Adaptive lattice Boltzmann methods in AMROC (Adaptive Mesh Refinement in Object-oriented C++)

Ralf Deiterding

Aerodynamics and Flight Mechanics Research Group
Department of Aeronautics & Astronautics
University of Southampton
Boldrewood Campus, Southampton SO16 7QF, UK
E-mail: r.deiterding@soton.ac.uk

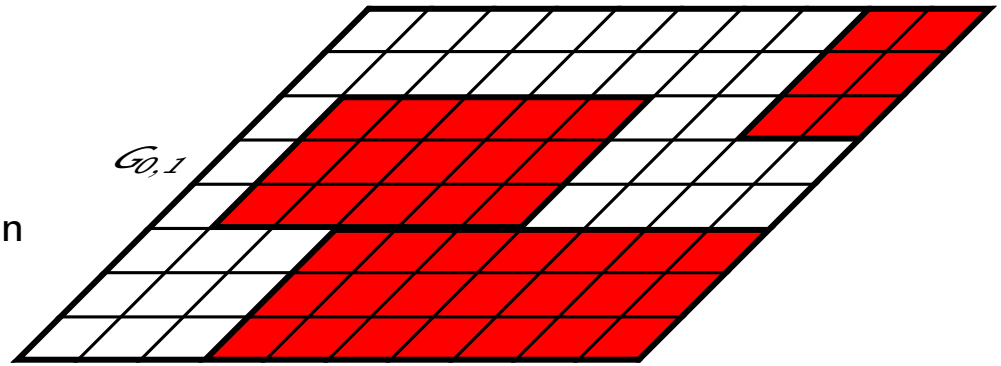
February 25, 2021

Outline

Block-structured adaptive mesh refinement (SAMR)

Originally for conservation laws $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

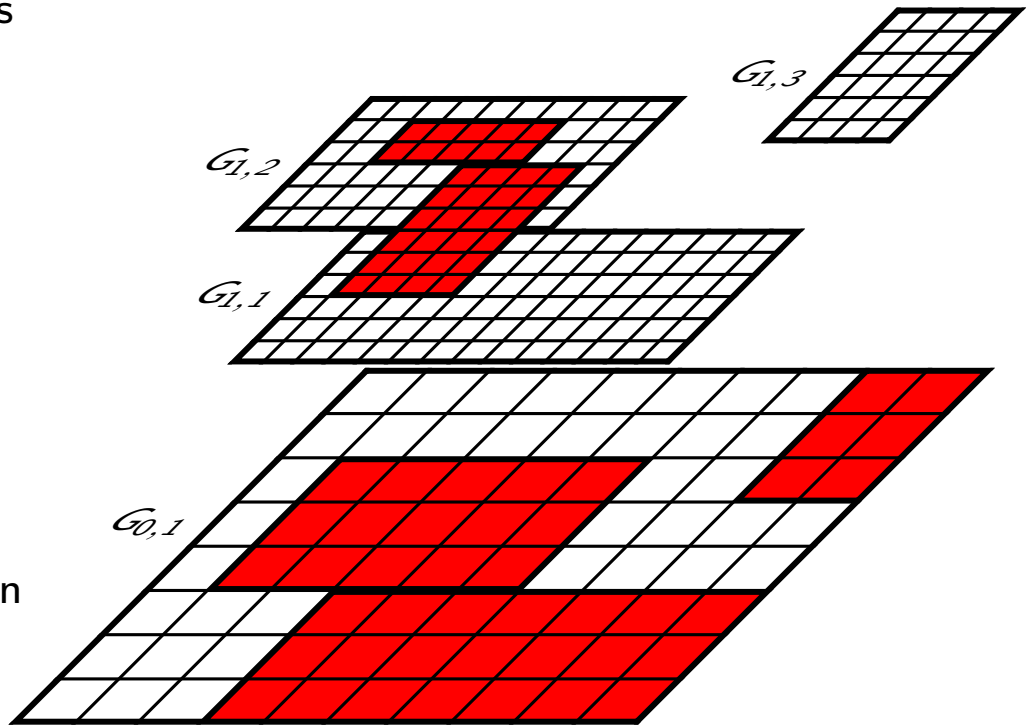
- Refined blocks overlay coarser ones
- Refinement in space *and time* by factor r_l [Berger and Colella, 1988]
- Block (aka patch) based data structures
- \+ Numerical scheme only for single patch necessary
- \+ Efficient cache-reuse / vectorization possible
- \- Cluster-algorithm necessary
- Implicit geometry representation with signed distance function(s)
- Level-set-type techniques for embedded/immersed boundary condition construction
- Papers: [Deiterding, 2011a, Deiterding et al., 2009b, Deiterding et al., 2007]



Block-structured adaptive mesh refinement (SAMR)

Originally for conservation laws $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

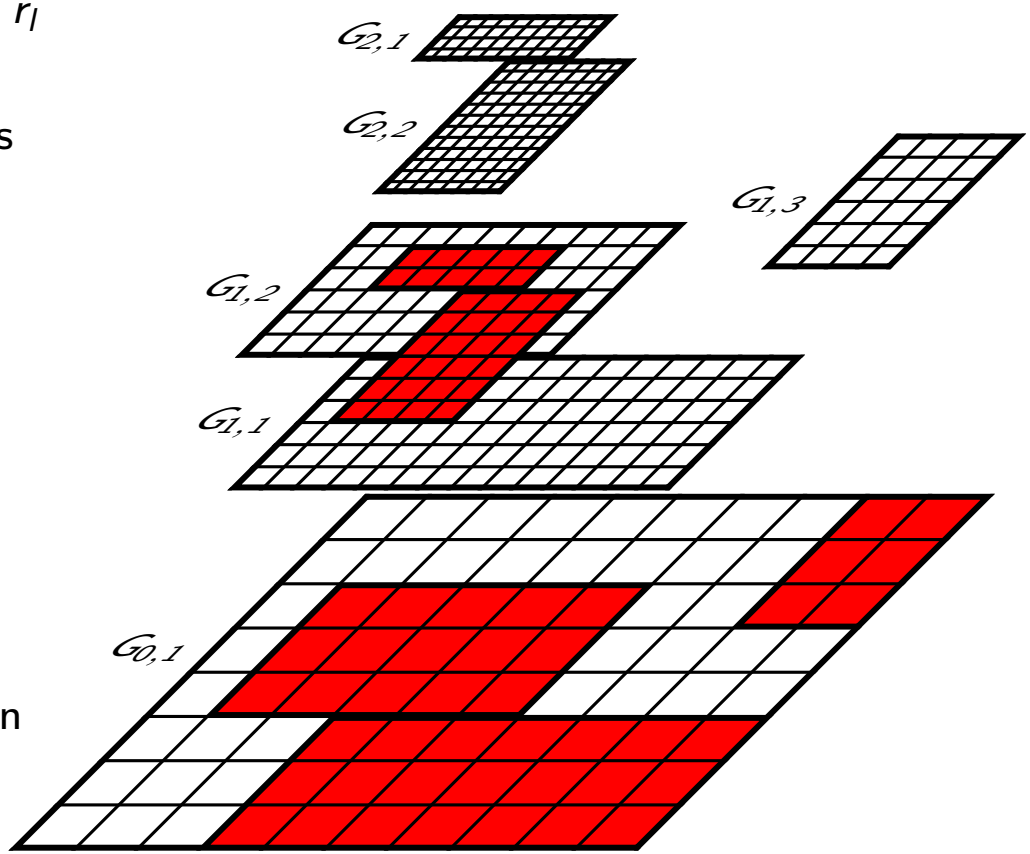
- Refined blocks overlay coarser ones
- Refinement in space *and time* by factor r_l [Berger and Colella, 1988]
- Block (aka patch) based data structures
- \+ Numerical scheme only for single patch necessary
- \+ Efficient cache-reuse / vectorization possible
- \- Cluster-algorithm necessary
- Implicit geometry representation with signed distance function(s)
- Level-set-type techniques for embedded/immersed boundary condition construction
- Papers: [Deiterding, 2011a, Deiterding et al., 2009b, Deiterding et al., 2007]



Block-structured adaptive mesh refinement (SAMR)

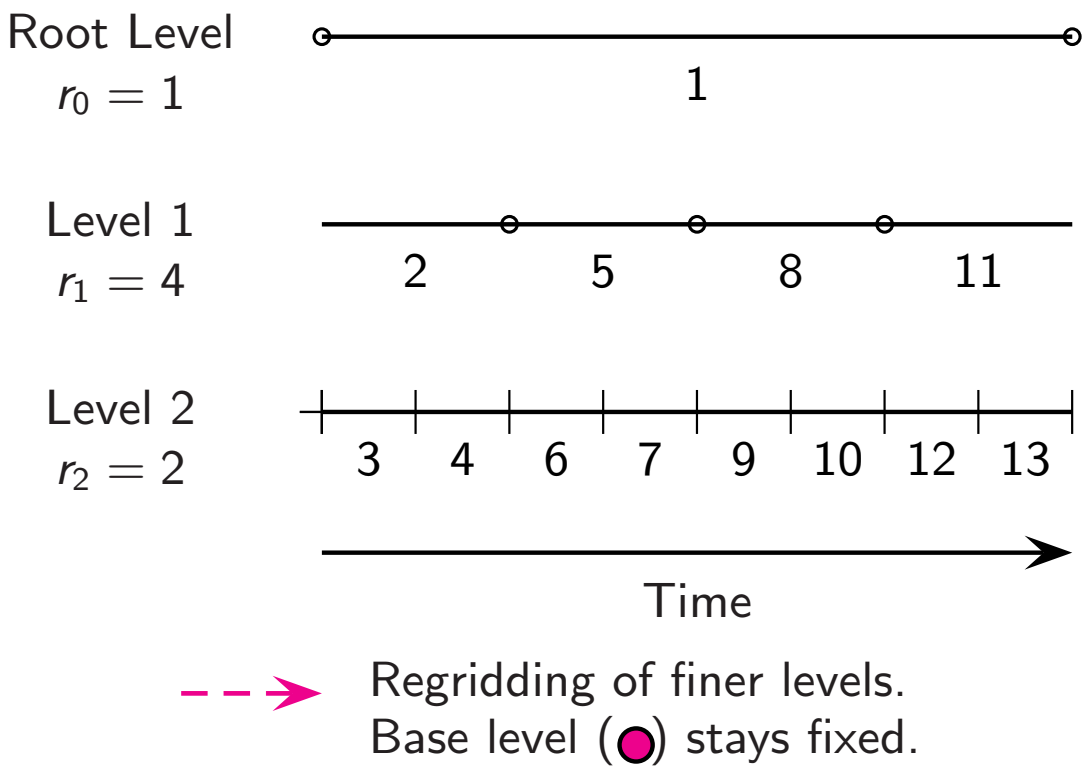
Originally for conservation laws $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

- Refined blocks overlay coarser ones
- Refinement in space *and time* by factor r_l [Berger and Colella, 1988]
- Block (aka patch) based data structures
- \+ Numerical scheme only for single patch necessary
- \+ Efficient cache-reuse / vectorization possible
- \- Cluster-algorithm necessary
- Implicit geometry representation with signed distance function(s)
- Level-set-type techniques for embedded/immersed boundary condition construction
- Papers: [Deiterding, 2011a, Deiterding et al., 2009b, Deiterding et al., 2007]



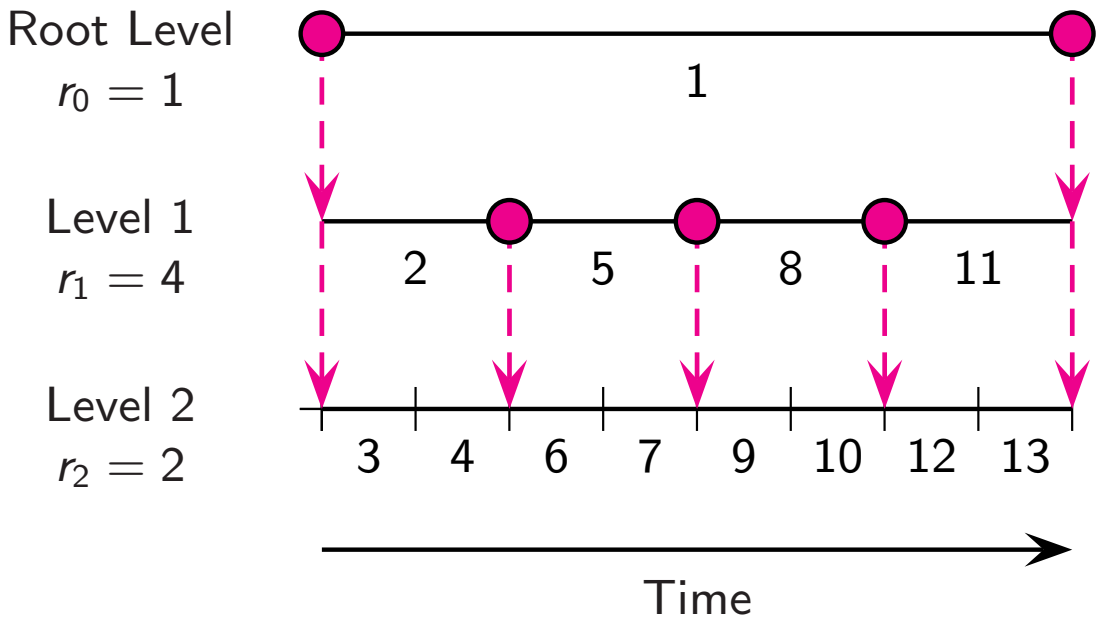
Recursive integration order

- Space-time interpolation of coarse data to set $I_l^s, l > 0$



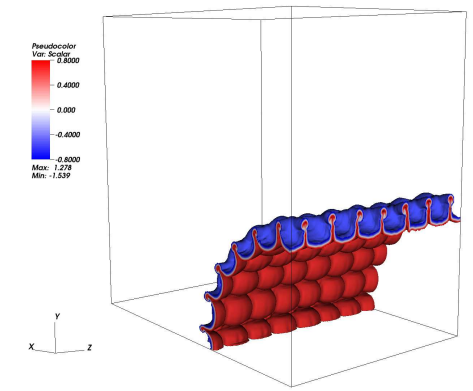
Recursive integration order

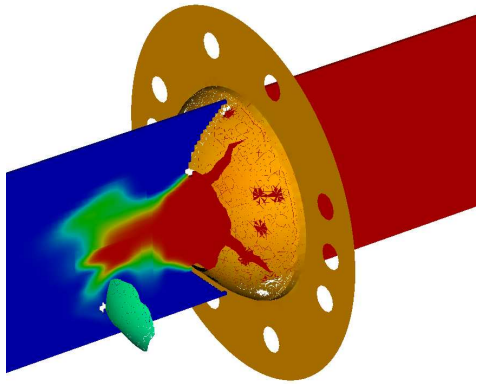
- Space-time interpolation of coarse data to set $I_l^s, l > 0$
- Regridding:
  - Creation of new grids, copy existing cells on level $l > 0$
  - Spatial interpolation to initialize new cells on level $l > 0$



AMROC framework and most important patch solvers

- Implements described algorithms and facilitates easy exchange of the block-based numerical scheme
- Shock-induced combustion with detailed chemistry: [Deiterding, 2003, Deiterding and Bader, 2005, Deiterding, 2011b, Cai et al., 2016, Cai et al., 2018]
- Hybrid WENO methods for LES and DNS: [Pantano et al., 2007, Lombardini and Deiterding, 2010, Ziegler et al., 2011, Cerminara et al., 2018]
- FSI deformation from water hammer: [Cirak et al., 2007, Deiterding et al., 2009a, Perotti et al., 2013, Wan et al., 2017]
- Level-set method for Eulerian solid mechanics: [Barton et al., 2013]
- Ideal magneto-hydrodynamics: [Gomes et al., 2015, Souza Lopes et al., 2018]

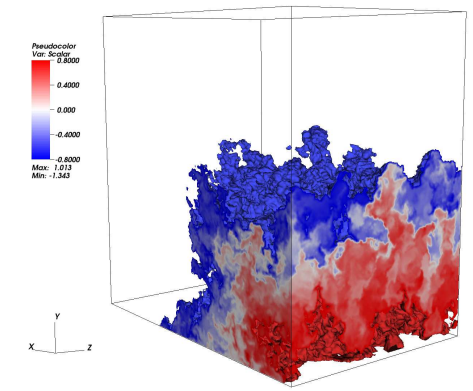


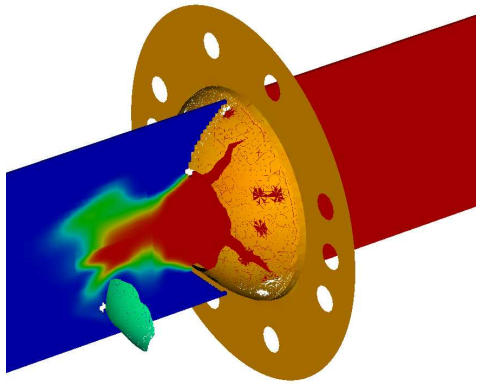


- $\sim 500,000$ LOC in C++, C, Fortran-77, Fortran-90
- V2.0 plus FSI coupling routines as open source at http://www.vtf.website
- Used here V3.0 with significantly enhanced parallelization (V2.1 not released)

AMROC framework and most important patch solvers

- Implements described algorithms and facilitates easy exchange of the block-based numerical scheme
- Shock-induced combustion with detailed chemistry: [Deiterding, 2003, Deiterding and Bader, 2005, Deiterding, 2011b, Cai et al., 2016, Cai et al., 2018]
- Hybrid WENO methods for LES and DNS: [Pantano et al., 2007, Lombardini and Deiterding, 2010, Ziegler et al., 2011, Cerminara et al., 2018]
- FSI deformation from water hammer: [Cirak et al., 2007, Deiterding et al., 2009a, Perotti et al., 2013, Wan et al., 2017]
- Level-set method for Eulerian solid mechanics: [Barton et al., 2013]
- Ideal magneto-hydrodynamics: [Gomes et al., 2015, Souza Lopes et al., 2018]

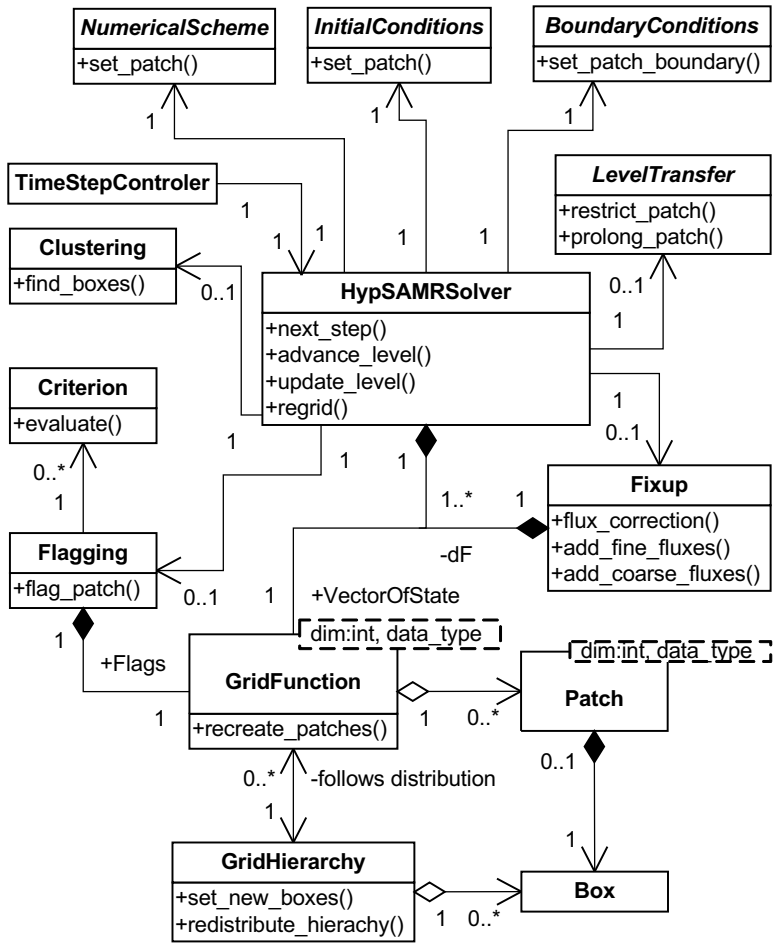




- $\sim 500,000$ LOC in C++, C, Fortran-77, Fortran-90
- V2.0 plus FSI coupling routines as open source at http://www.vtf.website
- Used here V3.0 with significantly enhanced parallelization (V2.1 not released)

UML design of AMROC

- Classical framework approach with generic main program(s) in C++
- $\sim 50,000$ LOC for C++ SAMR kernel, but very complex code
- Fortran patch solvers require only linking to F77 functions for initial and boundary conditions, source terms
- Applications of recent C++-only patch solvers (LBM, MHD) are customized in Problem.h
- Predefined, scheme-specific classes provided for standard simulations
- autoconf / automake environment with support for typical parallel high-performance system
- Expert usage (algorithm modification, advanced output, etc.) also in Problem.h by derivation from base classes and redefining virtual interface functions



Lattice Boltzmann method

Lattice Boltzmann equation $\partial_t f + \mathbf{u} \cdot \nabla f = \omega(f^{eq} - f)$ is solved with the steps

1.) Transport step solves $\partial_t f_\alpha + \mathbf{e}_\alpha \cdot \nabla f_\alpha = 0$
Operator: $\mathcal{T}$: $\tilde{f}_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}, t)$

2.) Collision step solves $\partial_t f_\alpha = \omega(f_\alpha^{eq} - f_\alpha)$
Operator $\mathcal{C}$: $f_\alpha(\cdot, t + \Delta t) = \tilde{f}_\alpha(\cdot, t + \Delta t) + \omega_L \Delta t \left(\tilde{f}_\alpha^{eq}(\cdot, t + \Delta t) - \tilde{f}_\alpha(\cdot, t + \Delta t) \right)$

State vector is a template parameter. Any LBM scheme on a block fits easily into the framework.

Lattice Boltzmann method

Lattice Boltzmann equation $\partial_t f + \mathbf{u} \cdot \nabla f = \omega(f^{eq} - f)$ is solved with the steps

1.) Transport step solves $\partial_t f_\alpha + \mathbf{e}_\alpha \cdot \nabla f_\alpha = 0$
Operator: $\mathcal{T}$: $\tilde{f}_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}, t)$

2.) Collision step solves $\partial_t f_\alpha = \omega(f_\alpha^{eq} - f_\alpha)$
Operator $\mathcal{C}$: $f_\alpha(\cdot, t + \Delta t) = \tilde{f}_\alpha(\cdot, t + \Delta t) + \omega_L \Delta t \left(\tilde{f}_\alpha^{eq}(\cdot, t + \Delta t) - \tilde{f}_\alpha(\cdot, t + \Delta t) \right)$

State vector is a template parameter. Any LBM scheme on a block fits easily into the framework.
Currently available implementations:

- SRT-D2Q9: Isothermal, SRT-D2Q9-D2Q4 with temp. field, SRT-D2Q9-D2Q9(-D2Q4) scalar transport with temp. field
- SRT-D3Q19: Isothermal, SRT-D3Q19-D3Q6 with temp. field, SRT-D3Q19-D3Q19(-D3Q6) scalar transport with temp. field
- SRT-RR-D2Q9, SRT-RR-D3Q27: Regularized recursive for isothermal
- Experimental: D2Q9 for shallow water, FV-D2Q9 on mapped meshes

Lattice Boltzmann method

Lattice Boltzmann equation $\partial_t f + \mathbf{u} \cdot \nabla f = \omega(f^{eq} - f)$ is solved with the steps

1.) Transport step solves $\partial_t f_\alpha + \mathbf{e}_\alpha \cdot \nabla f_\alpha = 0$
Operator: $\mathcal{T}$: $\tilde{f}_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}, t)$

2.) Collision step solves $\partial_t f_\alpha = \omega(f_\alpha^{eq} - f_\alpha)$
Operator $\mathcal{C}$: $f_\alpha(\cdot, t + \Delta t) = \tilde{f}_\alpha(\cdot, t + \Delta t) + \omega_L \Delta t \left(\tilde{f}_\alpha^{eq}(\cdot, t + \Delta t) - \tilde{f}_\alpha(\cdot, t + \Delta t) \right)$

State vector is a template parameter. Any LBM scheme on a block fits easily into the framework.
Currently available implementations:

- SRT-D2Q9: Isothermal, SRT-D2Q9-D2Q4 with temp. field, SRT-D2Q9-D2Q9(-D2Q4) scalar transport with temp. field
- SRT-D3Q19: Isothermal, SRT-D3Q19-D3Q6 with temp. field, SRT-D3Q19-D3Q19(-D3Q6) scalar transport with temp. field
- SRT-RR-D2Q9, SRT-RR-D3Q27: Regularized recursive for isothermal
- Experimental: D2Q9 for shallow water, FV-D2Q9 on mapped meshes

Currently available LES models:

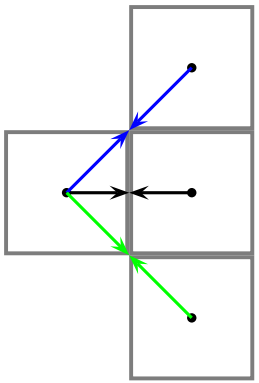
- Constant Smagorinsky: SRT-D2Q9, SRT-D3Q19, SRT-RR-D2Q9, SRT-RR-D3Q27 (in development for temp. field and passive scalar)
- WALE: SRT-D3Q19, SRT-RR-D3Q27
- Dynamic Smagorinsky: SRT-D3Q19

Initial and boundary conditions

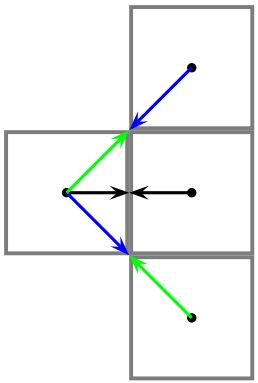
- Initial conditions are constructed as $f_\alpha^{eq}(\rho(t=0), \mathbf{u}(t=0))$

Boundary conditions on cell-centered data – applied before streaming step

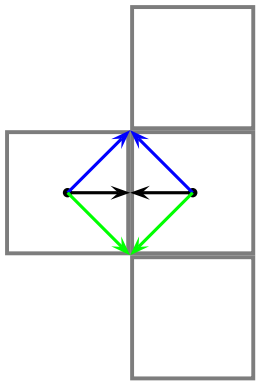
No-slip



Slip



Symmetry



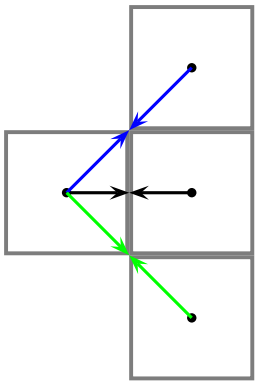
- Simple and characteristic outlet boundary conditions
- Inlet and pressure boundary conditions use f_α^{eq} [Guo et al., 2002]

Initial and boundary conditions

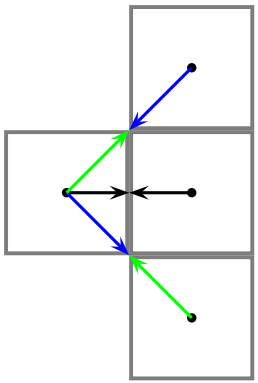
- Initial conditions are constructed as $f_\alpha^{eq}(\rho(t=0), \mathbf{u}(t=0))$

Boundary conditions on cell-centered data – applied before streaming step

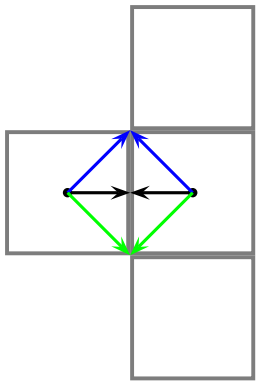
No-slip



Slip



Symmetry



- Simple and characteristic outlet boundary conditions
- Inlet and pressure boundary conditions use f_α^{eq} [Guo et al., 2002]

Complex geometry:

- Use level set method to construct macro-values in embedded boundary cells by interpolation / extrapolation.
- Construct macro-velocity in ghost cells for no-slip BC as $\mathbf{u}' = 2\mathbf{w} - \mathbf{u}$
- Then use $f_\alpha^{eq}(\rho', \mathbf{u}')$ or interpolated bounce-back [Bouzidi et al., 2001] to construct distributions in embedded ghost cells
- Wall function boundary conditions [Malaspinas and Sagaut, 2014] currently experimental

Dimensional vs. non-dimensional units

AMROC assumes dimensional units (normal for finite volume solvers).
LBM on patch is implemented on the unit lattice with $\Delta\tilde{x} = \Delta\tilde{t} = 1$

$$\frac{\Delta x}{l_0} = 1, \quad \frac{\Delta t}{t_0} = 1 \longrightarrow c = 1$$

Rescaling takes place in IO functions of the respective LBM patch solver class.
Velocity normalization factor: $u_0 = \frac{l_0}{t_0}$, density ρ_0

$$\mathrm{Re} = \frac{uL}{\nu} = \frac{u/u_0 \cdot l/l_0}{\nu/(u_0 l_0)} = \frac{\tilde{u}\tilde{l}}{\tilde{\nu}}$$

AMROC-specific trick for scheme acceleration: Use $\bar{u} = Su$ and $\bar{\nu} = S\nu$ which yields

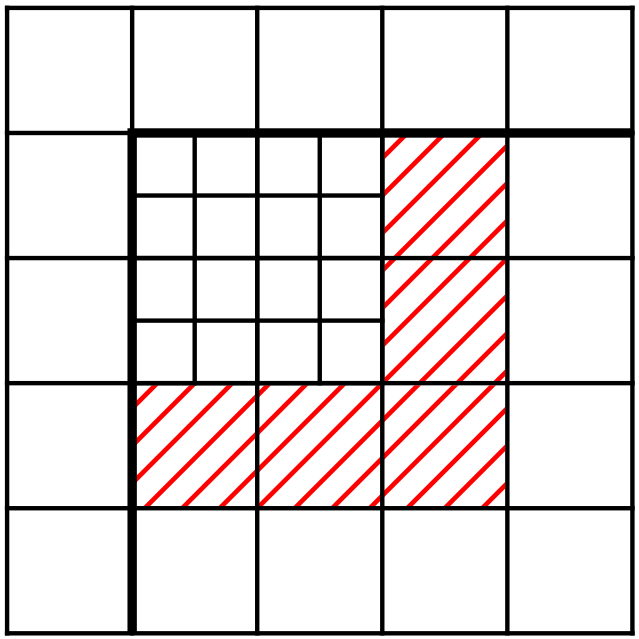
$$\bar{\omega}_L = \frac{c_s^2}{S\nu + \Delta t/S\, c_s^2/2}$$

For instance, the physical hydrodynamic pressure is then obtained for a caloric gas as

$$p = (\tilde{\rho} - 1)\tilde{c}_s^2 \frac{u_0^2}{S^2}\rho_0 + \frac{c_s^2 \rho_0}{\gamma}$$

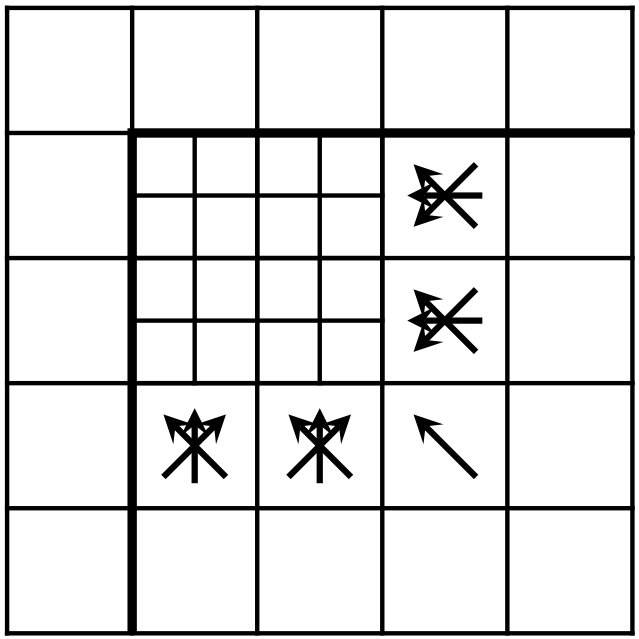
Adaptive LBM

1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$



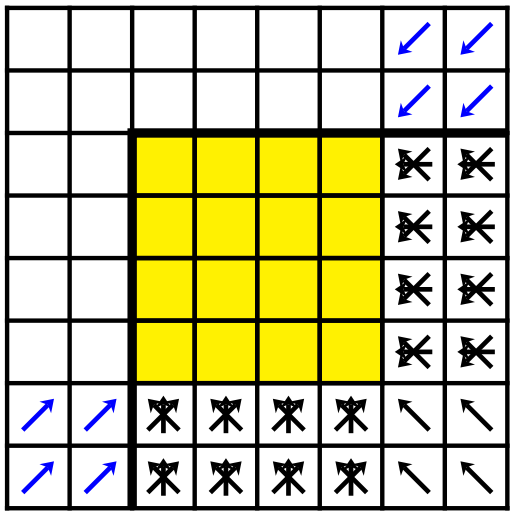
Adaptive LBM

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.



Adaptive LBM

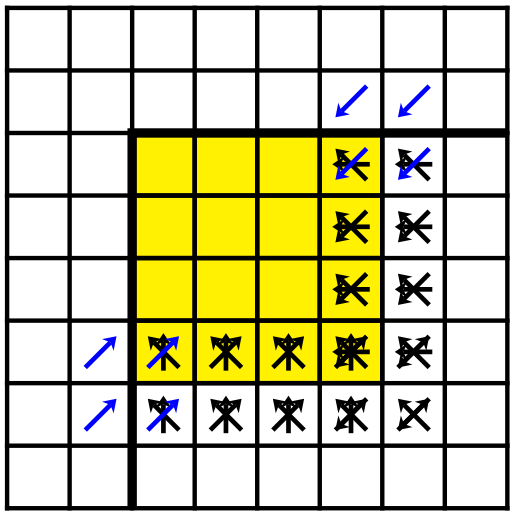
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.



$f_{\alpha,in}^{f,n}$

Adaptive LBM

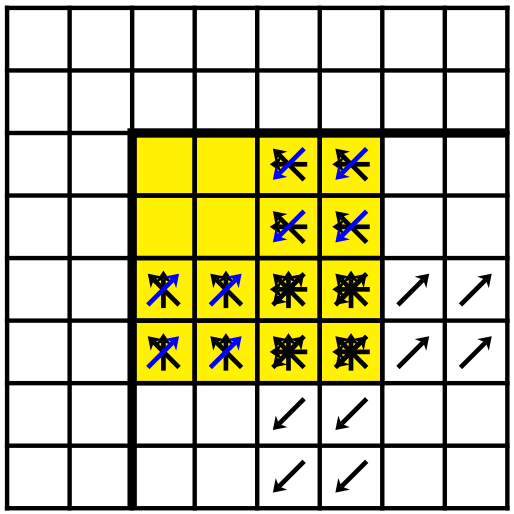
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



$\tilde{f}_{\alpha,in}^{f,n}$

Adaptive LBM

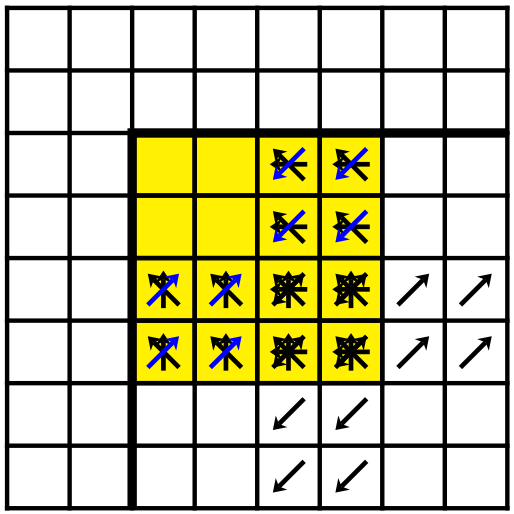
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



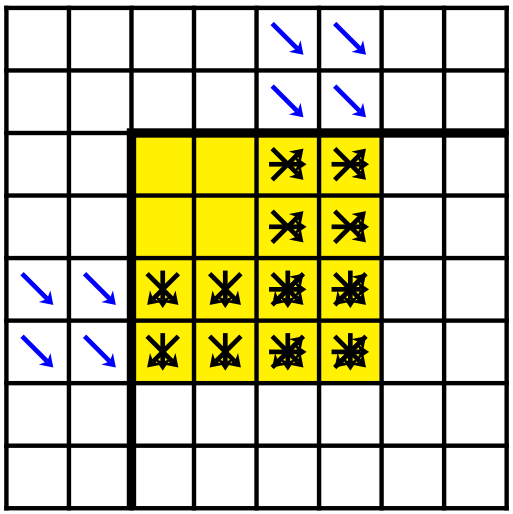
$\tilde{f}_{\alpha,in}^{f,n+1/2}$

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.



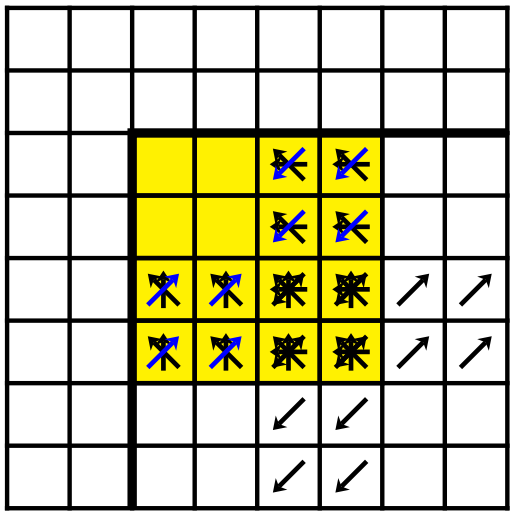
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



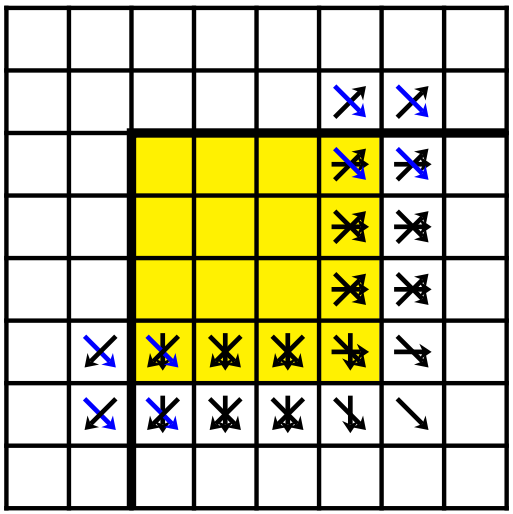
$f_{\alpha,out}^{f,n}$

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.



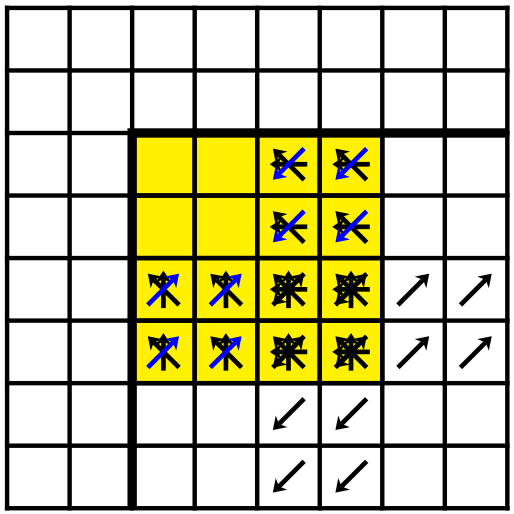
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



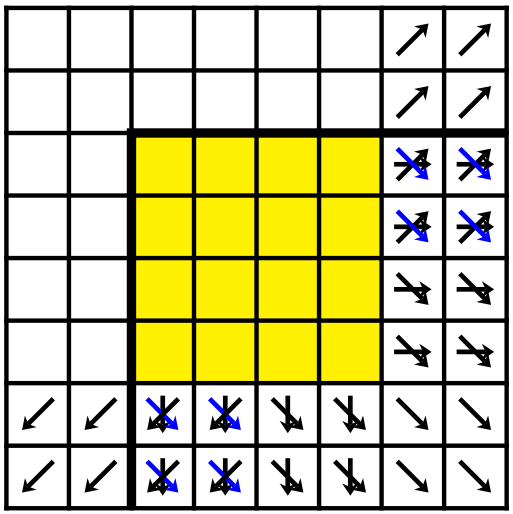
$\tilde{f}_{\alpha,out}^{f,n}$

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.



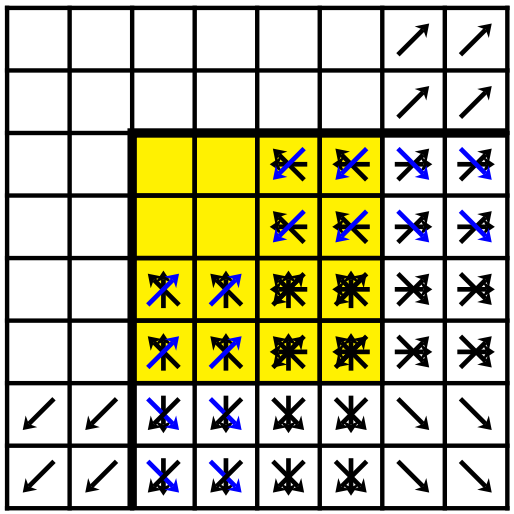
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



$\tilde{f}_{\alpha,out}^{f,n+1/2}$

Adaptive LBM

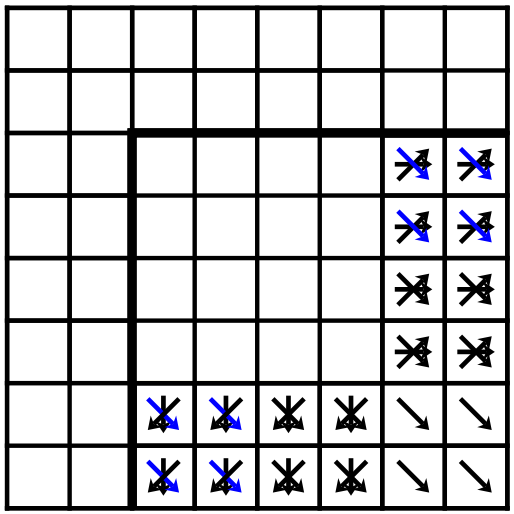
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



$\tilde{f}_{\alpha,out}^{f,n+1/2}, \tilde{f}_{\alpha,in}^{f,n+1/2}$

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.

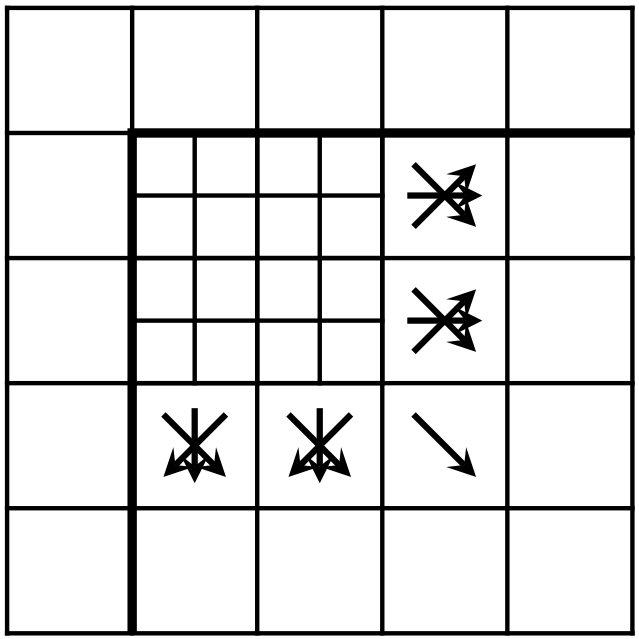


$\tilde{f}_{\alpha,out}^{f,n+1/2}$, $\tilde{f}_{\alpha,out}^{f,n}$

5. Average $\tilde{f}_{\alpha,out}^{f,n+1/2}$ (inner halo layer), $\tilde{f}_{\alpha,out}^{f,n}$ (outer halo layer) to obtain $\tilde{f}_{\alpha,out}^{C,n}$.

Adaptive LBM

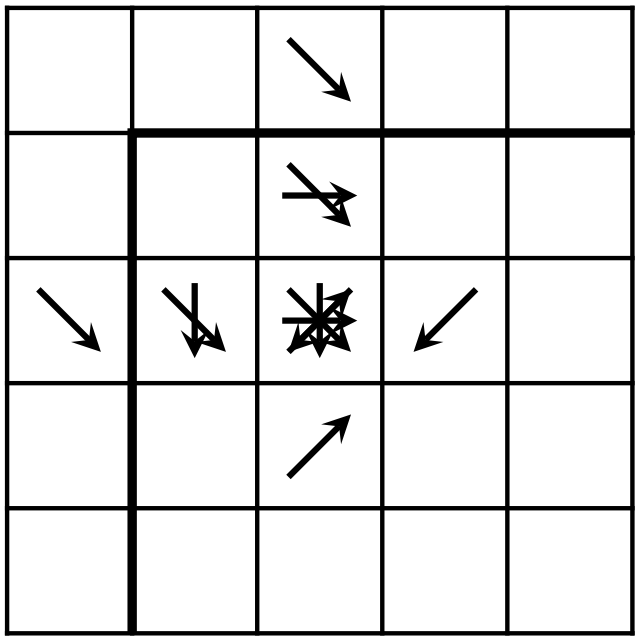
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

Adaptive LBM

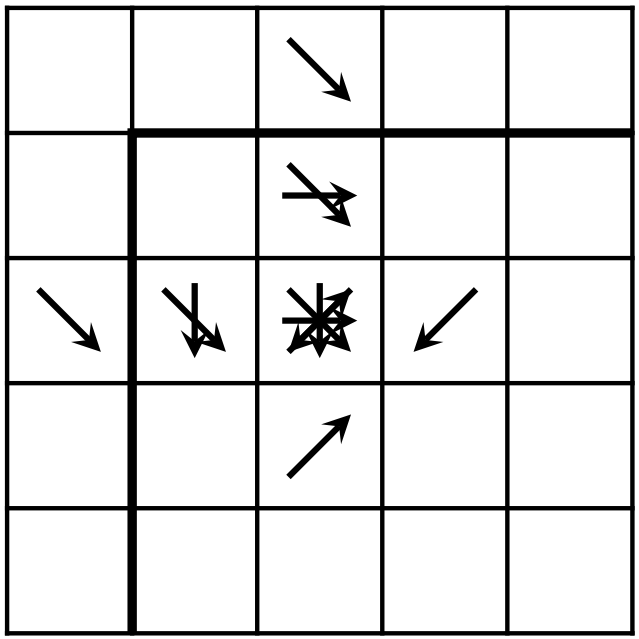
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

Adaptive LBM

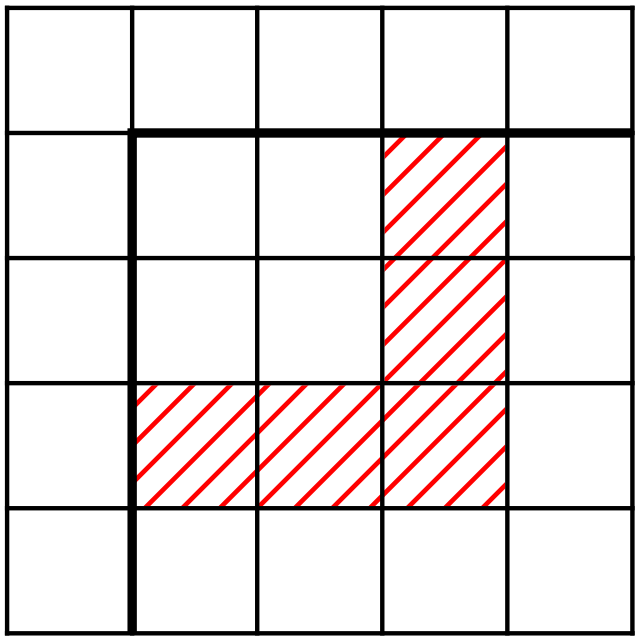
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

Adaptive LBM

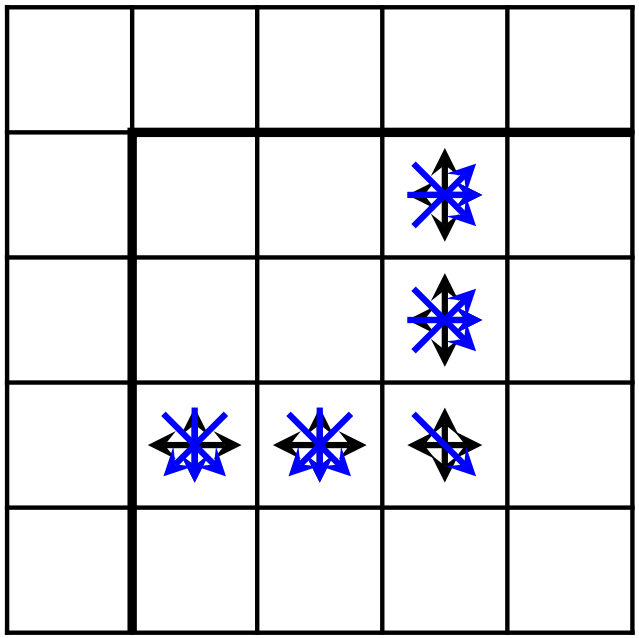
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

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.

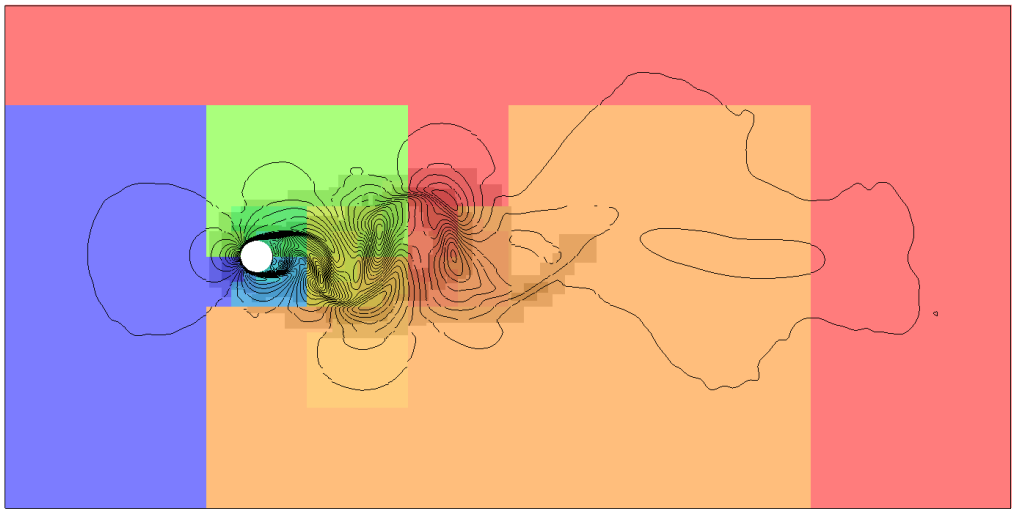


5. Average $\tilde{f}_{\alpha,out}^{f,n+1/2}$ (inner halo layer), $\tilde{f}_{\alpha,out}^{f,n}$ (outer halo layer) to obtain $\tilde{f}_{\alpha,out}^{C,n}$.
6. Revert transport into halos: $\bar{f}_{\alpha,out}^{C,n} := \mathcal{T}^{-1}(\tilde{f}_{\alpha,out}^{C,n})$
7. Parallel synchronization of $f_\alpha^{C,n}$, $\bar{f}_{\alpha,out}^{C,n}$
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].
[Deiterding and Wood, 2016]

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 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].

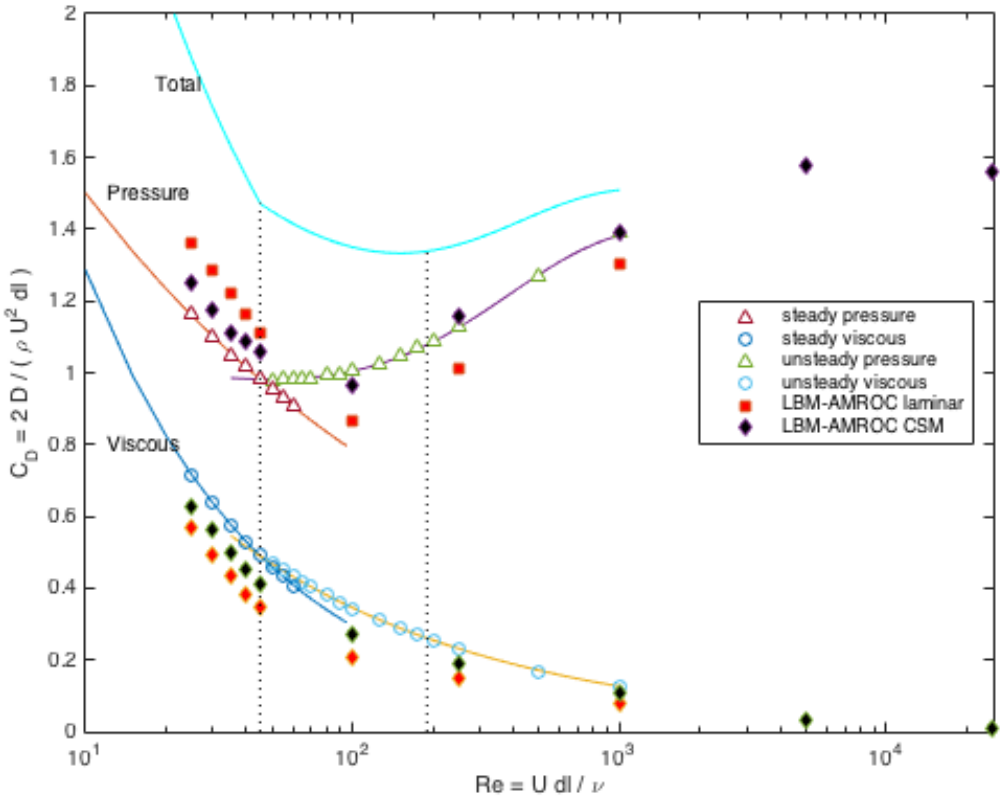
Flow over 2D cylinder, $d = 2\,\mathrm{cm}$

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

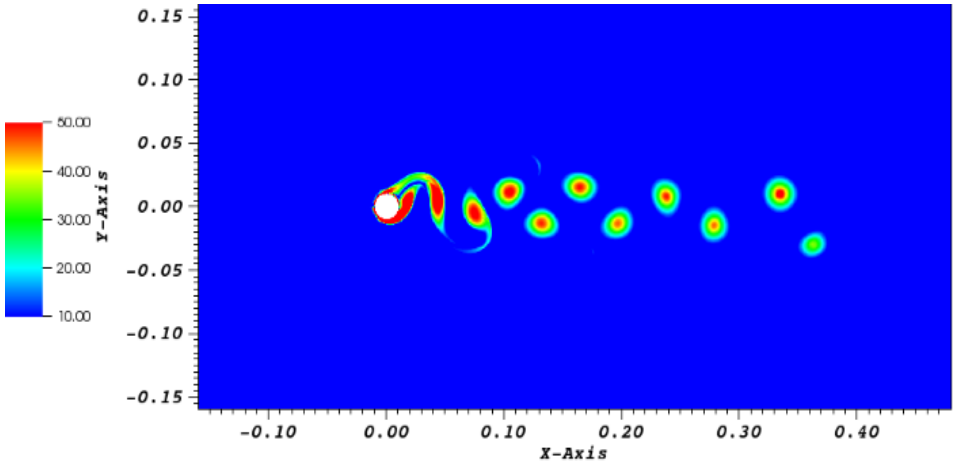
- Air with
  $\nu = 1.61 \cdot 10^{-5}\,\mathrm{m}^2/\mathrm{s}$,
  $\rho = 1.205\,\mathrm{kg}/\mathrm{m}^3$
- Domain size
  $[-8d, 24d] \times [-8d, 8d]$
- Dynamic refinement based on velocity. Last level to refine structure further.
- Inflow from left. Characteristic boundary conditions [Schlaffer,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



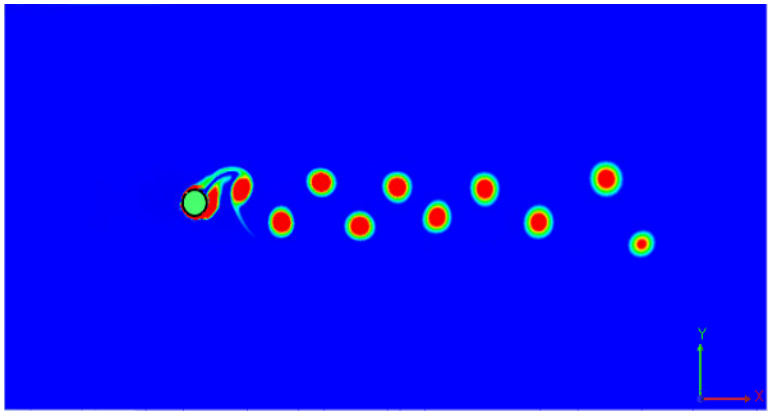
- Base lattice 320×160, 3 additional levels with factors $r_l = 2, 4, 4$.
- Resolution: ~ 320 points in diameter d
- Computation of C_D on 400 equidistant points along circle and averaged over time. Comparison above with [Henderson, 1995].

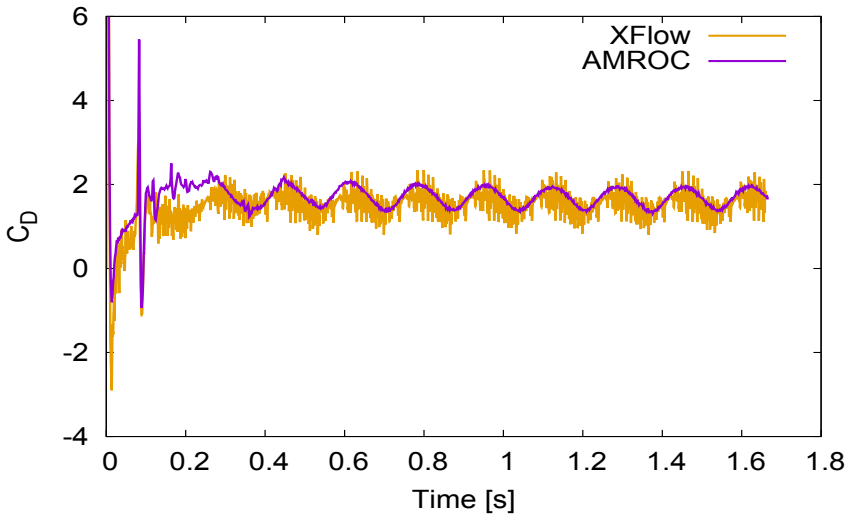
Oscillating cylinder, $V_R = 0.5$, $f_t = f_\theta = 3$, $\mathrm{Re} = 6310$

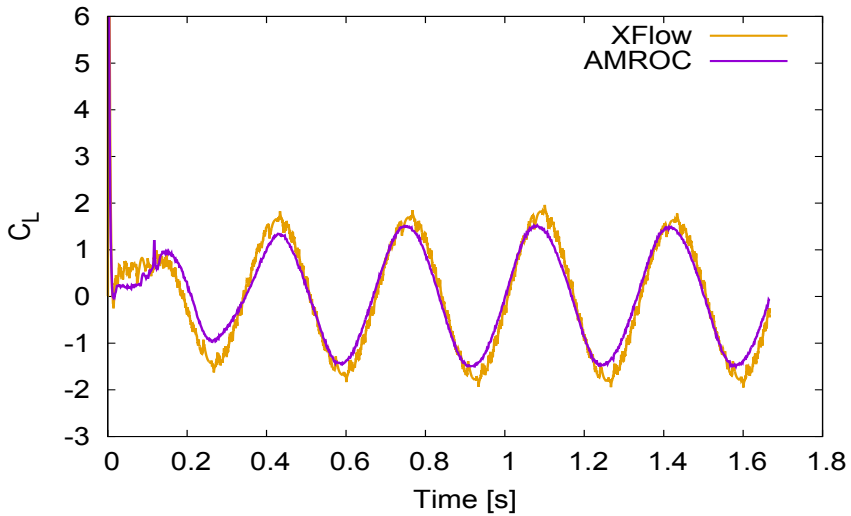
AMROC



XFlow

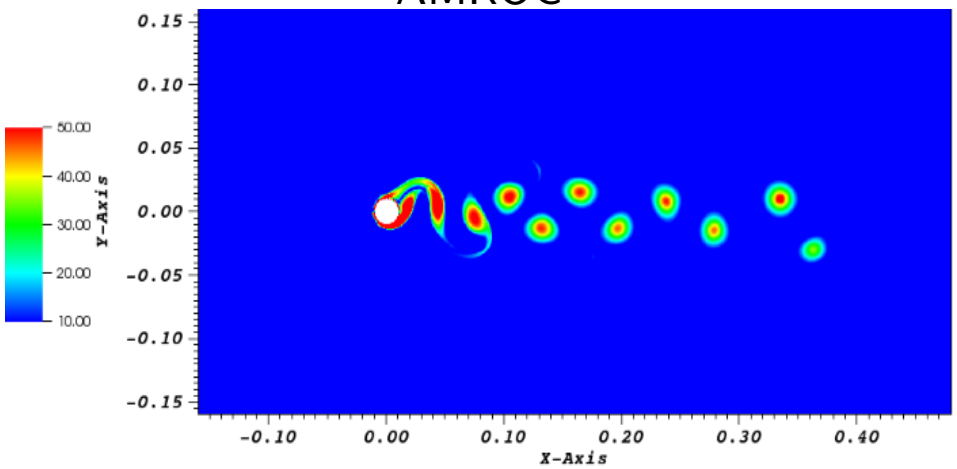




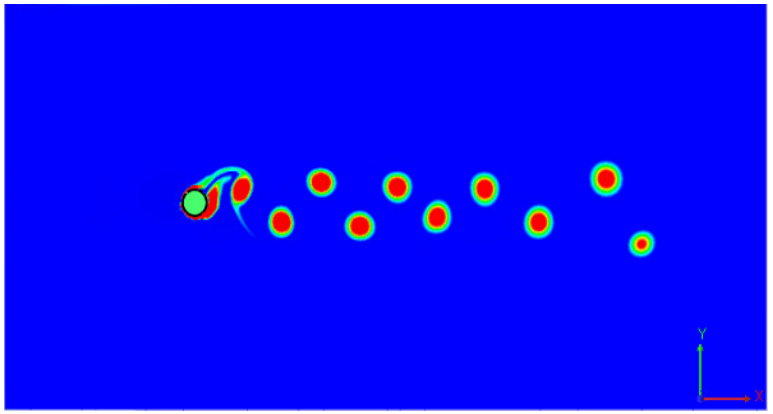


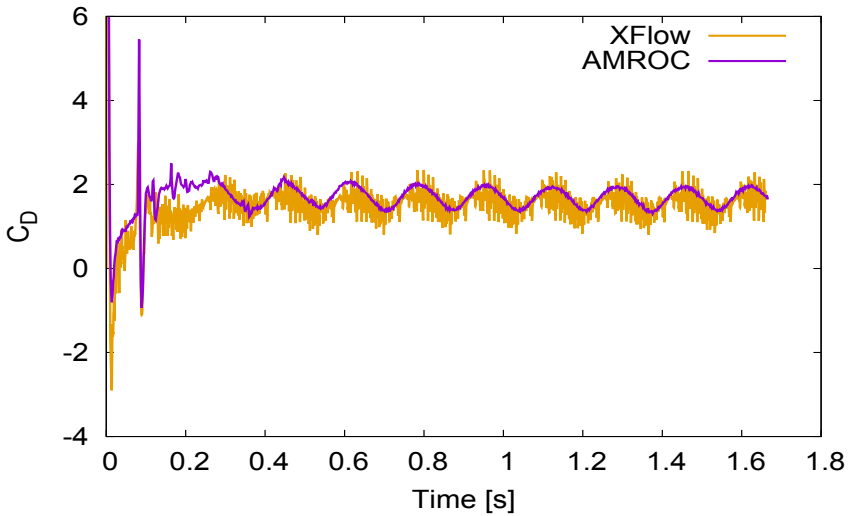
Oscillating cylinder, $V_R = 0.5$, $f_t = f_\theta = 3$, $\mathrm{Re} = 6310$

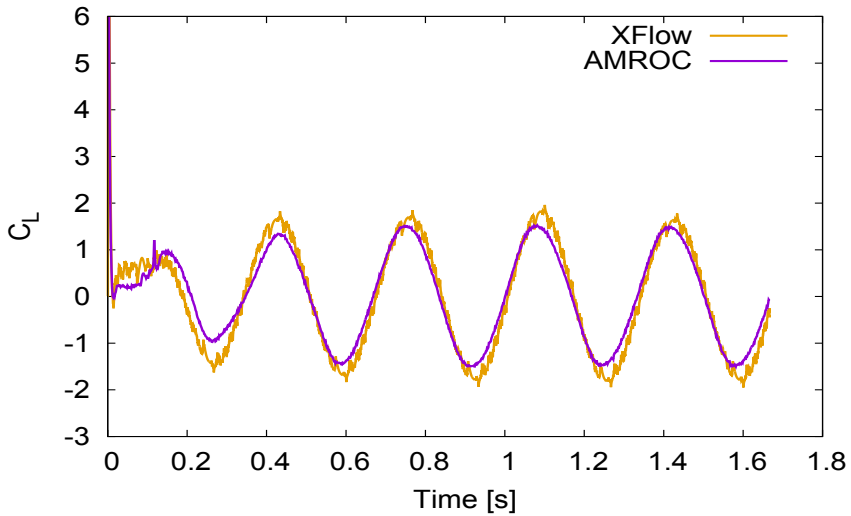
AMROC



XFlow







- Oscillation period: $T = 1/f_t = 0.33\,\mathrm{s}$. 10 regular vortices in $1.67\,\mathrm{s}$.
- CPU time on 6 cores for AMROC: $635.8\,s$, XFlow $\sim 50\,\%$ more expensive when normalized based on number of cells

Computational performance

Flow type	Case	Δt_0 [s]	Total cells		Δt_e [s]	Re	y^+	CPU time [s]	
			AMROC	XFlow				AMROC	XFlow
Laminar	**1a**	0.0015	85982	84778	3.33	1322	0	161.89	176
	1b	0.0015	91774	90488	3.33	1322	0	165.97	183
Turbulent	**2a**	0.00031	232840	216452	1.66	6310	2.4	635.8	887
	2b	0.00031	255582	246366	1.66	6310	2.6	933.2	1325

- [Laloglu and Deiterding, 2017]
- Intel-Xeon-3.50-GHz desktop workstation with 6 cores, communication through MPI
- Same base mesh and always three additional refinement levels
- AMROC: single-relaxation time LBM, block-based mesh adaptation
- XFlow: multi-relaxation time LBM, cell-based mesh adaptation

Computational performance

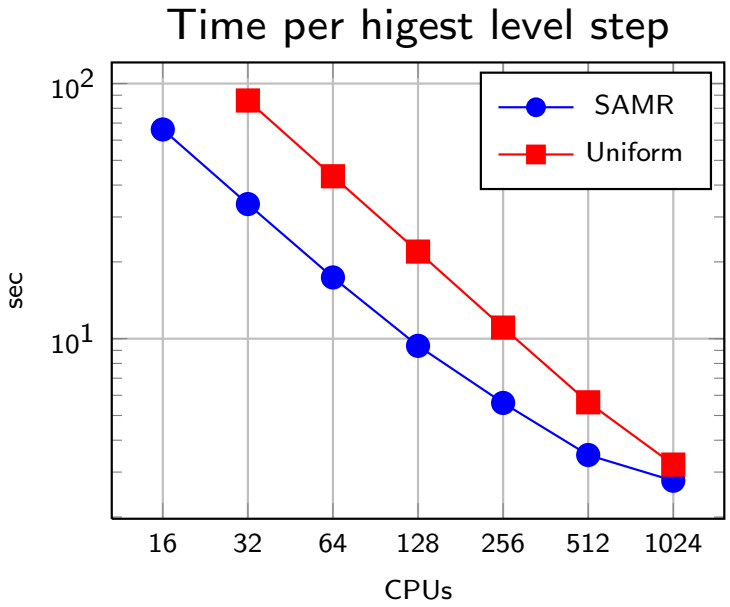
Flow type	Case	Δt_0 [s]	Total cells		Δt_e [s]	Re	y^+	CPU time [s]	
			AMROC	XFlow				AMROC	XFlow
Laminar	**1a**	0.0015	85982	84778	3.33	1322	0	161.89	176
	1b	0.0015	91774	90488	3.33	1322	0	165.97	183
Turbulent	**2a**	0.00031	232840	216452	1.66	6310	2.4	635.8	887
	2b	0.00031	255582	246366	1.66	6310	2.6	933.2	1325

- [Laloglu and Deiterding, 2017]
- Intel-Xeon-3.50-GHz desktop workstation with 6 cores, communication through MPI
- Same base mesh and always three additional refinement levels
- AMROC: single-relaxation time LBM, block-based mesh adaptation
- XFlow: multi-relaxation time LBM, cell-based mesh adaptation
- AMROC uses $\sim 7.5\,\%$ more cells on average more cells
- Normalized on cell number Case 2a is 50 % more expensive for XFlow than for AMROC-LBM
- Case 2b is 42 % more expensive in CPU time alone

AMROC strong scalability tests

3D wave propagation method with Roe scheme: spherical blast wave

- Tests run IBM BG/P (mode VN)



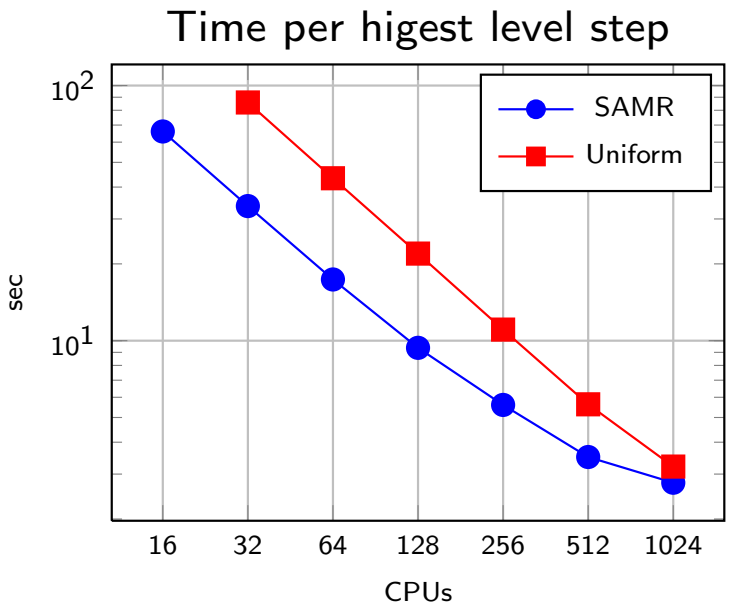
$64 \times 32 \times 32$ base grid, 2 additional levels with factors 2, 4; uniform $512 \times 256 \times 256 = 33.6 \cdot 10^6$ cells

Level	Grids	Cells
0	1709	65,536
1	1735	271,048
2	2210	7,190,208

AMROC strong scalability tests

3D wave propagation method with Roe scheme: spherical blast wave

- Tests run IBM BG/P (mode VN)

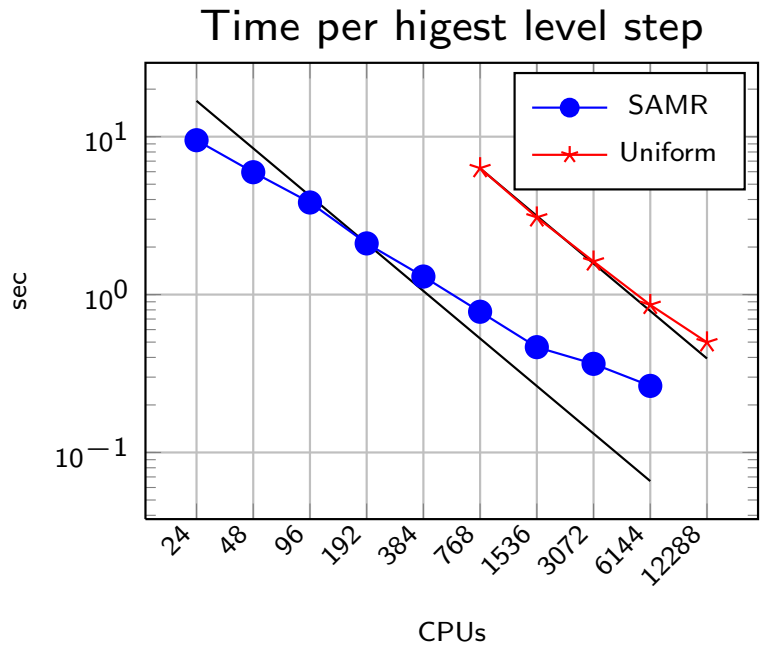


$64 \times 32 \times 32$ base grid, 2 additional levels with factors 2, 4; uniform $512 \times 256 \times 256 = 33.6 \cdot 10^6$ cells

Level	Grids	Cells
0	1709	65,536
1	1735	271,048
2	2210	7,190,208

3D SRT-D3Q19: flow over rough surface of $19 \times 13 \times 2$ spheres

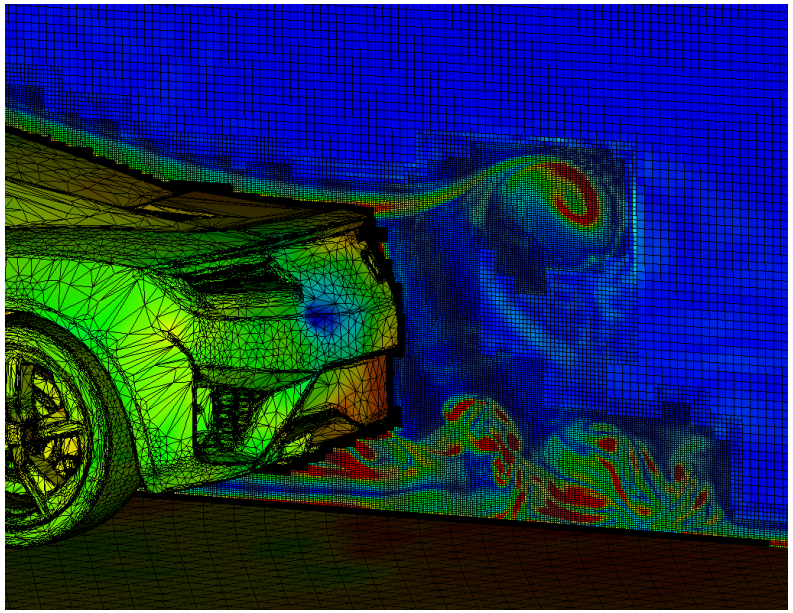
- Tests run Cray XC30m (Archer)

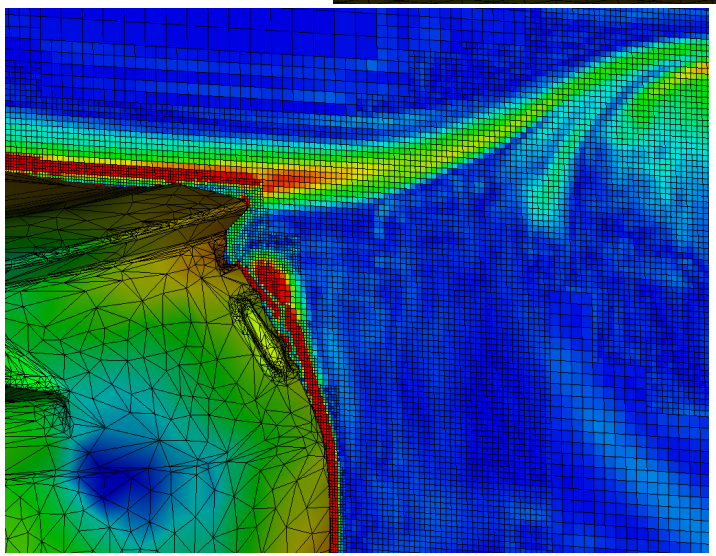


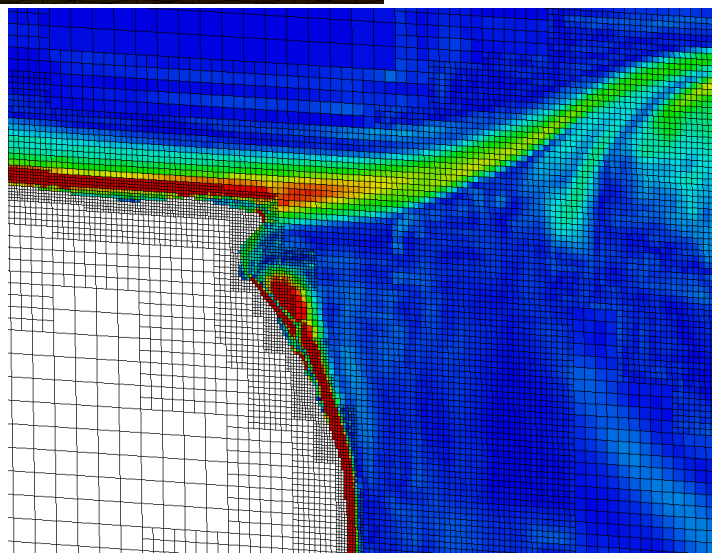
$360 \times 240 \times 108$ base grid, 2 additional levels with factors 2, 4; uniform $1440 \times 1920 \times 432 = 1.19 \cdot 10^9$ cells

Level	Grids	Cells
0	788	9,331,200
1	21367	24,844,504
2	1728	10,838,016

Mesh adaptation over a prototype car

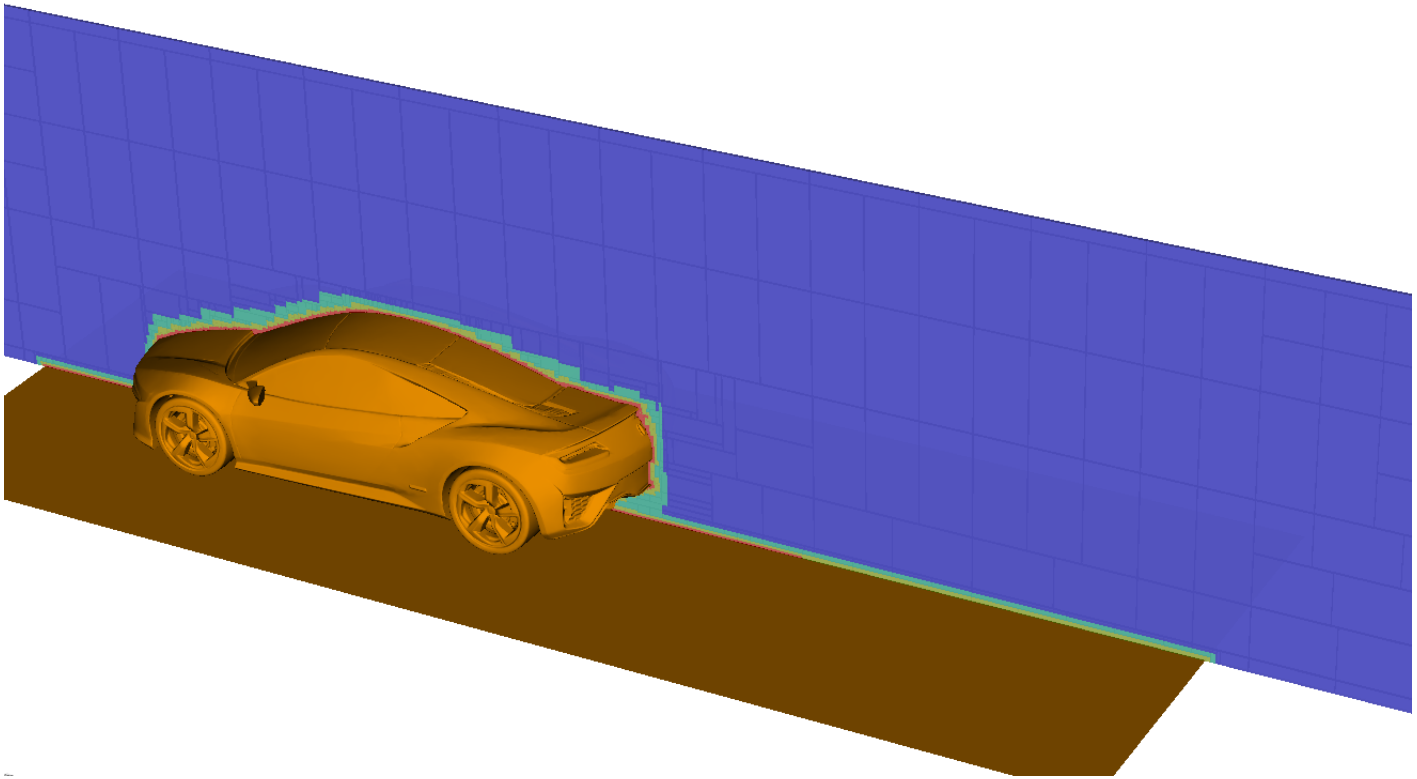






Mesh adaptation over a prototype car

Used refinement blocks and levels (indicated by color)



- SAMR base grid $600 \times 200 \times 132$ cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 3.125\,\mathrm{mm}$
- Adaptation based on level set and scaled gradient of magnitude of vorticity vector
- 236M cells vs. 8.1 billion (uniform) at $t = 0.4075\,\mathrm{s}$

Refinement at $t = 0.4075\,\mathrm{s}$

Level	Grids	Cells
0	11,605	15,840,000
1	11,513	23,646,984
2	31,382	144,447,872
3	21,221	52,388,336

Conclusions / Ongoing work

- AMROC-LBM provides a fully dimensional interface for real-world CFD.
- User has to create surface mesh files (e.g. Gmsh, Cubit, Centaur, Meshlab) in OBJ and use converter into ISS format.
- Typically copy and adjust suitable Problem.h in C++ and build the specific case.
- Visualization nowadays for instance with Paraview, VisIt (.vtk), Tecplot (.dat) using own converter tool from native hierarchical HDF4 data.

Conclusions / Ongoing work

- AMROC-LBM provides a fully dimensional interface for real-world CFD.
- User has to create surface mesh files (e.g. Gmsh, Cubit, Centaur, Meshlab) in OBJ and use converter into ISS format.
- Typically copy and adjust suitable Problem.h in C++ and build the specific case.
- Visualization nowadays for instance with Paraview, VisIt (.vtk), Tecplot (.dat) using own converter tool from native hierarchical HDF4 data.
- Reliable and validated fully for laminar flows.
- Incorporation of complex turbulence models and wall functions is still experimental.
- Excellent interface for incorporating LBM schemes of arbitrary state vector.
- Cartesian CFD with block-based AMR is faster than cell-cased AMR and tailored for massively parallel computer systems using MPI.
- Fast dynamic mesh adaptation in AMROC makes FSI problems with complex motion easily accessible. See talk tomorrow.

Conclusions / Ongoing work

- AMROC-LBM provides a fully dimensional interface for real-world CFD.
- User has to create surface mesh files (e.g. Gmsh, Cubit, Centaur, Meshlab) in OBJ and use converter into ISS format.
- Typically copy and adjust suitable Problem.h in C++ and build the specific case.
- Visualization nowadays for instance with Paraview, VisIt (.vtk), Tecplot (.dat) using own converter tool from native hierarchical HDF4 data.
- Reliable and validated fully for laminar flows.
- Incorporation of complex turbulence models and wall functions is still experimental.
- Excellent interface for incorporating LBM schemes of arbitrary state vector.
- Cartesian CFD with block-based AMR is faster than cell-cased AMR and tailored for massively parallel computer systems using MPI.
- Fast dynamic mesh adaptation in AMROC makes FSI problems with complex motion easily accessible. See talk tomorrow.
- Currently working on non-Cartesian and overlapping adaptive meshes for hypersonic finite volume schemes and FV-LBM.
- Experiments with CUDA and OpenMP GPU code versions of LBM so far outside of AMROC.
- More information about AMROC (including doxygen pages with LBM codes) can be found at *http* : *//rdeiterding.website/html/amrcourse.htm*

References I

[Barton et al., 2013] Barton, P. T., Deiterding, R., Meiron, D. I., and Pullin, D. I. (2013). Eulerian continuum model and adaptive finite-difference method for high-velocity impact and penetration problems. *J. Comput. Phys.*, 240:76–99.

[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, 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 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.

[Bouzidi et al., 2001] Bouzidi, M., Firdaouss, M., and Lallemand, P. (2001). Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of Fluids*, 13:3452.

[Cai et al., 2018] Cai, X., Deiterding, R., Liang, J., Sun, M., and Mahmoudi, Y. (2018). Experimental and numerical investigations on propagating modes of detonations: detonation wave/boundary layer interaction. *Combust. Flame*, 190:201–215.

[Cai et al., 2016] Cai, X., Liang, J., Deiterding, R., Che, Y., and Lin, Z. (2016). Adaptive mesh refinement based simulations of three-dimensional detonation combustion in supersonic combustible mixtures with a detailed reaction model. *Int. J. Hydrogen Energy*, 41:3222–3239.

[Cerminara et al., 2018] Cerminara, A., Deiterding, R., and Sandham, N. (2018). Dns of hypersonic flow over porous surfaces with a hybrid method. In *AIAA Aerospace Sciences Meeting, AIAA Science and Technology Forum and Exposition*.

[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.

[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.

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

References II

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

[Deiterding, 2011b] Deiterding, R. (2011b). High-resolution numerical simulation and analysis of Mach reflection structures in detonation waves in low-pressure $\mathrm{h_2} : \mathrm{o_2} : \mathrm{ar}$ mixtures: a summary of results obtained with adaptive mesh refinement framework AMROC. *J. Combustion*, 2011:738969.

[Deiterding and Bader, 2005] Deiterding, R. and Bader, G. (2005). High-resolution simulation of detonations with detailed chemistry. In Warnecke, G., editor, *Analysis and Numerics for Conservation Laws*, pages 69–91. Springer.

[Deiterding et al., 2009a] Deiterding, R., Cirak, F., and Mauch, S. P. (2009a). 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., 2007] Deiterding, R., Cirak, F., Mauch, S. P., and Meiron, D. I. (2007). A virtual test facility for simulating detonation-and shock-induced deformation and fracture of thin flexible shells. *Int. J. Multiscale Computational Engineering*, 5(1):47–63.

[Deiterding et al., 2009b] Deiterding, R., Domingues, M. O., Gomes, S. M., Roussel, O., and Schneider, K. (2009b). Adaptive multiresolution or adaptive mesh refinement? A case study for 2D Euler equations. *European Series in Applied and Industrial Mathematics: Proceedings*, 29:28–42.

[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, 2016] Deiterding, R. and Wood, S. L. (2016). An adaptive lattice Boltzmann method for predicting wake fields behind wind turbines. In Dillmann, A., Heller, G., Krämer, E., Wagner, C., and Breitsamter, C., editors, *New Results in Numerical and Experimental Fluid Mechanics X*, volume 132 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pages 845–857. Springer.

[Gomes et al., 2015] Gomes, A. K. F., Domingues, M. O., Schneider, K., Mendes, O., and Deiterding, R. (2015). An adaptive multiresolution method for ideal magnetohydrodynamics using divergence cleaning with parabolic-hyperbolic correction. *Applied Numerical Mathematics*, 95:199–213.

References III

[Guo et al., 2002] Guo, Z., Shi, B., and Zheng, C. (2002). A coupled lattice BGK model for the Boussinesq equations. *Int. J. Numerical Methods in Fluids*, 39:325–342.

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

[Laloglu and Deiterding, 2017] Laloglu, C. and Deiterding, R. (2017). Simulation of the flow around an oscillating cylinder with adaptive lattice Boltzmann methods. In Ivanyi, P. Topping, B. H. V. and Varady, G., editors, *Proc. 5th Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering*, page paper 19. Civil-Comp Press.

[Lombardini and Deiterding, 2010] Lombardini, M. and Deiterding, R. (2010). Large-eddy simulation of Richtmyer-Meshkov instability in a converging geometry. *Physics of Fluids*, 22(9):091112.

[Malaspinas and Sagaut, 2014] Malaspinas, O. and Sagaut, P. (2014). Wall model for the large-eddy simulation based on the lattice Boltzmann method. *J. Comput. Phys.*, 275:25–40.

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

[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.

[Perotti et al., 2013] Perotti, L. E., Deiterding, R., Inaba, 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(3-4):473–486.

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

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

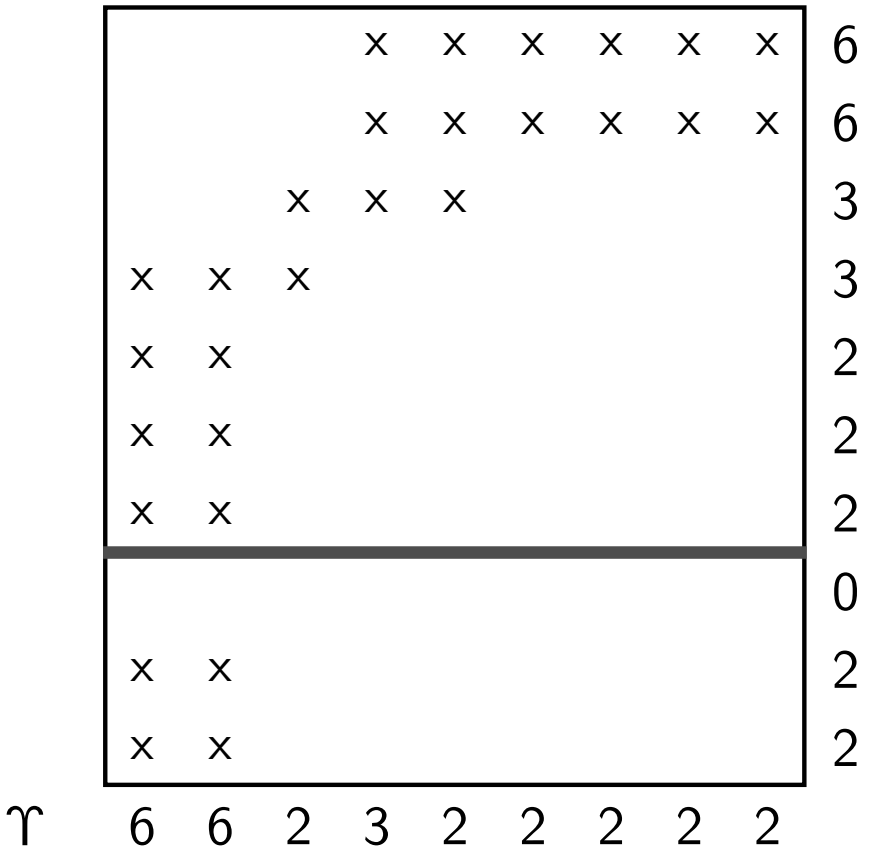
[Souza Lopes et al., 2018] Souza Lopes, M. M., Deiterding, R., Gomes, A. K. F., Mendes, O., and Domingues, M. O. (2018). An ideal compressible magnetohydrodynamic solver with parallel block-structured adaptive mesh refinement. *Computers & Fluids*. in press.

[Wan et al., 2017] Wan, Q., Jeon, H., Deiterding, R., and Eliasson, V. (2017). Numerical and experimental investigation of oblique shock wave reflection off a water wedge. *J. Fluid Mech.*, 826:732–758.

[Ziegler et al., 2011] 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.

References IV

Clustering by signatures

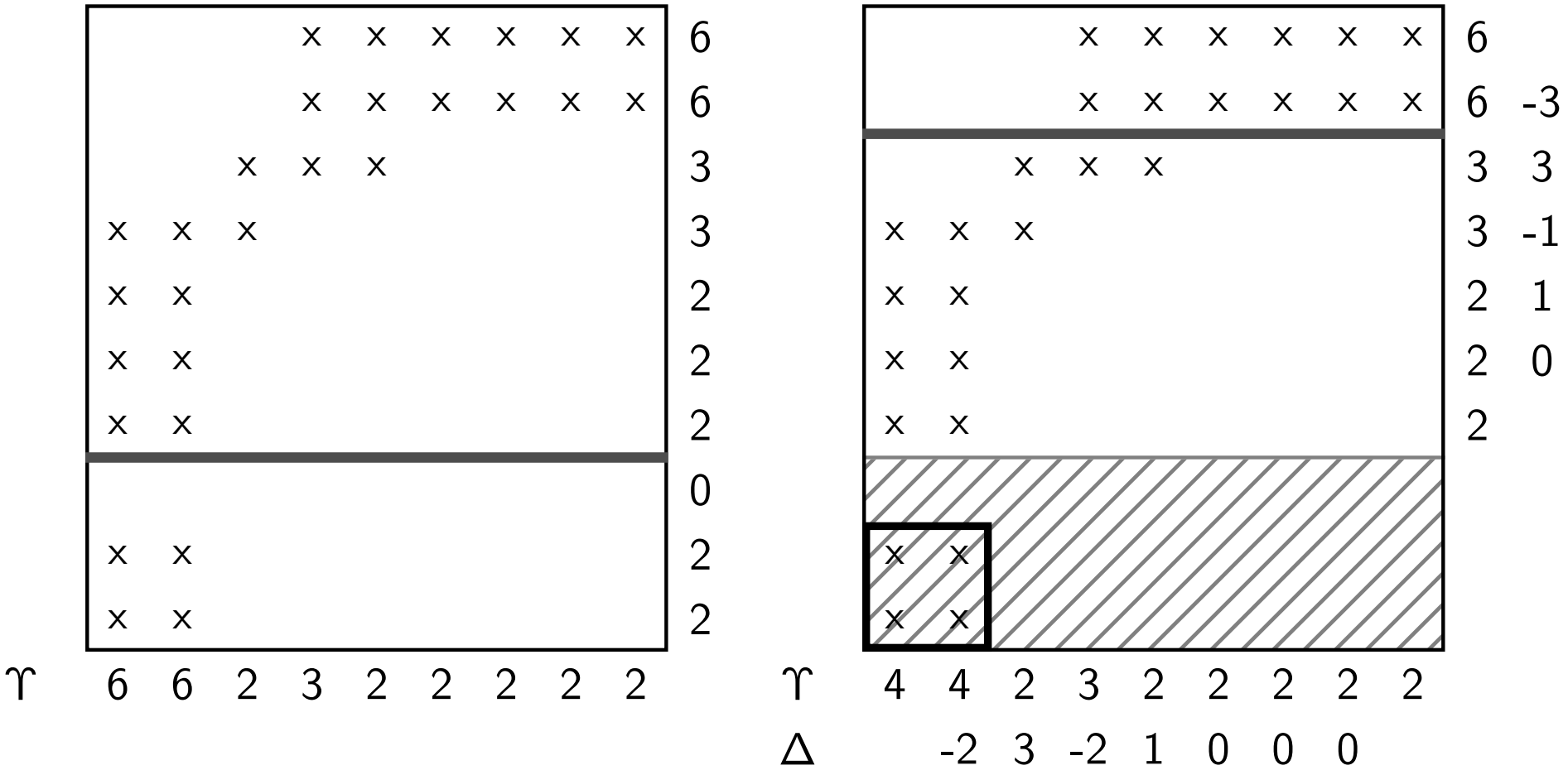


Υ Flagged cells per row/column

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]

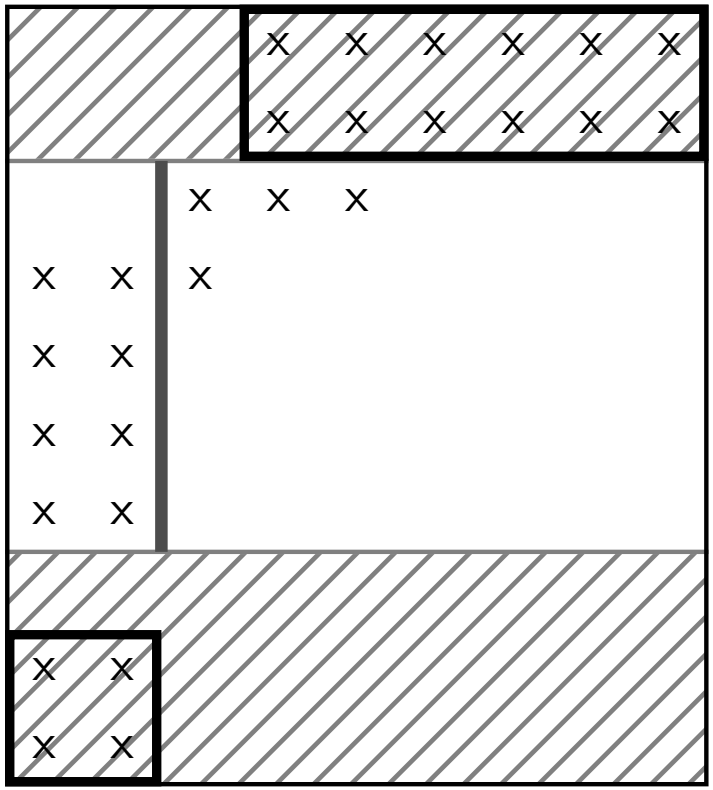
Clustering by signatures



ϒ Flagged cells per row/column

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

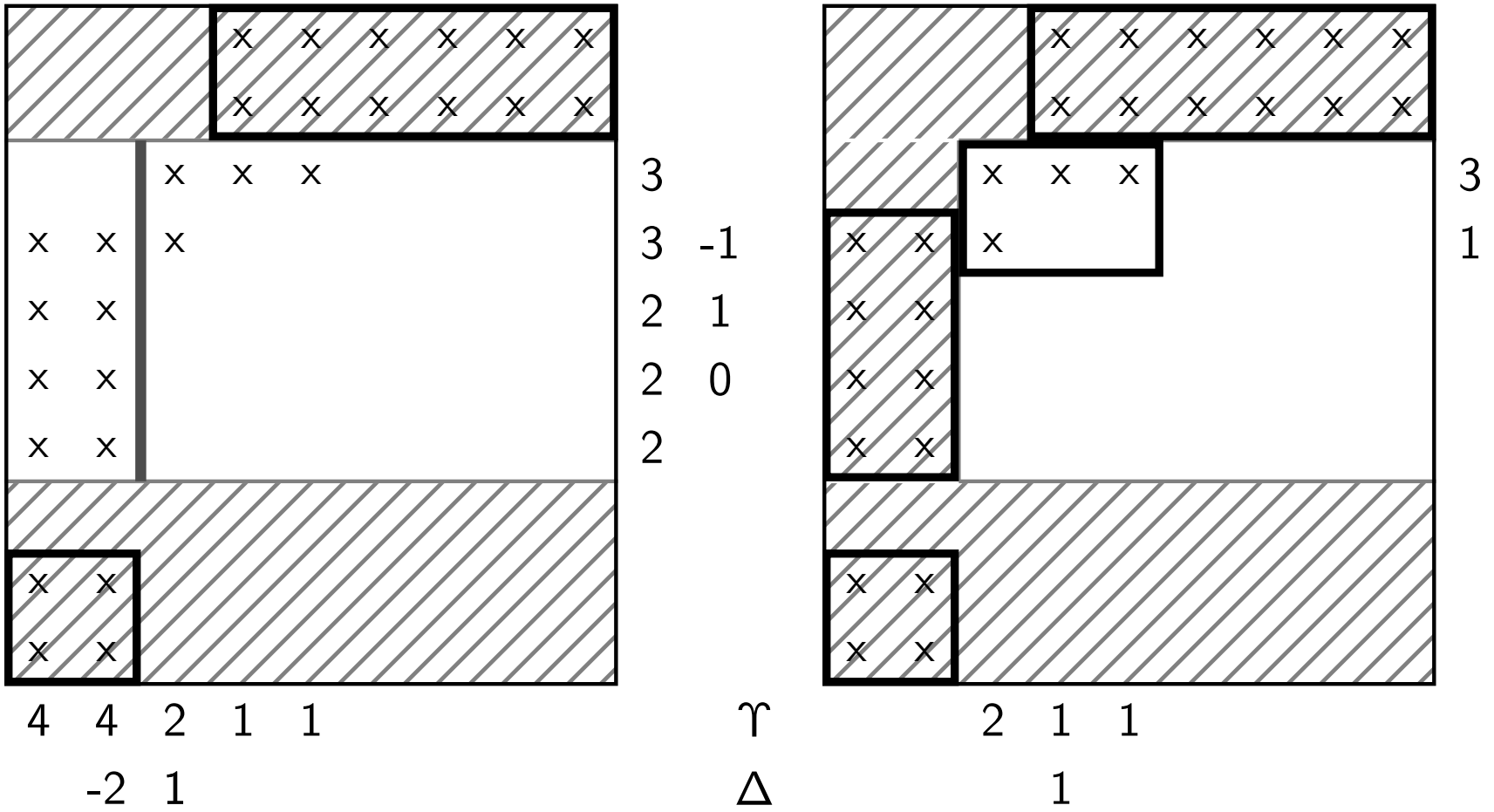
Technique from image detection: [Bell et al., 1994], see also [Berger and Rigoutsos, 1991], [Berger, 1986]



Υ	4	4	2	1	1		
Δ		-2	1				

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 $> \eta_{tol}$



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 $> \eta_{tol}$

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.

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]

Closest point transform algorithm

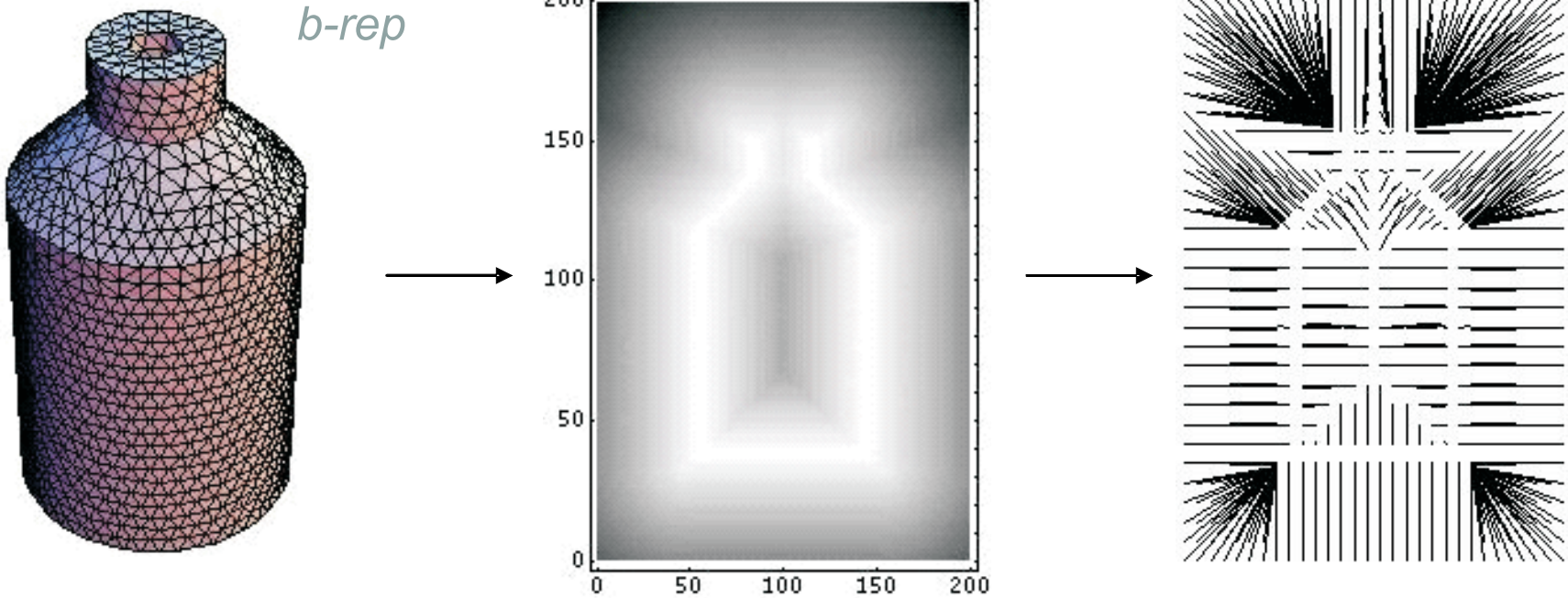
The signed distance φ to a surface $\mathcal{I}$ satisfies the eikonal equation [Sethian, 1999]

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

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]

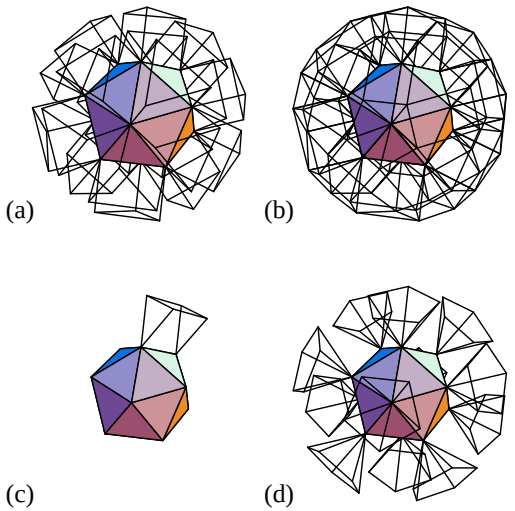


Surface mesh $\mathcal{I}$ Distance φ Normal to closest point

The characteristic / scan conversion algorithm

1. Build the characteristic polyhedrons for the surface mesh

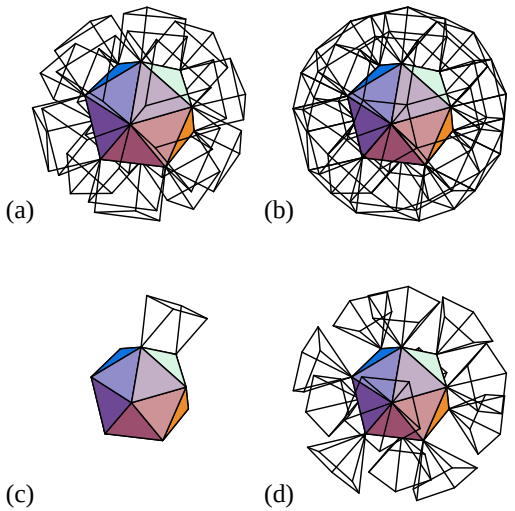
Characteristic polyhedra for faces, edges, and vertices



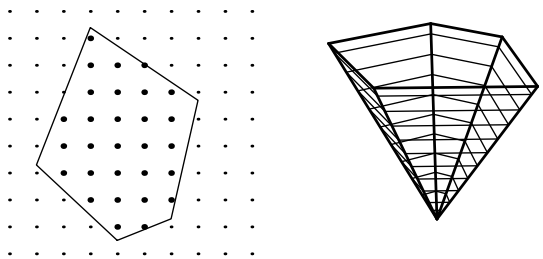
The characteristic / scan conversion algorithm

1. Build the characteristic polyhedrons for the surface mesh
2. For each face/edge/vertex
   2.1 Scan convert the polyhedron.

Characteristic polyhedra for faces, edges, and vertices



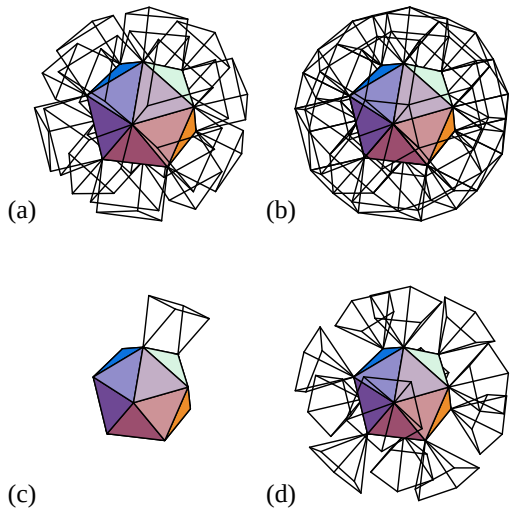
Slicing and scan conversion of apolygon



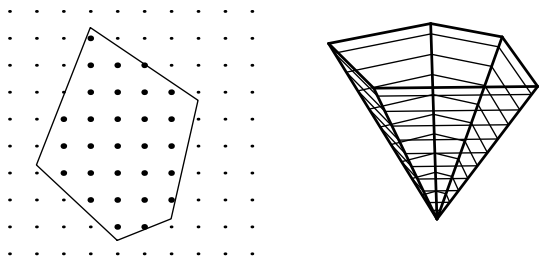
The characteristic / scan conversion algorithm

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

Characteristic polyhedra for faces, edges, and vertices



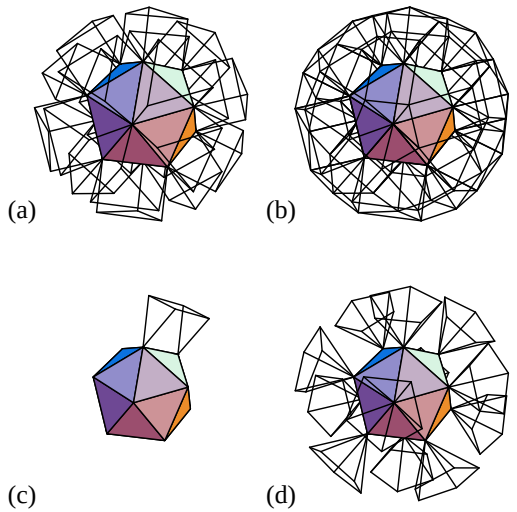
Slicing and scan conversion of apolygon



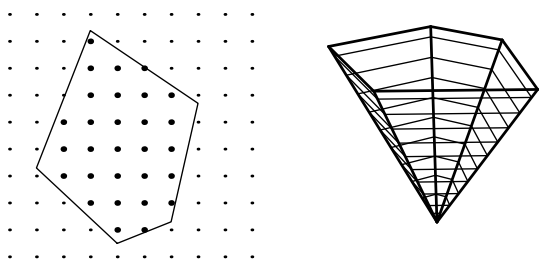
The characteristic / scan conversion algorithm

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.

Characteristic polyhedra for faces, edges, and vertices



Slicing and scan conversion of apolygon

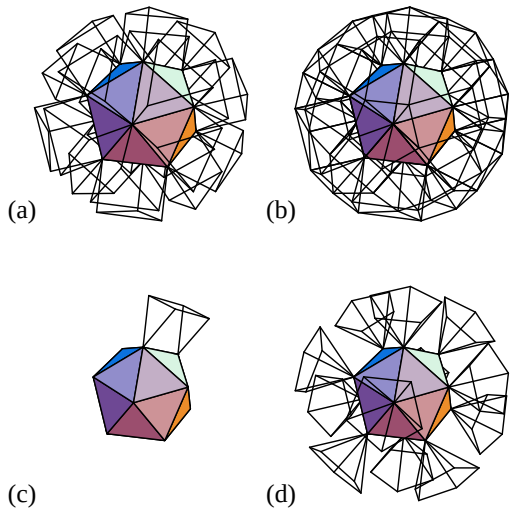


The characteristic / scan conversion algorithm

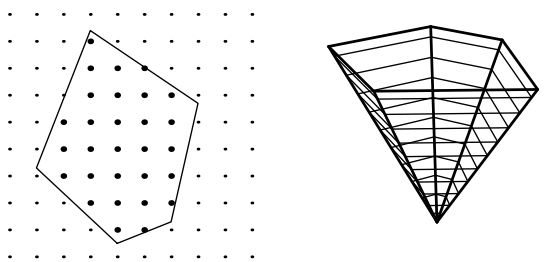
1. Build the characteristic polyhedrons for the surface mesh
2. For each face/edge/vertex
   - 2.1 Scan convert the polyhedron.
   - 2.2 Compute distance to that primitive for the scan converted points
3. Computational complexity.
   - $O(m)$ to build the b-rep and the polyhedra.
   - $O(n)$ to scan 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



Driven cavity - 3d cavity

- Intel Xeon-2.67 GHz 6-core (Westmere) dual-processor nodes with Qlogics interconnect
- Unigrid with 1 ghost cell

Cores	6	12	24	48	96
Time per step	2.80s	1.46s	0.73s	0.37s	0.18s
Par. Efficiency	100.00%	96.09%	95.33%	95.21%	94.82%
LBM Update	78.05%	77.08%	75.85%	74.50%	71.38%
Synchronization	7.25%	8.67%	10.00%	11.32%	14.35%
Phys. Boundary	0.51%	0.46%	0.45%	0.44%	0.44%
Misc	14.19%	13.79%	13.70%	13.73%	13.83%

- AMR with 4 ghost cells

Cores	6	12	24	48	96
Time per step	3.32s	1.90s	1.21s	0.54s	0.30s
Par. Efficiency	100.00%	87.42%	68.76%	77.02%	68.19%
LBM Update	43.44%	40.93%	31.33%	34.64%	30.11%
Synchronization	14.13%	18.26%	34.73%	25.76%	30.69%
Phys. Boundary	1.03%	0.98%	0.77%	0.86%	0.77%
Regridding	15.53%	16.02%	13.87%	18.72%	20.82%
Interpolation	16.74%	15.71%	11.95%	13.15%	11.51%
Fixup	2.89%	2.60%	2.02%	2.28%	2.03%
Misc	6.22%	5.50%	5.33%	4.59%	4.08%

Driven cavity - 3d cavity

- Intel Xeon-2.67 GHz 6-core (Westmere) dual-processor nodes with Qlogics interconnect
- Unigrid with 1 ghost cell

Cores	6	12	24	48	96
Time per step	2.80s	1.46s	0.73s	0.37s	0.18s
Par. Efficiency	100.00%	96.09%	95.33%	95.21%	94.82%
LBM Update	78.05%	77.08%	75.85%	74.50%	71.38%
Synchronization	7.25%	8.67%	10.00%	11.32%	14.35%
Phys. Boundary	0.51%	0.46%	0.45%	0.44%	0.44%
Misc	14.19%	13.79%	13.70%	13.73%	13.83%

- AMR with 4 ghost cells

Cores	6	12	24	48	96
Time per step	3.32s	1.90s	1.21s	0.54s	0.30s
Par. Efficiency	100.00%	87.42%	68.76%	77.02%	68.19%
LBM Update	43.44%	40.93%	31.33%	34.64%	30.11%
Synchronization	14.13%	18.26%	34.73%	25.76%	30.69%
Phys. Boundary	1.03%	0.98%	0.77%	0.86%	0.77%
Regridding	15.53%	16.02%	13.87%	18.72%	20.82%
Interpolation	16.74%	15.71%	11.95%	13.15%	11.51%
Fixup	2.89%	2.60%	2.02%	2.28%	2.03%
Misc	6.22%	5.50%	5.33%	4.59%	4.08%

- Expense for boundary is increased compared to FV methods because the algorithm uses few floating point operations but a large state vector!

Space-filling curve algorithm