Adaptive Cartesian lattice Boltzmann methods in the AMROC framework and comparison with a non-Cartesian approach

Ralf Deiterding, Christos Gkoudesnes, Juan Antonio Reyes Barraza

Aerodynamics and Flight Mechanics Research Group
University of Southampton
Highfield Campus, Southampton SO17 1BJ, UK
E-mail: r.deiterding@soton.ac.uk

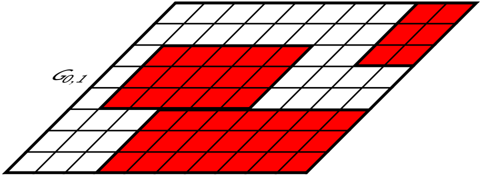
July 16, 2019

Outline

Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

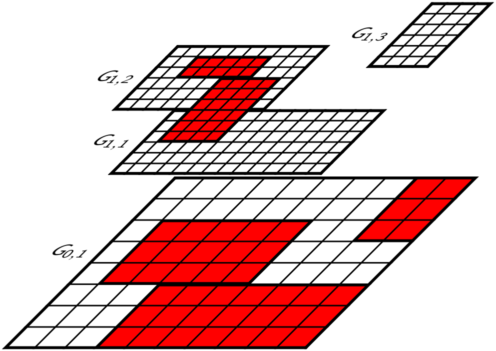
- Refined blocks overlay coarser ones



Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

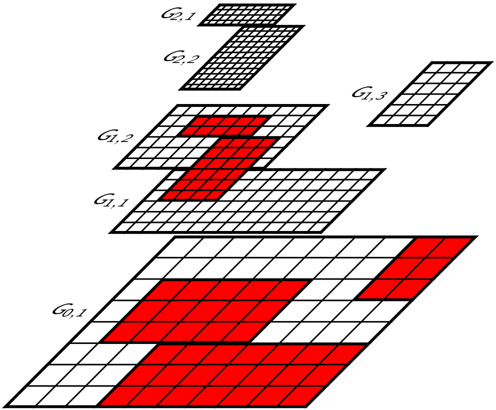
- Refined blocks overlay coarser ones



Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

- Refined blocks overlay coarser ones



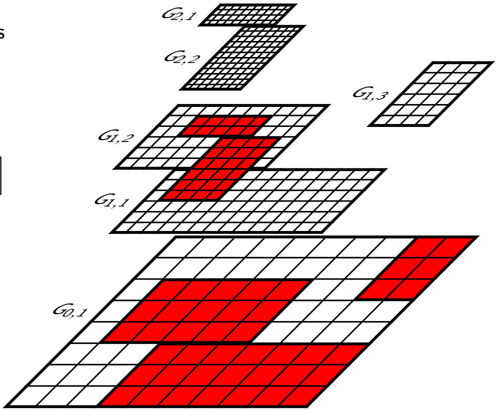
Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

- 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

$$\mathbf{Q}_{jk}^{n+1} = \mathbf{Q}_{jk}^{n} - \frac{\Delta t}{\Delta x}\left[\mathbf{F}_{j+\frac{1}{2},k} - \mathbf{F}_{j-\frac{1}{2},k}\right] - \frac{\Delta t}{\Delta y}\left[\mathbf{G}_{j,k+\frac{1}{2}} - \mathbf{G}_{j,k-\frac{1}{2}}\right]$$

only for single patch necessary



Block-structured adaptive mesh refinement (SAMR)

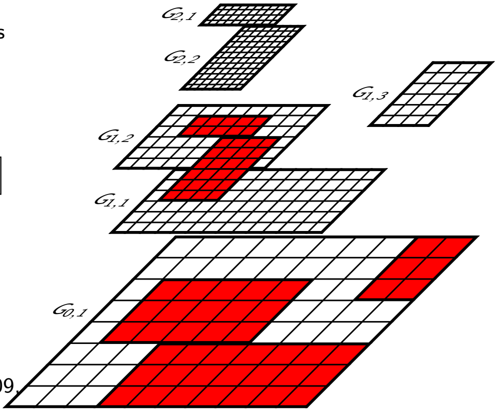
For simplicity $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

- 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

$$\mathbf{Q}_{jk}^{n+1} = \mathbf{Q}_{jk}^{n} - \frac{\Delta t}{\Delta x}\left[\mathbf{F}_{j+\frac{1}{2},k} - \mathbf{F}_{j-\frac{1}{2},k}\right] - \frac{\Delta t}{\Delta y}\left[\mathbf{G}_{j,k+\frac{1}{2}} - \mathbf{G}_{j,k-\frac{1}{2}}\right]$$

only for single patch necessary

- \+ Efficient cache-reuse / vectorization possible
- \- Cluster-algorithm necessary
- Papers: [Deiterding, 2011, Deiterding et al., 2009, Deiterding et al., 2007]

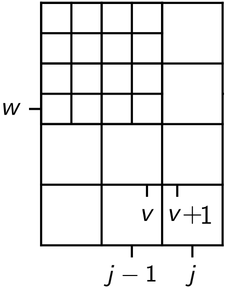


Conservative flux correction

Example: Cell j, k

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

Correction pass:



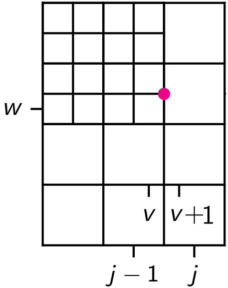
Conservative flux correction

Example: Cell j, k

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

Correction pass:

1. $\delta \mathbf{F}^{l+1}_{j-\frac{1}{2},k} := -\mathbf{F}^l_{j-\frac{1}{2},k}$



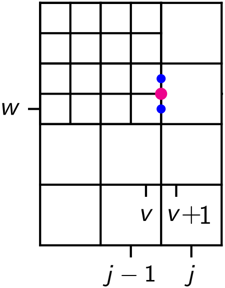
Conservative flux correction

Example: Cell $j,\ k$

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

Correction pass:

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



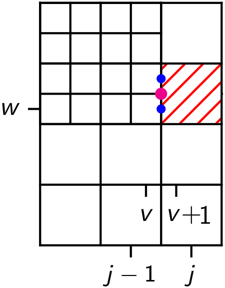
Conservative flux correction

Example: Cell j, k

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

Correction pass:

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



Approximation of Boltzmann equation

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

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

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

Approximation of Boltzmann equation

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

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

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

Equation is approximated with a splitting approach.

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

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

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

Discrete velocities:

$\mathbf{e}_0 = (0,0), \mathbf{e}_1 = (1,0)c, \mathbf{e}_2 = (-1,0)c, \mathbf{e}_3 = (0,1)c, \mathbf{e}_4 = (1,1)c, \ldots$

$c = \dfrac{\Delta x}{\Delta t}$, Physical speed of sound: $c_s = \dfrac{c}{\sqrt{3}}$

Approximation of Boltzmann equation

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

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

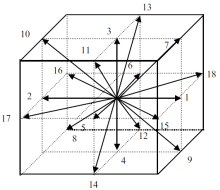
- $\mathrm{Kn} = l_f / L \ll 1$, where l_f is replaced with Δx
- Weak compressibilty and small Mach number assumed
- Assume a simplified phase space

Equation is approximated with a splitting approach.

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

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

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



Discrete velocities:

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

Approximation of equilibrium state

2.) Collision step solves $\partial_t f_\alpha = \omega(f_\alpha^{eq} - f_\alpha) + F_\alpha$
Operator $\mathcal{C}$:

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

with $F_\alpha = 3\rho t_\alpha \mathbf{e}_\alpha \mathbf{F}/c^2$

Approximation of equilibrium state

2.) Collision step solves $\partial_t f_\alpha = \omega(f_\alpha^{eq} - f_\alpha) + F_\alpha$
Operator $\mathcal{C}$:

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

with $F_\alpha = 3\rho t_\alpha \mathbf{e}_\alpha \mathbf{F}/c^2$ and equilibrium function

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

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

$\delta p = \sum_\alpha f_\alpha^{eq} c_s^2 = \rho c_s^2$. Dev. stress $\Sigma_{ij} = \left(1 - \frac{\omega_L \Delta t}{2}\right) \sum_\alpha e_{\alpha i} e_{\alpha j} (f_\alpha^{eq} - f_\alpha)$

Approximation of equilibrium state

2.) Collision step solves $\partial_t f_\alpha = \omega(f_\alpha^{eq} - f_\alpha) + F_\alpha$
Operator $\mathcal{C}$:

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

with $F_\alpha = 3\rho t_\alpha \mathbf{e}_\alpha \mathbf{F}/c^2$ and equilibrium function

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

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

$\delta p = \sum_\alpha f_\alpha^{eq} c_s^2 = \rho c_s^2$. Dev. stress $\Sigma_{ij} = \left(1 - \frac{\omega_L \Delta t}{2}\right) \sum_\alpha e_{\alpha i} e_{\alpha j} (f_\alpha^{eq} - f_\alpha)$

Approximation of equilibrium state

2.) Collision step solves $\partial_t f_\alpha = \omega(f_\alpha^{eq} - f_\alpha) + F_\alpha$
Operator $\mathcal{C}$:

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

with $F_\alpha = 3\rho t_\alpha \mathbf{e}_\alpha \mathbf{F}/c^2$ and equilibrium function

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

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

$\delta p = \sum_\alpha f_\alpha^{eq} c_s^2 = \rho c_s^2$. Dev. stress $\Sigma_{ij} = \left(1 - \frac{\omega_L \Delta t}{2}\right) \sum_\alpha e_{\alpha i} e_{\alpha j} (f_\alpha^{eq} - f_\alpha)$
Using the third-order equilibrium function

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

allows higher flow velocities (up to $M \approx 0.3 - 0.4$ vs. $M \approx 0.15 - 0.2$).

Approximation of equilibrium state

2.) Collision step solves $\partial_t f_\alpha = \omega(f_\alpha^{eq} - f_\alpha) + F_\alpha$
Operator $\mathcal{C}$:

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

with $F_\alpha = 3\rho t_\alpha \mathbf{e}_\alpha \mathbf{F}/c^2$ and equilibrium function

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

with $t_\alpha = \frac{1}{9}\left\{3, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \right\}$
$\delta p = \sum_\alpha f_\alpha^{eq} c_s^2 = \rho c_s^2$. Dev. stress $\Sigma_{ij} = \left(1 - \frac{\omega_L \Delta t}{2}\right) \sum_\alpha e_{\alpha i} e_{\alpha j} (f_\alpha^{eq} - f_\alpha)$
Using the third-order equilibrium function

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

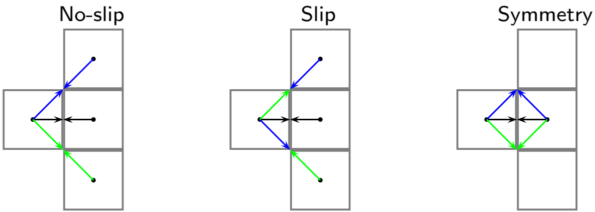
allows higher flow velocities (up to $M \approx 0.3 - 0.4$ vs. $M \approx 0.15 - 0.2$).
A Chapman-Enskog expansion shows

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

Initial and boundary conditions

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

Boundary conditions (applied before streaming step)

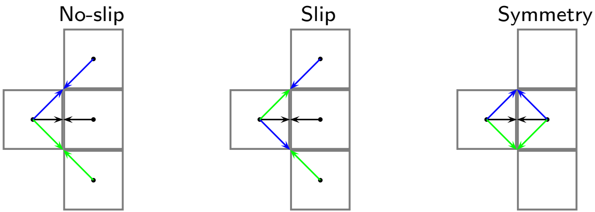


- Outlet basically copies all distributions (zero gradient)
- Inlet and pressure boundary conditions use f_α^{eq}

Initial and boundary conditions

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

Boundary conditions (applied before streaming step)



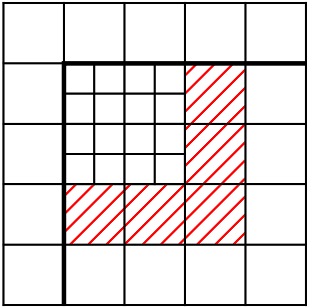
- Outlet basically copies all distributions (zero gradient)
- Inlet and pressure boundary conditions use f_α^{eq}

Complex geometry:

- Use level set method as before to construct macro-values in embedded boundary cells by interpolation / extrapolation [Deiterding, 2011].
- Distance function φ, normal $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$. Triangulated meshes use CPT algorithm [Mauch, 2003].
- 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

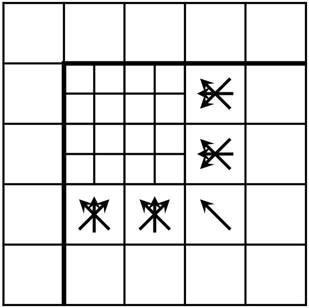
Adaptive LBM

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



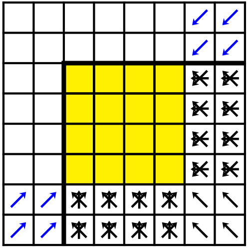
Adaptive LBM

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

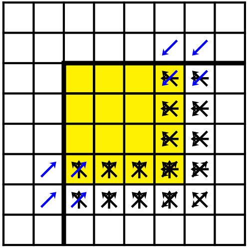
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.



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

Adaptive LBM

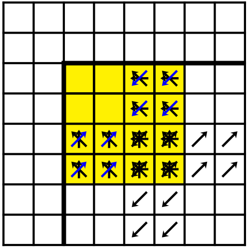
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



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

Adaptive LBM

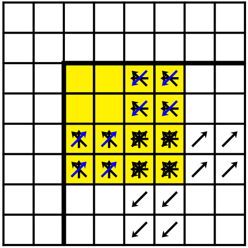
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



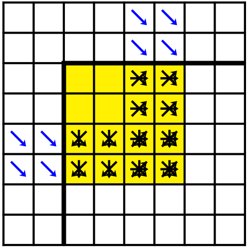
$\tilde{f}_{\alpha,in}^{f,n+1/2}$

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.



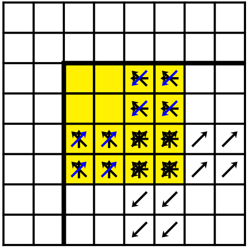
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



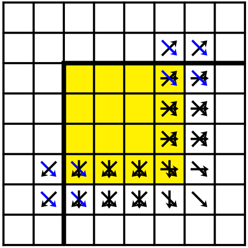
$f_{\alpha,out}^{f,n}$

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.



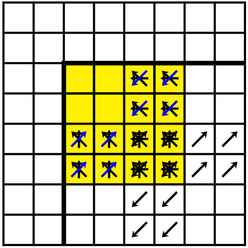
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



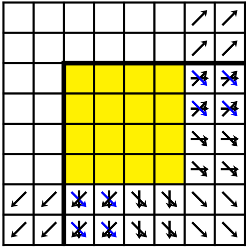
$\tilde{f}_{\alpha,out}^{f,n}$

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.



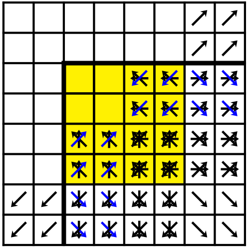
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



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

Adaptive LBM

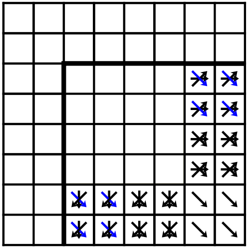
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

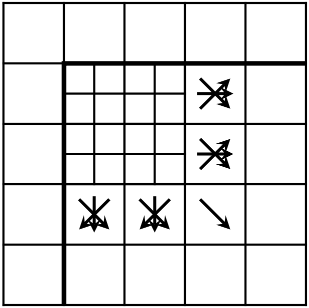


$\tilde{f}_{\alpha,out}^{f,n+1/2}$, $\tilde{f}_{\alpha,out}^{f,n}$

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

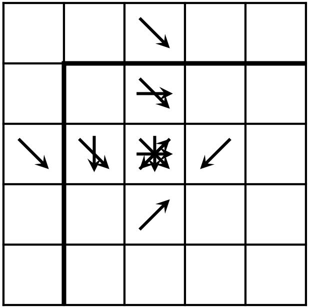
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

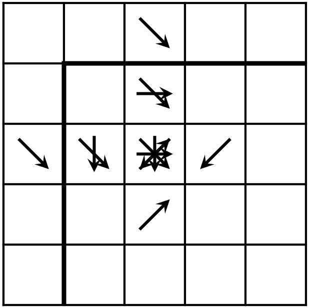
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

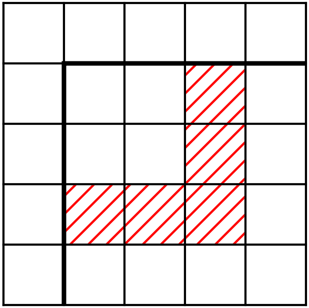
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

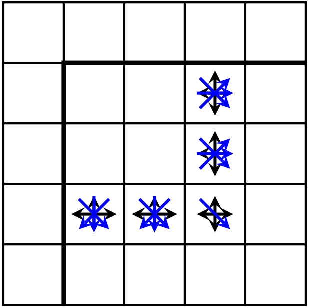
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

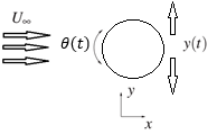


5. Average $\tilde{f}_{\alpha,out}^{f,n+1/2}$ (inner halo layer), $\tilde{f}_{\alpha,out}^{f,n}$ (outer halo layer) to obtain $\tilde{f}_{\alpha,out}^{C,n}$.
6. Revert transport into halos: $\bar{f}_{\alpha,out}^{C,n} := \mathcal{T}^{-1}(\tilde{f}_{\alpha,out}^{C,n})$
7. Parallel synchronization of $f_\alpha^{C,n},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

Oscillating cylinder – Setup

Motion imposed on cylinder



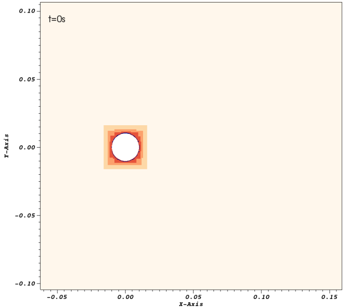
Case	A_t	$f_t = f_\theta$	V_R	U_∞	Re
1a	$D/4$	0.6	0.5	0.0606	1322
1b	$D/2$	0.6	1.0	0.0606	1322
2a	$D/4$	3.0	0.5	0.3030	6310
2b	$D/2$	3.0	1.0	0.3030	6310

$$y(t) = A_t \sin(2\pi f_t t), \qquad \theta(t) = A_\theta \sin(2\pi f_\theta t)$$

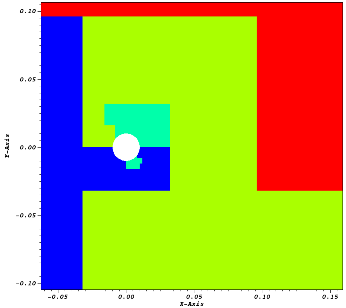
- Setup follows [Nazarinia et al., 2012], cf. [Laloglu and Deiterding, 2017]. Here $A_\theta = 1$ for all cases.
- Natural frequency of cylinder $f_N \approx 0.6154\,\mathrm{s}^{-1}$.
- Strouhal number $\mathrm{St}_t = f_t D/U_\infty \approx 0.198$ for all cases.
- Chosen here $D = 20\,\mathrm{mm}$
- Fluid is water with $c_s = 1482\,\mathrm{m/s}$, $\nu = 9.167 \cdot 10^{-7}\,\mathrm{m}^2/\mathrm{s}$, $\rho = 1016\,\mathrm{kg/m}^3$
- Constant coefficient model deactivated for Case 1, active for Case 2 with $C_{sm} = 0.2$

Case 1b, $V_R = 1$, $\mathrm{Re} = 1322$

Mesh refinement



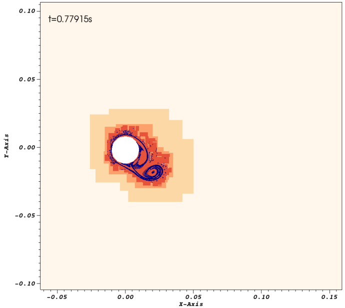
Distribution to 4 processors



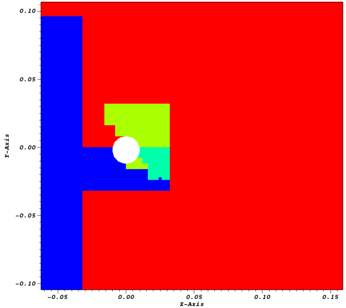
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

Case 1b, $V_R = 1$, $\mathrm{Re} = 1322$

Mesh refinement



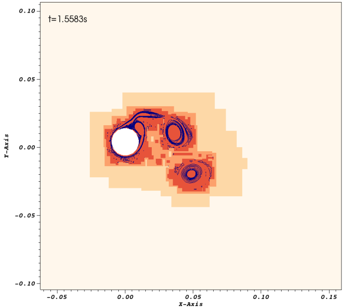
Distribution to 4 processors



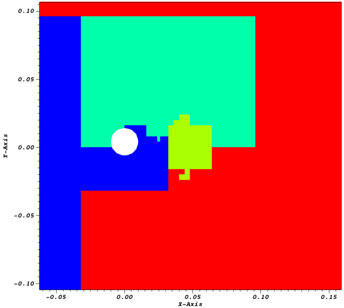
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

Case 1b, $V_R = 1$, $\mathrm{Re} = 1322$

Mesh refinement



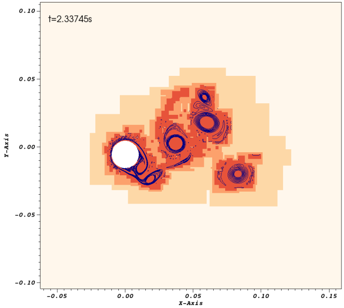
Distribution to 4 processors



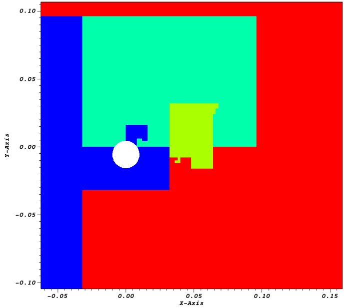
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

Case 1b, $V_R = 1$, $\mathrm{Re} = 1322$

Mesh refinement



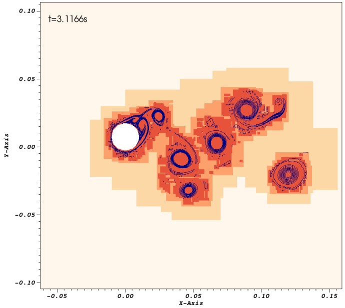
Distribution to 4 processors



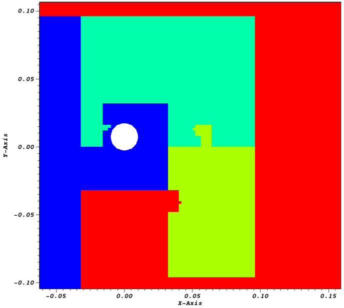
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

Case 1b, $V_R = 1$, $\mathrm{Re} = 1322$

Mesh refinement



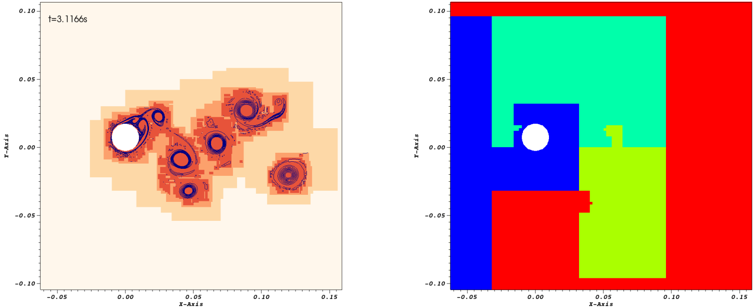
Distribution to 4 processors



- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

Case 1b, $V_R = 1$, $\mathrm{Re} = 1322$

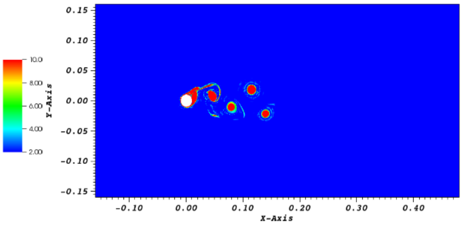
Mesh refinement | Distribution to 4 processors



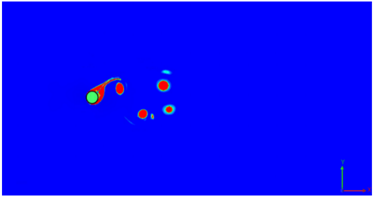
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$
- Basically identical setup in commercial code XFlow for comparison

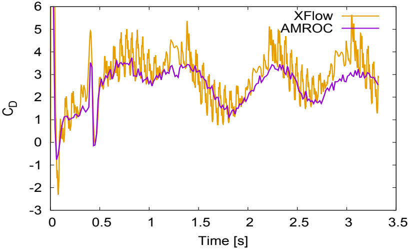
Oscillating cylinder, $V_R = 1$, $f_t = f_\theta = 0.6$, $\mathrm{Re} = 1322$

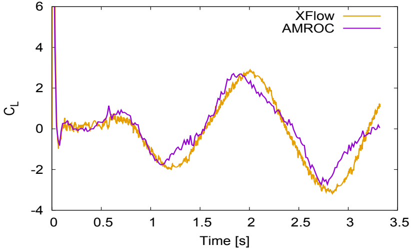
AMROC



XFlow

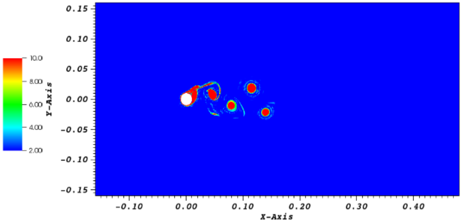




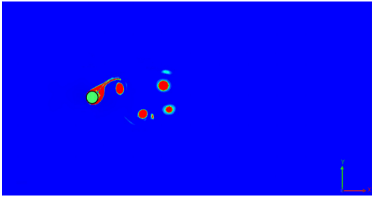


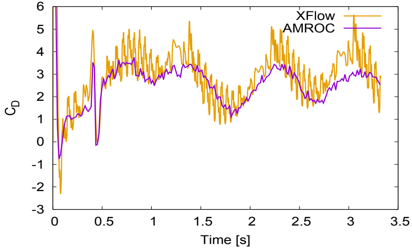
Oscillating cylinder, $V_R = 1,\ f_t = f_\theta = 0.6,\ \mathrm{Re} = 1322$

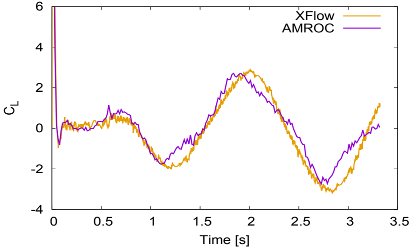
AMROC



XFlow



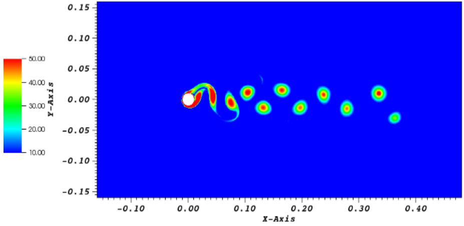




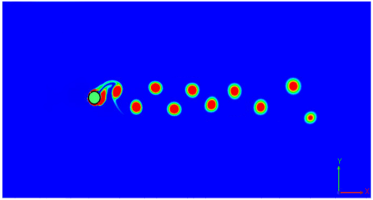
- Increase of rotational velocity leads to formation of a vortex pair plus single vortex. Drag and lift amplitude roughly doubled.
- Laminar results in good agreement with experiments of [Nazarinia et al., 2012].

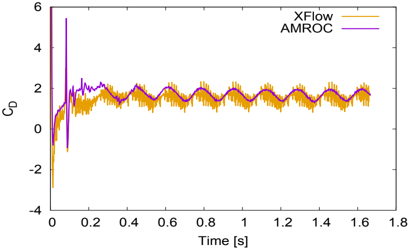
Oscillating cylinder, $V_R = 0.5,\ f_t = f_\theta = 3,\ \mathrm{Re} = 6310$

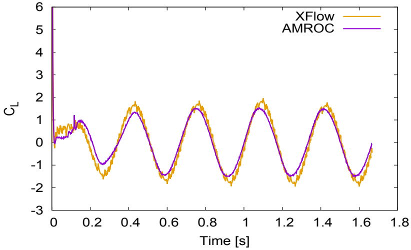
AMROC



XFlow

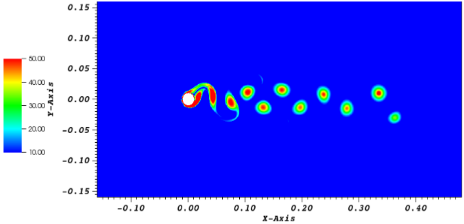




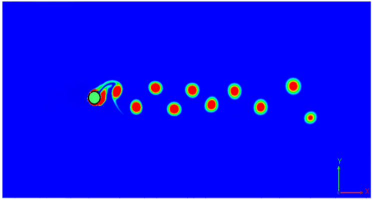


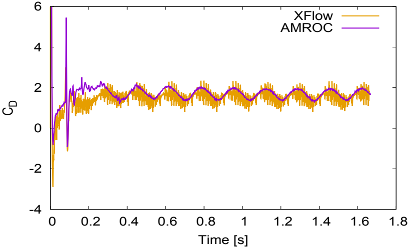
Oscillating cylinder, $V_R = 0.5,\ f_t = f_\theta = 3,\ \mathrm{Re} = 6310$

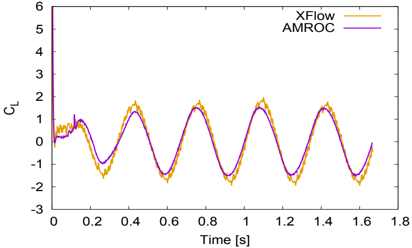
AMROC



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

Turbulence modeling

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

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

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

Turbulence modeling

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

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

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

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

Turbulence modeling

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

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

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

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

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

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

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

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

Turbulence modeling

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

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

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

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

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

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

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

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

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

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

Further LES models

Dynamic Smagorinsky model (DSMA)

$$C_{sm}(\mathbf{x},t)^2 = -\frac{1}{2}\frac{\langle L_{ij}M_{ij}\rangle}{\langle M_{ij}M_{ij}\rangle}$$

$$L_{ij} = T_{ij} - \widehat{\tau}_{ij} = \widehat{\overline{u}_i\overline{u}_j} - \hat{\overline{u}}_i\hat{\overline{u}}_j \qquad M_{ij} = \widehat{\Delta x}^2|\hat{\overline{\mathbf{S}}}|\hat{\overline{S}}_{ij} - \Delta x^2\widehat{|\overline{\mathbf{S}}|\overline{S}_{ij}}$$

No van Driest damping implemented yet!

Further LES models

Dynamic Smagorinsky model (DSMA)

$$C_{sm}(\mathbf{x},t)^2 = -\frac{1}{2}\frac{\langle L_{ij}M_{ij}\rangle}{\langle M_{ij}M_{ij}\rangle}$$

$$L_{ij} = T_{ij} - \widehat{\tau}_{ij} = \widehat{\overline{u}_i\overline{u}_j} - \hat{\overline{u}}_i\hat{\overline{u}}_j \qquad M_{ij} = \widehat{\Delta x}^2|\hat{\overline{\mathbf{S}}}|\hat{\overline{S}}_{ij} - \Delta x^2 \widehat{|\overline{\mathbf{S}}|\overline{S}_{ij}}$$

No van Driest damping implemented yet!

Wall-Adapting Local Eddy-viscosity model (WALE)

$$\nu_t = (C_w\Delta x)^2 OP_{WALE}, \quad \text{where } C_w = 0.5$$

WALE turbulence time-scale

$$OP_{WALE} = \frac{(\mathcal{J}_{ij}\mathcal{J}_{ij})^{\frac{3}{2}}}{(\overline{S}_{ij}\overline{S}_{ij})^{\frac{5}{2}} + (\mathcal{J}_{ij}\mathcal{J}_{ij})^{\frac{5}{4}}}$$

$$\mathcal{J}_{ij} = \overline{S}_{ik}\overline{S}_{kj} + \overline{\Omega}_{ik}\overline{\Omega}_{kj} - \frac{1}{3}\delta_{ij}(\overline{S}_{mn}\overline{S}_{mn} - \overline{\Omega}_{mn}\overline{\Omega}_{mn})$$

Effective relaxation time (see previous slide): $\tau_L^\star = \dfrac{(\nu+\nu_t) + \Delta t c_s^2/2}{c_s^2}$

Forced homogeneous isotropic turbulence

- Fourier representation
- Periodic boundaries, uniform mesh
- Use of external forcing term, i.e., result independent of initial conditions

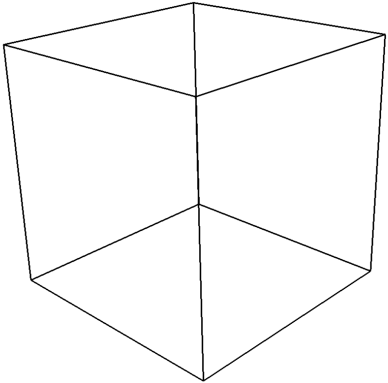
Forcing:

$$F_x = 2A\left(\frac{\kappa_y \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_y = -A\left(\frac{\kappa_x \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_z = -A\left(\frac{\kappa_x \kappa_y}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

Iso-surface $||\mathbf{u}||/\langle u_{rms}\rangle = 2$



with phase

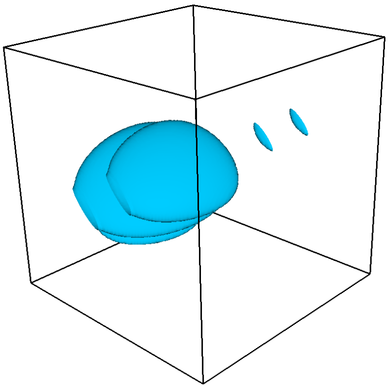
$$G(\kappa_x, \kappa_y, \kappa_z) = \sin\left(\frac{2\pi x}{L}\kappa_x + \frac{2\pi y}{L}\kappa_y + \frac{2\pi z}{L}\kappa_z + \phi\right) \text{ for } (0 < \kappa_i \leq 2) \text{ and } \phi$$

being a random phase value.

Forced homogeneous isotropic turbulence

- Fourier representation
- Periodic boundaries, uniform mesh
- Use of external forcing term, i.e., result independent of initial conditions

Iso-surface $||\mathbf{u}||/\langle u_{rms}\rangle = 2$



Forcing:

$$F_x = 2A\left(\frac{\kappa_y\kappa_z}{|\kappa|^2}\right)G(\kappa_x,\kappa_y,\kappa_z)$$

$$F_y = -A\left(\frac{\kappa_x\kappa_z}{|\kappa|^2}\right)G(\kappa_x,\kappa_y,\kappa_z)$$

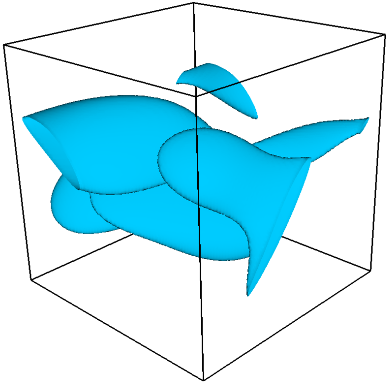
$$F_z = -A\left(\frac{\kappa_x\kappa_y}{|\kappa|^2}\right)G(\kappa_x,\kappa_y,\kappa_z)$$

with phase

$G(\kappa_x,\kappa_y,\kappa_z) = \sin\left(\frac{2\pi x}{L}\kappa_x + \frac{2\pi y}{L}\kappa_y + \frac{2\pi z}{L}\kappa_z + \phi\right)$ for $(0 < \kappa_i \le 2)$ and ϕ being a random phase value.

Forced homogeneous isotropic turbulence

Iso-surface $||\mathbf{u}||/\langle u_{rms}\rangle = 2$



- Fourier representation
- Periodic boundaries, uniform mesh
- Use of external forcing term, i.e., result independent of initial conditions

Forcing:

$$F_x = 2A\left(\frac{\kappa_y\kappa_z}{|\kappa|^2}\right)G(\kappa_x,\kappa_y,\kappa_z)$$

$$F_y = -A\left(\frac{\kappa_x\kappa_z}{|\kappa|^2}\right)G(\kappa_x,\kappa_y,\kappa_z)$$

$$F_z = -A\left(\frac{\kappa_x\kappa_y}{|\kappa|^2}\right)G(\kappa_x,\kappa_y,\kappa_z)$$

with phase

$G(\kappa_x,\kappa_y,\kappa_z) = \sin\left(\frac{2\pi x}{L}\kappa_x + \frac{2\pi y}{L}\kappa_y + \frac{2\pi z}{L}\kappa_z + \phi\right)$ for $(0 < \kappa_i \leq 2)$ and ϕ being a random phase value.

Forced homogeneous isotropic turbulence

- Fourier representation
- Periodic boundaries, uniform mesh
- Use of external forcing term, i.e., result independent of initial conditions

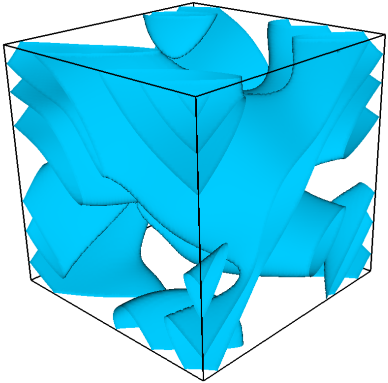
Forcing:

$$F_x = 2A\left(\frac{\kappa_y \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_y = -A\left(\frac{\kappa_x \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_z = -A\left(\frac{\kappa_x \kappa_y}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

Iso-surface $||\mathbf{u}||/\langle u_{rms}\rangle = 2$



with phase

$G(\kappa_x, \kappa_y, \kappa_z) = \sin\left(\frac{2\pi x}{L}\kappa_x + \frac{2\pi y}{L}\kappa_y + \frac{2\pi z}{L}\kappa_z + \phi\right)$ for $(0 < \kappa_i \leq 2)$ and ϕ being a random phase value.

Forced homogeneous isotropic turbulence

- Fourier representation
- Periodic boundaries, uniform mesh
- Use of external forcing term, i.e., result independent of initial conditions

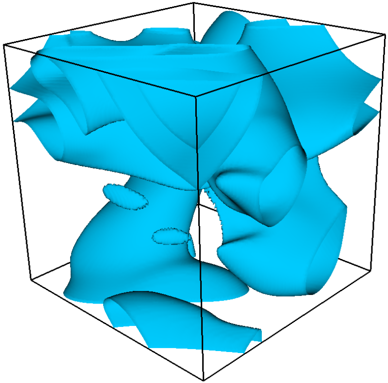
Forcing:

$$F_x = 2A\left(\frac{\kappa_y \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_y = -A\left(\frac{\kappa_x \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_z = -A\left(\frac{\kappa_x \kappa_y}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

Iso-surface $||\mathbf{u}||/\langle u_{rms}\rangle = 2$



with phase

$G(\kappa_x, \kappa_y, \kappa_z) = \sin\left(\frac{2\pi x}{L}\kappa_x + \frac{2\pi y}{L}\kappa_y + \frac{2\pi z}{L}\kappa_z + \phi\right)$ for $(0 < \kappa_i \leq 2)$ and ϕ being a random phase value.

Forced homogeneous isotropic turbulence

- Fourier representation
- Periodic boundaries, uniform mesh
- Use of external forcing term, i.e., result independent of initial conditions

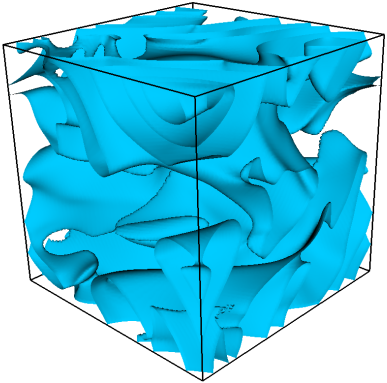
Forcing:

$$F_x = 2A\left(\frac{\kappa_y \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_y = -A\left(\frac{\kappa_x \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_z = -A\left(\frac{\kappa_x \kappa_y}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

Iso-surface $||\mathbf{u}||/\langle u_{rms}\rangle = 2$



with phase

$$G(\kappa_x, \kappa_y, \kappa_z) = \sin\left(\frac{2\pi x}{L}\kappa_x + \frac{2\pi y}{L}\kappa_y + \frac{2\pi z}{L}\kappa_z + \phi\right)$$ for $(0 < \kappa_i \le 2)$ and ϕ being a random phase value.

Forced homogeneous isotropic turbulence

- Fourier representation
- Periodic boundaries, uniform mesh
- Use of external forcing term, i.e., result independent of initial conditions

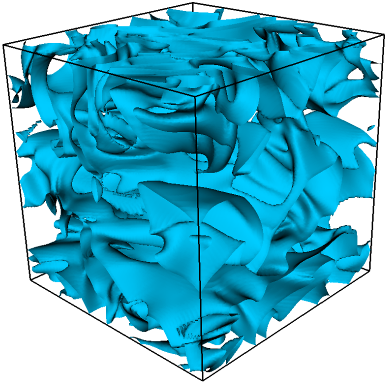
Forcing:

$$F_x = 2A\left(\frac{\kappa_y \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_y = -A\left(\frac{\kappa_x \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_z = -A\left(\frac{\kappa_x \kappa_y}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

Iso-surface $||\mathbf{u}||/\langle u_{rms}\rangle = 2$



with phase

$G(\kappa_x, \kappa_y, \kappa_z) = \sin\left(\frac{2\pi x}{L}\kappa_x + \frac{2\pi y}{L}\kappa_y + \frac{2\pi z}{L}\kappa_z + \phi\right)$ for $(0 < \kappa_i \leq 2)$ and ϕ being a random phase value.

Forced homogeneous isotropic turbulence

- Fourier representation
- Periodic boundaries, uniform mesh
- Use of external forcing term, i.e., result independent of initial conditions

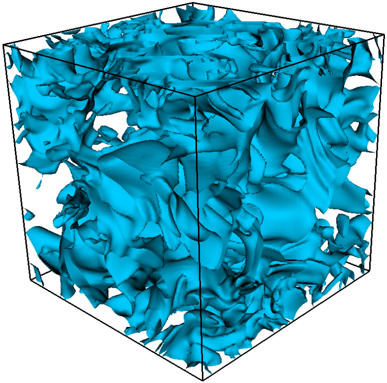
Forcing:

$$F_x = 2A\left(\frac{\kappa_y \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_y = -A\left(\frac{\kappa_x \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_z = -A\left(\frac{\kappa_x \kappa_y}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

Iso-surface $||\mathbf{u}||/\langle u_{rms}\rangle = 2$



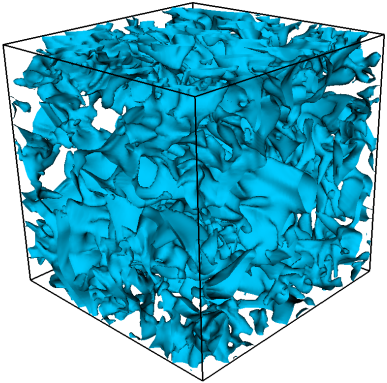
with phase

$G(\kappa_x, \kappa_y, \kappa_z) = \sin\left(\frac{2\pi x}{L}\kappa_x + \frac{2\pi y}{L}\kappa_y + \frac{2\pi z}{L}\kappa_z + \phi\right)$ for $(0 < \kappa_i \leq 2)$ and ϕ being a random phase value.

Forced homogeneous isotropic turbulence

- Fourier representation
- Periodic boundaries, uniform mesh
- Use of external forcing term, i.e., result independent of initial conditions

Iso-surface $||\mathbf{u}||/\langle u_{rms}\rangle = 2$



Forcing:

$$F_x = 2A\left(\frac{\kappa_y\kappa_z}{|\kappa|^2}\right)G(\kappa_x,\kappa_y,\kappa_z)$$

$$F_y = -A\left(\frac{\kappa_x\kappa_z}{|\kappa|^2}\right)G(\kappa_x,\kappa_y,\kappa_z)$$

$$F_z = -A\left(\frac{\kappa_x\kappa_y}{|\kappa|^2}\right)G(\kappa_x,\kappa_y,\kappa_z)$$

with phase

$G(\kappa_x,\kappa_y,\kappa_z) = \sin\left(\frac{2\pi x}{L}\kappa_x + \frac{2\pi y}{L}\kappa_y + \frac{2\pi z}{L}\kappa_z + \phi\right)$ for $(0 < \kappa_i \leq 2)$ and ϕ being a random phase value.

Forced homogeneous isotropic turbulence

- Fourier representation
- Periodic boundaries, uniform mesh
- Use of external forcing term, i.e., result independent of initial conditions

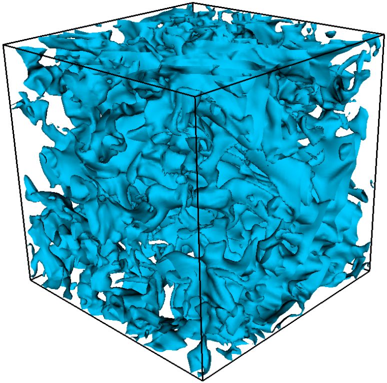
Forcing:

$$F_x = 2A\left(\frac{\kappa_y \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_y = -A\left(\frac{\kappa_x \kappa_z}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

$$F_z = -A\left(\frac{\kappa_x \kappa_y}{|\kappa|^2}\right) G(\kappa_x, \kappa_y, \kappa_z)$$

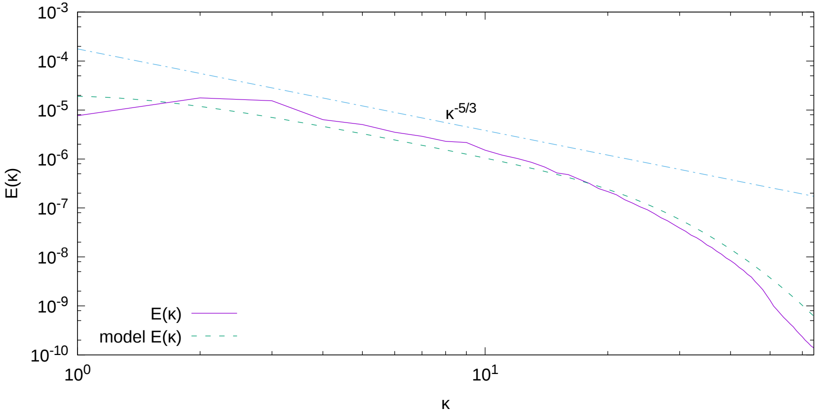
Iso-surface $||\mathbf{u}||/\langle u_{rms}\rangle = 2$



with phase

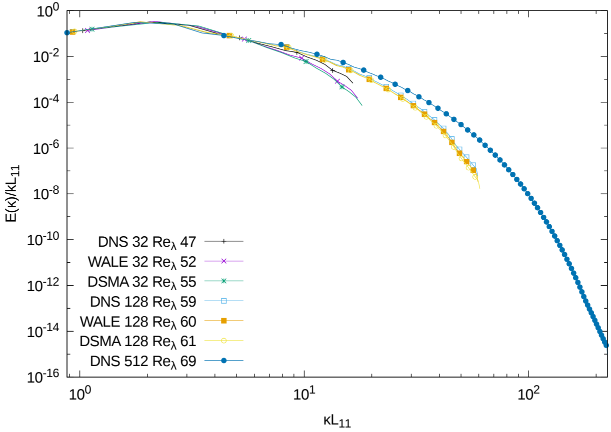
$G(\kappa_x, \kappa_y, \kappa_z) = \sin\left(\frac{2\pi x}{L}\kappa_x + \frac{2\pi y}{L}\kappa_y + \frac{2\pi z}{L}\kappa_z + \phi\right)$ for $(0 < \kappa_i \leq 2)$ and ϕ being a random phase value.

Comparison with model spectrum



Time-averaged energy spectrum (solid line) [$N = 128^3$ cells, $\nu = 3e^{-5}\ \mathrm{m}^2/\mathrm{s}$] against a modelled one (dashed line and the -5/3 power law (dot-dashed line).

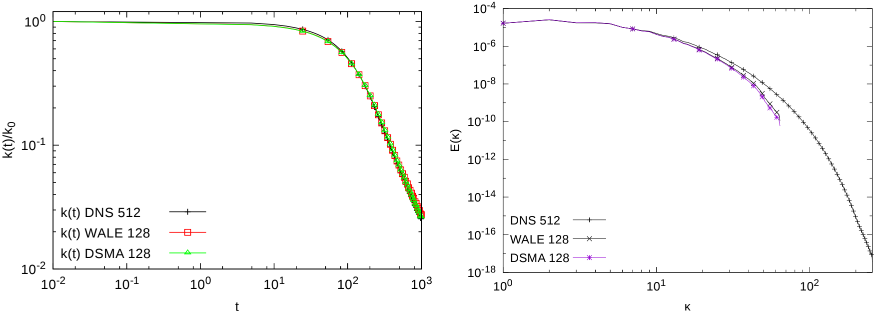
LES model spectra



Time-averaged energy spectra normalised by the turbulent kinetic energy k and the integral length scale L_{11} of LBM DNS and LES for two resolutions and DNS of the highest resolution for the viscosity value $\nu = 5 \cdot 10^{-5}$

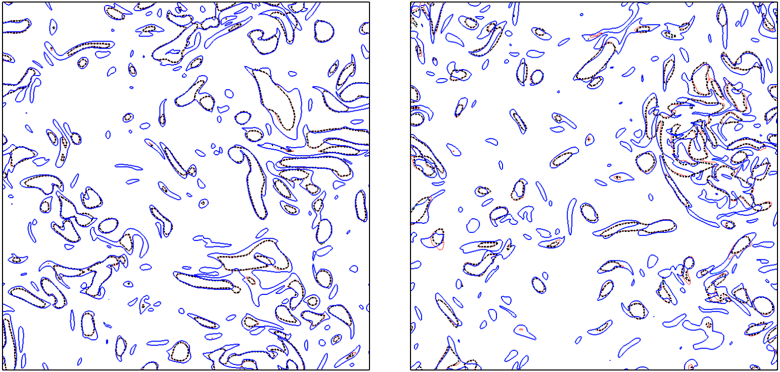
Decaying homogeneous isotropic turbulence

- Restart DNS of 512^3 resolution without forcing. Volume-averaging to 128^3 cells gives DSMA and WALE initial conditions



Evolution of the turbulent kinetic energy k (left) and energy spectra at $t = 68.72$ (right) for DNS of 512^3 against DSMA and WALE of 128^3 cells resolution.

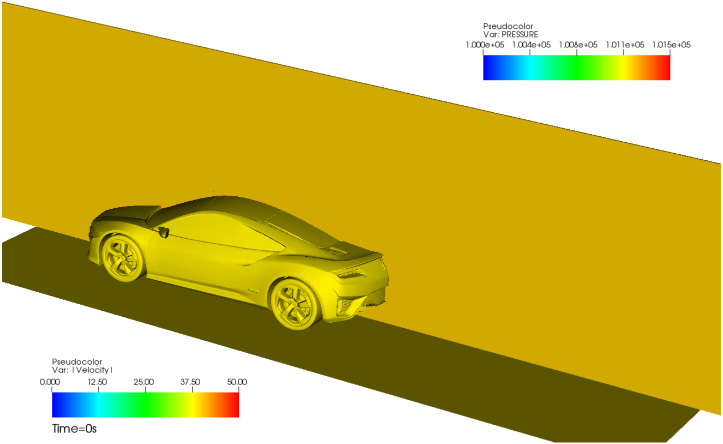
Flow field comparison



Contours of vorticity magnitude ($|\omega| = 0.18$) at $t = 4.91$ (left) and $t = 68.72$ (right) for DNS (thin blue lines) of 512^3 against DSMA (dotted black lines) and WALE (thick red lines) of 128^3 cells resolution

Wind tunnel simulation of a prototype car

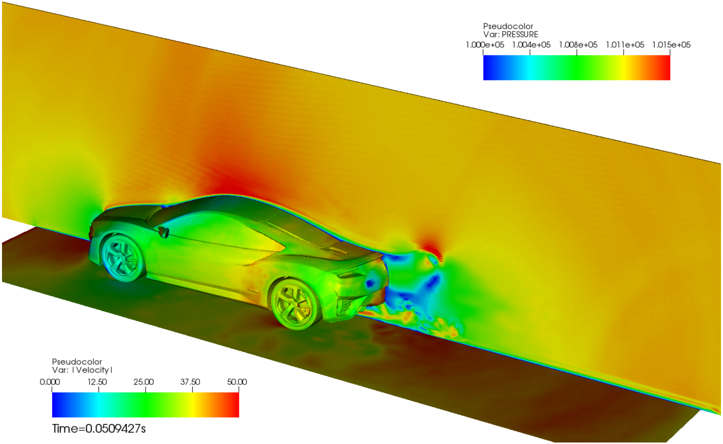
Fluid velocity and pressure on vehicle



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

Wind tunnel simulation of a prototype car

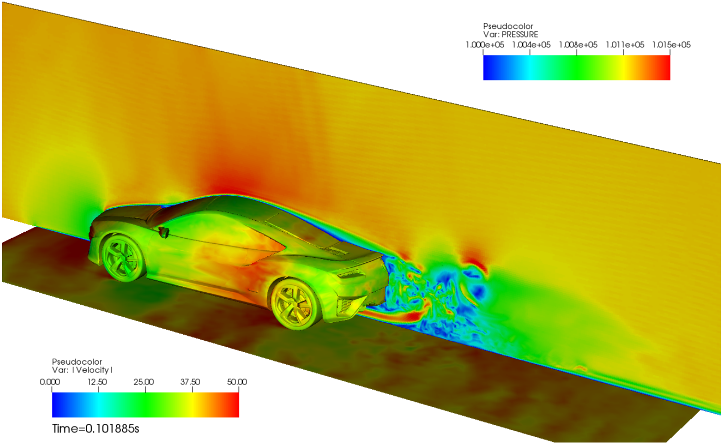
Fluid velocity and pressure on vehicle



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

Wind tunnel simulation of a prototype car

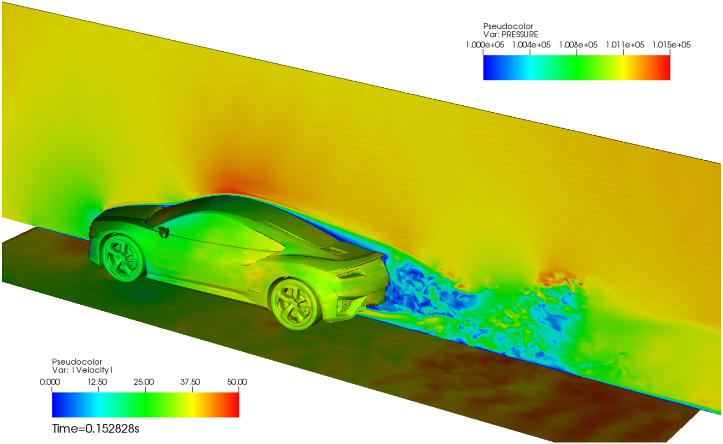
Fluid velocity and pressure on vehicle



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

Wind tunnel simulation of a prototype car

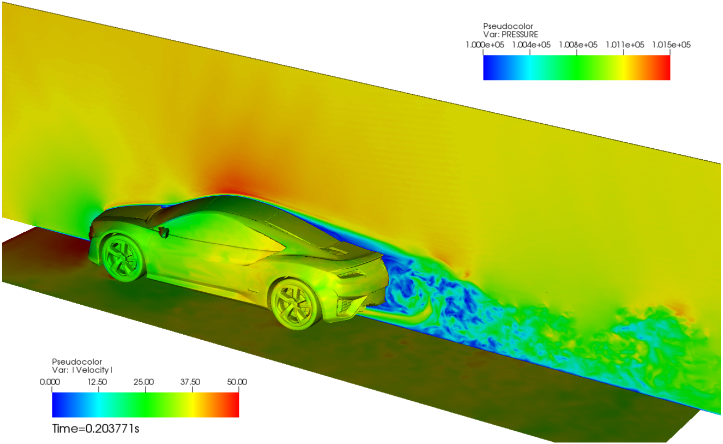
Fluid velocity and pressure on vehicle



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

Wind tunnel simulation of a prototype car

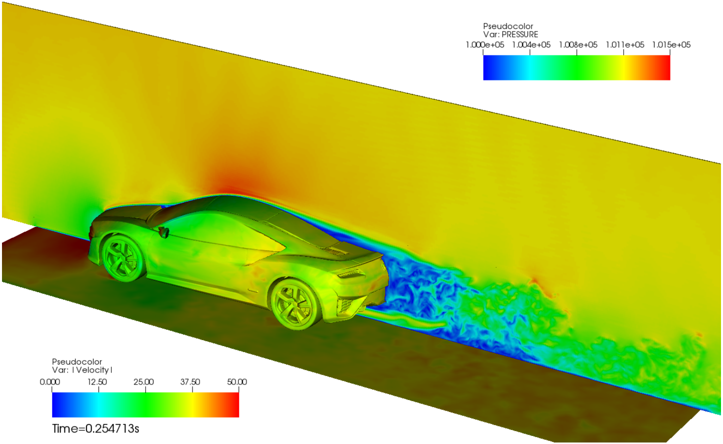
Fluid velocity and pressure on vehicle



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

Wind tunnel simulation of a prototype car

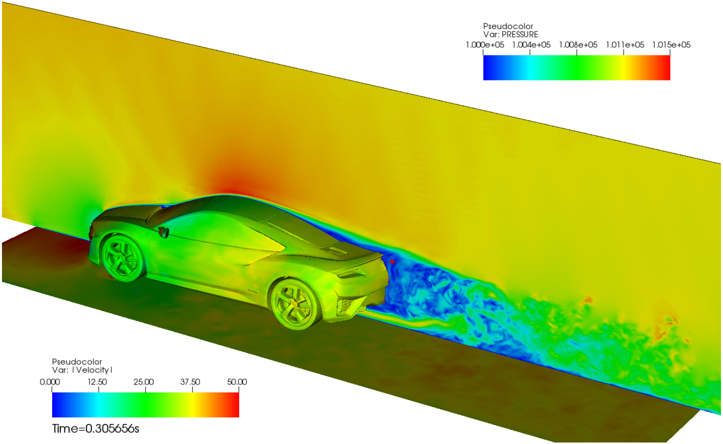
Fluid velocity and pressure on vehicle



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

Wind tunnel simulation of a prototype car

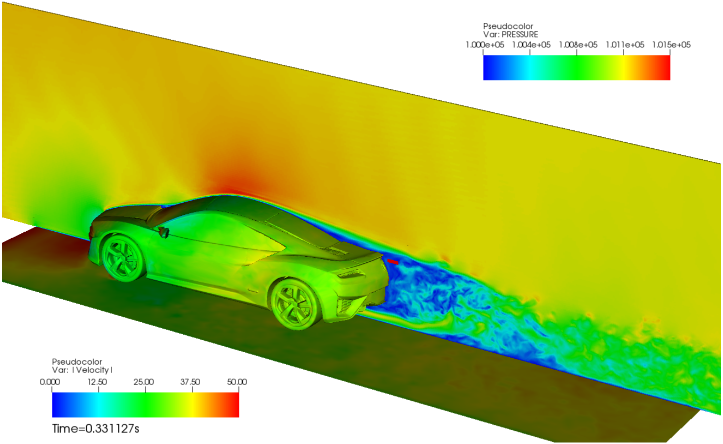
Fluid velocity and pressure on vehicle



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

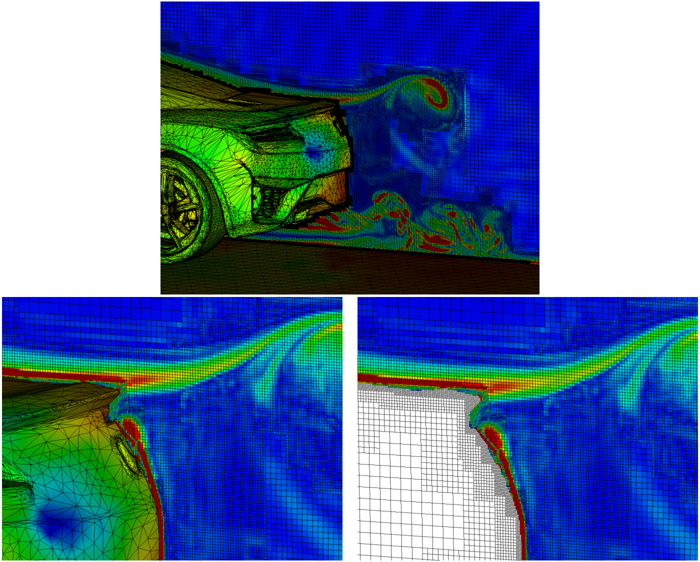
Wind tunnel simulation of a prototype car

Fluid velocity and pressure on vehicle



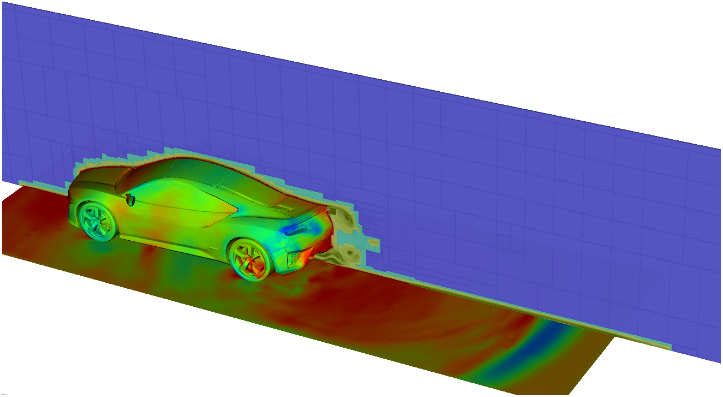
- Inflow 40 $\mathrm{m/s}$. CSMA LES model active. Characteristic boundary conditions.
- To $t = 0.5\,\mathrm{s}$ ($\sim$ 4 characteristic lengths) with 31,416 time steps on finest level in $\sim 37\,\mathrm{h}$ on 200 cores (7389 h CPU). Channel: $15\,\mathrm{m} \times 5\,\mathrm{m} \times 3.3\,\mathrm{m}$

Mesh adaptation



Mesh adaptation

Used refinement blocks and levels (indicated by color)



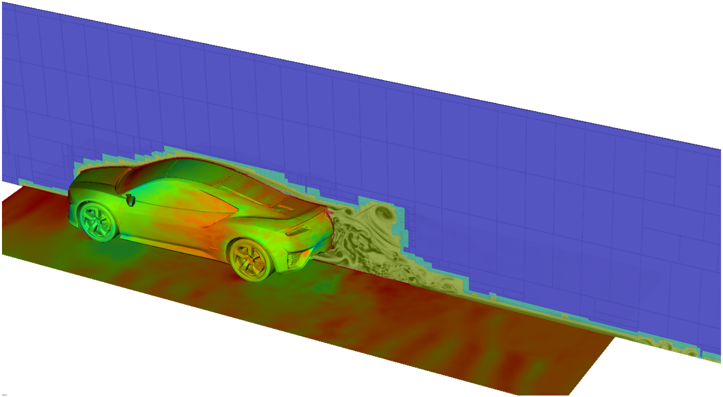
- SAMR base grid $600 \times 200 \times 132$ cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 3.125\,\mathrm{mm}$
- Adaptation based on level set and scaled gradient of magnitude of vorticity vector
- 236M cells vs. 8.1 billion (uniform) at $t = 0.4075\,\mathrm{s}$

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

Mesh adaptation

Used refinement blocks and levels (indicated by color)



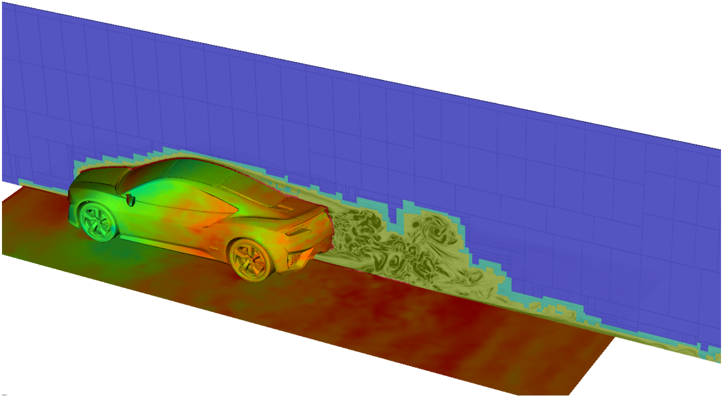
- SAMR base grid $600 \times 200 \times 132$ cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 3.125\,\mathrm{mm}$
- Adaptation based on level set and scaled gradient of magnitude of vorticity vector
- 236M cells vs. 8.1 billion (uniform) at $t = 0.4075\,\mathrm{s}$

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

Mesh adaptation

Used refinement blocks and levels (indicated by color)



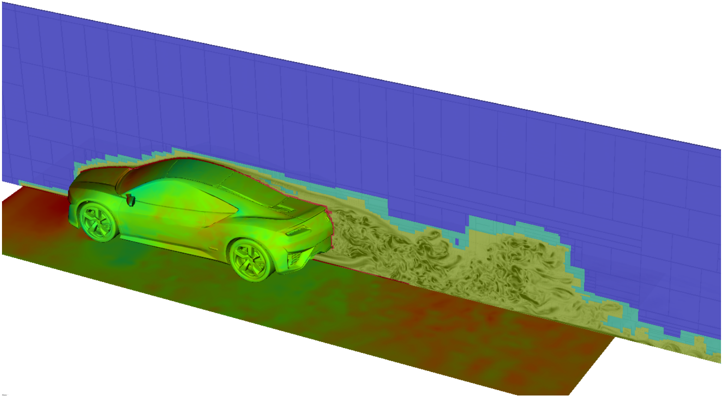
- SAMR base grid $600 \times 200 \times 132$ cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 3.125\,\mathrm{mm}$
- Adaptation based on level set and scaled gradient of magnitude of vorticity vector
- 236M cells vs. 8.1 billion (uniform) at $t = 0.4075\,\mathrm{s}$

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

Mesh adaptation

Used refinement blocks and levels (indicated by color)



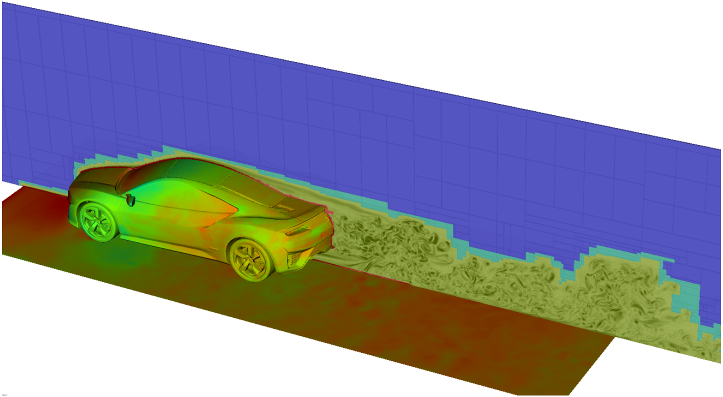
- SAMR base grid 600 × 200 × 132 cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 3.125\,\mathrm{mm}$
- Adaptation based on level set and scaled gradient of magnitude of vorticity vector
- 236M cells vs. 8.1 billion (uniform) at $t = 0.4075\,\mathrm{s}$

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

Mesh adaptation

Used refinement blocks and levels (indicated by color)



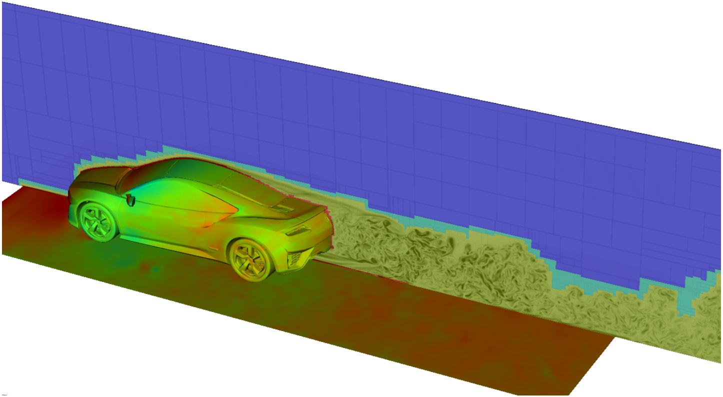
- SAMR base grid $600 \times 200 \times 132$ cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 3.125\,\mathrm{mm}$
- Adaptation based on level set and scaled gradient of magnitude of vorticity vector
- 236M cells vs. 8.1 billion (uniform) at $t = 0.4075\,\mathrm{s}$

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

Mesh adaptation

Used refinement blocks and levels (indicated by color)



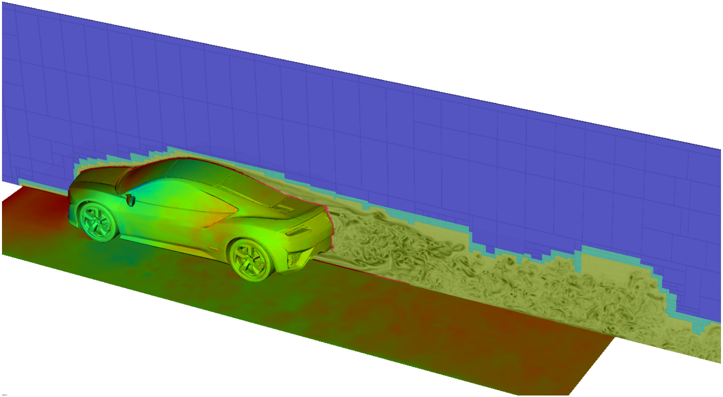
- SAMR base grid $600 \times 200 \times 132$ cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 3.125\,\mathrm{mm}$
- Adaptation based on level set and scaled gradient of magnitude of vorticity vector
- 236M cells vs. 8.1 billion (uniform) at $t = 0.4075\,\mathrm{s}$

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

Mesh adaptation

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

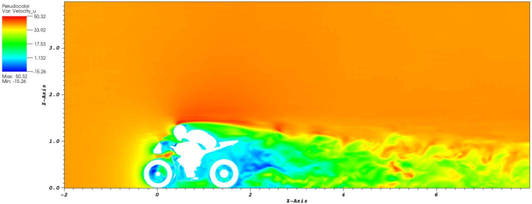
Refinement at $t = 0.4075\,\mathrm{s}$

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

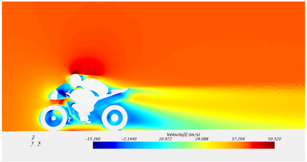
Flow over a motorcycle

- Inflow $40\,\mathrm{m/s}$. Bouzidi pressure boundary conditions at outflows. CSMA LES model active.
- SAMR base grid $200 \times 80 \times 80$ cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 6.25\,\mathrm{mm}$. 23560 time steps on finest level
- Forces in AMROC-LBM are time-averaged over interval $[0.5\mathrm{s}, 1\mathrm{s}]$
- Unstructured STAR-CCM+ mesh has significantly finer as well as coarser cells

AMROC-LBM LES at $t = 1\,\mathrm{s}$



STAR-CCM+ steady RANS



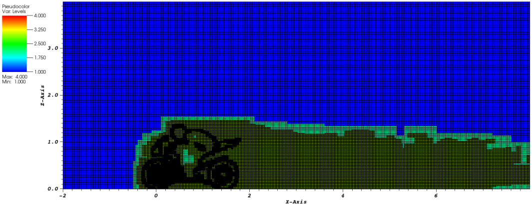
Velocity in flow direction

Variables	Forces (N) Drag	Sideforce	Lift	Total	Cores	Wall Time h	CPU Time h
STAR-CCM+	297	5	9	297	10	4.9	78
AMROC	297	10	23	298	64	10	635

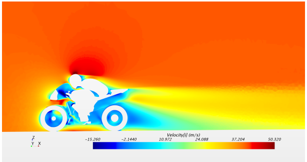
Flow over a motorcycle

- Inflow $40\,\mathrm{m/s}$. Bouzidi pressure boundary conditions at outflows. CSMA LES model active.
- SAMR base grid $200 \times 80 \times 80$ cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 6.25\,\mathrm{mm}$. 23560 time steps on finest level
- Forces in AMROC-LBM are time-averaged over interval $[0.5\mathrm{s}, 1\mathrm{s}]$
- Unstructured STAR-CCM+ mesh has significantly finer as well as coarser cells

AMROC-LBM LES at $t = 1\,\mathrm{s}$



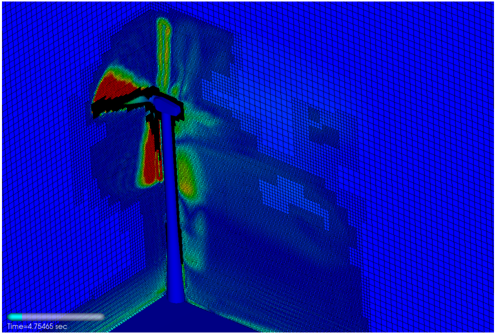
STAR-CCM+ steady RANS



Velocity in flow direction

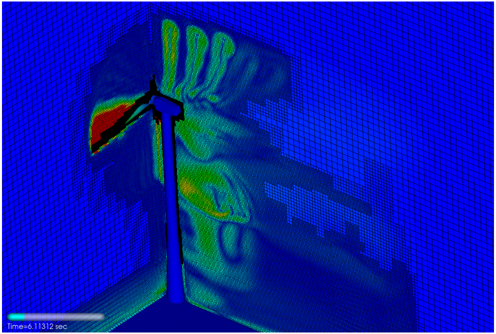
Variables	Forces (N) Drag	Sideforce	Lift	Total	Cores	Wall Time h	CPU Time h
STAR-CCM+	297	5	9	297	10	4.9	78
AMROC	297	10	23	298	64	10	635

Single Vestas V27



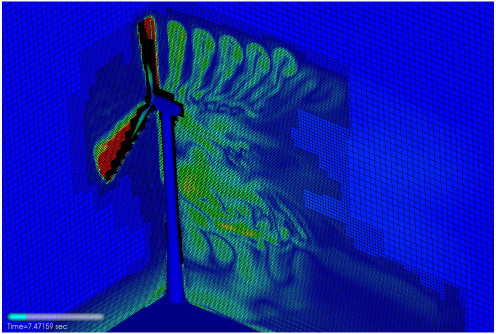
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$, TSR=5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set. CSMA LES model.
- ~ 24 time steps for 1^o rotation
- Validation results: [Deiterding and Wood, 2016]

Single Vestas V27



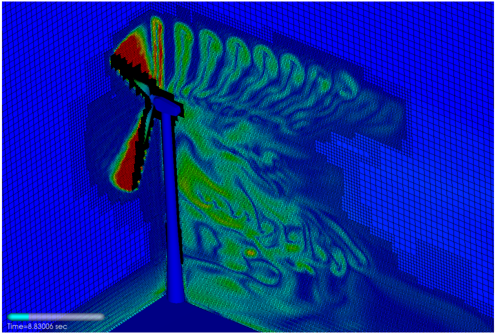
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$, TSR=5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set. CSMA LES model.
- ~ 24 time steps for 1^o rotation
- Validation results: [Deiterding and Wood, 2016]

Single Vestas V27



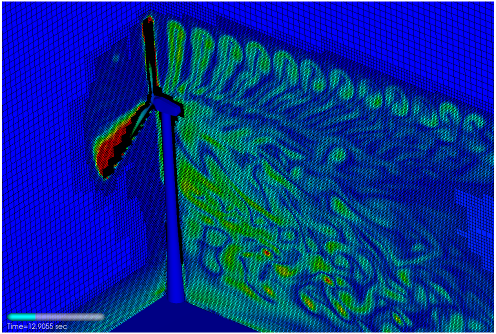
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$, TSR=5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set. CSMA LES model.
- $\sim$ 24 time steps for 1^o rotation
- Validation results: [Deiterding and Wood, 2016]

Single Vestas V27



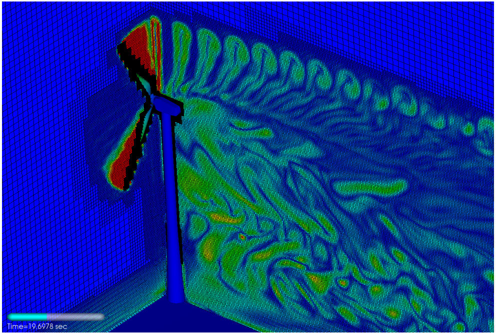
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$, TSR=5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set. CSMA LES model.
- ~ 24 time steps for 1^o rotation
- Validation results: [Deiterding and Wood, 2016]

Single Vestas V27



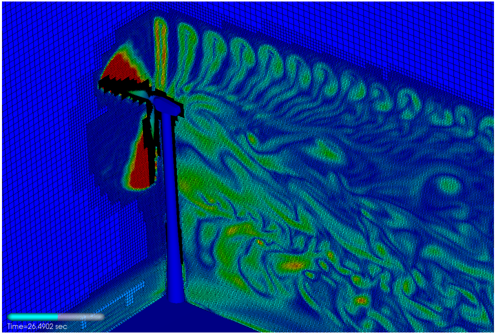
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_\mathrm{rpm} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$, TSR=5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set. CSMA LES model.
- ~ 24 time steps for 1^o rotation
- Validation results: [Deiterding and Wood, 2016]

Single Vestas V27



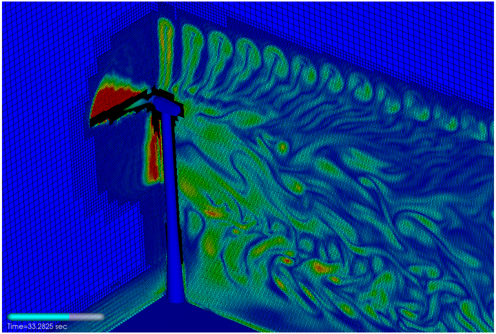
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$, TSR=5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set. CSMA LES model.
- ~ 24 time steps for 1^o rotation
- Validation results: [Deiterding and Wood, 2016]

Single Vestas V27



- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$, TSR=5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set. CSMA LES model.
- ~ 24 time steps for 1^o rotation
- Validation results: [Deiterding and Wood, 2016]

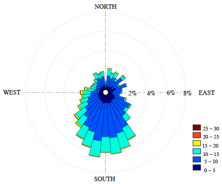
Single Vestas V27

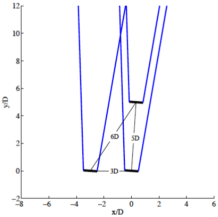


- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$, TSR=5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set. CSMA LES model.
- ~ 24 time steps for 1^o rotation
- Validation results: [Deiterding and Wood, 2016]

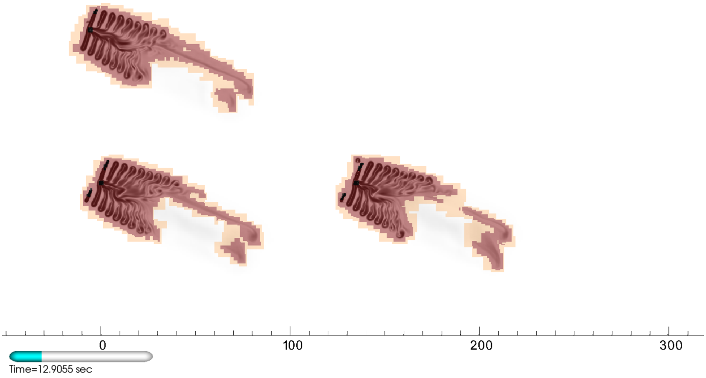
Simulation of the SWIFT array

- Three Vestas V27 turbines (geometric details prototypical). 225 kW power generation at wind speeds 14 to 25 $\mathrm{m/s}$ (then cut-off)
- Prescribed motion of rotor with 33 rpm. Inflow velocity 8 $\mathrm{m/s}$
- Simulation domain 448 $\mathrm{m} \times 240\,\mathrm{m} \times 100\,\mathrm{m}$
- Base mesh $448 \times 240 \times 100$ cells with refinement factors 2, 2,4. Resolution of rotor and tower $\Delta x = 6.25\,\mathrm{cm}$
- 94,224 highest level iterations to $t_e = 40\,\mathrm{s}$ computed, then statistics are gathered for 10 s [Deiterding and Wood, 2016]





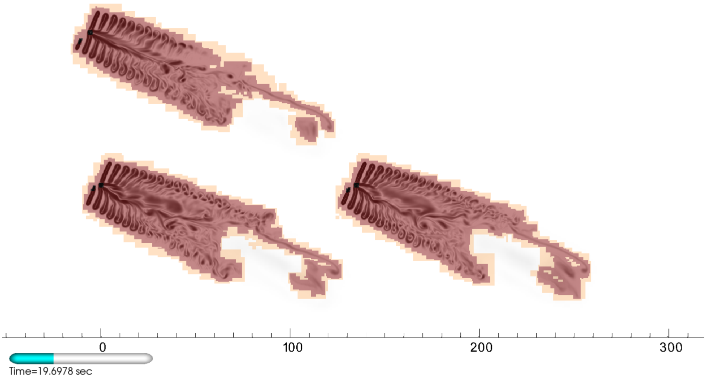
Levels – inflow at 30°, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



- At $63.8\,\mathrm{s}$ approximately 167M cells used vs. 44 billion (factor 264)
- $\sim 6.01\,\mathrm{h}$ per revolution ($961\,\mathrm{h}$ CPU) $\longrightarrow \sim 5.74\,\mathrm{h}$ CPU/1M cells/revolution
- 160 cores Intel-Xeon E5 2.6 GHz, $33.03\,\mathrm{h}$ wall time for $10\,\mathrm{s}$ interval

Level	Grids	Cells
0	2,463	10,752,000
1	6,464	20,674,760
2	39,473	131,018,832
3	827	4,909,632

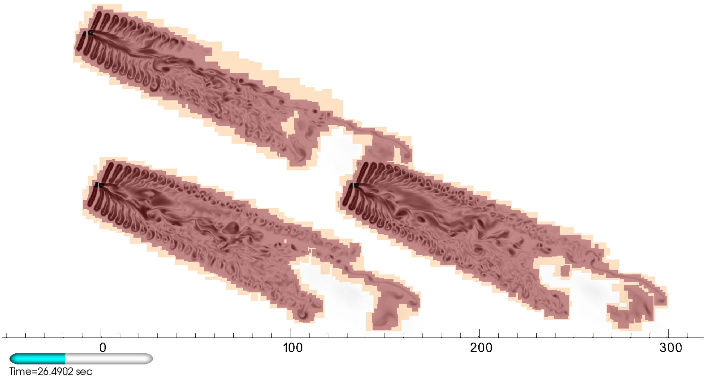
Levels – inflow at 30°, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



- At $63.8\,\mathrm{s}$ approximately 167M cells used vs. 44 billion (factor 264)
- $\sim 6.01\,\mathrm{h}$ per revolution ($961\,\mathrm{h}$ CPU) $\longrightarrow \sim 5.74\,\mathrm{h}$ CPU/1M cells/revolution
- 160 cores Intel-Xeon E5 2.6 GHz, $33.03\,\mathrm{h}$ wall time for $10\,\mathrm{s}$ interval

Level	Grids	Cells
0	2,463	10,752,000
1	6,464	20,674,760
2	39,473	131,018,832
3	827	4,909,632

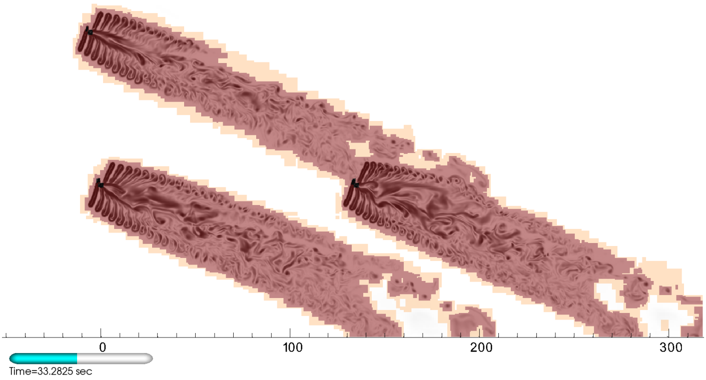
Levels – inflow at 30°, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



- At $63.8\,\mathrm{s}$ approximately 167M cells used vs. 44 billion (factor 264)
- $\sim 6.01\,\mathrm{h}$ per revolution ($961\,\mathrm{h}$ CPU) $\longrightarrow \sim 5.74\,\mathrm{h}$ CPU/1M cells/revolution
- 160 cores Intel-Xeon E5 2.6 GHz, $33.03\,\mathrm{h}$ wall time for $10\,\mathrm{s}$ interval

Level	Grids	Cells
0	2,463	10,752,000
1	6,464	20,674,760
2	39,473	131,018,832
3	827	4,909,632

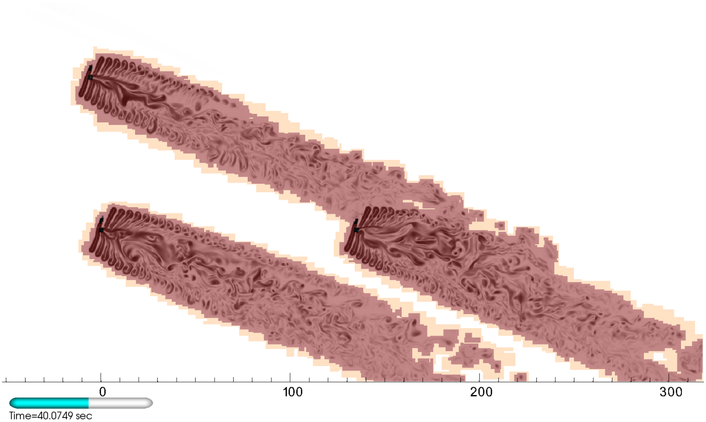
Levels – inflow at 30^o, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



- At $63.8\,\mathrm{s}$ approximately 167M cells used vs. 44 billion (factor 264)
- $\sim 6.01\,\mathrm{h}$ per revolution ($961\,\mathrm{h}$ CPU) $\longrightarrow \sim 5.74\,\mathrm{h}$ CPU/1M cells/revolution
- 160 cores Intel-Xeon E5 2.6 GHz, $33.03\,\mathrm{h}$ wall time for $10\,\mathrm{s}$ interval

Level	Grids	Cells
0	2,463	10,752,000
1	6,464	20,674,760
2	39,473	131,018,832
3	827	4,909,632

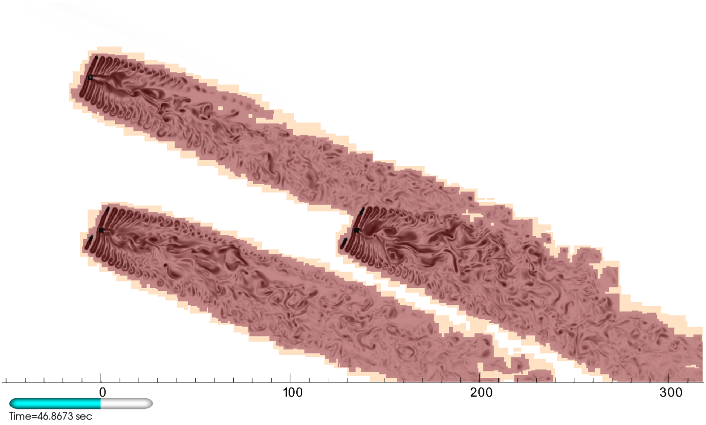
Levels – inflow at 30°, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



- At $63.8\,\mathrm{s}$ approximately 167M cells used vs. 44 billion (factor 264)
- $\sim$ 6.01 h per revolution (961 h CPU) $\longrightarrow$ $\sim$ 5.74 h CPU/1M cells/revolution
- 160 cores Intel-Xeon E5 2.6 GHz, 33.03 h wall time for 10 s interval

Level	Grids	Cells
0	2,463	10,752,000
1	6,464	20,674,760
2	39,473	131,018,832
3	827	4,909,632

Levels – inflow at 30°, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



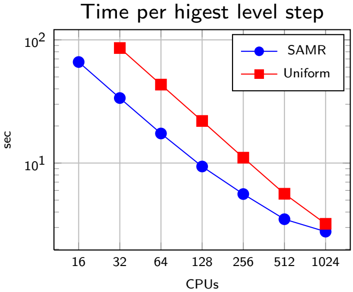
- At $63.8\,\mathrm{s}$ approximately 167M cells used vs. 44 billion (factor 264)
- $\sim$ 6.01 h per revolution (961 h CPU) $\longrightarrow$ $\sim$ 5.74 h CPU/1M cells/revolution
- 160 cores Intel-Xeon E5 2.6 GHz, 33.03 h wall time for 10 s interval

Level	Grids	Cells
0	2,463	10,752,000
1	6,464	20,674,760
2	39,473	131,018,832
3	827	4,909,632

AMROC strong scalability tests

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

- Tests run IBM BG/P (mode VN)



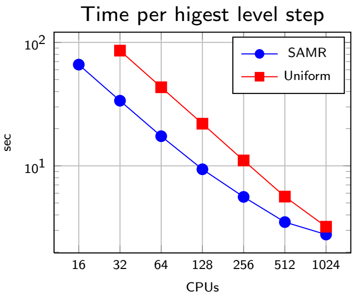
$64 \times 32 \times 32$ base grid, 2 additional levels with factors 2, 4; uniform $512 \times 256 \times 256 = 33.6 \cdot 10^6$ cells

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)

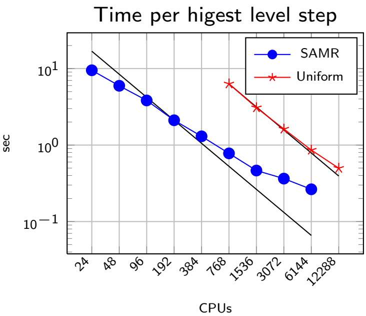


$64 \times 32 \times 32$ base grid, 2 additional levels with factors 2, 4; uniform $512 \times 256 \times 256 = 33.6 \cdot 10^6$ cells

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

3D SRT-lattice Boltzmann scheme: 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

Lattice Boltzmann equation in mapped coordinates

Consider mapping from Cartesian to non-Cartesian coordinates

$$\xi = \xi(x,y),\; \eta = \eta(x,y)$$

with

$$\frac{\partial}{\partial x} = \frac{\partial}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial}{\partial \eta}\frac{\partial \eta}{\partial x},\quad \frac{\partial}{\partial y} = \frac{\partial}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial}{\partial \eta}\frac{\partial \eta}{\partial y}.$$

Under this transformation the convection term reads

$$\begin{aligned}
\mathbf{e}_\alpha \cdot \nabla f_\alpha &= e_{\alpha x}\frac{\partial f_\alpha}{\partial x} + e_{\alpha y}\frac{\partial f_\alpha}{\partial y} \\
&= e_{\alpha x}\left(\frac{\partial f_\alpha}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial f_\alpha}{\partial \eta}\frac{\partial \eta}{\partial x}\right) + e_{\alpha y}\left(\frac{\partial f_\alpha}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial f_\alpha}{\partial \eta}\frac{\partial \eta}{\partial y}\right) \\
&= \left(e_{\alpha x}\frac{\partial \xi}{\partial x} + e_{\alpha y}\frac{\partial \xi}{\partial y}\right)\frac{\partial f_\alpha}{\partial \xi} + \left(e_{\alpha x}\frac{\partial \eta}{\partial x} + e_{\alpha y}\frac{\partial \eta}{\partial y}\right)\frac{\partial f_\alpha}{\partial \eta} \\
&= \tilde{e}_{\alpha\xi}\frac{\partial f_\alpha}{\partial \xi} + \tilde{e}_{\alpha\eta}\frac{\partial f_\alpha}{\partial \eta},
\end{aligned}$$

and hence the lattice Boltzmann equation becomes

$$\frac{\partial f}{\partial t} + \tilde{e}_{\alpha\xi}\frac{\partial f_\alpha}{\partial \xi} + \tilde{e}_{\alpha\eta}\frac{\partial f_\alpha}{\partial \eta} = -\frac{1}{\tau}\left(f_\alpha - f_\alpha^{eq}\right).$$

Scheme construction

Currently using the explicit 4th-order Runge-Kutta scheme

$$f_\alpha^1 = f_\alpha^t,\ f_\alpha^2 = f_\alpha^1 + \frac{\Delta t}{4} R_\alpha^1,$$
$$f_\alpha^3 = f_\alpha^1 + \frac{\Delta t}{3} R_\alpha^2, f_\alpha^4 = f_\alpha^1 + \frac{\Delta t}{2} R_\alpha^3,$$
$$f_\alpha^{t+\Delta t} = f_\alpha^1 + \Delta t R_\alpha^4.$$

with

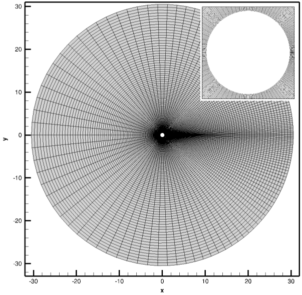
$$R_{\alpha(i,j)} = -\left(\tilde{e}_{\alpha\xi_{(i,j)}} \frac{f_{\alpha(i+1,j)} - f_{\alpha(i-1,j)}}{2\Delta\xi} + \tilde{e}_{\alpha\eta_{(i,j)}} \frac{f_{\alpha(i,j+1)} - f_{\alpha(i,j-1)}}{2\Delta\eta}\right) - \frac{1}{\tau}\left(f_{\alpha(i,j)} - f^{eq}_{\alpha(i,j)}\right)$$

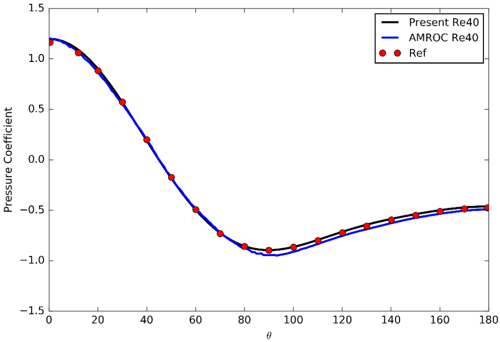
for the solution, 2nd-order central differences to approximate derivatives.
A 4th-order dissipation term

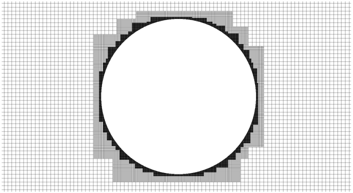
$$D = -\epsilon\left((\Delta\xi)^4 \frac{\partial^4 f_\alpha}{\partial \xi^4} + (\Delta\eta)^4 \frac{\partial^4 f_\alpha}{\partial \eta^4}\right)$$

is added for stabilization [Hejranfar and Hajihassanpour, 2017].
Prototype implementation is presently on finite difference meshes!

2d cylinder study



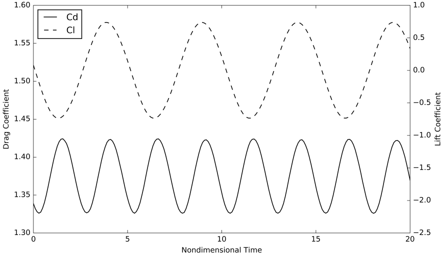




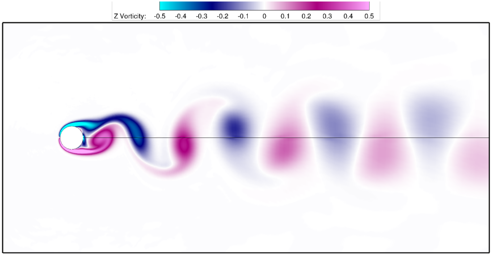
Re	Author(s)	C_d	$C_p(0)$	$C_p(180)$	$2L/D$
20	[Tritton, 1959]	2.20	-	-	-
	[Henderson, 1995]	2.06	-	-0.60	-
	[Dennis and Chang, 1970]	2.05	1.27	-0.58	1.88
	[Hejranfar and Ezzatneshan, 2014]	2.02	1.25	-0.59	1.84
	AMROC-LBM	1.98	1.26	-0.59	1.85
	Present	2.02	1.31	-0.55	1.85
40	[Tritton, 1959]	1.65	-	-	-
	[Henderson, 1995]	1.55	-	-0.53	-
	[Dennis and Chang, 1970]	1.52	1.14	-0.50	4.69
	[Hejranfar and Ezzatneshan, 2014]	1.51	1.15	-0.48	4.51
	AMROC-LBM	1.45	1.19	-0.49	4.66
	Present	1.51	1.19	-0.46	4.60

$2L/D$ is normalized length of wake behind cylinder

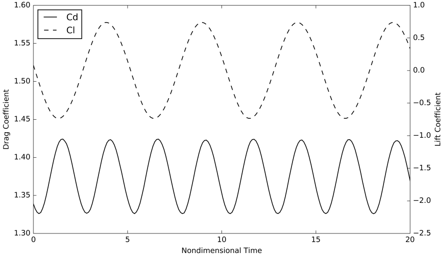
2d cylinder study – unsteady flow case



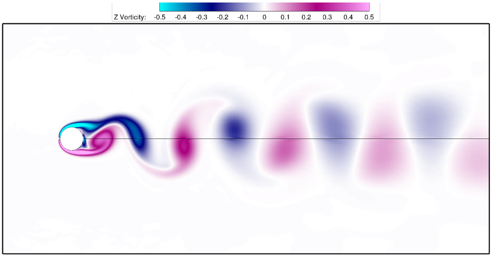
Re	Author(s)	St	$\overline{C_d}$	C_l'
100	[Chiu et al., 2010]	0.167	1.35	0.30
	AMROC-LBM	0.166	1.28	0.32
	Present	0.165	1.36	0.35
200	[Chiu et al., 2010]	0.198	1.37	0.71
	AMROC-LBM	0.196	1.26	0.70
	Present	0.196	1.37	0.73



2d cylinder study – unsteady flow case



Re	Author(s)	St	$\overline{C_d}$	C_l'
100	[Chiu et al., 2010]	0.167	1.35	0.30
	AMROC-LBM	0.166	1.28	0.32
	Present	0.165	1.36	0.35
200	[Chiu et al., 2010]	0.198	1.37	0.71
	AMROC-LBM	0.196	1.26	0.70
	Present	0.196	1.37	0.73



Re		CPU-time	Mesh
20	AMROC-LBM	24:55:21	297796
	Present	06:08:41	65536
40	AMROC-LBM	27:10:08	317732
	Present	05:57:17	65536
100	AMROC-LBM	113:15:37	1026116
	Present	05:58:49	65536
200	AMROC-LBM	130:37:18	1130212
	Present	06:03:42	65536

Conclusions

- Cartesian LBM is a very efficient low-dissipation method and especially suitable for DNS and LES
- Cartesian CFD with block-based AMR is faster than cell-cased AMR and tailored for modern massively parallel computer systems
- Fast dynamic mesh adaptation in AMROC makes FSI problems with complex motion easily accessible. Time-explicit approach leads to very tight coupling
- For high Reynolds number flows around complex bodies an LES turbulence model is vital for stability (so are higher-order in- and outflow boundary conditions)
- Currently validating and extending (dynamic) Smagorinsky with wall-near damping and WALE model for realistic problems
- Turbulent wall function boundary condition model under development

Conclusions

- Cartesian LBM is a very efficient low-dissipation method and especially suitable for DNS and LES
- Cartesian CFD with block-based AMR is faster than cell-cased AMR and tailored for modern massively parallel computer systems
- Fast dynamic mesh adaptation in AMROC makes FSI problems with complex motion easily accessible. Time-explicit approach leads to very tight coupling
- For high Reynolds number flows around complex bodies an LES turbulence model is vital for stability (so are higher-order in- and outflow boundary conditions)
- Currently validating and extending (dynamic) Smagorinsky with wall-near damping and WALE model for realistic problems
- Turbulent wall function boundary condition model under development
- Accurate simulation of thin, wall-resolved boundary layers is dramatically more efficient with the non-Cartesian LBM approach, despite the availability of AMR in AMROC
  - Develop non-Cartesian version of AMROC-LBM as near-term goal
  - Chimera technique within AMROC-LBM might be long-term goal

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.

[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.

[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.

[Chiu et al., 2010] Chiu, P. H., Lin, R. K., and Sheu, T. W. (2010). A differentially interpolated direct forcing immersed boundary method for predicting incompressible Navier Stokes equations in time-varying complex geometries. *Journal of Computational Physics*.

[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., 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., 2009] Deiterding, R., Domingues, M. O., Gomes, S. M., Roussel, O., and Schneider, K. (2009). 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.

[Dennis and Chang, 1970] Dennis, S. C. R. and Chang, G. (1970). Numerical solutions for steady flow past a circular cylinder at Reynolds numbers up to 100. *Journal of Fluid Mechanics*, 42(03):471.

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

References II

[Hejranfar and Ezzatneshan, 2014] Hejranfar, K. and Ezzatneshan, E. (2014). Implementation of a high-order compact finite-difference lattice Boltzmann method in generalized curvilinear coordinates. *Journal of Computational Physics*, 267:28–49.

[Hejranfar and Hajihassanpour, 2017] Hejranfar, K. and Hajihassanpour, M. (2017). Chebyshev collocation spectral lattice Boltzmann method in generalized curvilinear coordinates. *Computers and Fluids*.

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

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

[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.

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

[Nazarinia et al., 2012] Nazarinia, M., Jacono, D. L., Thompson, M. C., and Sheridan, J. (2012). Flow over a cylinder subjected to combined translational and rotational oscillations. *J. Fluids and Structures*, 32:135–145.

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

[Tritton, 1959] Tritton, D. (1959). Experiments on the flow past a circular cylinder at low Reynolds numbers. *Journal of Fluid Mechanics*, 6(4):547–567.

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

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

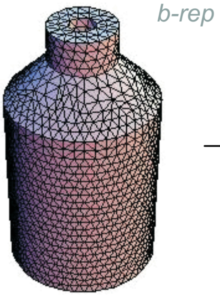
The signed distance φ to a surface $\mathcal{I}$ satisfies the eikonal equation [Sethian, 1999]

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

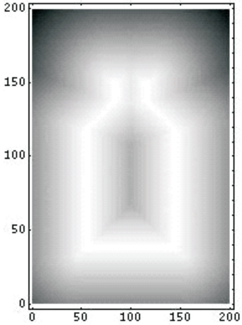
Solution smooth but non-diferentiable across characteristics.

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

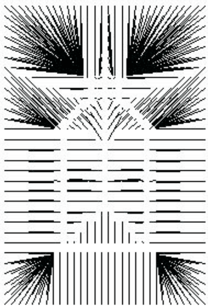
- Geometric solution approach with plosest-point-transform algorithm [Mauch, 2003]



Surface mesh $\mathcal{I}$



Distance φ

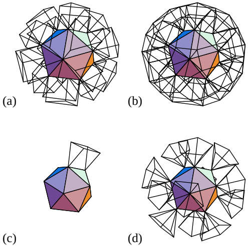


Normal to closest point

The characteristic / scan conversion algorithm

1. Build the characteristic polyhedrons for the surface mesh

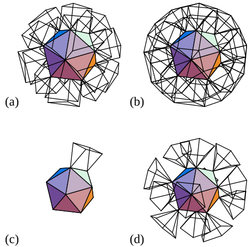
Characteristic polyhedra for faces, edges, and vertices



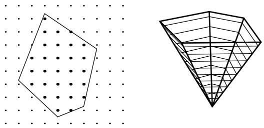
The characteristic / scan conversion algorithm

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



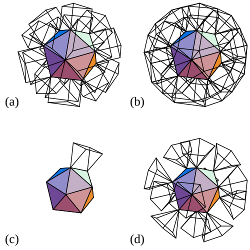
Slicing and scan conversion of apolygon



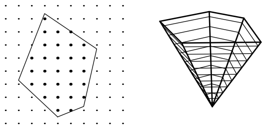
The characteristic / scan conversion algorithm

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



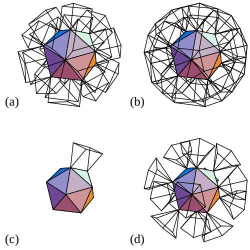
Slicing and scan conversion of apolygon



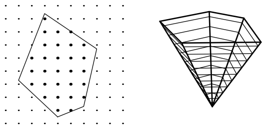
The characteristic / scan conversion algorithm

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

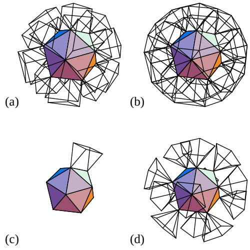


The characteristic / scan conversion algorithm

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