A block-structured parallel adaptive Lattice-Boltzmann method for rotating geometries

Ralf Deiterding* & Stephen Wood+

*Deutsches Zentrum für Luft- und Raumfahrt
Bunsenstr. 10, Göttingen, Germany
E-mail: ralf.deiterding@dlr.de

+University of Tennessee - Knoxville
The Bredesen Center, Knoxville, TN 37996

SIAM Conference on Parallel Processing for Scientific Computing
February 19, 2014

Outline

Outline

Outline

Outline

S. Wood was partially sponsored by TN-Score and the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725.

AMROC

- Cartesian adaptive fluid solver framework for explicit finite volume methods. Implements for instance Berger-Collela-type AMR.
- Many shock-capturing methods (MUSCL, (hybrid) WENO, etc.) implemented for complex flux functions.

AMROC

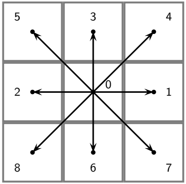
- Cartesian adaptive fluid solver framework for explicit finite volume methods. Implements for instance Berger-Collela-type AMR.
- Many shock-capturing methods (MUSCL, (hybrid) WENO, etc.) implemented for complex flux functions.
- Used to drive Virtual Test Facility (VTF) FSI software.
- Targets strongly driven problems (shocks, blast, detonations)
- Geometry embedding via ghost fluid techniques and level set functions. Distance computation with CPT algorithm [Mauch, 2000].

AMROC

- Cartesian adaptive fluid solver framework for explicit finite volume methods. Implements for instance Berger-Collela-type AMR.
- Many shock-capturing methods (MUSCL, (hybrid) WENO, etc.) implemented for complex flux functions.
- Used to drive Virtual Test Facility (VTF) FSI software.
- Targets strongly driven problems (shocks, blast, detonations)
- Geometry embedding via ghost fluid techniques and level set functions. Distance computation with CPT algorithm [Mauch, 2000].
- $\sim 430,000$ LOC in C++, C, Fortran-77, Fortran-90.
- Version V2.0 at http://www.cacr.caltech.edu/asc. V1.1 (no complex boundaries) still at http://amroc.sourceforge.net.
- Version used here V3.0 with significantly enhanced parallelization (V2.1 not released).
- Papers: [Deiterding, 2011, Deiterding and Wood, 2013, Deiterding et al., 2009, Deiterding et al., 2007, Deiterding et al., 2006] and at http://www.rdeiterding.de

Lattice Boltzmann method

Boltzmann equation: $\partial_t f + \mathbf{u} \cdot \nabla f = \omega(f^{eq} - f)$
Two-dimensional LBM for weakly compressible flows
Formulated on FV grids! ($\rightarrow$ boundary conditions!)

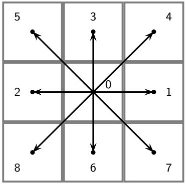


$$\rho(\mathbf{x},t) = \sum_{\alpha=0}^{8} f_\alpha(\mathbf{x},t), \quad \rho(\mathbf{x},t) u_i(\mathbf{x},t) = \sum_{\alpha=0}^{8} \mathbf{e}_{\alpha i} f_\alpha(\mathbf{x},t)$$

cf. [Hähnel, 2004]

Lattice Boltzmann method

Boltzmann equation: $\partial_t f + \mathbf{u} \cdot \nabla f = \omega(f^{eq} - f)$
Two-dimensional LBM for weakly compressible flows
Formulated on FV grids! ($\rightarrow$ boundary conditions!)



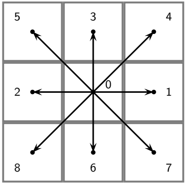
$$\rho(\mathbf{x},t) = \sum_{\alpha=0}^{8} f_\alpha(\mathbf{x},t), \quad \rho(\mathbf{x},t)u_i(\mathbf{x},t) = \sum_{\alpha=0}^{8} \mathbf{e}_{\alpha i} f_\alpha(\mathbf{x},t)$$

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

cf. [Hähnel, 2004]

Lattice Boltzmann method

Boltzmann equation: $\partial_t f + \mathbf{u} \cdot \nabla f = \omega(f^{eq} - f)$
Two-dimensional LBM for weakly compressible flows
Formulated on FV grids! ($\rightarrow$ boundary conditions!)



$$\rho(\mathbf{x},t) = \sum_{\alpha=0}^{8} f_\alpha(\mathbf{x},t), \quad \rho(\mathbf{x},t) u_i(\mathbf{x},t) = \sum_{\alpha=0}^{8} \mathbf{e}_{\alpha i} f_\alpha(\mathbf{x},t)$$

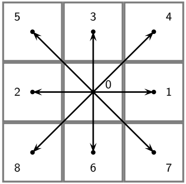
1.) Transport step $\mathcal{T}$: $\tilde{f}_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}, t)$
2.) Collision step $\mathcal{C}$:

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

cf. [Hähnel, 2004]

Lattice Boltzmann method

Boltzmann equation: $\partial_t f + \mathbf{u} \cdot \nabla f = \omega(f^{eq} - f)$
Two-dimensional LBM for weakly compressible flows
Formulated on FV grids! ($\rightarrow$ boundary conditions!)



$$\rho(\mathbf{x},t) = \sum_{\alpha=0}^{8} f_\alpha(\mathbf{x},t), \quad \rho(\mathbf{x},t) u_i(\mathbf{x},t) = \sum_{\alpha=0}^{8} \mathbf{e}_{\alpha i} f_\alpha(\mathbf{x},t)$$

1.) Transport step $\mathcal{T}$: $\tilde{f}_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}, t)$
2.) Collision step $\mathcal{C}$:

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

with equilibrium function

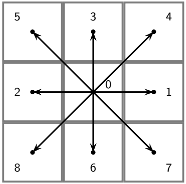
$$f_\alpha^{eq}(\rho, \mathbf{u}) = \rho t_\alpha \left[1 + \frac{\mathbf{e}_\alpha \mathbf{u}}{c_s^2} + \frac{(\mathbf{e}_\alpha \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^4} \right]$$

mit $t_\alpha = \frac{1}{9}\left\{4, 1, 1, 1, \frac{1}{4}, \frac{1}{4}, 1, \frac{1}{4}, \frac{1}{4}\right\}$

cf. [Hähnel, 2004]

Lattice Boltzmann method

Boltzmann equation: $\partial_t f + \mathbf{u} \cdot \nabla f = \omega(f^{eq} - f)$
Two-dimensional LBM for weakly compressible flows
Formulated on FV grids! ($\rightarrow$ boundary conditions!)



$$\rho(\mathbf{x},t) = \sum_{\alpha=0}^{8} f_\alpha(\mathbf{x},t), \quad \rho(\mathbf{x},t)u_i(\mathbf{x},t) = \sum_{\alpha=0}^{8} \mathbf{e}_{\alpha i} f_\alpha(\mathbf{x},t)$$

1.) Transport step $\mathcal{T}$: $\tilde{f}_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x},t)$
2.) Collision step $\mathcal{C}$:

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

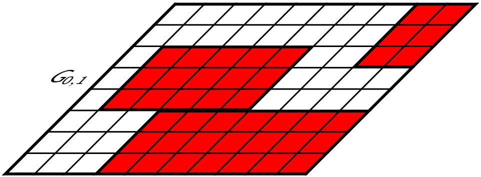
with equilibrium function

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

mit $t_\alpha = \frac{1}{9}\left\{4, 1, 1, 1, \frac{1}{4}, \frac{1}{4}, 1, \frac{1}{4}, \frac{1}{4}\right\}$
Lattice speed of sound: $c_s = \frac{1}{\sqrt{3}} \frac{\Delta x}{\Delta t}$, pressure $p = \sum_\alpha f_\alpha^{eq} c_s^2 = \rho c_s^2 = \rho R T$
Collision frequency vs. kinematic viscosity: $\omega = \frac{c_s^2}{\nu + \Delta t c_s^2/2}$ cf. [Hähnel, 2004]

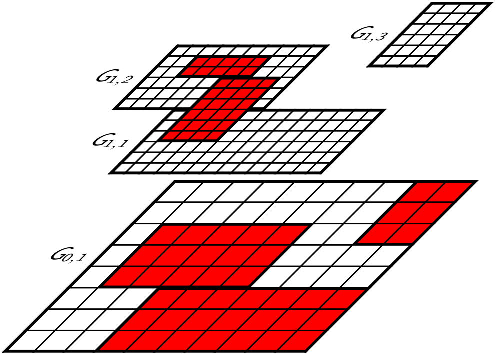
Block-structured adaptive mesh refinement (SAMR)

- Refined blocks overlay coarser ones



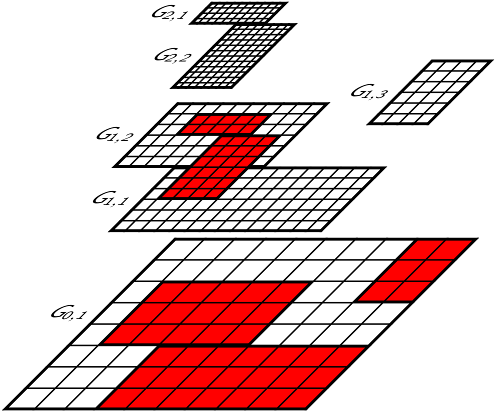
Block-structured adaptive mesh refinement (SAMR)

- Refined blocks overlay coarser ones



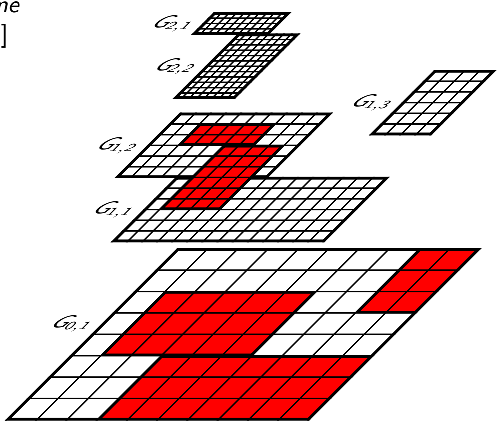
Block-structured adaptive mesh refinement (SAMR)

- Refined blocks overlay coarser ones



Block-structured adaptive mesh refinement (SAMR)

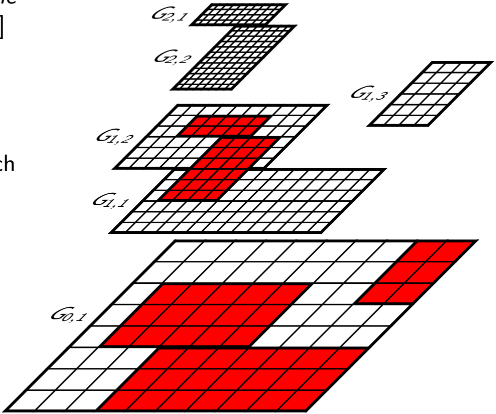
- Refined blocks overlay coarser ones
- Recursive refinement in space *and time* by factor r_l [Berger and Colella, 1988] ideal for LBM



Block-structured adaptive mesh refinement (SAMR)

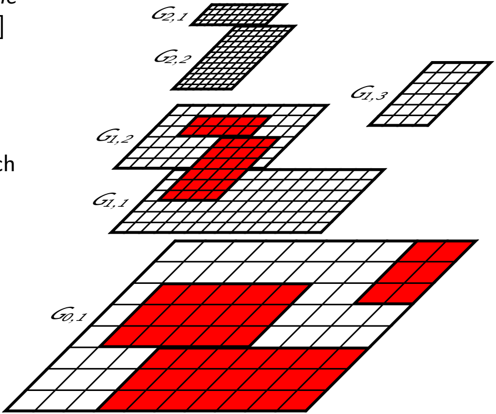
- Refined blocks overlay coarser ones
- Recursive refinement in space *and time* by factor r_l [Berger and Colella, 1988] ideal for LBM
- Block (aka patch) based data structures

+ Numerical scheme only for single patch necessary
+ Most efficient LBM implementation with patch-wise for-loops



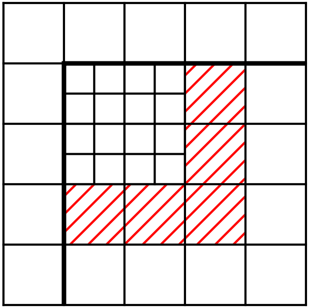
Block-structured adaptive mesh refinement (SAMR)

- Refined blocks overlay coarser ones
- Recursive refinement in space *and time* by factor r_l [Berger and Colella, 1988] ideal for LBM
- Block (aka patch) based data structures
- \+ Numerical scheme only for single patch necessary
- \+ Most efficient LBM implementation with patch-wise for-loops
- \+ Cache efficient
- Spatial interpolation and averaging can be used unaltered
- \- Cluster-algorithm necessary



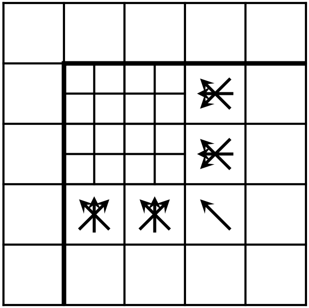
Adaptive LBM

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



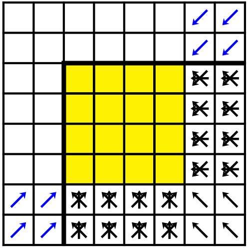
Adaptive LBM

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

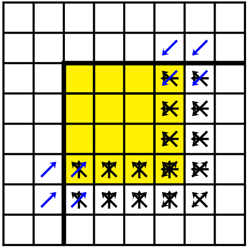
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.



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

Adaptive LBM

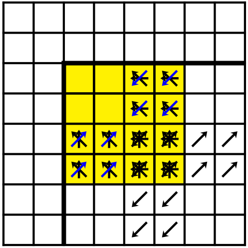
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



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

Adaptive LBM

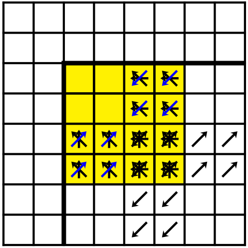
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



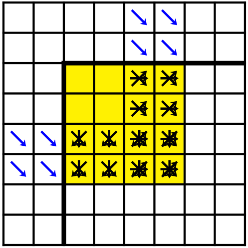
$\tilde{f}_{\alpha,in}^{f,n+1/2}$

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.



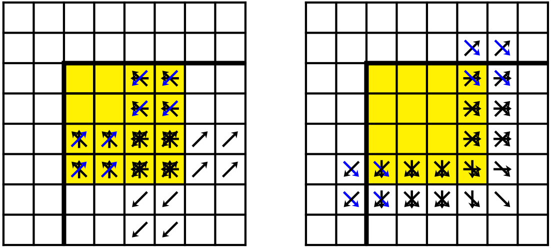
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



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

Adaptive LBM

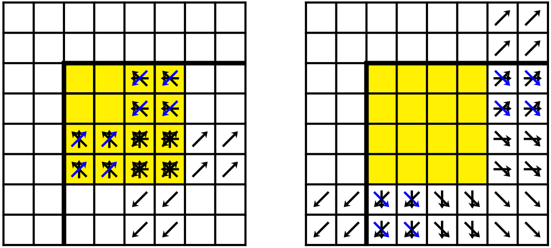
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



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

Adaptive LBM

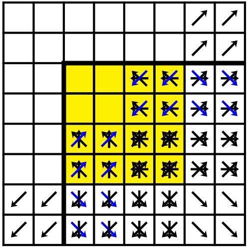
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

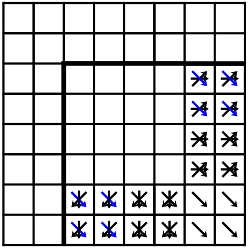
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

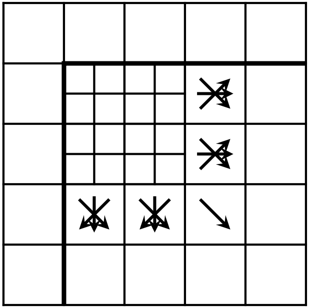


$\tilde{f}_{\alpha,out}^{f,n+1/2}$, $\tilde{f}_{\alpha,out}^{f,n}$

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

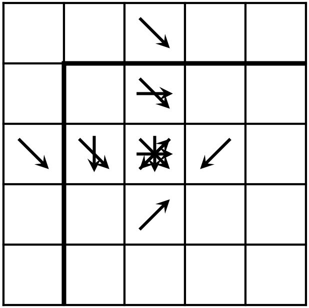
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

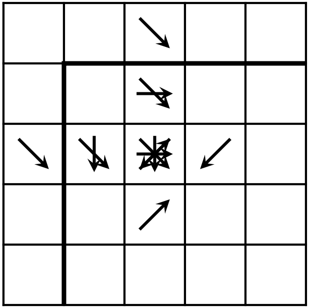
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

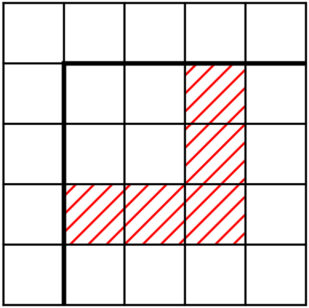
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

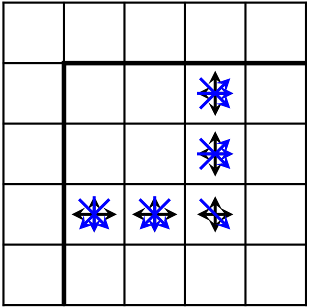
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

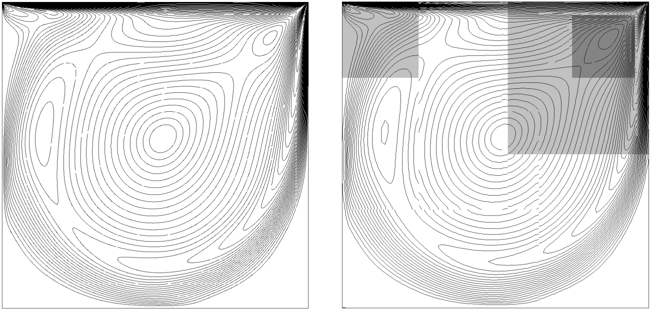


5. Average $\tilde{f}_{\alpha,out}^{f,n+1/2}$ (inner halo layer), $\tilde{f}_{\alpha,out}^{f,n}$ (outer halo layer) to obtain $\tilde{f}_{\alpha,out}^{C,n}$.
6. Revert transport into halos: $\bar{f}_{\alpha,out}^{C,n} := \mathcal{T}^{-1}(\tilde{f}_{\alpha,out}^{C,n})$
7. Parallel synchronization of $f_\alpha^{C,n},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

Verification - driven cavity

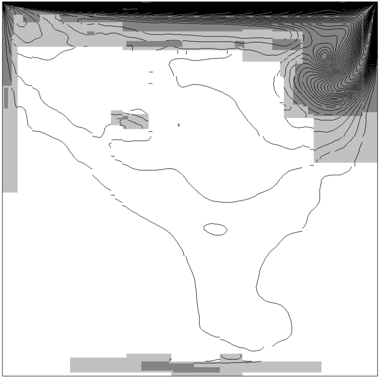
- $\mathrm{Re} = 1500$ in air, $\nu = 1.5 \cdot 10^{-5}\,\mathrm{m}^2/\mathrm{s}$, $u = 22.5\,\mathrm{m/s}$.
- Domain size $1\,\mathrm{mm} \times 1\,\mathrm{mm}$.
- Reference computation uses 800×800 lattice.
- 588,898 time steps to $t_e = 5 \cdot 10^{-3}\,\mathrm{s}$, $\sim 35\,\mathrm{h}$ CPU.
- Statically adaptive computation uses 100×100 lattice with $r_{1,2} = 2$.
- 294,452 time steps to $t_e = 5 \cdot 10^{-3}\,\mathrm{s}$ on finest level.

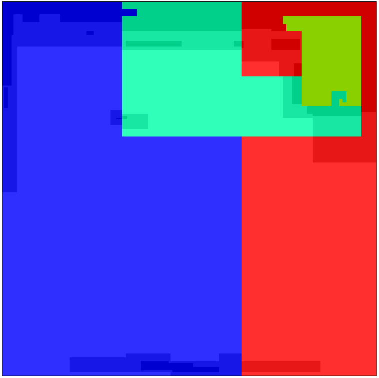


Isolines of density. Left: reference, right on refinement at t_e.

Driven cavity - dynamic refinement

- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution

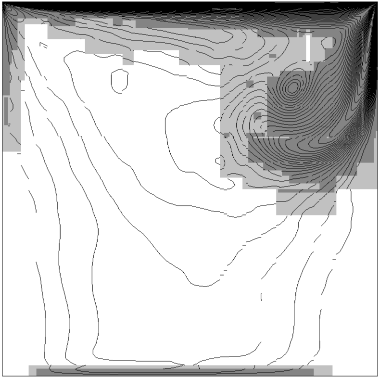




Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

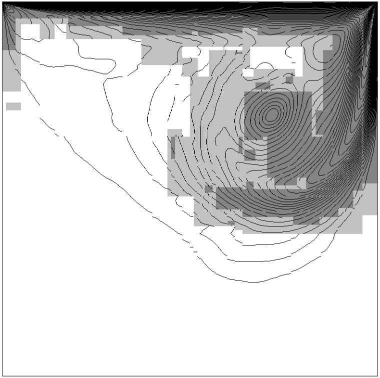
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

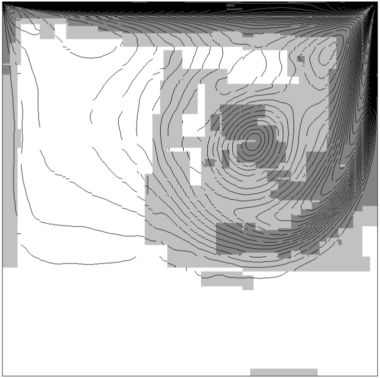
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

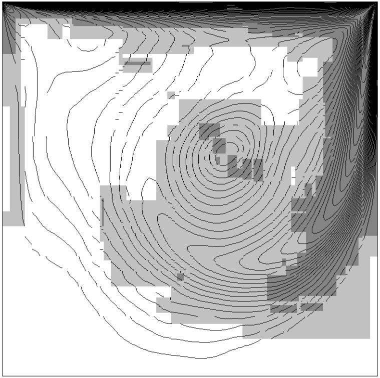
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution

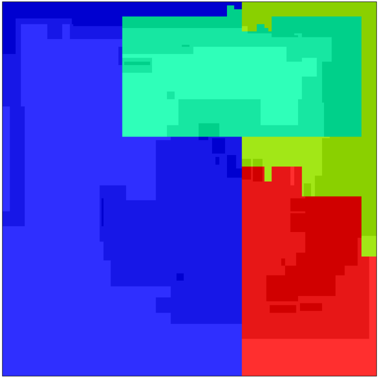


Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution

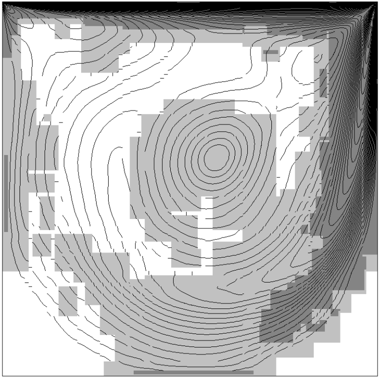


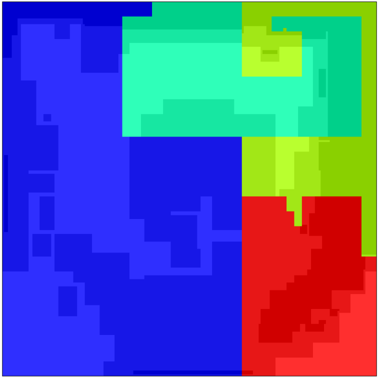


Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution

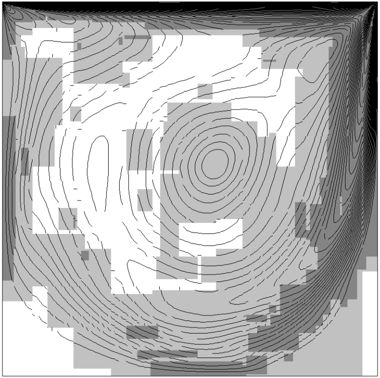


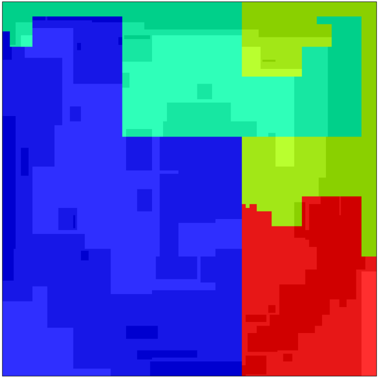


Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution

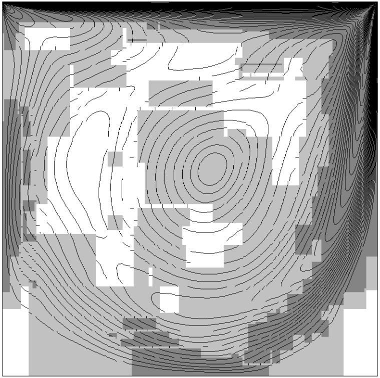




Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

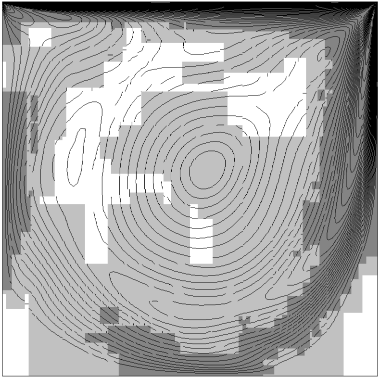
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

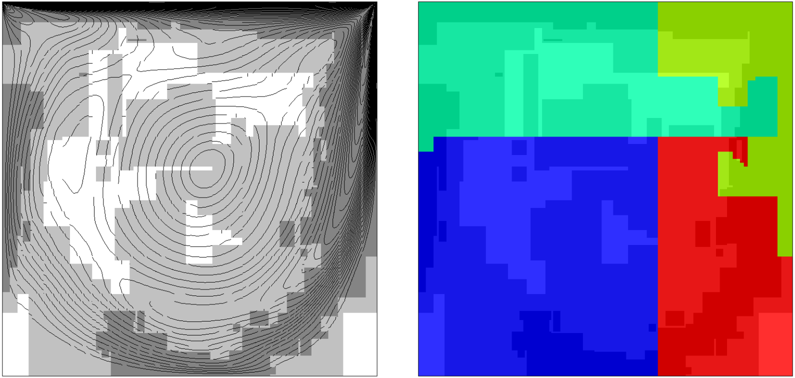
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

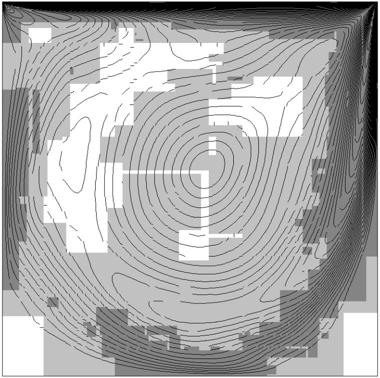
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

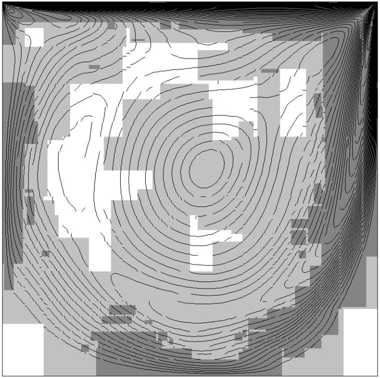
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

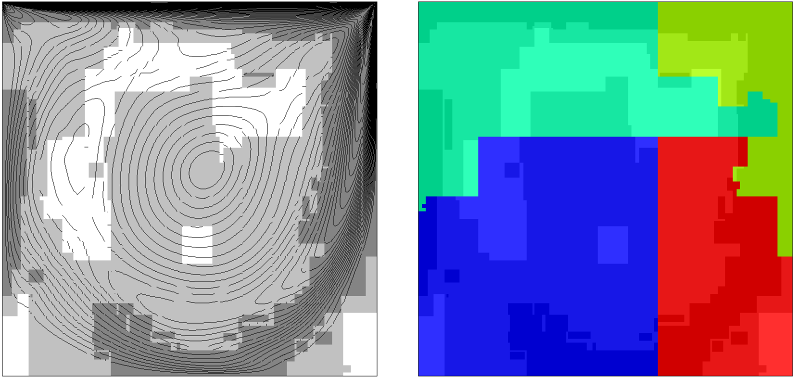
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - 3d cavity

- Similar setup as in 2d. No-slip wall everywhere except at lid. $\mathrm{Re} = 1000$ in air, $u = 15\,\mathrm{m/s}$.
- AMR 64^3 base mesh with $r_{1,2} = 2$. Regridding and repartition only at every 2nd base level step.
- 95 time steps on coarsest level benchmarked.
- Uniform grid $256^3 = 16.8 \cdot 10^6$ cells.

Level	Grids	Cells
0	178	262,144
1	668	1,538,912
2	2761	7,842,872

Grid and cells used on 24 cores

Cores	6	12	24	48	96
Time per step	1.82s	0.94s	0.50s	0.28s	0.16s
LBM Update	44.97%	42.83%	39.64%	35.37%	31.10%
Error Estimation	1.37%	1.30%	1.20%	1.07%	0.94%
Regridding	14.59%	14.79%	15.60%	16.75%	19.14%
Fixup	4.18%	3.96%	3.74%	3.42%	3.07%
Interp. Boundaries	9.34%	9.15%	8.30%	7.17%	6.13%
Interp. Regridding	3.53%	3.23%	3.02%	2.73%	2.44%
Sync Boundaries	8.69%	11.20%	14.28%	18.26%	21.07%
Sync Fixup	2.41%	3.41%	4.70%	6.50%	7.99%
Sync Regridding	0.77%	0.72%	0.74%	0.83%	0.99%
Phys. Boundaries	0.69%	0.68%	0.63%	0.56%	0.49%
Clustering	0.55%	0.48%	0.44%	0.40%	0.36%
Misc	8.90%	8.25%	7.72%	6.95%	6.26%

Driven cavity - 3d cavity

- Intel Xeon-2.67 GHz 6-core (Westmere) dual-processor nodes with Qlogics interconnect
- Unigrid with 2 ghost cells

Cores	6	12	24	48	96
Time per step	3.44s	1.81s	0.92s	0.47s	0.24s
Par. Efficiency	100.00%	95.04%	93.34%	92.40%	91.38%
LBM Update	74.86%	74.60%	72.48%	70.78%	67.58%
Synchronization	12.42%	13.32%	15.48%	17.35%	20.52%
Phys. Boundary	0.78%	0.74%	0.69%	0.69%	0.65%
Misc	11.94%	11.34%	11.34%	11.18%	11.25%

- AMR with 2 ghost cells

Cores	6	12	24	48	96
Time per step	1.82s	0.94s	0.50s	0.28s	0.16s
Par. Efficiency	100.00%	96.47%	90.00%	81.68%	73.04%
LBM Update	46.34%	44.13%	40.84%	36.44%	32.04%
Synchronization	11.87%	15.33%	19.72%	25.58%	30.06%
Phys. Boundary	0.69%	0.68%	0.63%	0.56%	0.49%
Regridding	14.59%	14.79%	15.60%	16.75%	19.14%
Interpolation	12.88%	12.38%	11.31%	9.90%	8.57%
Fixup	4.18%	3.96%	3.74%	3.42%	3.07%
Misc	9.45%	8.73%	8.16%	7.36%	6.62%

Driven cavity - 3d cavity

- Intel Xeon-2.67 GHz 6-core (Westmere) dual-processor nodes with Qlogics interconnect
- Unigrid with 2 ghost cells

Cores	6	12	24	48	96
Time per step	3.44s	1.81s	0.92s	0.47s	0.24s
Par. Efficiency	100.00%	95.04%	93.34%	92.40%	91.38%
LBM Update	74.86%	74.60%	72.48%	70.78%	67.58%
Synchronization	12.42%	13.32%	15.48%	17.35%	20.52%
Phys. Boundary	0.78%	0.74%	0.69%	0.69%	0.65%
Misc	11.94%	11.34%	11.34%	11.18%	11.25%

- AMR with 2 ghost cells

Cores	6	12	24	48	96
Time per step	1.82s	0.94s	0.50s	0.28s	0.16s
Par. Efficiency	100.00%	96.47%	90.00%	81.68%	73.04%
LBM Update	46.34%	44.13%	40.84%	36.44%	32.04%
Synchronization	11.87%	15.33%	19.72%	25.58%	30.06%
Phys. Boundary	0.69%	0.68%	0.63%	0.56%	0.49%
Regridding	14.59%	14.79%	15.60%	16.75%	19.14%
Interpolation	12.88%	12.38%	11.31%	9.90%	8.57%
Fixup	4.18%	3.96%	3.74%	3.42%	3.07%
Misc	9.45%	8.73%	8.16%	7.36%	6.62%

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

Driven cavity - 3d cavity

- Unigrid with 1 ghost cell

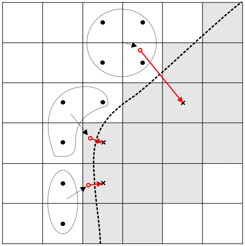
Cores	6	12	24	48	96
Time per step	2.80s	1.46s	0.73s	0.37s	0.18s
Par. Efficiency	100.00%	96.09%	95.33%	95.21%	94.82%
LBM Update	78.05%	77.08%	75.85%	74.50%	71.38%
Synchronization	7.25%	8.67%	10.00%	11.32%	14.35%
Phys. Boundary	0.51%	0.46%	0.45%	0.44%	0.44%
Misc	14.19%	13.79%	13.70%	13.73%	13.83%

- 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%

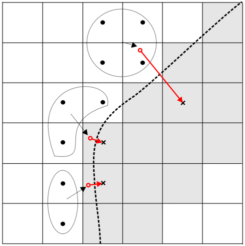
- Basically linear dependency on number of ghost cells used

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 macro-values in embedded boundary cells by interpolation / extrapolation.

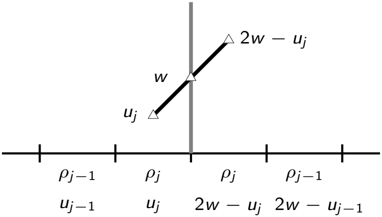
Level-set method for boundary embedding



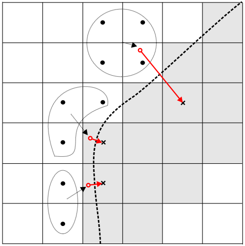
- Implicit boundary representation via distance function φ, normal $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$.
- Complex boundary moving with local velocity $\mathbf{w}$, treat interface as moving rigid wall.
- Construction of macro-values in embedded boundary cells by interpolation / extrapolation.

Interpolate / constant value extrapolate values at

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



Level-set method for boundary embedding



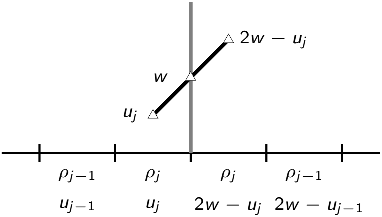
- Implicit boundary representation via distance function φ, normal $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$.
- Complex boundary moving with local velocity $\mathbf{w}$, treat interface as moving rigid wall.
- Construction of macro-values in embedded boundary cells by interpolation / extrapolation.

Interpolate / constant value extrapolate values at

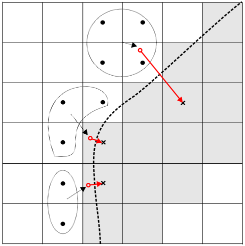
$$\tilde{\mathbf{x}} = \mathbf{x} + 2\varphi\mathbf{n}$$

Macro-velocity in ghost cells:
No-slip: $\mathbf{u}' = 2\mathbf{w} - \mathbf{u}$
Slip:

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



Level-set method for boundary embedding



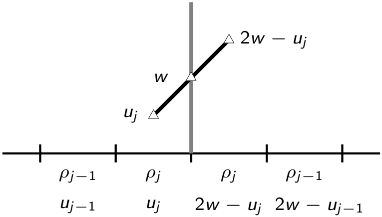
- Implicit boundary representation via distance function φ, normal $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$.
- Complex boundary moving with local velocity $\mathbf{w}$, treat interface as moving rigid wall.
- Construction of macro-values in embedded boundary cells by interpolation / extrapolation.
- Then use $f_\alpha^{eq}(\rho', \mathbf{u}')$ to construct distributions in embedded ghost cells.
- 2nd order improvements possible, cf. [Peng and Luo, 2008].

Interpolate / constant value extrapolate values at

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

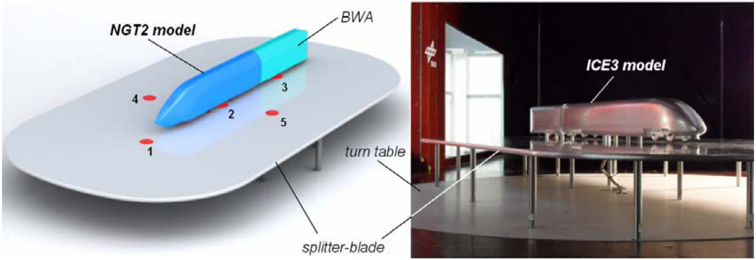
Macro-velocity in ghost cells:
No-slip: $\mathbf{u}' = 2\mathbf{w} - \mathbf{u}$
Slip:

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



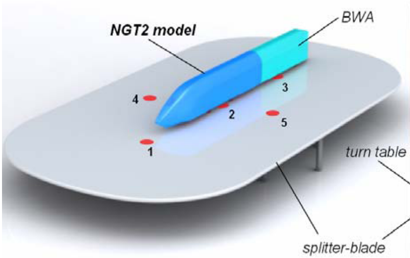
Side-wind investigation for a train model

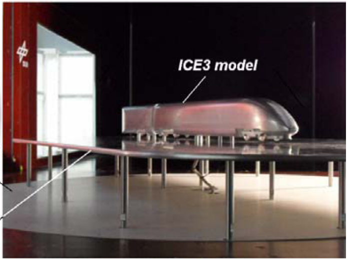
- 1:25 train model represented with 74,670 triangles (41,226 front body, 12,398 back body, 21,006 blade)



Side-wind investigation for a train model

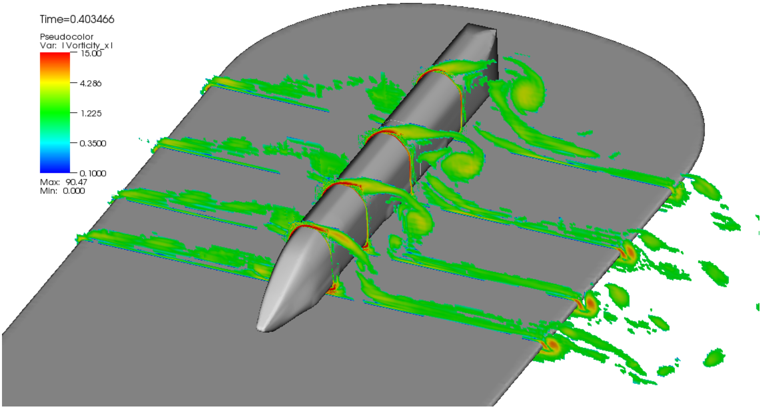
- 1:25 train model represented with 74,670 triangles (41,226 front body, 12,398 back body, 21,006 blade)
- Wind tunnel conditions: air at room temperature with 60.25 $\mathrm{m/s}$ ($M = 0.18$), $\mathrm{Re} = 450,000$
- Systematic side wind investigation with $0 \geq \beta \geq 30^o$ to obtain lift, drag and roll moment coefficients
- Instationary, turbulent flow conditions make replacing/supplementing experiments with simulations very challenging. Typical DLR problem and good real-world CFD benchmark.





Flow prediction, $\mathrm{Re} = 450,000$, $\beta = 30^o$

- Domain $10\,\mathrm{m} \times 2.4\,\mathrm{m} \times 1.6\,\mathrm{m}$
- Computation started in 3 steps. Full resolution after 5889 coarsest level steps or $t \geq 0.4\,\mathrm{s}$
- $\sim$ 1140 coarsest level steps in $24\,\mathrm{h}$ on 96 cores shown above. Overall cost $\sim 4600\,\mathrm{h}$ CPU.

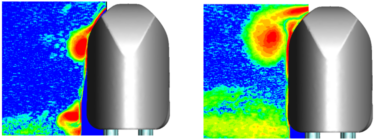


Vorticity vector component perpendicular to middle axis.

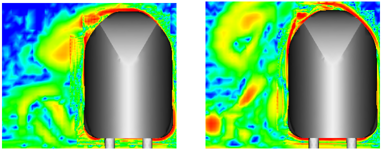
Flow prediction, $\mathrm{Re} = 450,000$, $\beta = 30^o$

- Domain $10\,\mathrm{m} \times 2.4\,\mathrm{m} \times 1.6\,\mathrm{m}$
- Computation started in 3 steps. Full resolution after 5889 coarsest level steps or $t \geq 0.4\,\mathrm{s}$
- $\sim$ 1140 coarsest level steps in $24\,\mathrm{h}$ on 96 cores shown above. Overall cost $\sim$ $4600\,\mathrm{h}$ CPU.

Experiment (time-averaged)



AMROC-LBM Simulation (instantaneous snapshots)

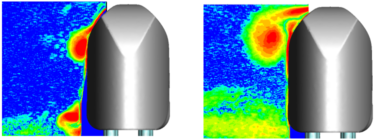


Vorticity component (seen from behind) in axial direction $80\,\mathrm{mm}$ and $290\,\mathrm{mm}$ away from model tip.

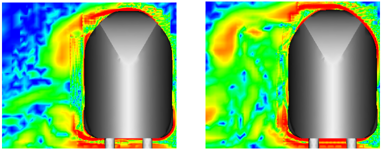
Flow prediction, $\mathrm{Re} = 450,000$, $\beta = 30^o$

- Domain $10\,\mathrm{m} \times 2.4\,\mathrm{m} \times 1.6\,\mathrm{m}$
- Computation started in 3 steps. Full resolution after 5889 coarsest level steps or $t \geq 0.4\,\mathrm{s}$
- $\sim$ 1140 coarsest level steps in $24\,\mathrm{h}$ on 96 cores shown above. Overall cost $\sim$ $4600\,\mathrm{h}$ CPU.

Experiment (time-averaged)



AMROC-LBM Simulation (instantaneous snapshots)

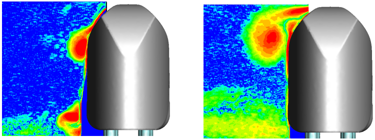


Vorticity component (seen from behind) in axial direction $80\,\mathrm{mm}$ and $290\,\mathrm{mm}$ away from model tip.

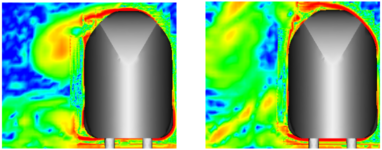
Flow prediction, $\mathrm{Re} = 450,000$, $\beta = 30^o$

- Domain $10\,\mathrm{m} \times 2.4\,\mathrm{m} \times 1.6\,\mathrm{m}$
- Computation started in 3 steps. Full resolution after 5889 coarsest level steps or $t \geq 0.4\,\mathrm{s}$
- $\sim$ 1140 coarsest level steps in $24\,\mathrm{h}$ on 96 cores shown above. Overall cost $\sim 4600\,\mathrm{h}$ CPU.

Experiment (time-averaged)



AMROC-LBM Simulation (instantaneous snapshots)

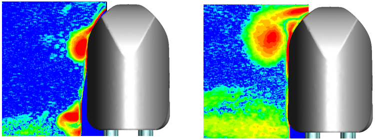


Vorticity component (seen from behind) in axial direction $80\,\mathrm{mm}$ and $290\,\mathrm{mm}$ away from model tip.

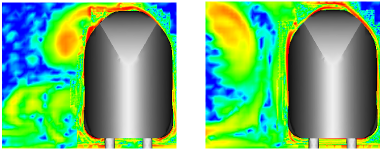
Flow prediction, $\mathrm{Re} = 450,000$, $\beta = 30^o$

- Domain $10\,\mathrm{m} \times 2.4\,\mathrm{m} \times 1.6\,\mathrm{m}$
- Computation started in 3 steps. Full resolution after 5889 coarsest level steps or $t \geq 0.4\,\mathrm{s}$
- $\sim$ 1140 coarsest level steps in $24\,\mathrm{h}$ on 96 cores shown above. Overall cost $\sim 4600\,\mathrm{h}$ CPU.

Experiment (time-averaged)



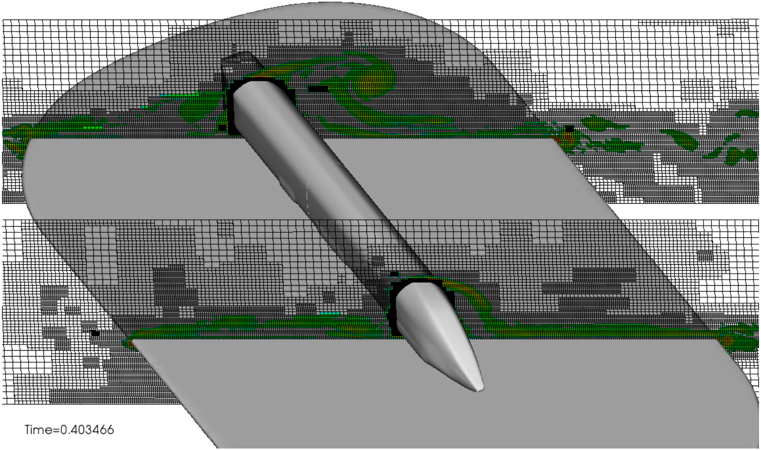
AMROC-LBM Simulation (instantaneous snapshots)



Vorticity component (seen from behind) in axial direction $80\,\mathrm{mm}$ and $290\,\mathrm{mm}$ away from model tip.

Dynamic mesh adaptation

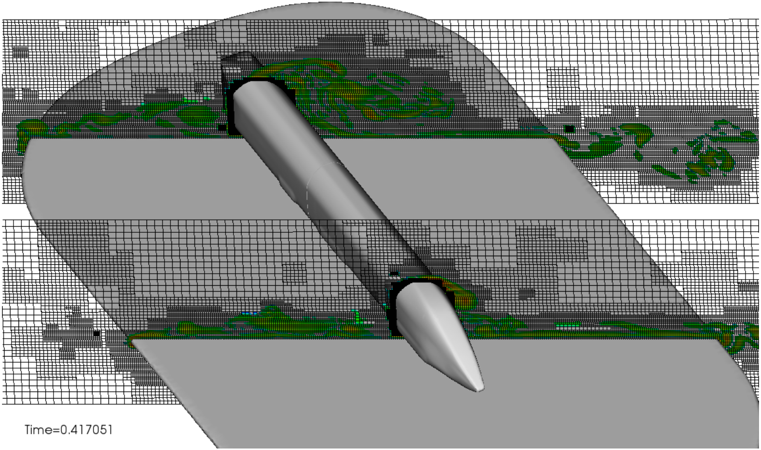
- Base mesh $500 \times 120 \times 80$ cells, refinement factors 2,2,4.
- Refinement based on error estimation of $|\mathbf{u}|$ up to second highest level.
- Highest level reserved to geometry refinement with $\Delta x = 1.25\,\mathrm{mm}$.



Dynamically adapting mesh. View in wind direction.

Dynamic mesh adaptation

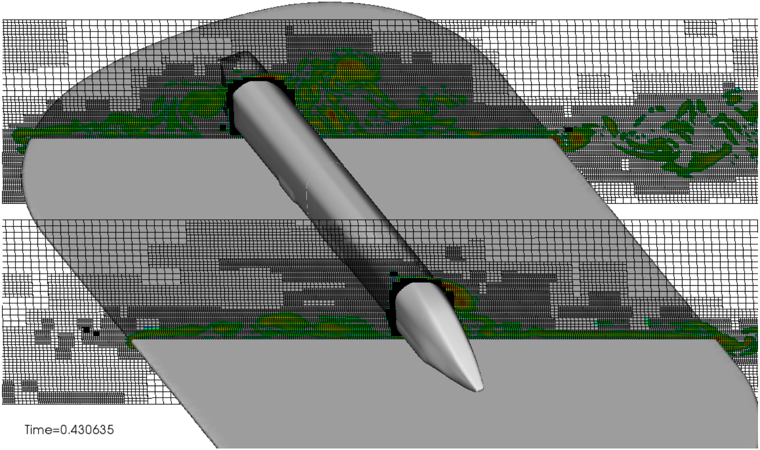
- Base mesh $500 \times 120 \times 80$ cells, refinement factors 2,2,4.
- Refinement based on error estimation of $|\mathbf{u}|$ up to second highest level.
- Highest level reserved to geometry refinement with $\Delta x = 1.25\,\mathrm{mm}$.



Dynamically adapting mesh. View in wind direction.

Dynamic mesh adaptation

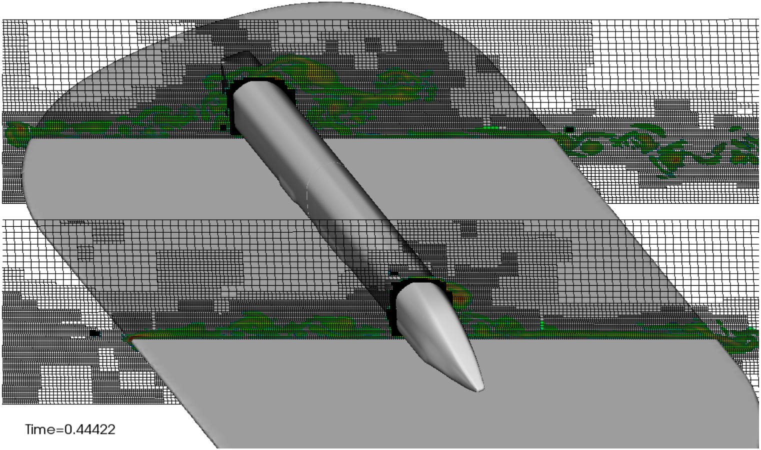
- Base mesh $500 \times 120 \times 80$ cells, refinement factors 2,2,4.
- Refinement based on error estimation of $|\mathbf{u}|$ up to second highest level.
- Highest level reserved to geometry refinement with $\Delta x = 1.25\,\mathrm{mm}$.



Dynamically adapting mesh. View in wind direction.

Dynamic mesh adaptation

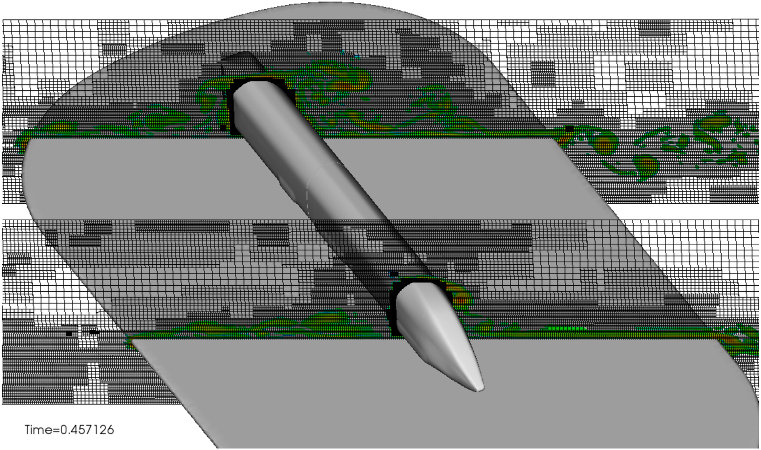
- Base mesh $500 \times 120 \times 80$ cells, refinement factors 2,2,4.
- Refinement based on error estimation of $|\mathbf{u}|$ up to second highest level.
- Highest level reserved to geometry refinement with $\Delta x = 1.25\,\mathrm{mm}$.



Dynamically adapting mesh. View in wind direction.

Dynamic mesh adaptation

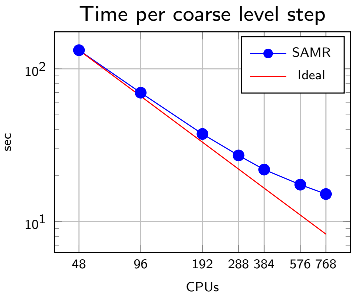
- Base mesh $500 \times 120 \times 80$ cells, refinement factors 2,2,4.
- Refinement based on error estimation of $|\mathbf{u}|$ up to second highest level.
- Highest level reserved to geometry refinement with $\Delta x = 1.25\,\mathrm{mm}$.



Dynamically adapting mesh. View in wind direction.

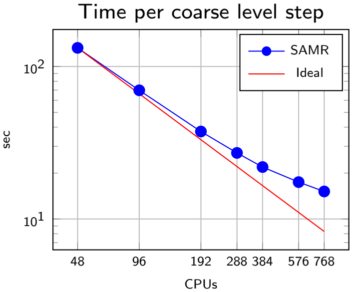
Strong scalability test

- Computation is restarted from disk checkpoint at $t = 0.526408\,\mathrm{s}$.
- 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).



Strong scalability test

- Computation is restarted from disk checkpoint at $t = 0.526408\,\mathrm{s}$.
- 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).



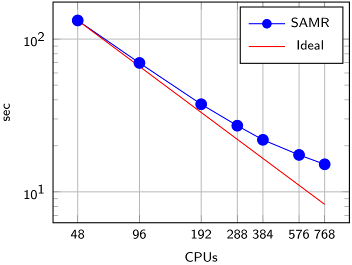
Time in % spent in main operations

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

Strong scalability test

- Computation is restarted from disk checkpoint at $t = 0.526408\,\mathrm{s}$.
- 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

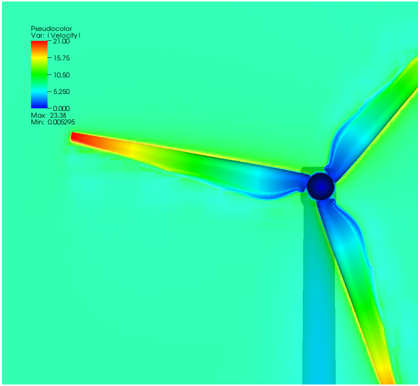


Time in % spent in main operations

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

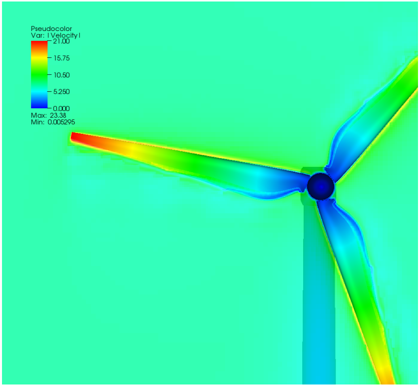
Simulation of a single turbine

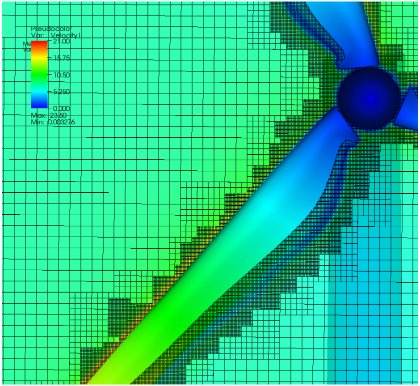
- Geometry from realistic Vestas V27 turbine. Rotor diameter $27\,\mathrm{m}$, tower height $\sim 35\,\mathrm{m}$. Ground considered.
- Prescribed motion of rotor with $15\,\mathrm{rpm}$. Inflow velocity $7\,\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- 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.



Simulation of a single turbine

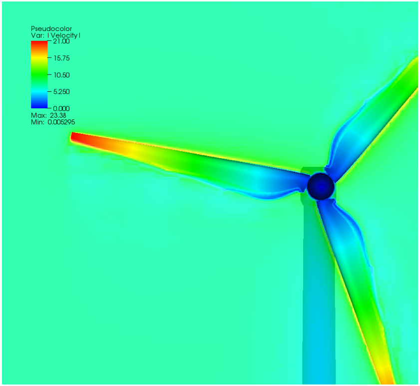
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh $400 \times 200 \times 200$ 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.

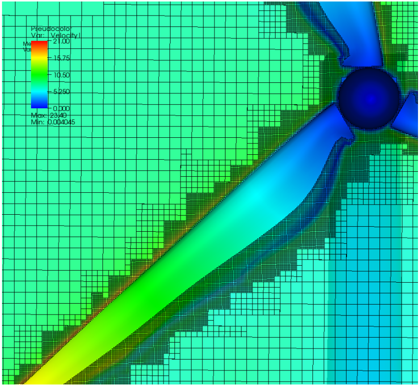




Simulation of a single turbine

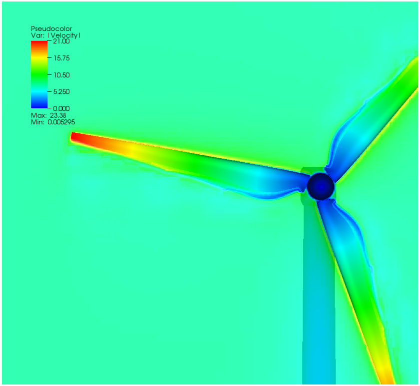
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh $400 \times 200 \times 200$ 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.

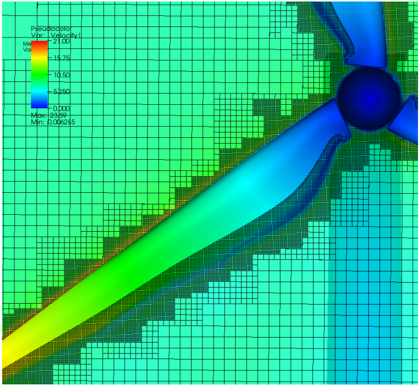




Simulation of a single turbine

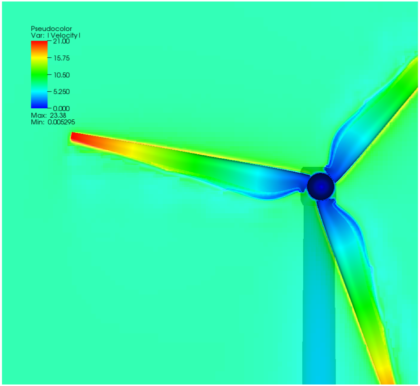
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 m/s.
- Simulation domain 200 m $\times$ 100 m $\times$ 100 m.
- Base mesh 400 $\times$ 200 $\times$ 200 cells with refinement factors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

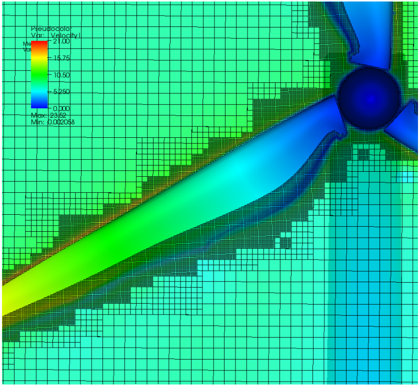




Simulation of a single turbine

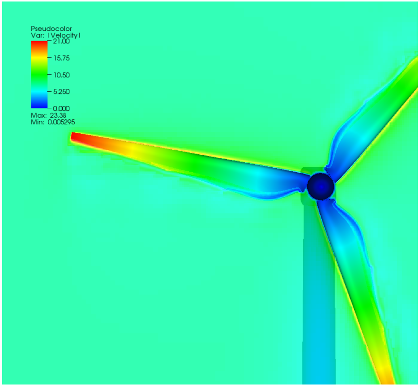
- Geometry from realistic Vestas V27 turbine. Rotor diameter $27\,\mathrm{m}$, tower height $\sim 35\,\mathrm{m}$. Ground considered.
- Prescribed motion of rotor with $15\,\mathrm{rpm}$. Inflow velocity $7\,\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- 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.

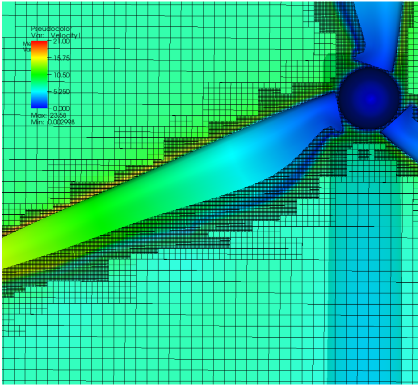




Simulation of a single turbine

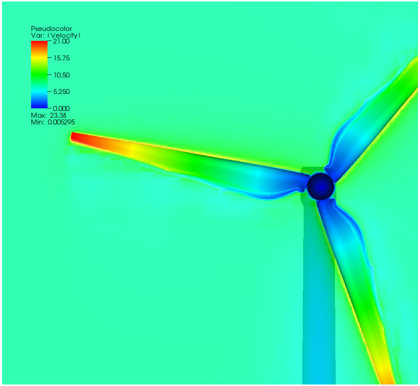
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 m/s.
- Simulation domain 200 m $\times$ 100 m $\times$ 100 m.
- Base mesh 400 $\times$ 200 $\times$ 200 cells with refinement factors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

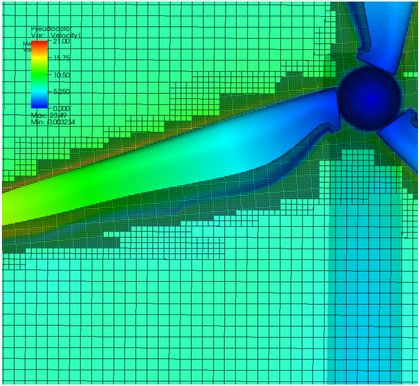




Simulation of a single turbine

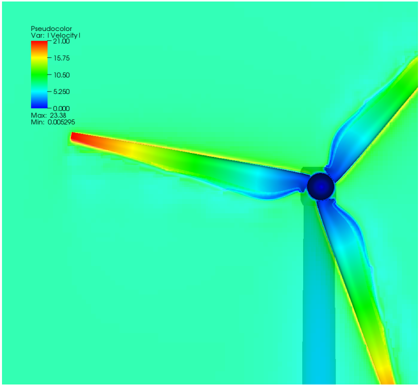
- Geometry from realistic Vestas V27 turbine. Rotor diameter $27\,\mathrm{m}$, tower height $\sim 35\,\mathrm{m}$. Ground considered.
- Prescribed motion of rotor with $15\,\mathrm{rpm}$. Inflow velocity $7\,\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- 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.

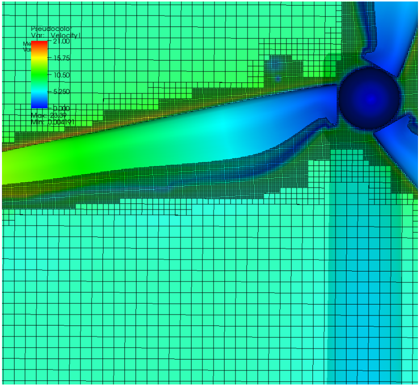




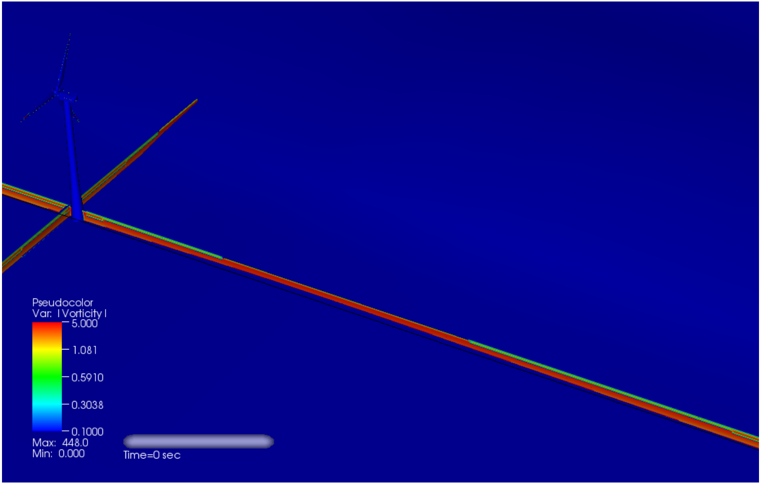
Simulation of a single turbine

- Geometry from realistic Vestas V27 turbine. Rotor diameter $27\,\mathrm{m}$, tower height $\sim 35\,\mathrm{m}$. Ground considered.
- Prescribed motion of rotor with $15\,\mathrm{rpm}$. Inflow velocity $7\,\mathrm{m/s}$.
- 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.



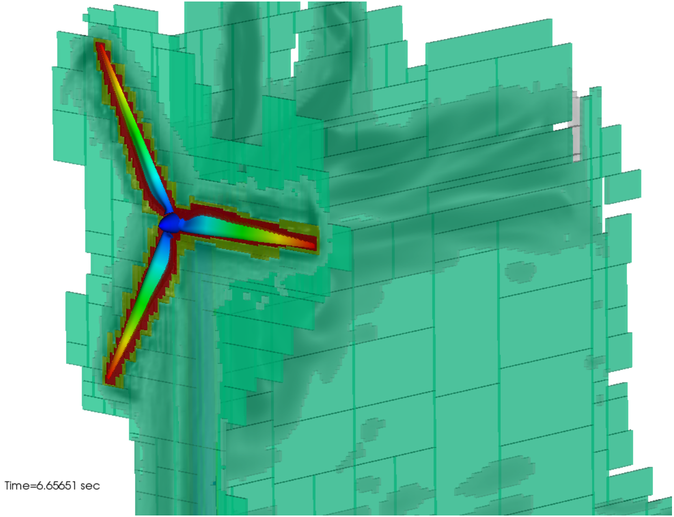


Wake field behind turbine



- Simulation on 96 cores Intel Xeon-Westmere. $\sim 10,400\,\mathrm{h}$ CPU.
- Error estimation in $|\mathbf{u}|$ refines wake up to level 1 ($\Delta x = 25\,\mathrm{cm}$).
- Rotation starts at $t = 4\,\mathrm{s}$.

Adaptive refinement

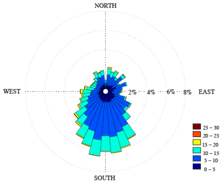


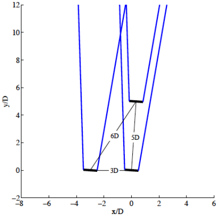
Dynamic evolution of refinement blocks (indicated by color).

Simulation of the SWIFT array

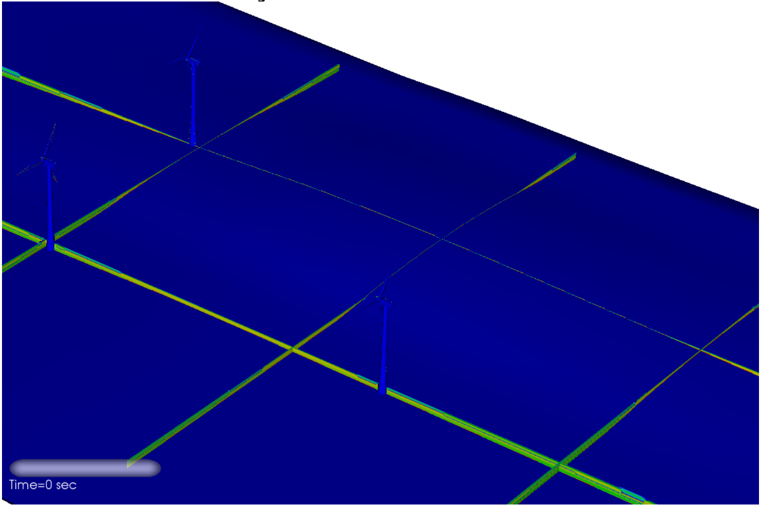
- Three Vestas V27 turbines. 225 kW power generation at wind speeds 14 to 25 $\mathrm{m/s}$ (then cut-off).
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$ (power generation 52.5 kW).

- Simulation domain 488 $\mathrm{m} \times 240\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh 448 $\times$ 240 $\times$ 100 cells with refinement factors 2,2,2. Resolution of rotor and tower $\Delta x = 12.5\,\mathrm{cm}$.
- 47,120 highest level iterations to $t_e = 40\,\mathrm{s}$ computed.





Wakes in SWIFT array



- Simulation on 288 cores Intel Xeon-Westmere. $\sim 140,000\,\mathrm{h}$ CPU.
- Refinement of wake up to level 2 ($\Delta x = 25\,\mathrm{cm}$).
- Rotation starts at $t = 4\,\mathrm{s}$, full refinement at $t = 8\,\mathrm{s}$ to avoid refining initial acoustic waves.

Conclusions

- Developed and demonstrated a first version of a block-based dynamically adaptive LBM for real-world CFD with moving boundaries.
- Reuse of templatized AMROC classes from previous finite volume methods already provides robust real-world capabilities.

Conclusions

- Developed and demonstrated a first version of a block-based dynamically adaptive LBM for real-world CFD with moving boundaries.
- Reuse of templatized AMROC classes from previous finite volume methods already provides robust real-world capabilities.
- Improve refinement criteria (e.g., vorticity-based) to capture wake fields reliably.

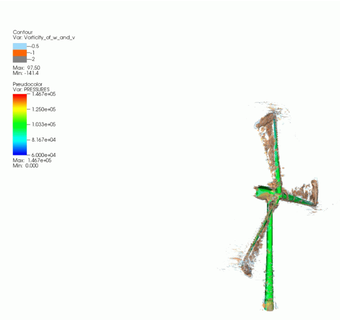
Conclusions

- Developed and demonstrated a first version of a block-based dynamically adaptive LBM for real-world CFD with moving boundaries.
- Reuse of templatized AMROC classes from previous finite volume methods already provides robust real-world capabilities.
- Improve refinement criteria (e.g., vorticity-based) to capture wake fields reliably.
- Performance for moderate core count is reasonable, some improvements for larger core count still desirable.
  - Reduce communication width to a single halo layer.
  - Consider workload due to embedded boundary method in partitioning algorithm.
  - Allow other than rigorous domain decomposition.

Conclusions

- Developed and demonstrated a first version of a block-based dynamically adaptive LBM for real-world CFD with moving boundaries.
- Reuse of templatized AMROC classes from previous finite volume methods already provides robust real-world capabilities.
- Improve refinement criteria (e.g., vorticity-based) to capture wake fields reliably.
- Performance for moderate core count is reasonable, some improvements for larger core count still desirable.
  - Reduce communication width to a single halo layer.
  - Consider workload due to embedded boundary method in partitioning algorithm.
  - Allow other than rigorous domain decomposition.
- Use system for understanding turbine-turbine interactions.
- Realistic turbine model with dynamic pitch angle, nacelle rotation, etc. under development.

NREL 5 MW turbine



References I

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

[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.

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

[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.

References II

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

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

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

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

[Mauch, 2000] Mauch, S. (2000). A fast algorithm for computing the closest point and distance transform. *SIAM J. Scientific Comput.*

[Peng and Luo, 2008] Peng, Y. and Luo, L.-S. (2008). A comparative study of immersed-boundary and interpolated bounce-back methods in lbe. *Prog. Comp. Fluid Dynamics*, 8(1-4):156–167.