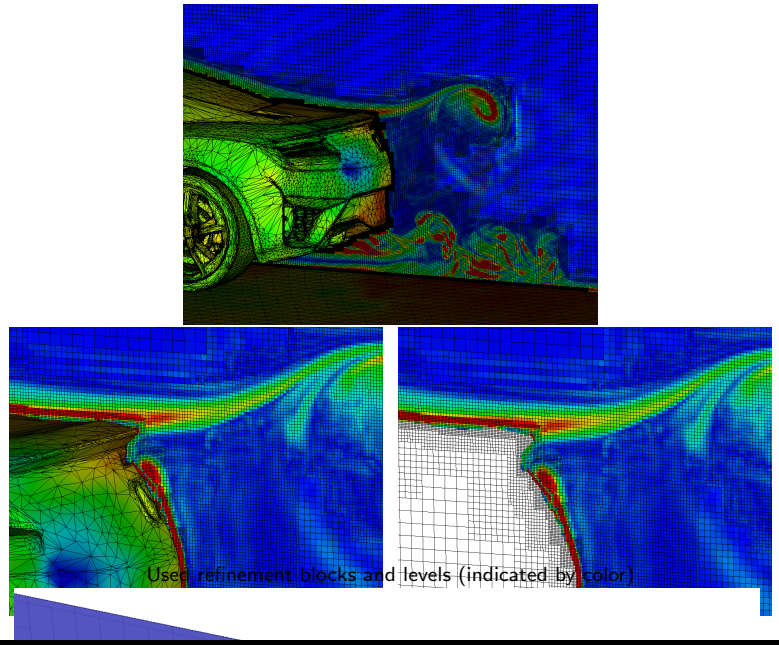
Fluid velocity and pressure on vehicle



- Inflow 40 $\mathrm{m/s}$. LES model active. Characteristic boundary conditions.
- To $t = 0.5\,\mathrm{s}$ ($\sim$ 4 characteristic lengths) with 31,416 time steps on finest level in $\sim 37\,\mathrm{h}$ on 200 cores (7389 h CPU). Channel: $15\,\mathrm{m} \times 5\,\mathrm{m} \times 3.3\,\mathrm{m}$

Outline

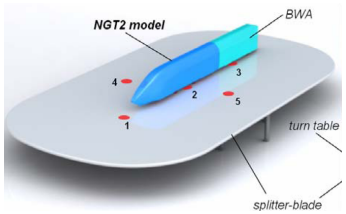
Mesh adaptation

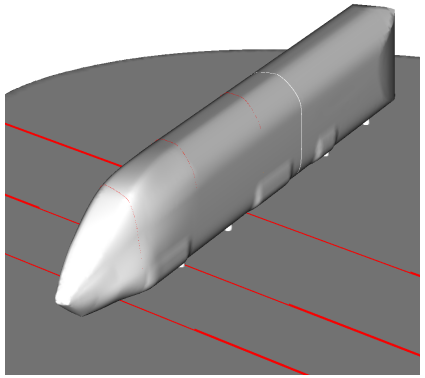


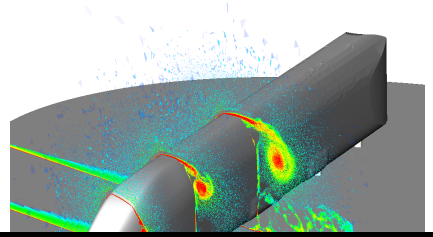
Used refinement blocks and levels (indicated by color)

Next Generation Train (NGT)

- 1:25 train model of 74,670 triangles
- Wind tunnel: air at room temperature with $33.48\,\mathrm{m/s}$, $\mathrm{Re} = 250,000$, yaw angle 30°
- Comparison between LBM (fluid air) and incompressible OpenFOAM solvers



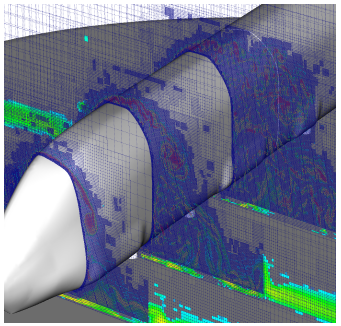




NGT model

- LBM-AMR computation with 5 additional levels, factor 2 refinement (uniform: 120.4e9 cells)
- Dynamic AMR until $T_c = 34$, then static for $\sim 12\,T_C$ to obtain average coefficients
- OpenFOAM simulations by M. Fragner (DLR)

Simulation	Mesh	CFX	CFY	CMX
Wind tunnel	–	-0.06	-5.28	-3.46
DDES	low	-0.40	-5.45	-3.61
Σ only	low	0.10	-0.04	-0.05
LES	high	-0.45	-6.07	-4.14
DDES	high	-0.43	-5.72	-3.77
LBM - p only	–	-0.30	-5.09	-3.46



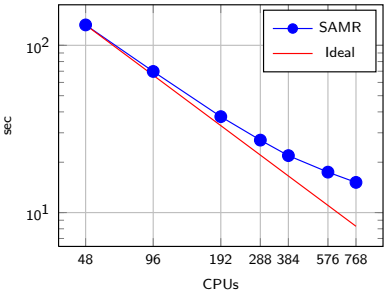
	LBM	DDES(l)	LES	DDES(h)
Cells	147M	34.1M	219M	219M
y^+	43	3.2	1.7	1.7
x^+, z^+	43	313	140	140
Δx wake [mm]	0.936	3.0	1.5	1.5
Runtime $[T_C]$	34	35.7	10.3	9.2
Processors	200	80	280	280
CPU [h]	34,680	49,732	194,483	164,472
$T_C/\Delta t$	1790	1325	1695	1695
CPU [h]/T_C/1M cells	5.61	39.75	86.4	81.36

Adaptive LBM code 16x faster than OpenFOAM with PISO algorithm on static mesh!

Strong scalability test (1:25 train)

- Computation is restarted from disk checkpoint at $t = 0.526408\,\mathrm{s}$ from 96 core run.
- Time for initial re-partitioning removed from benchmark.
- 200 coarse level time steps computed.
- Regridding and re-partitioning every 2nd level-0 step.
- Computation starts with 51.8M cells (l3: 10.2M, l2: 15.3M, l1: 21.5M, l0= 4.8M) vs. 19.66 billion (uniform).
- Portions for parallel communication quite considerable (4 ghost cells still used).

Time per coarse level step



Time in % spent in main operations

Cores	48	96	192	288	384	576	768
Time per step	132.43s	69.79s	37.47s	27.12s	21.91s	17.45s	15.15s
Par. Efficiency	100.0	94.88	88.36	81.40	75.56	63.24	54.63
LBM Update	5.91	5.61	5.38	4.92	4.50	3.73	3.19
Regridding	15.44	12.02	11.38	10.92	10.02	8.94	8.24
Partitioning	4.16	2.43	1.16	1.02	1.04	1.16	1.34
Interpolation	3.76	3.53	3.33	3.05	2.83	2.37	2.06
Sync Boundaries	54.71	59.35	59.73	56.95	54.54	52.01	51.19
Sync Fixup	9.10	10.41	12.25	16.62	20.77	26.17	28.87
Level set	0.78	0.93	1.21	1.37	1.45	1.48	1.47
Interp./Extrap.	3.87	3.81	3.76	3.49	3.26	2.75	2.39
Misc	2.27	1.91	1.79	1.67	1.58	1.38	1.25

Motion solver

Based on the Newton-Euler method solution of dynamics equation of kinetic chains [Tsai, 1999]

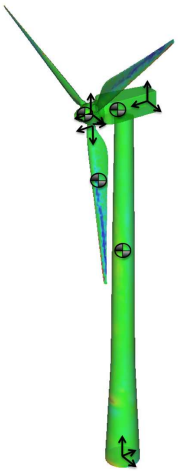
$$\begin{pmatrix} \mathbf{F} \\ \boldsymbol{\tau}_\mathrm{P} \end{pmatrix} = \begin{pmatrix} m\mathbf{1} & -m[\mathbf{c}]^\times \\ m[\mathbf{c}]^\times \mathbf{I}_\mathrm{cm} & -m[\mathbf{c}]^\times[\mathbf{c}]^\times \end{pmatrix} \begin{pmatrix} \mathbf{a}_\mathrm{P} \\ \boldsymbol{\alpha} \end{pmatrix} + \begin{pmatrix} m[\boldsymbol{\omega}]^\times[\boldsymbol{\omega}]^\times\mathbf{c} \\ [\boldsymbol{\omega}]^\times(\mathbf{I}_\mathrm{cm} - m[\mathbf{c}]^\times[\mathbf{c}]^\times)\,\boldsymbol{\omega} \end{pmatrix}.$$

m = mass of the body, 1 = the 4×4 homogeneous identity matrix,
$\mathbf{a}_p$ = acceleration of link frame with origin at $\mathbf{p}$ in the preceding link's frame,
$\mathbf{I}_\mathrm{cm}$ = moment of inertia about the center of mass,
$\boldsymbol{\omega}$ = angular velocity of the body,
$\boldsymbol{\alpha}$ = angular acceleration of the body,
$\mathbf{c}$ is the location of the body's center of mass,
and $[\mathbf{c}]^\times$, $[\boldsymbol{\omega}]^\times$ denote skew-symmetric cross product matrices.

Here, we additionally define the total force and torque acting on a body,
$\mathbf{F} = (\mathbf{F}_{FSI} + \mathbf{F}_{prescribed}) \cdot \mathcal{C}_{xyz}$ and

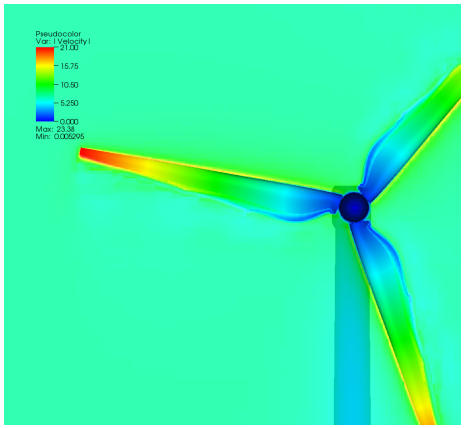
$\tau = (\tau_{FSI} + \tau_{prescribed}) \cdot \mathcal{C}_{\alpha\beta\gamma}$ respectively.

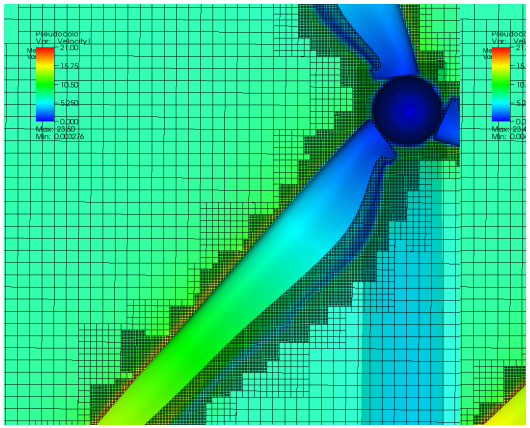
Where $\mathcal{C}_{xyz}$ and $\mathcal{C}_{\alpha\beta\gamma}$ are the translational and rotational constraints, respectively.



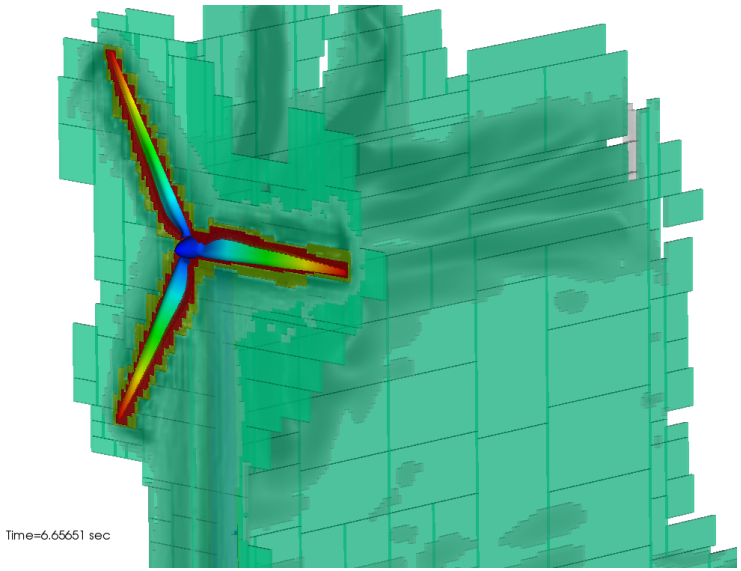
Simulation of a single turbine

- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height ~ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 m/s.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh $400 \times 200 \times 200$ cells with refinement factors 2,2,4. Resolution of rotor and tower $\Delta x = 3.125\,\mathrm{cm}$.
- 141,344 highest level iterations to $t_e = 30\,\mathrm{s}$ computed.





Adaptive refinement

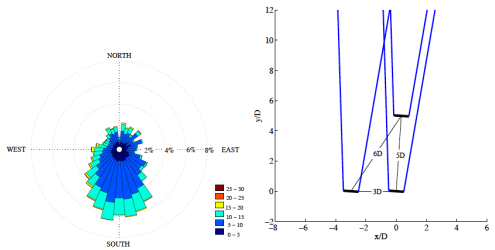


Dynamic evolution of refinement blocks (indicated by color).

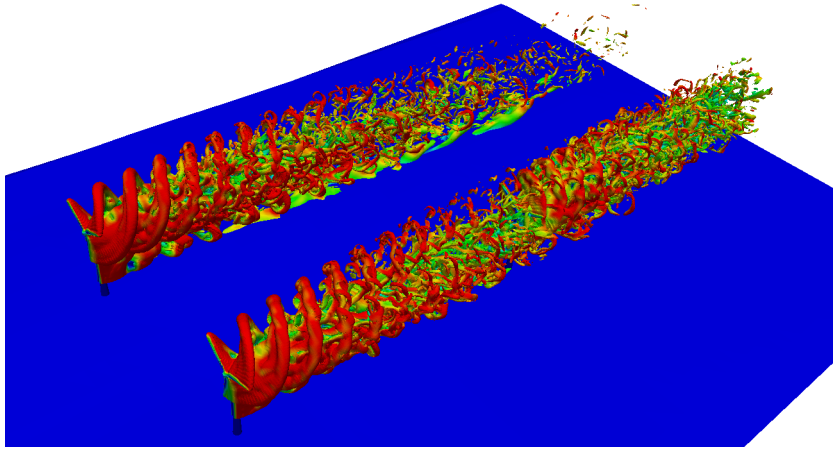
`code/doc/html/capps/motion-amroc_2WindTurbine__Terrain_2src_2FluidProblem_8h_source.html`, `code/doc/html/capps/motion-amroc_2WindTurbine__Terrain_2src_2SolidProblem_8h_source.html`, `code/doc/html/capps/Terrain_2src_2Terrain_8h_source.html`

Simulation of the SWIFT array

- Three Vestas V27 turbines. 225 kW power generation at wind speeds 14 to 25 m/s (then cut-off)
- Prescribed motion of rotor with 33 and 43 rpm. Inflow velocity 8 and 25 m/s
- TSR: 5.84 and 2.43, $\mathrm{Re}_r \approx 919,700$ and 1,208,000
- Simulation domain $448\,\mathrm{m} \times 240\,\mathrm{m} \times 100\,\mathrm{m}$
- Base mesh $448 \times 240 \times 100$ cells with refinement factors 2,2,4. Resolution of rotor and tower $\Delta x = 6.25\,\mathrm{cm}$
- 94,224 highest level iterations to $t_e = 40\,\mathrm{s}$ computed, then statistics are gathered for 10 s [Deiterding and Wood, 2015]



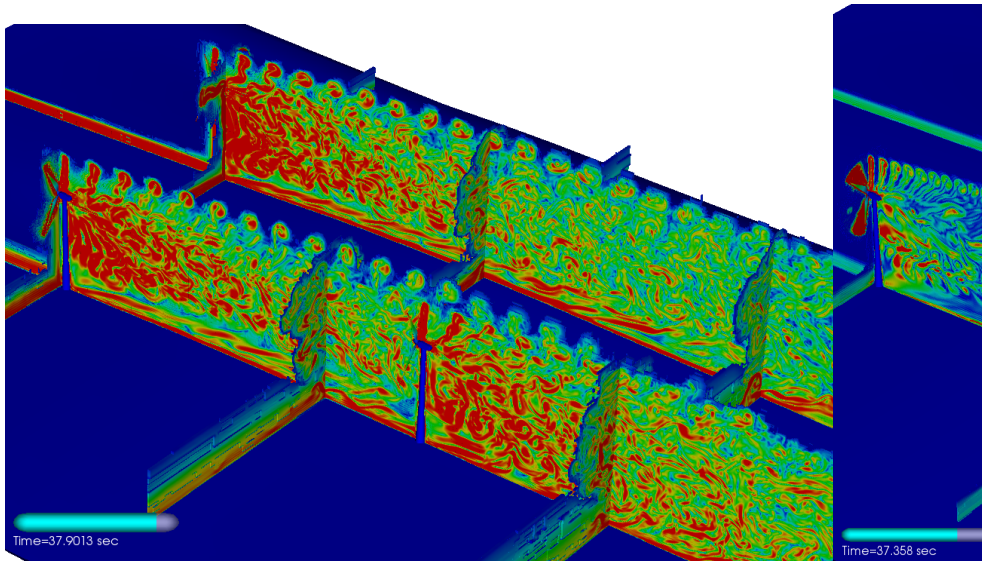
Wake propagation through array – 25 m/s, 43 rpm



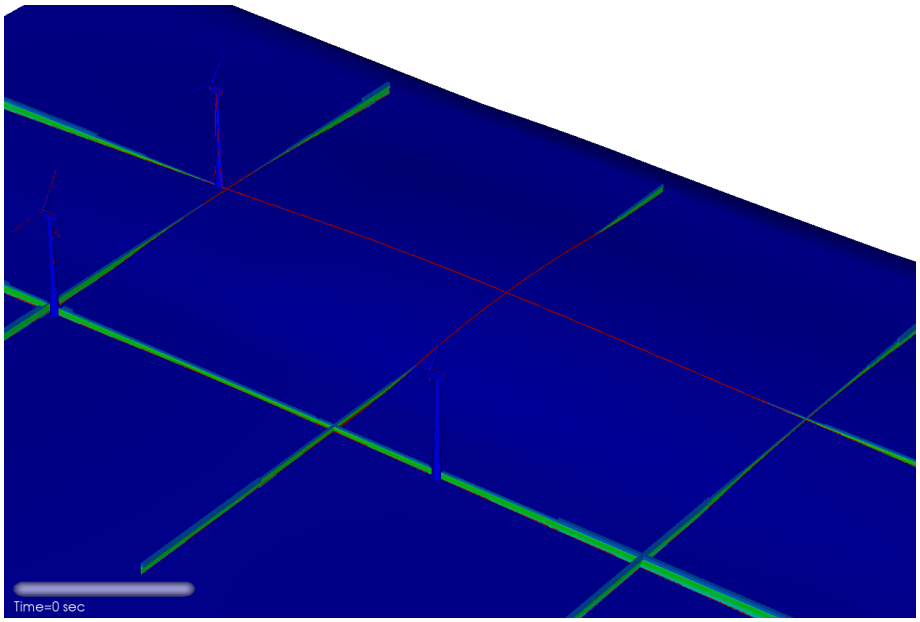
- On 288 cores Intel Xeon-Ivybride 10 s in 38.5 h (11,090 h CPU)
- Only levels 0 and 1 used for iso-surface visualization
- At t_e approximately 140M cells used vs. 44 billion (factor 315)
- Only levels 0 and 1 used for iso-surface visualization

Level	Grids	Cells
0	3,234	10,752,000
1	11,921	21,020,256
2	66,974	102,918,568
3	896	5,116,992

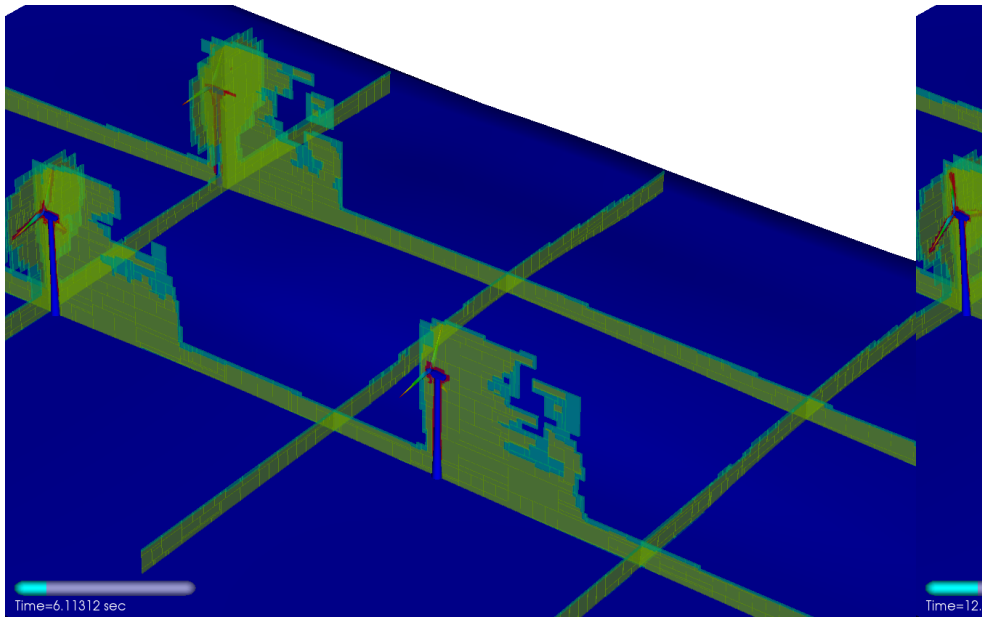
Vorticity generation - $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$$u = 25\,\mathrm{m/s}$, $43\,\mathrm{rpm}$



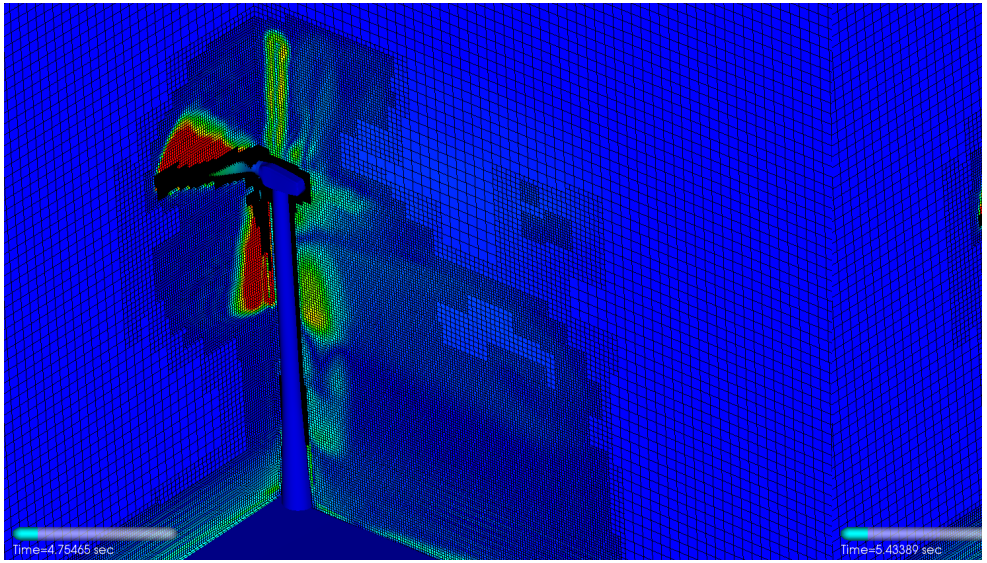
Vorticity development - $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



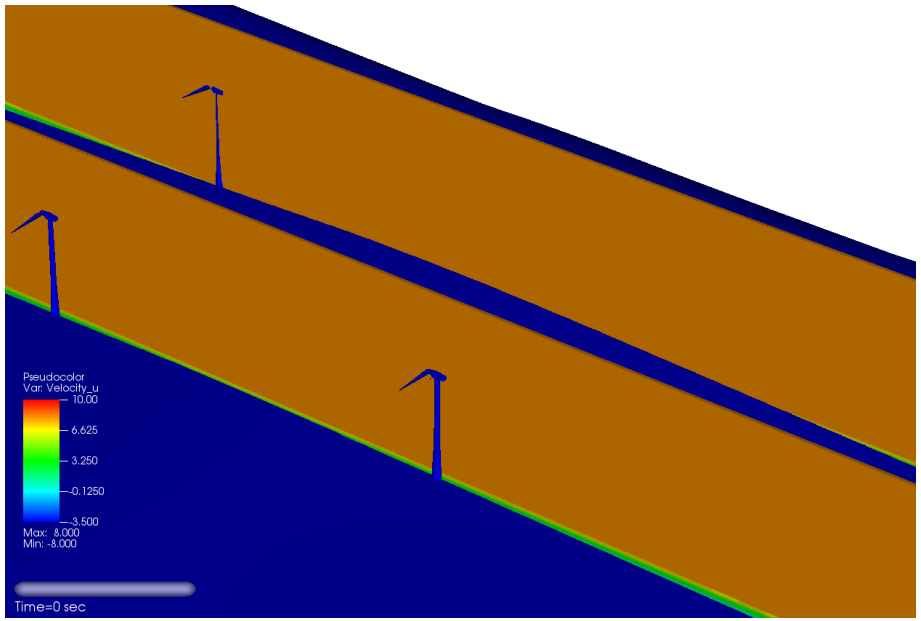
Refinement $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



Wake refinement behind a leading turbine

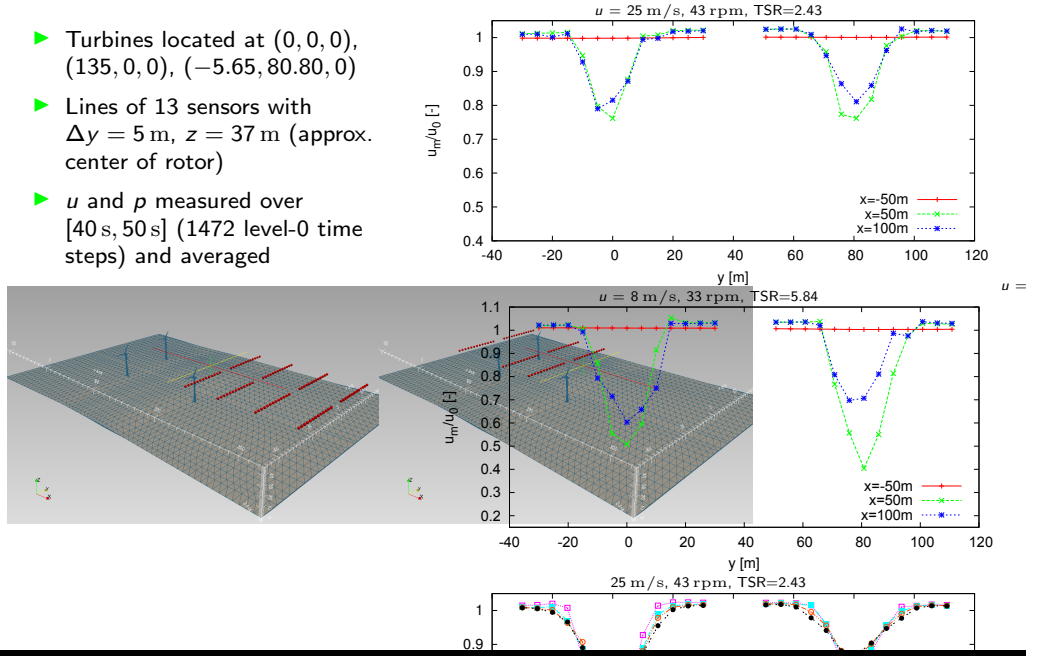


Axial velocity - $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



Mean point values

- Turbines located at $(0, 0, 0)$, $(135, 0, 0)$, $(-5.65, 80.80, 0)$
- Lines of 13 sensors with $\Delta y = 5\,\mathrm{m}$, $z = 37\,\mathrm{m}$ (approx. center of rotor)
- u and p measured over $[40\,\mathrm{s}, 50\,\mathrm{s}]$ (1472 level-0 time steps) and averaged



Outline

Poisson equation

$$\begin{aligned} \Delta q(\mathbf{x}) &= \psi(\mathbf{x}), \quad \mathbf{x} \in \Omega \subset \mathbb{R}^d, \quad q \in \mathrm{C}^2(\Omega), \quad \psi \in \mathrm{C}^0(\Omega) \\ q &= \psi^\Gamma(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega \end{aligned}$$

Discrete Poisson equation in 2D:

$$\frac{Q_{j+1,k} - 2Q_{jk} + Q_{j-1,k}}{\Delta x_1^2} + \frac{Q_{j,k+1} - 2Q_{jk} + Q_{j,k-1}}{\Delta x_2^2} = \psi_{jk}$$

Operator

$$\mathcal{A}(Q_{\Delta x_1, \Delta x_2}) = \begin{bmatrix} & \frac{1}{\Delta x_2^2} & \\ \frac{1}{\Delta x_1^2} & -\left(\frac{2}{\Delta x_1^2} + \frac{2}{\Delta x_2^2}\right) & \frac{1}{\Delta x_2^2} \\ & \frac{1}{\Delta x_2^2} & \end{bmatrix} Q(x_{1,j}, x_{2,k}) = \psi_{jk}$$

$$Q_{jk} = \frac{1}{\sigma}\left[(Q_{j+1,k} + Q_{j-1,k})\Delta x_2^2 + (Q_{j,k+1} + Q_{j,k-1})\Delta x_1^2 - \Delta x_1^2 \Delta x_2^2 \psi_{jk}\right]$$

with $\sigma = \dfrac{2\Delta x_1^2 + 2\Delta x_2^2}{\Delta x_1^2 \Delta x_2^2}$

Iterative methods

Jacobi iteration

$$Q_{jk}^{m+1} = \frac{1}{\sigma}\left[(Q_{j+1,k}^{m} + Q_{j-1,k}^{m})\Delta x_2^2 + (Q_{j,k+1}^{m} + Q_{j,k-1}^{m})\Delta x_1^2 - \Delta x_1^2 \Delta x_2^2 \psi_{jk}\right]$$

Lexicographical Gauss-Seidel iteration (use updated values when they become available)

$$Q_{jk}^{m+1} = \frac{1}{\sigma}\left[(Q_{j+1,k}^{m} + Q_{j-1,k}^{m+1})\Delta x_2^2 + (Q_{j,k+1}^{m} + Q_{j,k-1}^{m+1})\Delta x_1^2 - \Delta x_1^2 \Delta x_2^2 \psi_{jk}\right]$$

Efficient parallelization / patch-wise application not possible!

Checker-board or Red-Black Gauss Seidel iteration

1. $Q_{jk}^{m+1} = \frac{1}{\sigma}\left[(Q_{j+1,k}^{m} + Q_{j-1,k}^{m})\Delta x_2^2 + (Q_{j,k+1}^{m} + Q_{j,k-1}^{m})\Delta x_1^2 - \Delta x_1^2 \Delta x_2^2 \psi_{jk}\right]$ for $j + k \bmod 2 = 0$
2. $Q_{jk}^{m+1} = \frac{1}{\sigma}\left[(Q_{j+1,k}^{m+1} + Q_{j-1,k}^{m+1})\Delta x_2^2 + (Q_{j,k+1}^{m+1} + Q_{j,k-1}^{m+1})\Delta x_1^2 - \Delta x_1^2 \Delta x_2^2 \psi_{jk}\right]$ for $j + k \bmod 2 = 1$

Gauss-Seidel methods require $\sim 1/2$ of iterations than Jacobi method, however, iteration count still proportional to number of unknowns [Hackbusch, 1994]

Smoothing vs. solving

ν iterations with iterative linear solver

$$Q^{m+\nu} = \mathcal{S}(Q^m, \psi, \nu)$$

Defect after m iterations

$$d^m = \psi - \mathcal{A}(Q^m)$$

Defect after $m + \nu$ iterations

$$d^{m+\nu} = \psi - \mathcal{A}(Q^{m+\nu}) = \psi - \mathcal{A}(Q^m + v_\nu^m) = d^m - \mathcal{A}(v_\nu^m)$$

with correction

$$v_\nu^m = \mathcal{S}(\vec{0}, d^m, \nu)$$

Neglecting the sub-iterations in the smoother we write

$$Q^{n+1} = Q^n + v = Q^n + \mathcal{S}(d^n)$$

Observation: Oscillations are damped faster on coarser grid.

Coarse grid correction:

$$Q^{n+1} = Q^n + v = Q^n + \mathcal{P}\mathcal{S}\mathcal{R}(d^n)$$

where $\mathcal{R}$ is suitable restriction operator and $\mathcal{P}$ a suitable prolongation operator

Two-grid correction method

Relaxation on current grid:

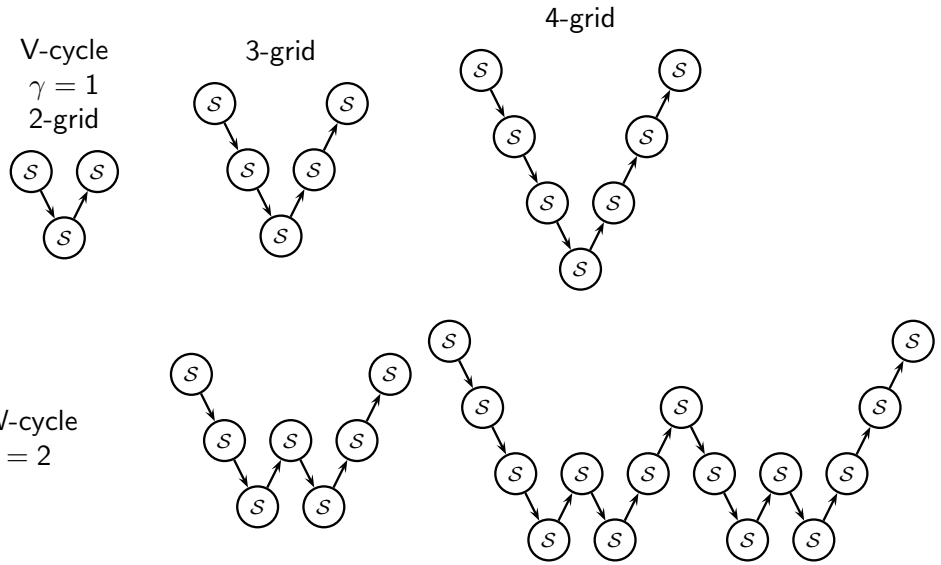
$$\bar{Q} = \mathcal{S}(Q^n, \psi, \nu)$$

$$Q^{n+1} = \bar{Q} + \mathcal{P}\mathcal{S}(\vec{0}, \cdot, \mu)\mathcal{R}(\psi - \mathcal{A}(\bar{Q}))$$

Algorithm:	with smoothing:	with pre- and post-iteration:
$\bar{Q} := \mathcal{S}(Q^n, \psi, \nu)$	$d := \psi - \mathcal{A}(Q)$	$d := \psi - \mathcal{A}(Q)$
$d := \psi - \mathcal{A}(\bar{Q})$	$v := \mathcal{S}(0, d, \nu)$	$v := \mathcal{S}(0, d, \nu_1)$
	$r := d - \mathcal{A}(v)$	$r := d - \mathcal{A}(v)$
$d_c := \mathcal{R}(d)$	$d_c := \mathcal{R}(r)$	$d_c := \mathcal{R}(r)$
$v_c := \mathcal{S}(0, d_c, \mu)$	$v_c := \mathcal{S}(0, d_c, \mu)$	$v_c := \mathcal{S}(0, d_c, \mu)$
$v := \mathcal{P}(v_c)$	$v := v + \mathcal{P}(v_c)$	$v := v + \mathcal{P}(v_c)$
$Q^{n+1} := \bar{Q} + v$	$Q^{n+1} := Q + v$	$d := d - \mathcal{A}(v)$
		$r := \mathcal{S}(0, d, \nu_2)$
		$Q^{n+1} := Q + v + r$

[Hackbusch, 1985]

Multi-level methods and cycles



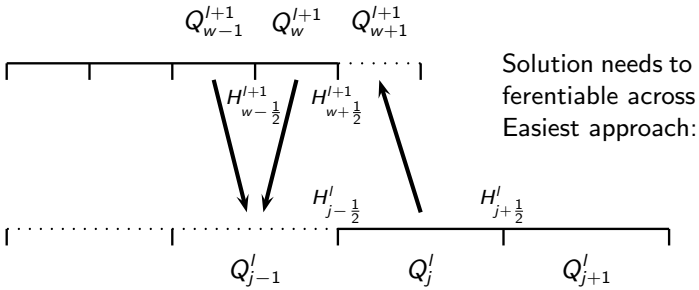
[Hackbusch, 1985] [Wesseling, 1992] ...

Stencil modification at coarse-fine boundaries in 1D

1D Example: Cell j, $\psi - \nabla \cdot \nabla q = 0$

$$d_j^l = \psi_j - \frac{1}{\Delta x_l}\left(\frac{1}{\Delta x_l}(Q_{j+1}^l - Q_j^l) - \frac{1}{\Delta x_l}(Q_j^l - Q_{j-1}^l)\right) = \psi_j - \frac{1}{\Delta x_l}\left(H_{j+\frac{1}{2}}^l - H_{j-\frac{1}{2}}^l\right)$$

H is approximation to *derivative* of Q^l.
Consider 2-level situation with $r_{l+1} = 2$:



Solution needs to be continuously differentiable across interface.
Easiest approach: $H_{w+\frac{1}{2}}^{l+1} \equiv H_{j-\frac{1}{2}}^l$

No specific modification necessary for 1D vertex-based stencils, cf. [Bastian, 1996]

Stencil modification at coarse-fine boundaries in 1D II

Set $H_{w+\frac{1}{2}}^{l+1} = H_{\mathcal{I}}$. Inserting Q gives

$$\frac{Q_{w+1}^{l+1} - Q_w^{l+1}}{\Delta x_{l+1}} = \frac{Q_j^l - Q_w^{l+1}}{\frac{3}{2}\Delta x_{l+1}}$$

from which we readily derive

$$Q_{w+1}^{l+1} = \frac{2}{3}Q_j^l + \frac{1}{3}Q_w^{l+1}$$

for the boundary cell on $l+1$. We use the flux correction procedure to enforce $H_{w+\frac{1}{2}}^{l+1} \equiv H_{j-\frac{1}{2}}^l$ and thereby $H_{j-\frac{1}{2}}^l \equiv H_{\mathcal{I}}$.

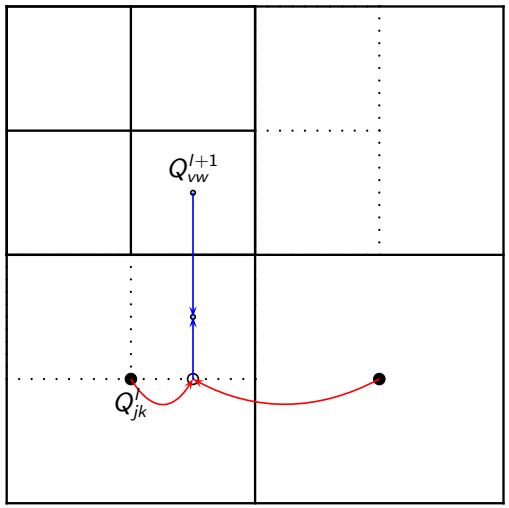
Correction pass [Martin, 1998]

1. $\delta H_{j-\frac{1}{2}}^{l+1} := -H_{j-\frac{1}{2}}^l$
2. $\delta H_{j-\frac{1}{2}}^{l+1} := \delta H_{j-\frac{1}{2}}^{l+1} + H_{w+\frac{1}{2}}^{l+1} = -H_{j-\frac{1}{2}}^l + (Q_j^l - Q_w^{l+1})/\frac{3}{2}\Delta x_{l+1}$
3. $\check{d}_j^l := d_j^l + \frac{1}{\Delta x_l}\delta H_{j-\frac{1}{2}}^{l+1}$

yields

$$\check{d}_j^l = \psi_j - \frac{1}{\Delta x_l}\left(\frac{1}{\Delta x_l}(Q_{j+1}^l - Q_j^l) - \frac{2}{3\Delta x_{l+1}}(Q_j^l - Q_w^{l+1})\right)$$

Stencil modification at coarse-fine boundaries: 2D



$$Q_{v,w-1}^{l+1} = \frac{1}{3}Q_{vw}^{l+1} + \frac{2}{3}\left(\frac{3}{4}Q_{jk}^l + \frac{1}{4}Q_{j+1,k}^l\right)$$

In general:

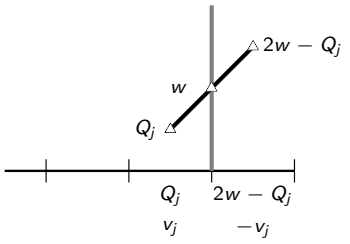
$$Q_{v,w-1}^{l+1} = \left(1 - \frac{2}{r_{l+1}+1}\right)Q_{vw}^{l+1} + \frac{2}{r_{l+1}+1}\left((1-f)Q_{jk}^l + fQ_{j+1,k}^l\right)$$

with

$$f = \frac{x_{1,l+1}^v - x_{1,l}^j}{\Delta x_{1,l}}$$

Components of an SAMR multigrid method

- Stencil operators
  - Application of defect $d^l = \psi^l - \mathcal{A}(Q^l)$ on each grid $G_{l,m}$ of level l
  - Computation of correction $v^l = \mathcal{S}(0, d^l, \nu)$ on each grid of level l
- Boundary (ghost cell) operators
  - Synchronization of Q^l and v^l on $\tilde{S}_l^1$
  - Specification of Dirichlet boundary conditions for a finite volume discretization for $Q^l \equiv w$ and $v^l \equiv w$ on $\tilde{P}_l^1$
  - Specification of $v^l \equiv 0$ on $\tilde{I}_l^1$
  - Specification of $Q_l = \frac{(r_l-1)Q^{l+1}+2Q^k}{r_l+1}$ on $\tilde{I}_l^1$



- Coarse-fine boundary flux accumulation and application of δH^{l+1} on defect d^l
- Standard prolongation and restriction on grids between adjacent levels
- Adaptation criteria
  - E.g., standard restriction to project solution on 2x coarsended grid, then use local error estimation
- Looping instead of time steps and check of convergence

Additive geometric multigrid algorithm

`AdvanceLevelMG(`l`) - Correction Scheme`

Set ghost cells of Q^l	
Calculate defect d^l from Q^l, ψ^l	$d^l := \psi^l - \mathcal{A}(Q^l)$
If $(l < l_{max})$	
Calculate updated defect r^{l+1} from v^{l+1}, d^{l+1}	$r^{l+1} := d^{l+1} - \mathcal{A}(v^{l+1})$
Restrict d^{l+1} onto d^l	$d^l := \mathcal{R}_l^{l+1}(r^{l+1})$
Do ν_1 smoothing steps to get correction v^l	$v^l := \mathcal{S}(0, d^l, \nu_1)$
If $(l > l_{min})$	
Do $\gamma > 1$ times	
AdvanceLevelMG$(l-1)$	
Set ghost cells of v^{l-1}	
Prolongate and add v^{l-1} to v^l	$v^l := v^l + \mathcal{P}_l^{l-1}(v^{l-1})$
If $(\nu_2 > 0)$	
Set ghost cells of v^l	
Update defect d^l according to v^l	$d^l := d^l - \mathcal{A}(v^l)$
Do ν_2 post-smoothing steps to get r^l	$r^l := \mathcal{S}(v^l, d^l, \nu_2)$
Add addional correction r^l to v^l	$v^l := v^l + r^l$
Add correction v^l to Q^l	$Q^l := Q^l + v^l$

Additive Geometric Multiplicative Multigrid Algorithm

`Start - Start iteration on level` l_{max}

For $l = l_{max}$ Downto $l_{min} + 1$ Do
Restrict Q^l onto Q^{l-1}
Regrid(0)
AdvanceLevelMG(l_{max})

See also: [Trottenberg et al., 2001], [Canu and Ritzdorf, 1994]
Vertex-based: [Brandt, 1977], [Briggs et al., 2001]

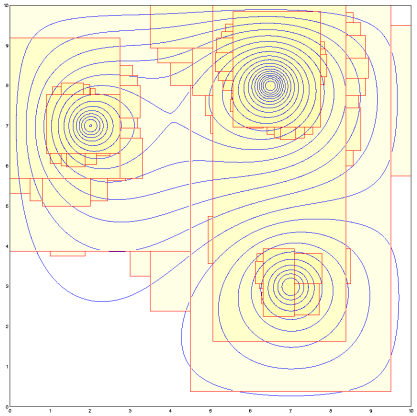
Example

On $\Omega = [0, 10] \times [0, 10]$ use hat function

$$\psi = \begin{cases} -A_n \cos\left(\frac{\pi r}{2R_n}\right), & r < R_n \\ 0 & \text{elsewhere} \end{cases}$$

with $r = \sqrt{(x_1 - X_n)^2 + (x_2 - Y_n)^2}$ to define three sources with

n	A_n	R_n	X_n	Y_n
1	0.3	0.3	6.5	8.0
2	0.2	0.3	2.0	7.0
3	-0.1	0.4	7.0	3.0



	128×128	1024×1024	1024×1024
l_{max}	3	0	0
l_{min}	-4	-7	-4
ν_1	5	5	5
ν_2	5	5	5
V-Cycles	15	16	341
Time [sec]	9.4	27.7	563

Stop at $\|d^l\|_{max} < 10^{-7}$ for $l \geq 0$, $\gamma = 1$, $r_l = 2$

Some comments on parabolic problems

- Consequences of time step refinement
- Level-wise elliptic solves vs. global solve
- If time step refinement is used an elliptic flux correction is unavoidable.
- The correction is explained in Bell, J. (2004). Block-structured adaptive mesh refinement. Lecture 2. Available at https://ccse.lbl.gov/people/jbb/shortcourse/lecture2.pdf.

References I

[Bastian, 1996] Bastian, P. (1996). *Parallele adaptive Mehrgitterverfahren*. Teubner Skripten zur Numerik. B. G. Teubner, Stuttgart.

[Brandt, 1977] Brandt, A. (1977). Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computations*, 31(183):333–390.

[Briggs et al., 2001] Briggs, W. L., Henson, V. E., and McCormick, S. F. (2001). *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, 2nd edition.

[Canu and Ritzdorf, 1994] Canu, J. and Ritzdorf, H. (1994). Adaptive, block-structured multigrid on local memory machines. In Hackbuch, W. and Wittum, G., editors, *Adaptive Methods-Algorithms, Theory and Applications*, pages 84–98, Braunschweig/Wiesbaden. Proceedings of the Ninth GAMM-Seminar, Vieweg & Sohn.

[Chen et al., 2006] Chen, H., Filippova, O., Hoch, J., Molvig, K., Shock, R., Teixeira, C., and Zhang, R. (2006). Grid refinement in lattice Boltzmann methods based on volumetric formulation. *Physica A*, 362:158–167.

References II

[Deiterding and Wood, 2015] Deiterding, R. and Wood, S. L. (2015). An adaptive lattice boltzmann method for predicting wake fields behind wind turbines. In Breitsamer, C. e. a., editor, *Proc. 19th DGLR-Fachsymposium der STAB, Munich, 2014*, Notes on Numerical Fluid Mechanics and Multidisciplinary Design. Springer. in press.

[Hackbusch, 1985] Hackbusch, W. (1985). *Multi-Grid Methods and Applications*. Springer Verlag, Berlin, Heidelberg.

[Hackbusch, 1994] Hackbusch, W. (1994). *Iterative solution of large sparse systems of equations*. Springer Verlag, New York.

[Hähnel, 2004] Hähnel, D. (2004). *Molekulare Gasdynamik*. Springer.

[Henderson, 1995] Henderson, R. D. (1995). Details of the drag curve near the onset of vortex shedding. *Phys. Fluids*, 7:2102–2104.

[Hou et al., 1996] Hou, S., Sterling, J., Chen, S., and Doolen, G. D. (1996). A lattice Boltzmann subgrid model for high Reynolds number flows. In Lawniczak, A. T. and Kapral, R., editors, *Pattern formation and lattice gas automata*, volume 6, pages 151–166. Fields Inst Comm.

References III

[Martin, 1998] Martin, D. F. (1998). *A cell-centered adaptive projection method for the incompressible Euler equations*. PhD thesis, University of California at Berkeley.

[Schlaffer, 2013] Schlaffer, M. B. (2013). *Non-reflecting boundary conditions for the lattice Boltzmann method*. PhD thesis, Technical University Munich.

[Trottenberg et al., 2001] Trottenberg, U., Oosterlee, C., and Schüller, A. (2001). *Multigrid*. Academic Press, San Antonio.

[Tsai, 1999] Tsai, L. (1999). *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*. Wiley.

[Wesseling, 1992] Wesseling, P. (1992). *An introduction to multigrid methods*. Wiley, Chichester.

[Yu, 2004] Yu, H. (2004). *Lattice Boltzmann equation simulations of turbulence, mixing, and combustion*. PhD thesis, Texas A&M University.