Ralf Deiterding

1992-1998 Study of Industrial Mathematics, Technical University Clausthal, Germany

1998-2003 PhD in Applied Mathematics, Technical University Cottbus, Germany

2003-2006 Postdoc in Applied and Computational Mathematics, California Institute of Technology

2006-2008 Householder Fellowship in Scientific Computing, Oak Ridge National Laboratory

2008-2013 Staff-level Computational Scientist, Oak Ridge National Laboratory

2013-2015 Group Leader for Computational Fluid Dynamics, German Aerospace Center

2015- Associate Professor in Fluid Dynamics at University of Southampton

- Research topics: Innovative numerical methods for CFD, adaptive mesh refinement, hypersonics (currently EPSRC funded), combustion and detonation dynamics, fluid-structure interaction, multi-phase flows
- $\sim$ 60 paper indexed by Web of Science, $>$ 1200 citations in Google scholar
- Main developer of AMROC and Virtual Test Facility (VTF) software
- Teaches numerical methods for supersonic flows, Aerothermodynamics, Hypersonics & high temperature gas dynamics
- Current group size: 2 postdocs, 3 PhD students

Adaptive lattice Boltzmann methods for high-fidelity aerodynamics simulation with moving boundaries

Ralf Deiterding

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

September 21, 2017

Outline

Collaboration on lattice Boltzmann methods with

- Stephen Wood (Joint Insitute of Computational Science, University Tennessee Knoxville, USA)
- Kai Feldhusen, Claus Wagner (German Aerospace Center – DLR)
- Moritz Fragner (Technical University Cottbus, Germany)
- Cinar Laloglu (Marmara University, Turkey)

Approximation of Boltzmann equation

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

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

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

Approximation of Boltzmann equation

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

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

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

Equation is approximated with a splitting approach.

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

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

Approximation of Boltzmann equation

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

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

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

Equation is approximated with a splitting approach.

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

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

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

Approximation of Boltzmann equation

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

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

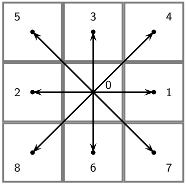
- $\mathrm{Kn} = l_f / L \ll 1$, where l_f is replaced with Δx
- Weak compressibilty and small Mach number assumed
- Assume a simplified phase space

Equation is approximated with a splitting approach.

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

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

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



Discrete velocities:

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

Approximation of Boltzmann equation

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

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

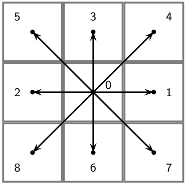
- $\mathrm{Kn} = l_f / L \ll 1$, where l_f is replaced with Δx
- Weak compressibilty and small Mach number assumed
- Assume a simplified phase space

Equation is approximated with a splitting approach.

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

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

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



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

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

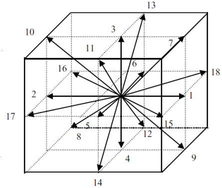
- $\mathrm{Kn} = l_f / L \ll 1$, where l_f is replaced with Δx
- Weak compressibilty and small Mach number assumed
- Assume a simplified phase space

Equation is approximated with a splitting approach.

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

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

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



Discrete velocities:

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

Approximation of equilibrium state

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

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

Approximation of equilibrium state

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

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

with equilibrium function

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

with $t_\alpha = \frac{1}{9}\left\{4, 1, 1, 1, \frac{1}{4}, \frac{1}{4}, 1, \frac{1}{4}, \frac{1}{4}\right\}$
Pressure $\delta p = \sum_\alpha f_\alpha^{eq} c_s^2 = \rho c_s^2$.
Dev. stress $\Sigma_{ij} = \left(1 - \frac{\omega_L \Delta t}{2}\right) \sum_\alpha \mathbf{e}_{\alpha i} \mathbf{e}_{\alpha j} (f_\alpha^{eq} - f_\alpha)$

Approximation of equilibrium state

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

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

with equilibrium function

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

with $t_\alpha = \frac{1}{9}\left\{3, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \right\}$
Pressure $\delta p = \sum_\alpha f_\alpha^{eq} c_s^2 = \rho c_s^2$.
Dev. stress $\Sigma_{ij} = \left(1 - \frac{\omega_L \Delta t}{2}\right) \sum_\alpha \mathbf{e}_{\alpha i} \mathbf{e}_{\alpha j} (f_\alpha^{eq} - f_\alpha)$

Approximation of equilibrium state

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

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

with equilibrium function

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

with $t_\alpha = \frac{1}{9}\left\{3, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \right\}$
Pressure $\delta p = \sum_\alpha f_\alpha^{eq} c_s^2 = \rho c_s^2$.
Dev. stress $\Sigma_{ij} = \left(1 - \frac{\omega_L \Delta t}{2}\right) \sum_\alpha \mathbf{e}_{\alpha i} \mathbf{e}_{\alpha j} (f_\alpha^{eq} - f_\alpha)$
Is derived by assuming a Maxwell-Boltzmann distribution of f_α^{eq} and approximating the involved exp() function with a Taylor series to second-order accuracy.

Approximation of equilibrium state

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

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

with equilibrium function

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

with $t_\alpha = \frac{1}{9}\left\{3, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \right\}$
Pressure $\delta p = \sum_\alpha f_\alpha^{eq} c_s^2 = \rho c_s^2$.
Dev. stress $\Sigma_{ij} = \left(1 - \frac{\omega_L \Delta t}{2}\right) \sum_\alpha \mathbf{e}_{\alpha i} \mathbf{e}_{\alpha j} (f_\alpha^{eq} - f_\alpha)$
Is derived by assuming a Maxwell-Boltzmann distribution of f_α^{eq} and approximating the involved exp() function with a Taylor series to second-order accuracy.
Using the third-order equilibrium function

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

allows higher flow velocities.

Relation to Navier-Stokes equations

Inserting a Chapman-Enskog expansion, that is,

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

and using

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

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

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

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

Relation to Navier-Stokes equations

Inserting a Chapman-Enskog expansion, that is,

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

and using

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

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

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

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

Kinematic viscosity and collision time are connected by

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

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

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

Turbulence modeling

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

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

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

Turbulence modeling

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

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

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

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 $\bar{f}_\alpha$ 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: $\nu^\star = \nu + \nu_t = \dfrac{1}{3}\left(\dfrac{\tau^\star_L}{\Delta t} - \dfrac{1}{2}\right) c\Delta x \quad \text{with} \quad \tau^\star_L = \tau_L + \tau_t$

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

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

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

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

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

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

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

Turbulence modeling

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

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

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

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

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

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

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

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

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

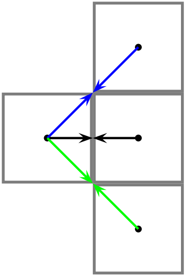
$$\tau_t = \frac{1}{2}\left(\sqrt{\tau_L^2 + 18\sqrt{2}(\rho_0 c^2)^{-1}C_{sm}^2\Delta x\bar{S}} - \tau_L\right)$$

Initial and boundary conditions

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

Boundary conditions (applied before streaming step)

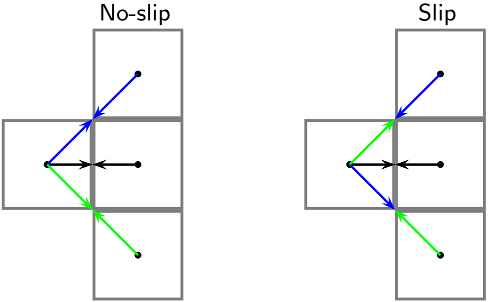
No-slip



Initial and boundary conditions

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

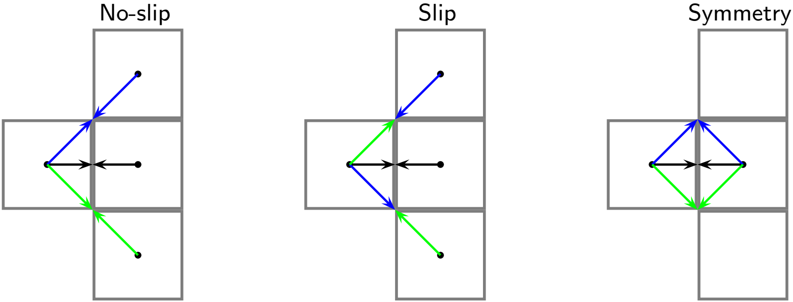
Boundary conditions (applied before streaming step)



Initial and boundary conditions

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

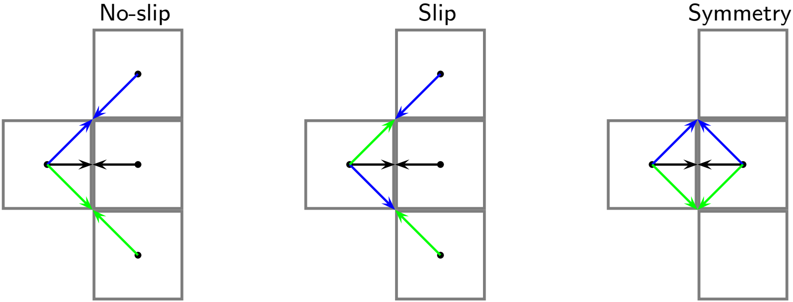
Boundary conditions (applied before streaming step)



Initial and boundary conditions

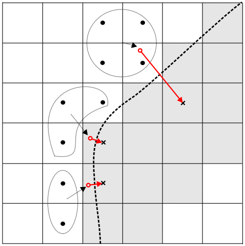
- Initial conditions are constructed as $f_\alpha^{eq}(\rho(t=0), \mathbf{u}(t=0))$

Boundary conditions (applied before streaming step)



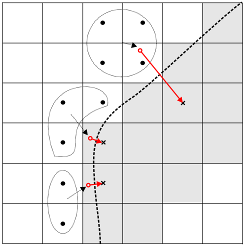
- Outlet basically copies all distributions (zero gradient)
- Inlet and pressure boundary conditions use f_α^{eq}

Level-set method for boundary embedding



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

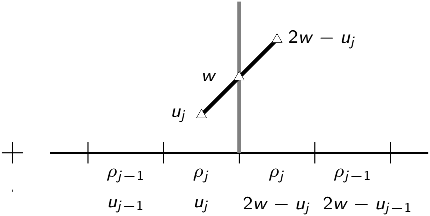
Level-set method for boundary embedding



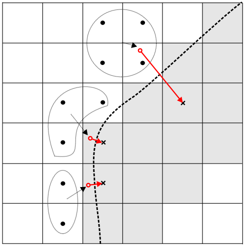
- Implicit boundary representation via distance function φ, normal $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$.
- Complex boundary moving with local velocity $\mathbf{w}$, treat interface as moving rigid wall.
- Construction of macro-values in embedded boundary cells by interpolation / extrapolation.

Interpolate / constant value extrapolate values at

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



Level-set method for boundary embedding



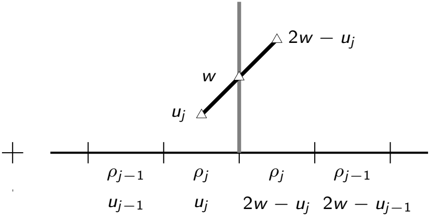
- Implicit boundary representation via distance function φ, normal $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$.
- Complex boundary moving with local velocity $\mathbf{w}$, treat interface as moving rigid wall.
- Construction of macro-values in embedded boundary cells by interpolation / extrapolation.

Interpolate / constant value extrapolate values at

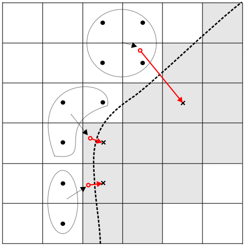
$$\tilde{\mathbf{x}} = \mathbf{x} + 2\varphi\mathbf{n}$$

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

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



Level-set method for boundary embedding



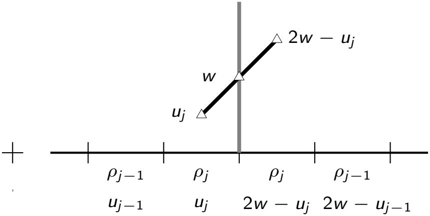
- Implicit boundary representation via distance function φ, normal $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$.
- Complex boundary moving with local velocity $\mathbf{w}$, treat interface as moving rigid wall.
- Construction of macro-values in embedded boundary cells by interpolation / extrapolation.
- Then use $f^{eq}_\alpha(\rho', \mathbf{u}')$ to construct distributions in embedded ghost cells.

Interpolate / constant value extrapolate values at

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

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

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



Normalization

The method is implemented on the unit lattice with $\Delta\tilde{x} = \Delta\tilde{t} = 1$

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

Normalization

The method is implemented on the unit lattice with $\Delta\tilde{x} = \Delta\tilde{t} = 1$

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

Lattice viscosity $\tilde{\nu} = \frac{1}{3}\left(\tau - \frac{1}{2}\right)$ and lattice sound speed $\tilde{c}_s = \frac{1}{\sqrt{3}}$ yield again

$$\omega_L = \frac{\tilde{c}_s^2}{\nu' + \tilde{c}_s^2/2} = \frac{c_s^2}{\nu + \Delta t c_s^2/2}$$

Normalization

The method is implemented on the unit lattice with $\Delta\tilde{x} = \Delta\tilde{t} = 1$

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

Lattice viscosity $\tilde{\nu} = \frac{1}{3}\left(\tau - \frac{1}{2}\right)$ and lattice sound speed $\tilde{c}_s = \frac{1}{\sqrt{3}}$ yield again

$$\omega_L = \frac{\tilde{c}_s^2}{\nu' + \tilde{c}_s^2/2} = \frac{c_s^2}{\nu + \Delta t c_s^2/2}$$

Velocity normalization factor: $u_0 = \frac{l_0}{t_0}$, density ρ_0

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

Normalization

The method is implemented on the unit lattice with $\Delta\tilde{x} = \Delta\tilde{t} = 1$

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

Lattice viscosity $\tilde{\nu} = \frac{1}{3}\left(\tau - \frac{1}{2}\right)$ and lattice sound speed $\tilde{c}_s = \frac{1}{\sqrt{3}}$ yield again

$$\omega_L = \frac{\tilde{c}_s^2}{\nu' + \tilde{c}_s^2/2} = \frac{c_s^2}{\nu + \Delta t c_s^2/2}$$

Velocity normalization factor: $u_0 = \frac{l_0}{t_0}$, density ρ_0

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

Trick for scheme acceleration: Use $\bar{u} = Su$ and $\bar{\nu} = S\nu$ which yields

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

Normalization

The method is implemented on the unit lattice with $\Delta\tilde{x} = \Delta\tilde{t} = 1$

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

Lattice viscosity $\tilde{\nu} = \frac{1}{3}\left(\tau - \frac{1}{2}\right)$ and lattice sound speed $\tilde{c}_s = \frac{1}{\sqrt{3}}$ yield again

$$\omega_L = \frac{\tilde{c}_s^2}{\nu' + \tilde{c}_s^2/2} = \frac{c_s^2}{\nu + \Delta t c_s^2/2}$$

Velocity normalization factor: $u_0 = \frac{l_0}{t_0}$, density ρ_0

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

Trick for scheme acceleration: Use $\bar{u} = Su$ and $\bar{\nu} = S\nu$ which yields

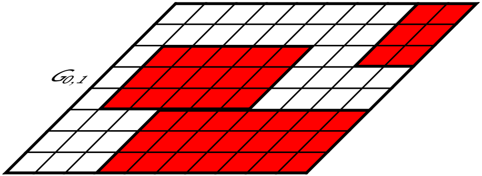
$$\bar{\omega}_L = \frac{c_s^2}{S\nu + \Delta t/S\, c_s^2/2}$$

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

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

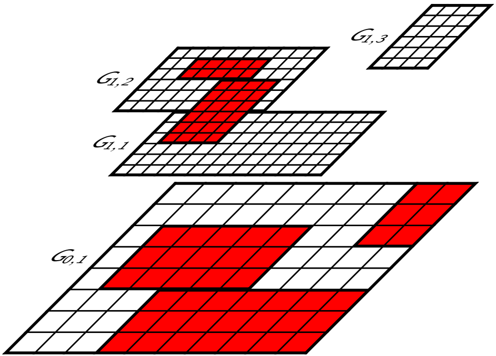
Block-structured adaptive mesh refinement (SAMR)

- Refined blocks overlay coarser ones



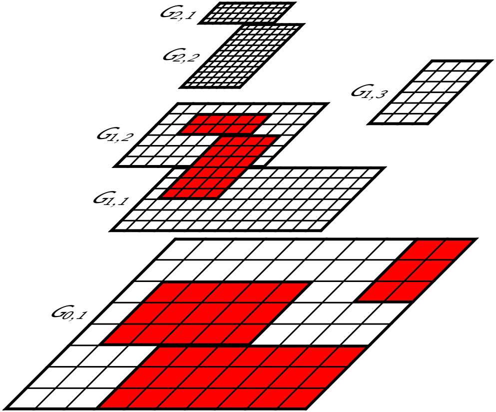
Block-structured adaptive mesh refinement (SAMR)

- Refined blocks overlay coarser ones



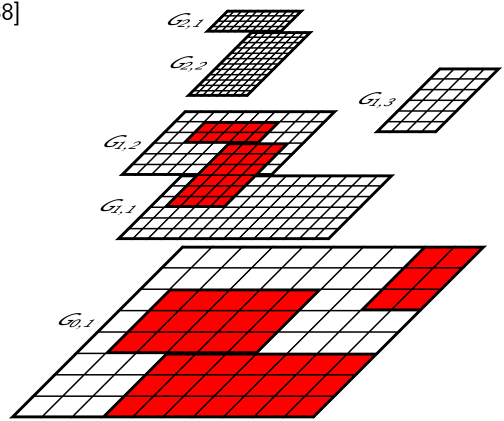
Block-structured adaptive mesh refinement (SAMR)

- Refined blocks overlay coarser ones



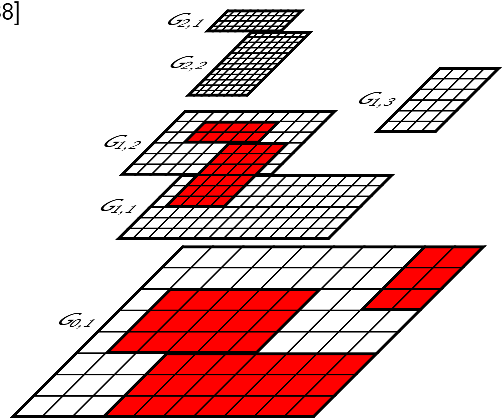
Block-structured adaptive mesh refinement (SAMR)

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



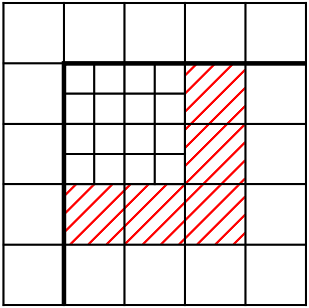
Block-structured adaptive mesh refinement (SAMR)

- Refined blocks overlay coarser ones
- Recursive refinement in space *and time* by factor r_l [Berger and Colella, 1988] ideal for LBM
- Block (aka patch) based data structures
- Most efficient LBM implementation with patch-wise for-loops
- LBM implemented on finite volume grids
- AMROC V3.0 with significantly enhanced parallelization.
- Papers: [Deiterding, 2011, Deiterding et al., 2007, Deiterding et al., 2006]



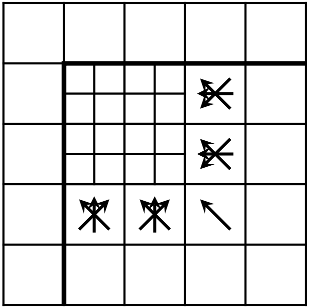
Adaptive LBM

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



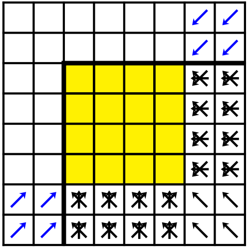
Adaptive LBM

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

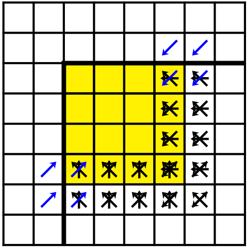
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.



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

Adaptive LBM

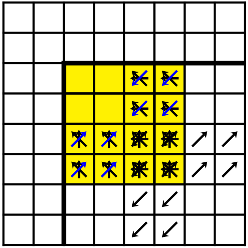
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



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

Adaptive LBM

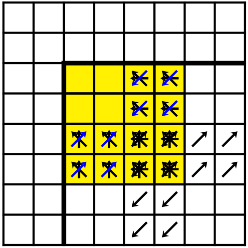
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



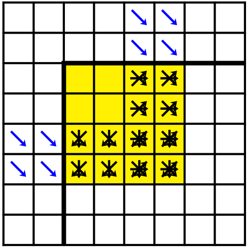
$\tilde{f}_{\alpha,in}^{f,n+1/2}$

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.



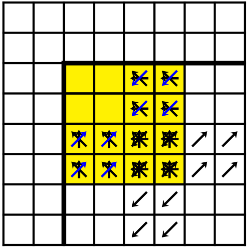
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



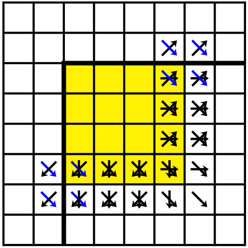
$f_{\alpha,out}^{f,n}$

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.



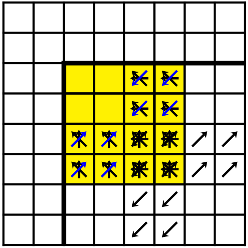
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



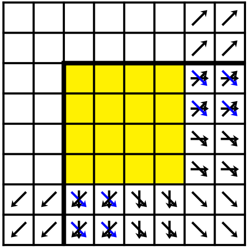
$\tilde{f}_{\alpha,out}^{f,n}$

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.



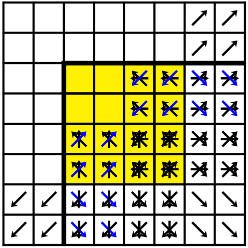
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



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

Adaptive LBM

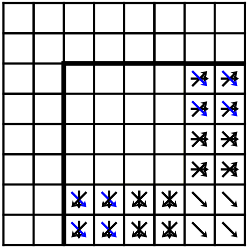
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

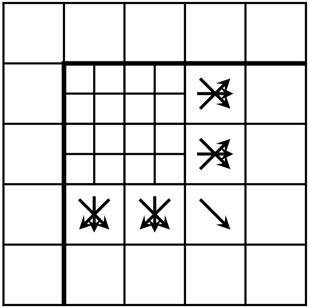


$\tilde{f}_{\alpha,out}^{f,n+1/2}$, $\tilde{f}_{\alpha,out}^{f,n}$

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

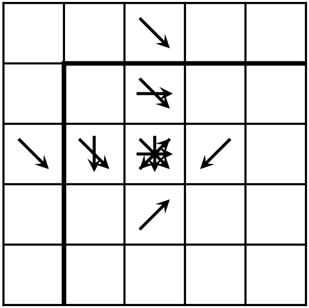
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

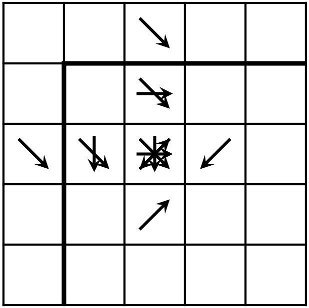
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

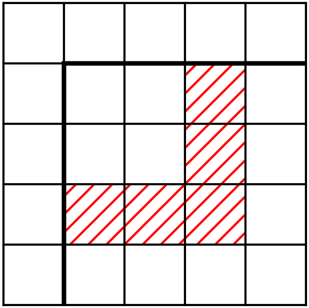
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

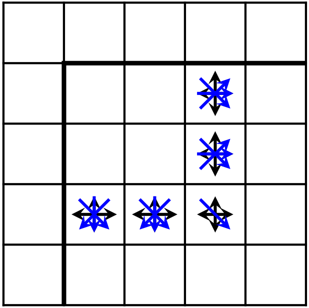
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

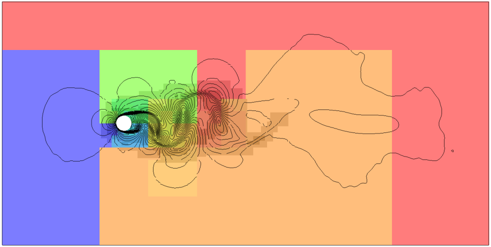


5. Average $\tilde{f}_{\alpha,out}^{f,n+1/2}$ (inner halo layer), $\tilde{f}_{\alpha,out}^{f,n}$ (outer halo layer) to obtain $\tilde{f}_{\alpha,out}^{C,n}$.
6. Revert transport into halos: $\bar{f}_{\alpha,out}^{C,n} := \mathcal{T}^{-1}(\tilde{f}_{\alpha,out}^{C,n})$
7. Parallel synchronization of $f_\alpha^{C,n},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

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

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



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

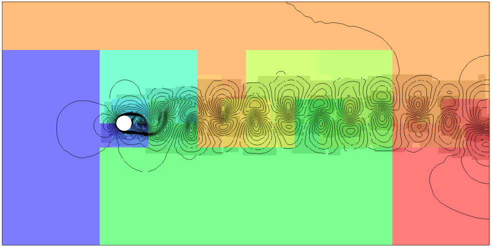
Flow over 2D cylinder, $d = 2\,\mathrm{cm}$

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

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

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

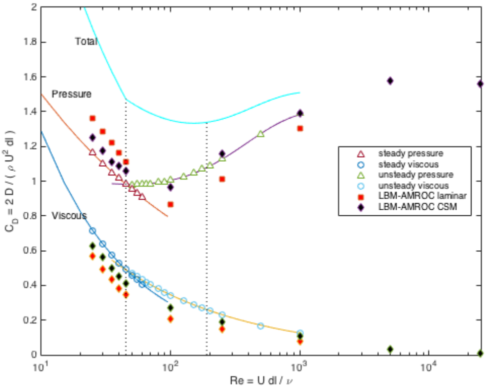
- Air with $\nu = 1.61 \cdot 10^{-5}\,\mathrm{m}^2/\mathrm{s}$, $\rho = 1.205\,\mathrm{kg}/\mathrm{m}^3$
- Domain size $[-8d, 24d] \times [-8d, 8d]$
- Dynamic refinement based on velocity. Last level to refine structure further.
- Inflow from left. Characteristic boundary conditions [Schlaffer, 2013]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



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

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

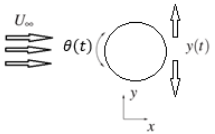
- Air with $\nu = 1.61 \cdot 10^{-5}\,\mathrm{m}^2/\mathrm{s}$, $\rho = 1.205\,\mathrm{kg}/\mathrm{m}^3$
- Domain size $[-8d, 24d] \times [-8d, 8d]$
- Dynamic refinement based on velocity. Last level to refine structure further.
- Inflow from left. Characteristic boundary conditions [Schlaffer, 2013]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



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

Oscillating cylinder – 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

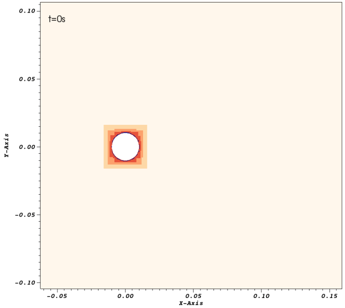
$$y(t) = A_t \sin(2\pi f_t t), \qquad \theta(t) = A_\theta \sin(2\pi f_\theta t)$$

- Setup follows [Nazarinia et al., 2012]. 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 $c_s = 1482\,\mathrm{m/s}$, $\nu = 9.167 \cdot 10^{-7}\,\mathrm{m^2/s}$, $\rho = 1016\,\mathrm{kg/m^3}$
- Constant coefficient model deactivated for Case 1, active for Case 2 with $C_{sm} = 0.2$

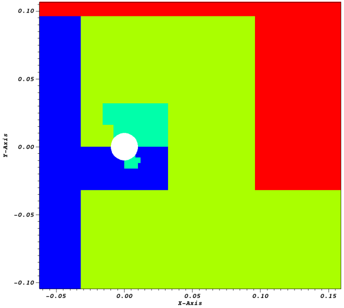
C. Laloglu, RD. Proc. 5th Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering, Civil-Comp Press, 2017.

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

Mesh refinement



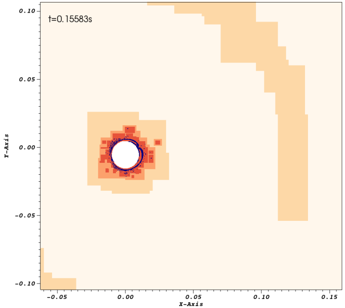
Distribution to 4 processors



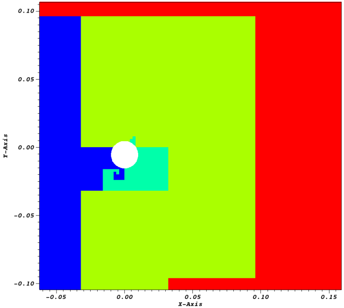
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



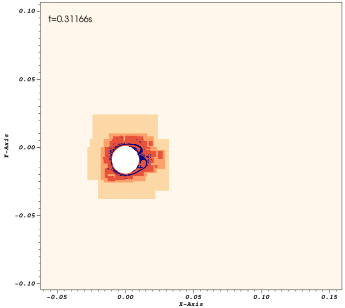
Distribution to 4 processors



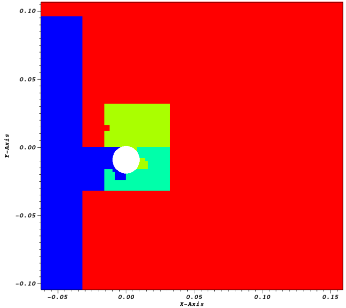
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



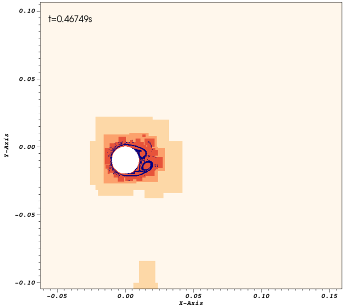
Distribution to 4 processors



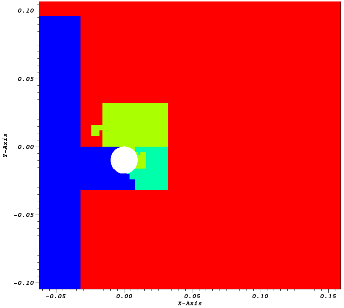
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



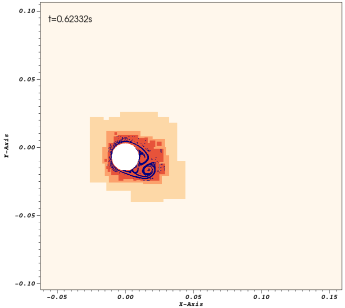
Distribution to 4 processors



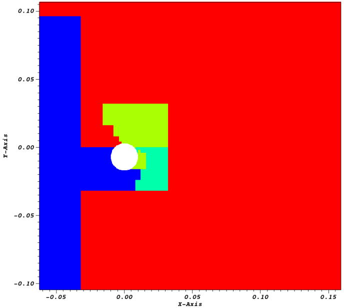
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



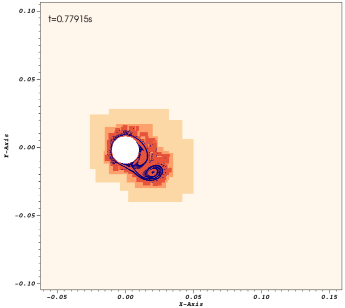
Distribution to 4 processors



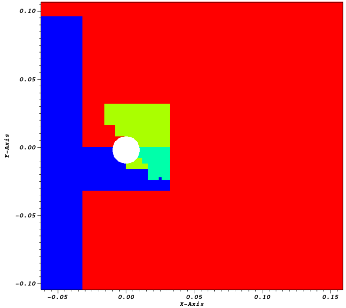
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



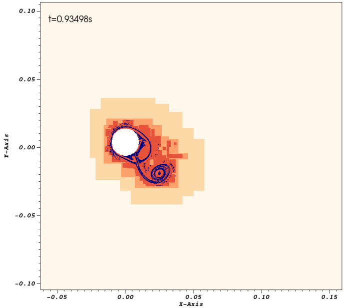
Distribution to 4 processors



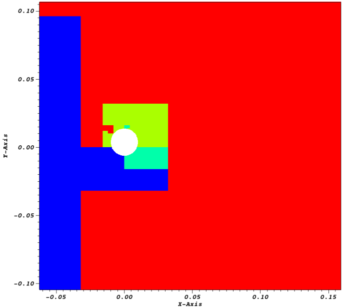
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



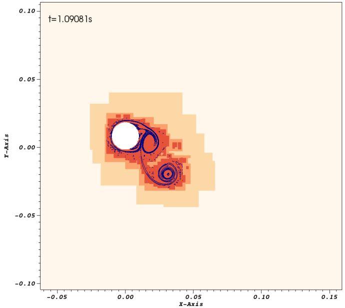
Distribution to 4 processors



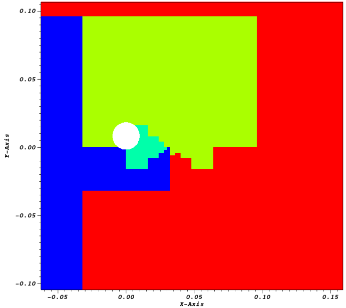
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



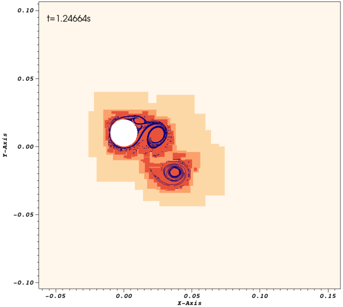
Distribution to 4 processors



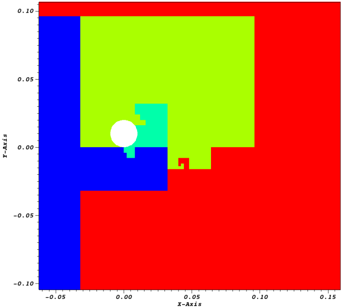
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



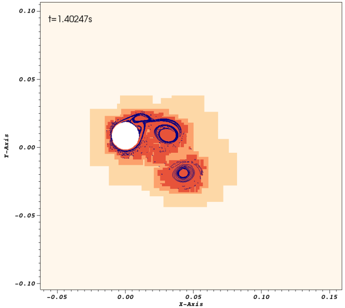
Distribution to 4 processors



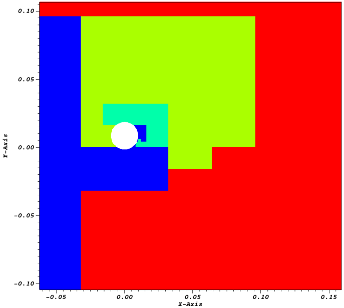
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



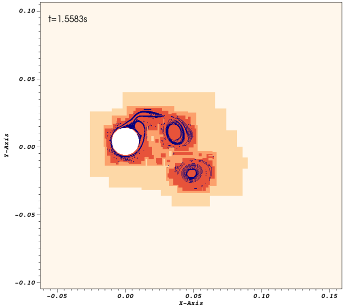
Distribution to 4 processors



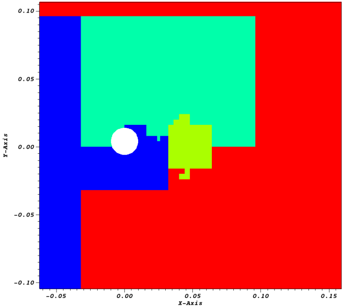
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



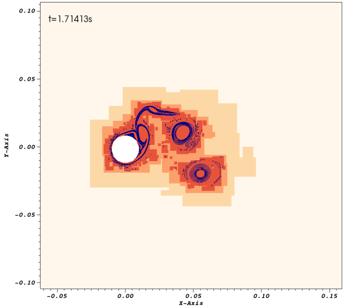
Distribution to 4 processors



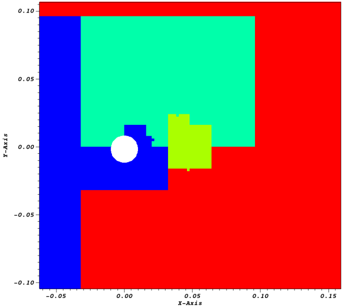
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



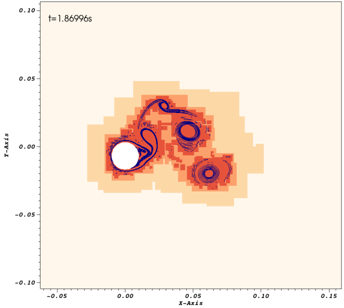
Distribution to 4 processors



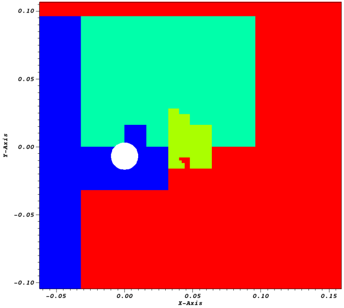
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



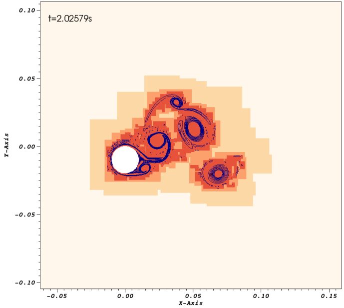
Distribution to 4 processors



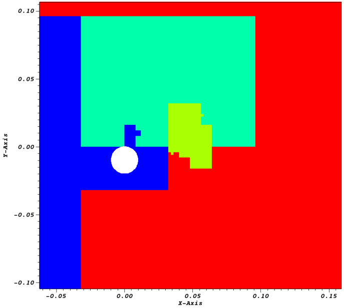
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



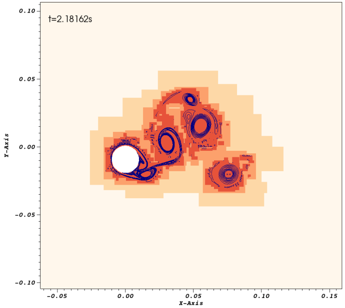
Distribution to 4 processors



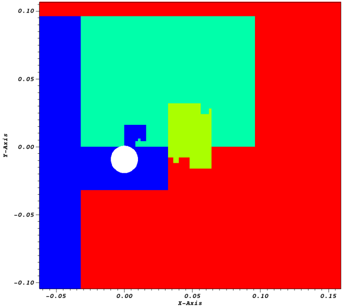
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



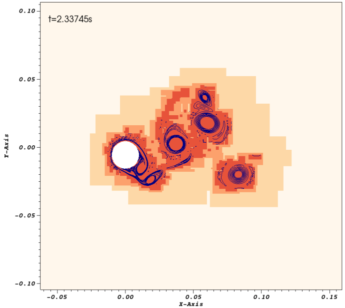
Distribution to 4 processors



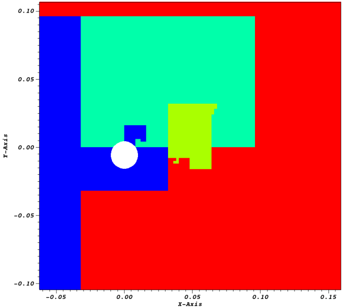
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



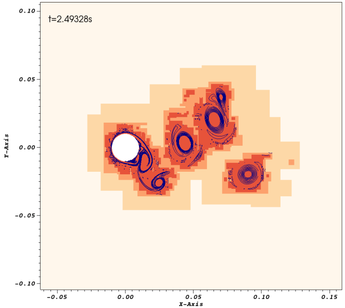
Distribution to 4 processors



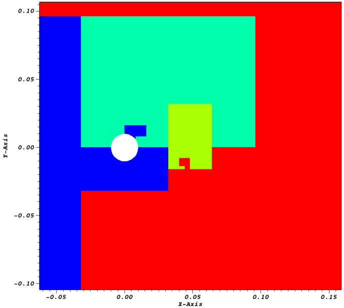
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



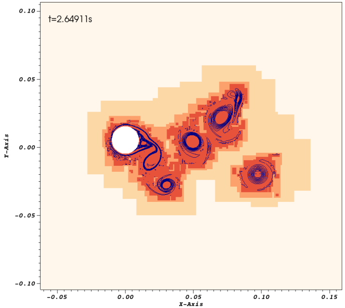
Distribution to 4 processors



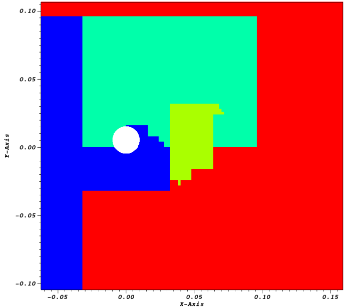
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



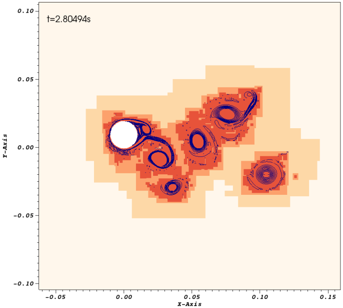
Distribution to 4 processors



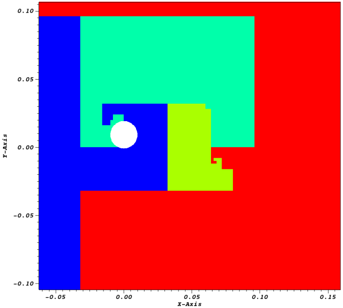
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



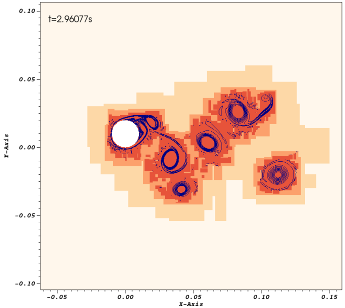
Distribution to 4 processors



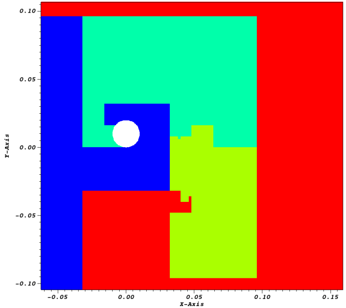
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



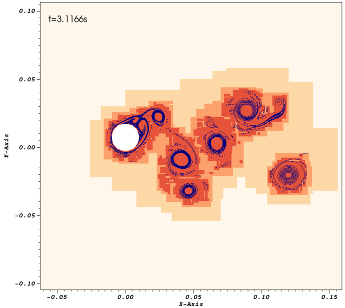
Distribution to 4 processors



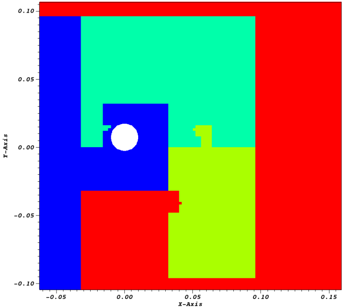
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



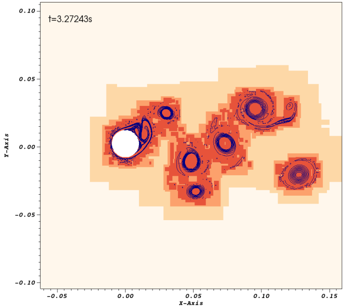
Distribution to 4 processors



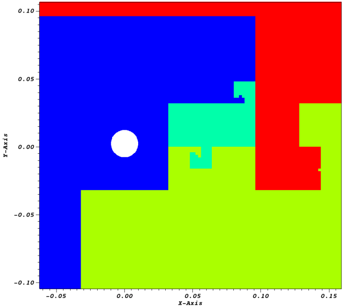
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



Distribution to 4 processors

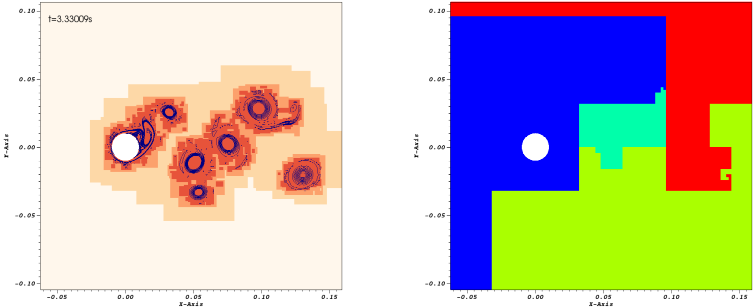


- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement

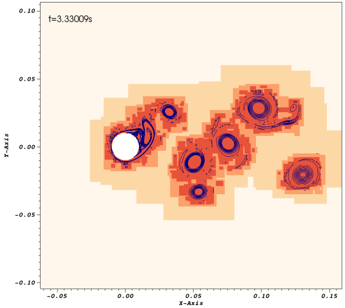
Distribution to 4 processors



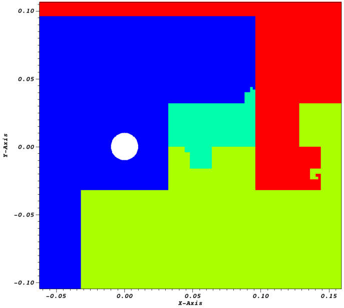
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$

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

Mesh refinement



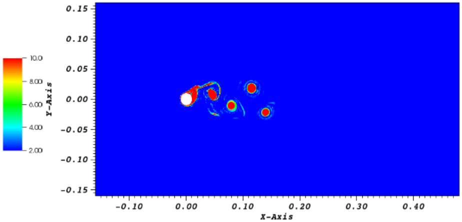
Distribution to 4 processors



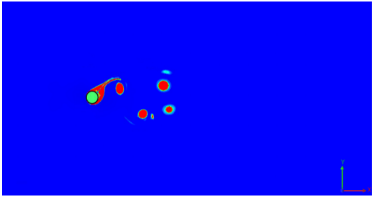
- Visualization enlargement of cylinder region
- Base mesh is discretized with 320×160 cells, 3 additional levels with factor $r_l = 2, 2, 2$
- 80 cells within D on highest level
- Speedup $S = 2000$
- Basically identical setup in commercial code XFlow for comparison

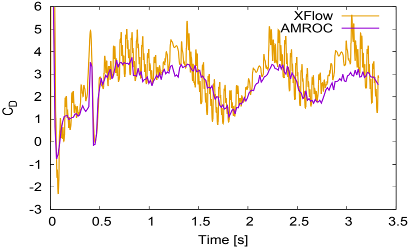
Case 1b, $V_R = 1$, $f_t = f_\theta = 0.6$, $\mathrm{Re} = 1322$

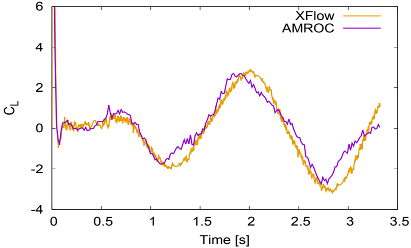
AMROC



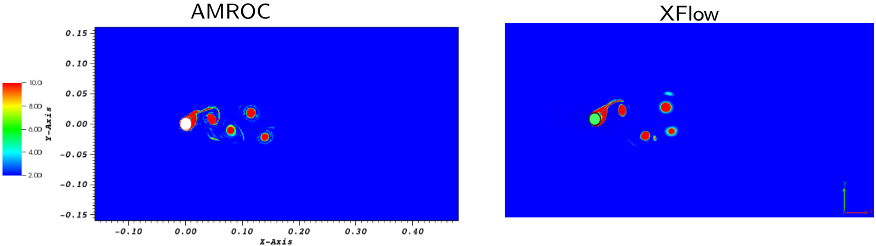
XFlow

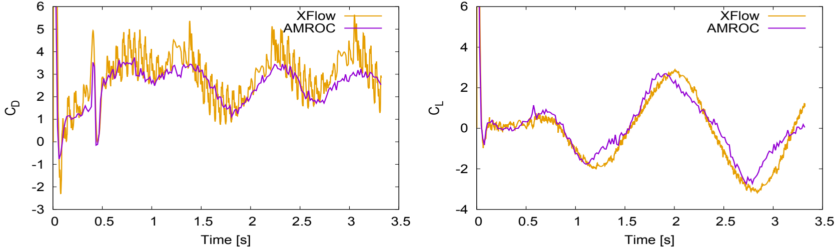






Case 1b, $V_R = 1$, $f_t = f_\theta = 0.6$, $\mathrm{Re} = 1322$

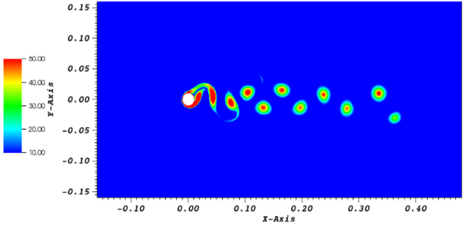




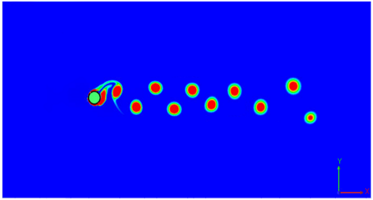
- Increase of rotational velocity leads to formation of a vortex pair plus single vortex. Drag and lift amplitude roughly doubled.
- Laminar results in good agreement with experiments of [Nazarinia et al., 2012].

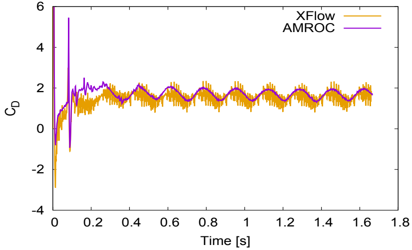
Case 2a, $V_R = 0.5$, $f_t = f_\theta = 3$, $\mathrm{Re} = 6310$

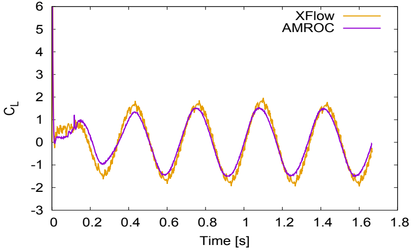
AMROC



XFlow

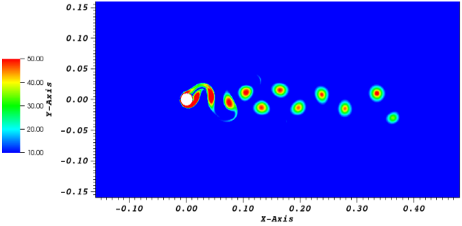




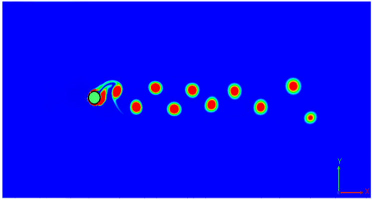


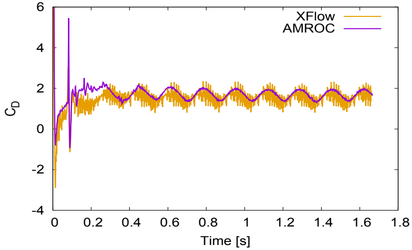
Case 2a, $V_R = 0.5,\ f_t = f_\theta = 3,\ \mathrm{Re} = 6310$

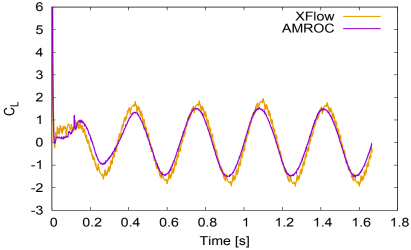
AMROC



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

An LBM for thermal transport

Consider the Navier-Stokes equations under Boussinesq approximation

$$\nabla \cdot \mathbf{u} = 0$$
$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F}$$
$$\frac{\partial T}{\partial t} + \nabla \cdot (\mathbf{u} T) = \mathcal{D} \nabla^2 T$$

with $\mathbf{F} = \mathbf{g}\beta\,(T - T_{ref})$.

An LBM for thermal transport

Consider the Navier-Stokes equations under Boussinesq approximation

$$\nabla \cdot \mathbf{u} = 0$$
$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F}$$
$$\frac{\partial T}{\partial t} + \nabla \cdot (\mathbf{u} T) = \mathcal{D} \nabla^2 T$$

with $\mathbf{F} = \mathbf{g}\beta\left(T - T_{ref}\right)$.
An LBM for this system needs to use two distribution functions f_α and g_α.
1.) Transport step $\mathcal{T}$:

$$\tilde{f}_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}, t), \quad \tilde{g}_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = g_\alpha(\mathbf{x}, t)$$

An LBM for thermal transport

Consider the Navier-Stokes equations under Boussinesq approximation

$$\nabla \cdot \mathbf{u} = 0$$
$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F}$$
$$\frac{\partial T}{\partial t} + \nabla \cdot (\mathbf{u} T) = \mathcal{D} \nabla^2 T$$

with $\mathbf{F} = \mathbf{g}\beta\,(T - T_{ref})$.

An LBM for this system needs to use two distribution functions f_α and g_α.

1.) Transport step $\mathcal{T}$:

$$\tilde{f}_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}, t), \quad \tilde{g}_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = g_\alpha(\mathbf{x}, t)$$

2.) Collision step $\mathcal{C}$:

$$f_\alpha(\cdot, t + \Delta t) = \tilde{f}_\alpha(\cdot, t + \Delta t) + \omega_{L,\nu} \Delta t \left(\tilde{f}^{eq}_\alpha(\cdot, t + \Delta t) - \tilde{f}_\alpha(\cdot, t + \Delta t) \right) + \Delta t \mathbf{F}_\alpha$$

$$g_\alpha(\cdot, t + \Delta t) = \tilde{g}_\alpha(\cdot, t + \Delta t) + \omega_{L,\mathcal{D}} \Delta t \left(\tilde{g}^{eq}_\alpha(\cdot, t + \Delta t) - \tilde{g}_\alpha(\cdot, t + \Delta t) \right)$$

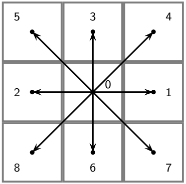
with collision frequencies

$$\omega_{L,\nu} = \frac{c_s^2}{\nu + c_s^2 \Delta t/2}, \quad \omega_{L,\mathcal{D}} = \frac{\frac{3}{2} c_s^2}{\mathcal{D} + \frac{3}{2} c_s^2 \Delta t/2}$$

Equilibrium operators

This incompressible method uses in 2D [Guo et al., 2002]

$$f_\alpha^{(eq)} = \begin{cases} -4\sigma_0 p - s_\alpha(\mathbf{u}), & \text{for } \alpha = 0, \\ \sigma_\alpha p + s_\alpha(\mathbf{u}), & \text{for } \alpha = 1, \ldots, 8, \end{cases}$$



where

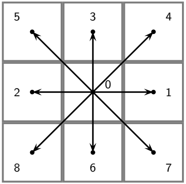
$$s_\alpha(\mathbf{u}) = t_\alpha \left[\frac{3\mathbf{e}_\alpha \mathbf{u}}{c^2} + \frac{9(\mathbf{e}_\alpha \mathbf{u})^2}{2c^4} - \frac{3\mathbf{u}^2}{2c^2} \right]$$

with $t_\alpha = \frac{1}{9}\left\{4, 1, 1, 1, \frac{1}{4}, \frac{1}{4}, 1, \frac{1}{4}, \frac{1}{4}\right\}$ and $\sigma_\alpha = \frac{1}{3}\left\{-5, 1, 1, 1, \frac{1}{4}, \frac{1}{4}, 1, \frac{1}{4}, \frac{1}{4}\right\}$

Equilibrium operators

This incompressible method uses in 2D [Guo et al., 2002]

$$f_\alpha^{(eq)} = \begin{cases} -4\sigma_0 p - s_\alpha(\mathbf{u}), & \text{for } \alpha = 0, \\ \sigma_\alpha p + s_\alpha(\mathbf{u}), & \text{for } \alpha = 1, \dots, 8, \end{cases}$$



where

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

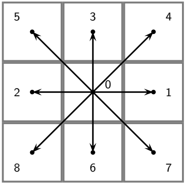
with $t_\alpha = \frac{1}{9}\left\{4, 1, 1, 1, \frac{1}{4}, \frac{1}{4}, 1, \frac{1}{4}, \frac{1}{4}\right\}$ and $\sigma_\alpha = \frac{1}{3}\left\{-5, 1, 1, 1, \frac{1}{4}, \frac{1}{4}, 1, \frac{1}{4}, \frac{1}{4}\right\}$

$$g_\alpha^{(eq)} = \frac{T}{4}\left[1 + 2\mathbf{e}_\alpha \cdot \mathbf{u}\right] \quad \text{for } \alpha = 1, \dots, 4$$

Equilibrium operators

This incompressible method uses in 2D [Guo et al., 2002]

$$f_\alpha^{(eq)} = \begin{cases} -4\sigma_0 p - s_\alpha(\mathbf{u}), & \text{for } \alpha = 0, \\ \sigma_\alpha p + s_\alpha(\mathbf{u}), & \text{for } \alpha = 1, \dots, 8, \end{cases}$$



where

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

with $t_\alpha = \frac{1}{9}\left\{4, 1, 1, 1, \frac{1}{4}, \frac{1}{4}, 1, \frac{1}{4}, \frac{1}{4}\right\}$ and $\sigma_\alpha = \frac{1}{3}\left\{-5, 1, 1, 1, \frac{1}{4}, \frac{1}{4}, 1, \frac{1}{4}, \frac{1}{4}\right\}$

$$g_\alpha^{(eq)} = \frac{T}{4}\left[1 + 2\mathbf{e}_\alpha \cdot \mathbf{u}\right] \quad \text{for } \alpha = 1, \dots, 4$$

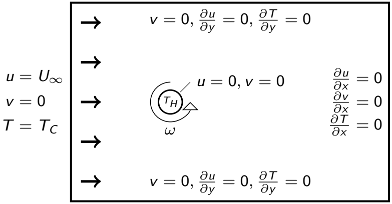
Forces are applied in y-direction only:

$$F_\alpha = \frac{1}{2}\left(\delta_{i3} - \delta_{i6}\right)\mathbf{e}_i \cdot \mathbf{F}$$

Moments: $\mathbf{u} = \sum_{\alpha>0} \mathbf{e}_i f_\alpha, \quad p = \frac{1}{4\sigma}\left[\sum_{\alpha>0} f_\alpha + s_0(\mathbf{u})\right], \quad T = \sum_{\alpha=1}^{4} g_\alpha$

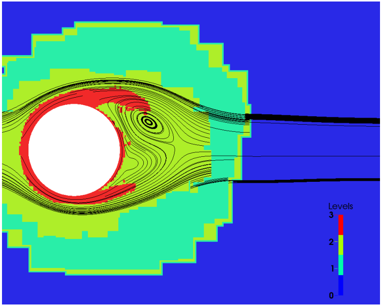
Heated rotating cylinder

- $R = 15$, domain: $[-6R, 16R] \times [-8R, 8R]$
- $\mathrm{Re} = 2U_\infty R/\nu = 200$, $U_\infty = 0.01$
- Peripheral velocity $V = \Omega R$, $V/U_\infty = 0.5$
- Base grid 288×240 refined by three levels with $r_1 = 2$, $r_{2,3} = 4$ using scaled gradients of u, v, T

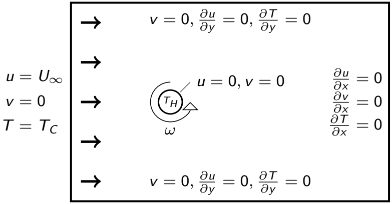


Heated rotating cylinder

- $R = 15$, domain: $[-6R, 16R] \times [-8R, 8R]$
- $\mathrm{Re} = 2U_\infty R/\nu = 200$, $U_\infty = 0.01$
- Peripheral velocity $V = \Omega R$, $V/U_\infty = 0.5$
- Base grid 288×240 refined by three levels with $r_1 = 2$, $r_{2,3} = 4$ using scaled gradients of u, v, T

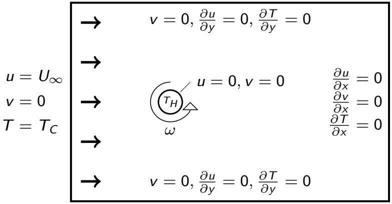


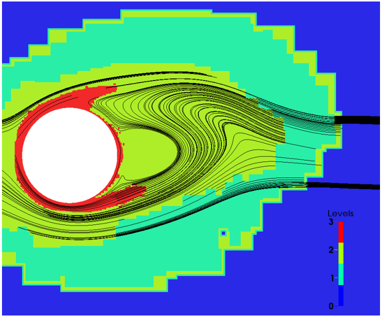
$t = 3$



Heated rotating cylinder

- $R = 15$, domain: $[-6R, 16R] \times [-8R, 8R]$
- $\mathrm{Re} = 2U_\infty R/\nu = 200$, $U_\infty = 0.01$
- Peripheral velocity $V = \Omega R$, $V/U_\infty = 0.5$
- Base grid 288×240 refined by three levels with $r_1 = 2$, $r_{2,3} = 4$ using scaled gradients of u, v, T

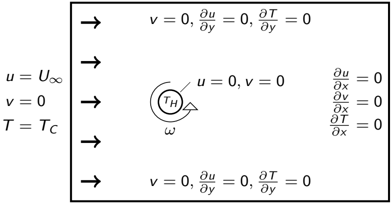


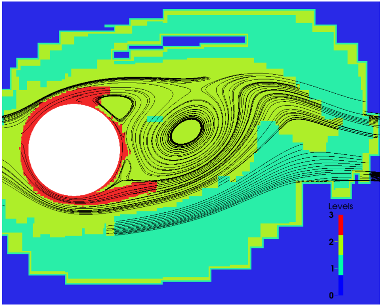


$t = 6$

Heated rotating cylinder

- $R = 15$, domain: $[-6R, 16R] \times [-8R, 8R]$
- $\mathrm{Re} = 2U_\infty R/\nu = 200$, $U_\infty = 0.01$
- Peripheral velocity $V = \Omega R$, $V/U_\infty = 0.5$
- Base grid 288×240 refined by three levels with $r_1 = 2$, $r_{2,3} = 4$ using scaled gradients of u, v, T

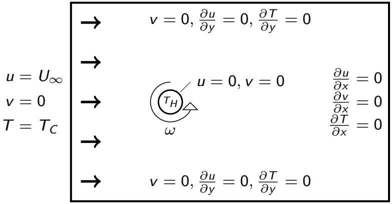


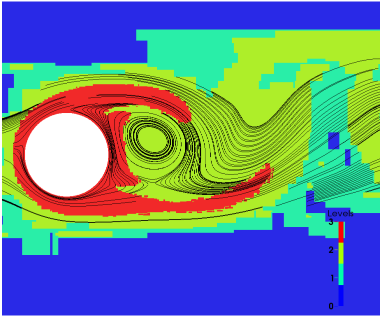


$t = 8$

Heated rotating cylinder

- $R = 15$, domain: $[-6R, 16R] \times [-8R, 8R]$
- $\mathrm{Re} = 2U_\infty R/\nu = 200$, $U_\infty = 0.01$
- Peripheral velocity $V = \Omega R$, $V/U_\infty = 0.5$
- Base grid 288×240 refined by three levels with $r_1 = 2$, $r_{2,3} = 4$ using scaled gradients of u, v, T

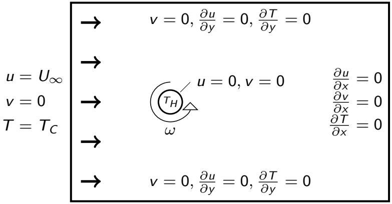


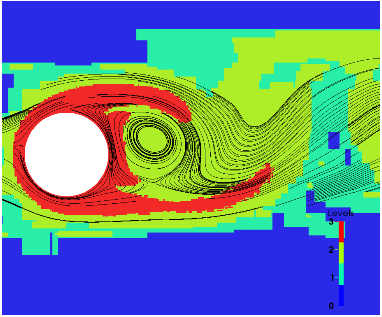


$t = 12$

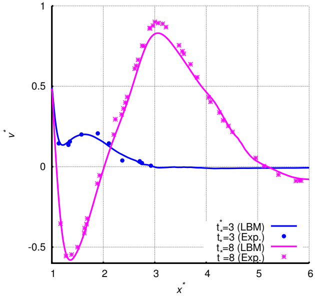
Heated rotating cylinder

- $R = 15$, domain: $[-6R, 16R] \times [-8R, 8R]$
- $\mathrm{Re} = 2U_\infty R/\nu = 200$, $U_\infty = 0.01$
- Peripheral velocity $V = \Omega R$, $V/U_\infty = 0.5$
- Base grid 288×240 refined by three levels with $r_1 = 2$, $r_{2,3} = 4$ using scaled gradients of u, v, T





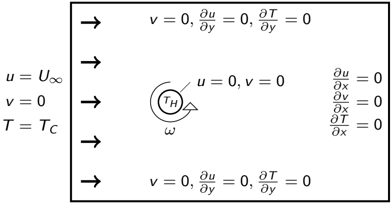
$t = 12$

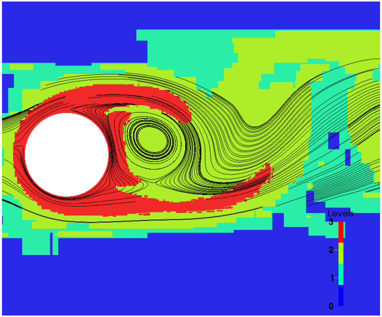


v velocity along x-axis

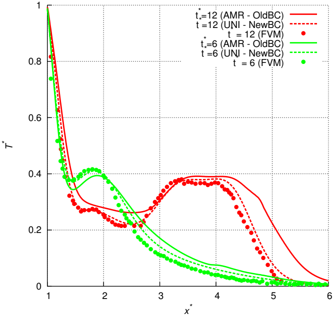
Heated rotating cylinder

- $R = 15$, domain: $[-6R, 16R] \times [-8R, 8R]$
- $\mathrm{Re} = 2U_\infty R/\nu = 200$, $U_\infty = 0.01$
- Peripheral velocity $V = \Omega R$, $V/U_\infty = 0.5$
- Base grid 288×240 refined by three levels with $r_1 = 2$, $r_{2,3} = 4$ using scaled gradients of u, v, T





$t = 12$



Temperature T along x-axis

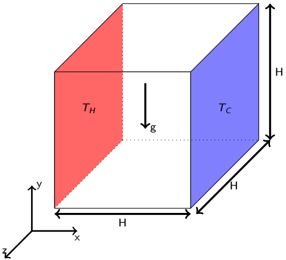
Natural convection

Characterized by

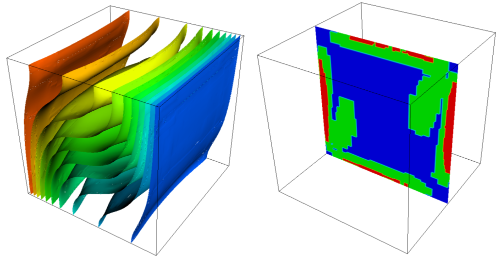
$$\mathrm{Ra} = \frac{g\beta\Delta TH^3}{\nu\mathcal{D}}$$

a = AMROC-LBM,
b = [Fusegi et al., 1991] (NS - uniform)

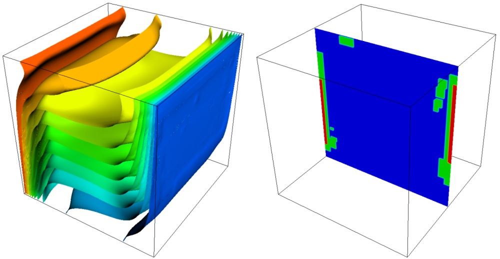
	Ref.	$u_{\max}$	$y_{\max}$	$v_{\max}$	$x_{\max}$	$\mathrm{Nu}_{\mathrm{ave}}$
$\mathrm{Ra} = 10^3$	a	0.132	0.195	0.132	0.829	1.099
	b	0.131	0.200	0.132	0.833	1.105
$\mathrm{Ra} = 10^4$	a	0.197	0.194	0.220	0.887	2.270
	b	0.201	0.183	0.225	0.883	2.302
$\mathrm{Ra} = 10^5$	a	0.141	0.152	0.242	0.935	4.583
	b	0.147	0.145	0.247	0.935	4.646



Isosurfaces of temperature and refinement levels



$\mathrm{Ra} = 10^4$



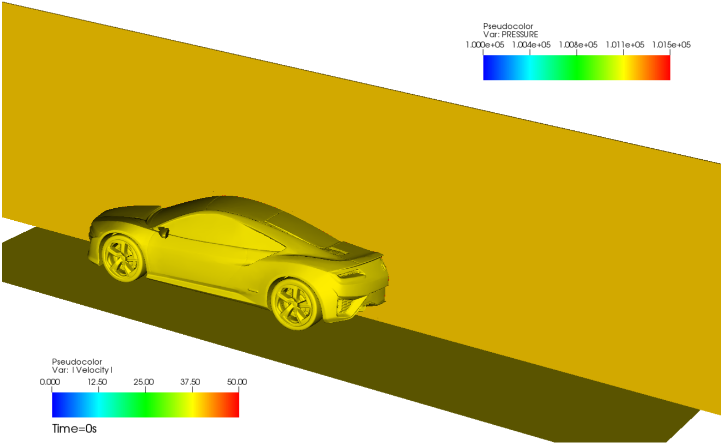
$\mathrm{Ra} = 10^5$

K. Feldhusen, RD, C. Wagner. J. Applied Math. Comp. Science 26(4): 735–747, 2016.

Outline

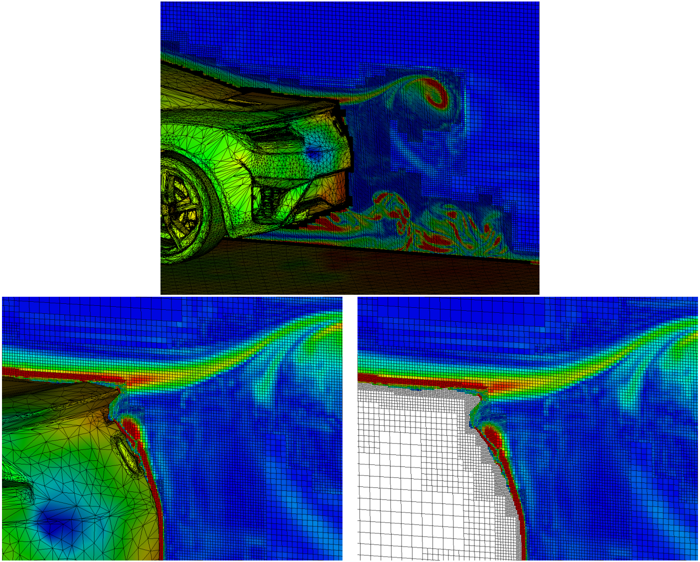
Wind tunnel simulation of a prototype car

Fluid velocity and pressure on vehicle



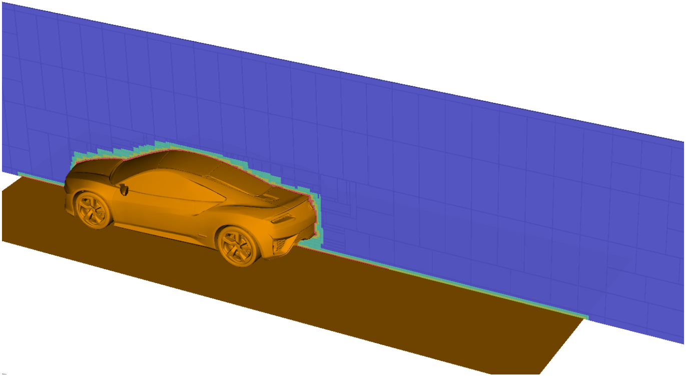
- Inflow 40 $\mathrm{m/s}$. LES model active. Characteristic boundary conditions.
- To $t = 0.5\,\mathrm{s}$ ($\sim$ 4 characteristic lengths) with 31,416 time steps on finest level in $\sim$ 37 h on 200 cores (7389 h CPU). Channel: 15 m $\times$ 5 m $\times$ 3.3 m

Mesh adaptation



Mesh adaptation

Used refinement blocks and levels (indicated by color)



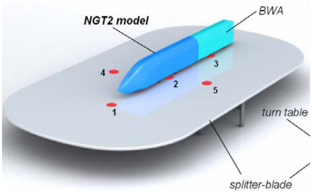
- SAMR base grid 600 × 200 × 132 cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 3.125\,\mathrm{mm}$
- Adaptation based on level set and scaled gradient of magnitude of vorticity vector
- 236M cells vs. 8.1 billion (uniform) at $t = 0.4075\,\mathrm{s}$

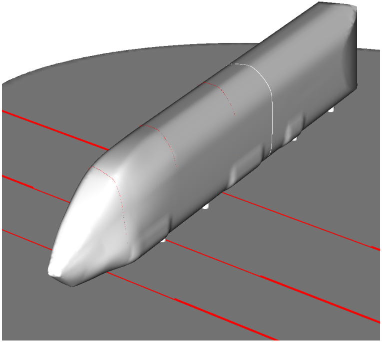
Refinement at $t = 0.4075\,\mathrm{s}$

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

Next Generation Train (NGT)

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

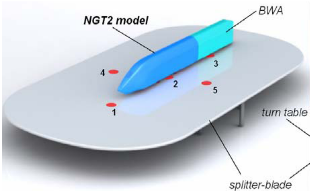


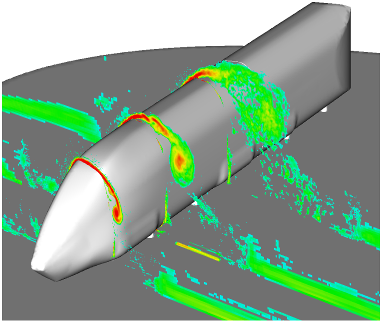


M. M. Fragner, RD. Int. J. Comput. Fluid Dynamics, 30(6): 402–407, 2016.

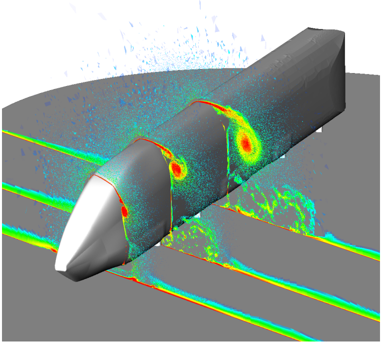
Next Generation Train (NGT)

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





Averaged vorticity LBM-LES



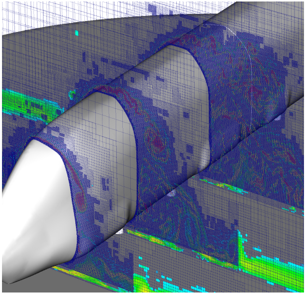
Averaged vorticity OpenFOAM-LES

M. M. Fragner, RD. Int. J. Comput. Fluid Dynamics, 30(6): 402–407, 2016.

NGT model

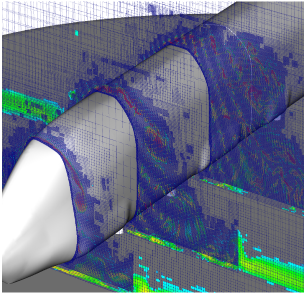
- LBM-AMR computation with 5 additional levels, factor 2 refinement (uniform: 120.4e9 cells)
- Dynamic AMR until $T_c = 34$, then static for $\sim 12 T_C$ to obtain average coefficients
- OpenFOAM simulations by M. Fragner (DLR)

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



NGT model

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

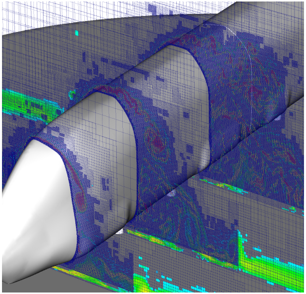


Simulation	Mesh	CFX	CFY	CMX
Wind tunnel	–	-0.06	-5.28	-3.46
DDES	low	-0.40	-5.45	-3.61
Σ only	low	0.10	-0.04	-0.05
LES	high	-0.45	-6.07	-4.14
DDES	high	-0.43	-5.72	-3.77
LBM - p only	–	-0.30	-5.09	-3.46

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

NGT model

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



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

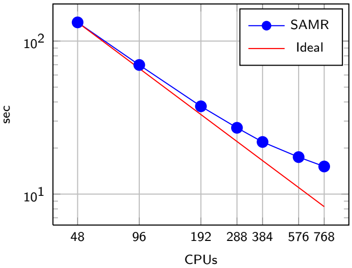
	LBM	DDES(l)	LES	DDES(h)
Cells	147M	34.1M	219M	219M
y^+	43	3.2	1.7	1.7
x^+, z^+	43	313	140	140
Δx wake [mm]	0.936	3.0	1.5	1.5
Runtime [T_C]	34	35.7	10.3	9.2
Processors	200	80	280	280
CPU [h]	34,680	49,732	194,483	164,472
$T_C/\Delta t$	1790	1325	1695	1695
CPU [h]/T_C/1M cells	5.61	39.75	86.4	81.36

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

Strong scalability test (1:25 train)

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

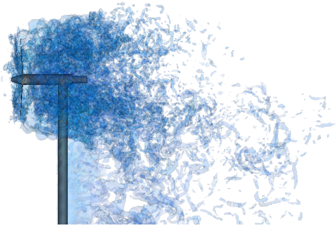
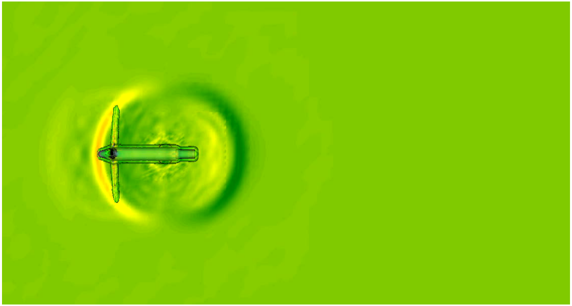
Time per coarse level step



Time in % spent in main operations

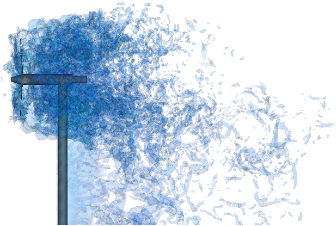
Cores	48	96	192	288	384	576	768
Time per step	132.43s	69.79s	37.47s	27.12s	21.91s	17.45s	15.15s
Par. Efficiency	100.0	94.88	88.36	81.40	75.56	63.24	54.63
LBM Update	5.91	5.61	5.38	4.92	4.50	3.73	3.19
Regridding	15.44	12.02	11.38	10.92	10.02	8.94	8.24
Partitioning	4.16	2.43	1.16	1.02	1.04	1.16	1.34
Interpolation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

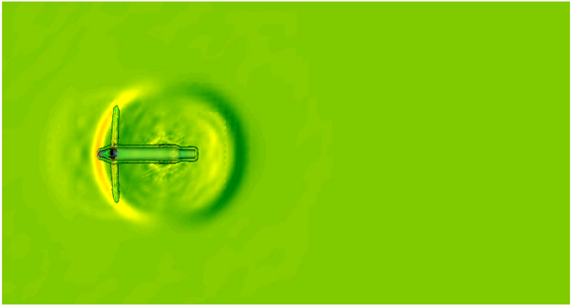
Mexico experimental turbine – 0^o inflow

- Setup and measurements by Energy Research Centre of the Netherlands (ECN) and the Technical University of Denmark (DTU) [Schepers and Boorsma, 2012]
- Inflow velocity 14.93 $\mathrm{m/s}$ in wind tunnel of 9.5 $\mathrm{m} \times$ 9.5 m cross section.
- Rotor diameter $D = 4.5\,\mathrm{m}$. Prescribed motion with 424.5 rpm: tip speed 100 $\mathrm{m/s}$, $\mathrm{Re}_r \approx 75839$ TSR 6.70
- Simulation with three additional levels with factors 2, 2, 4. Resolution of rotor and tower $\Delta x = 1.6\,\mathrm{cm}$
- 149.5 h on 120 cores Intel-Xeon (17490 h CPU) for 10 s

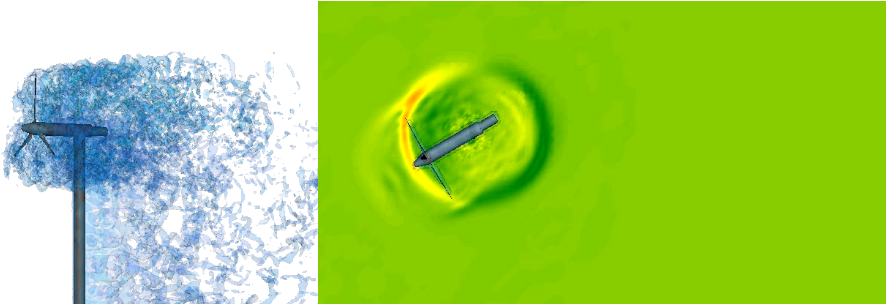
Mexico experimental turbine – 0^o inflow





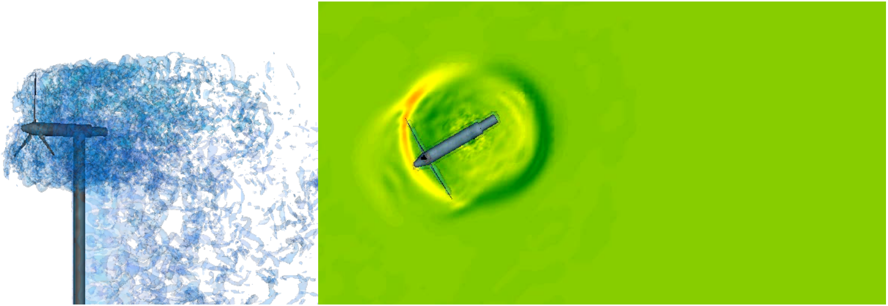
- Setup and measurements by Energy Research Centre of the Netherlands (ECN) and the Technical University of Denmark (DTU) [Schepers and Boorsma, 2012]
- Inflow velocity $14.93\,\mathrm{m/s}$ in wind tunnel of $9.5\,\mathrm{m} \times 9.5\,\mathrm{m}$ cross section.
- Rotor diameter $D = 4.5\,\mathrm{m}$. Prescribed motion with $424.5\,\mathrm{rpm}$: tip speed $100\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 75839$ TSR 6.70
- Simulation with three additional levels with factors 2, 2, 4. Resolution of rotor and tower $\Delta x = 1.6\,\mathrm{cm}$
- $149.5\,\mathrm{h}$ on 120 cores Intel-Xeon ($17490\,\mathrm{h}$ CPU) for $10\,\mathrm{s}$
- Data collected as average during $t \in [5, 10]$. Load on blade 1 as it passes through $\theta = 0^o$ (pointing vertically upwards), 35 rotations

Mexico experimental turbine – 30^o yaw



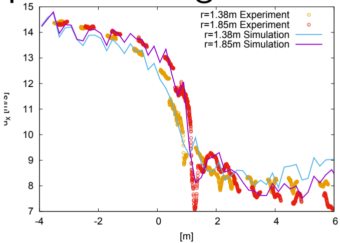
- 157.6 h on 120 cores Intel-Xeon for 10 s (70.75 revolutions) $\longrightarrow \sim$ 22.25 h CPU/1M cells/revolution
- $\sim$ 12 M cells in total – level 0: 768,000, level 1: $\sim$ 1.5 M, level 2: $\sim$ 6.8 M, level 3: $\sim$ 3.0 M

Mexico experimental turbine – 30^o yaw

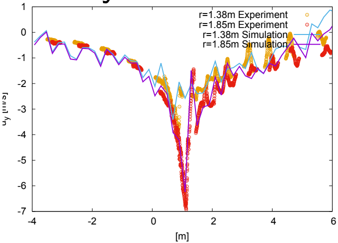


- 157.6 h on 120 cores Intel-Xeon for 10 s (70.75 revolutions) $\longrightarrow \sim$ 22.25 h CPU/1M cells/revolution
- $\sim$ 12 M cells in total – level 0: 768,000, level 1: $\sim$ 1.5 M, level 2: $\sim$ 6.8 M, level 3: $\sim$ 3.0 M
- For comparison [Schepers and Boorsma, 2012]:
- Wind Multi-Block Liverpool University (34 M cells): 209 h CPU/1M cells/revolution
- EllipSys3D (28.3 M cell mesh): $\sim$ 40.7 h CPU/1M cells/revolution, but $\sim$ 15% error in F_x and T_x already for 0° inflow [Sørensen et al., 2014]

Comparison along transects – 30^o yaw



u_x

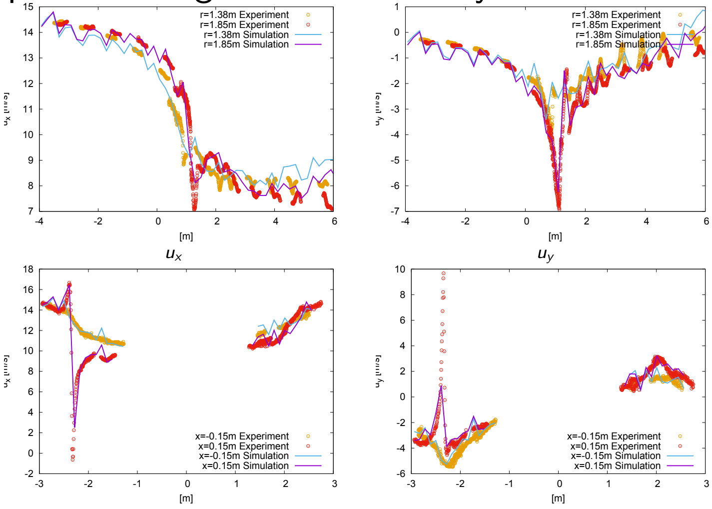


u_y

axial

RD, S. L. Wood. Proc. of TORQUE 2016. *J. Phys. Conference Series* 753: 082005, 2016.

Comparison along transects – 30^o yaw

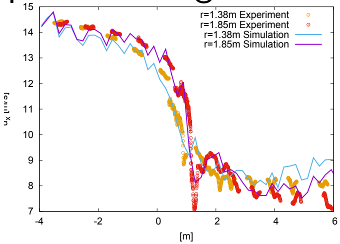


u_x u_y axial

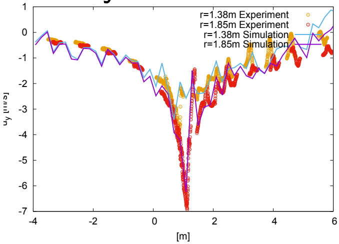
u_x u_y radial

RD, S. L. Wood. Proc. of TORQUE 2016. *J. Phys. Conference Series* 753: 082005, 2016.

Comparison along transects – 30^o yaw

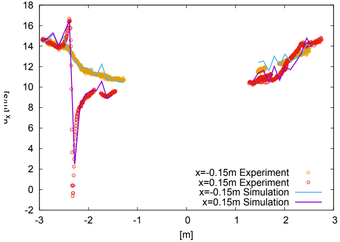


u_x

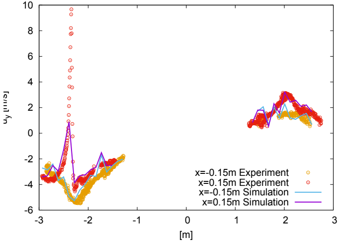


u_y

axial



u_x



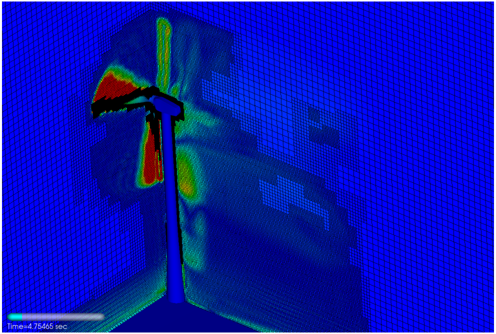
u_y

radial

- Blade loads: F_x: $\mathrm{Ref} = 13.66\,\mathrm{N}$, $\mathrm{cur.} = 14.8\,\mathrm{N}$ (8.3%)
- T_x: $\mathrm{Ref} = 7.72\,\mathrm{Nm}$, $\mathrm{cur.} = 8.36\,\mathrm{Nm}$ (8.3%)

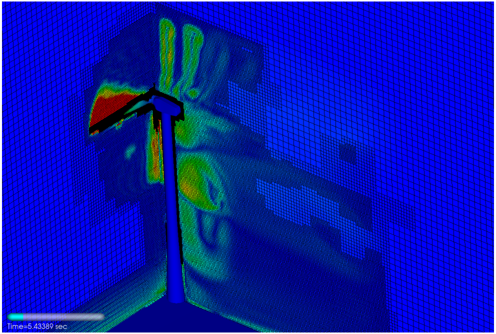
RD, S. L. Wood. Proc. of TORQUE 2016. *J. Phys. Conference Series* 753: 082005, 2016.

Single Vestas V27



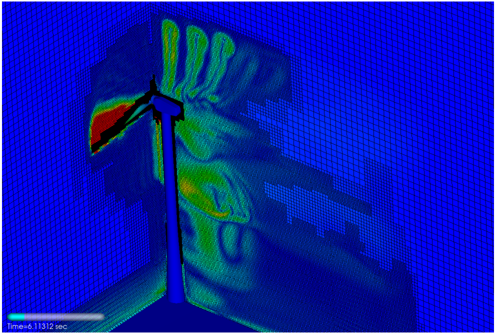
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- ~ 24 time steps for 1^o rotation

Single Vestas V27



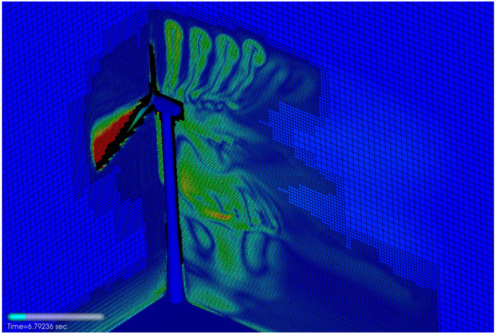
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- ~ 24 time steps for 1^o rotation

Single Vestas V27



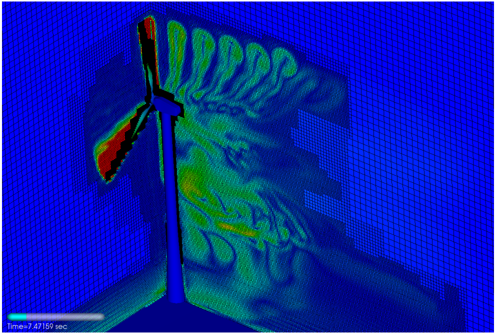
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- ~ 24 time steps for 1^o rotation

Single Vestas V27



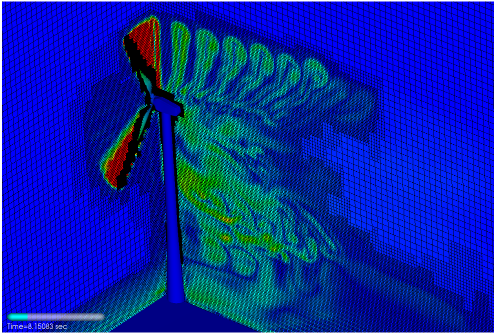
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_\mathrm{rpm} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- $\sim$ 24 time steps for 1^o rotation

Single Vestas V27



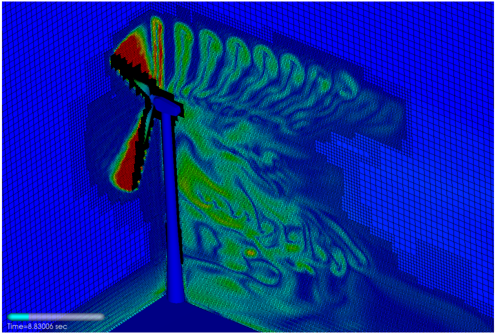
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- ~ 24 time steps for 1^o rotation

Single Vestas V27



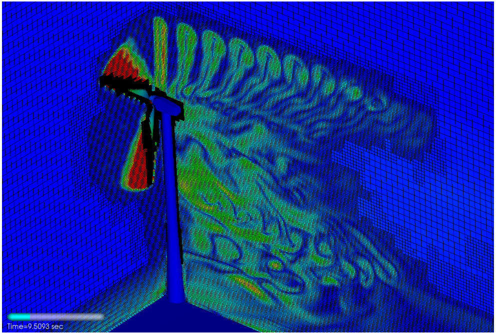
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- ~ 24 time steps for 1^o rotation

Single Vestas V27



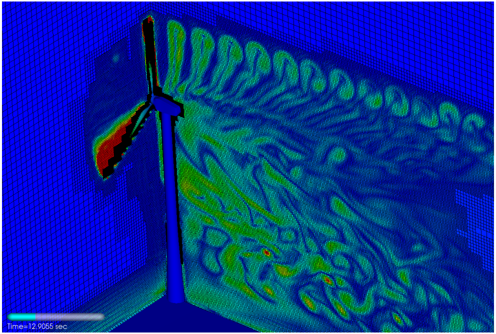
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_\mathrm{rpm} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- $\sim$ 24 time steps for 1^o rotation

Single Vestas V27



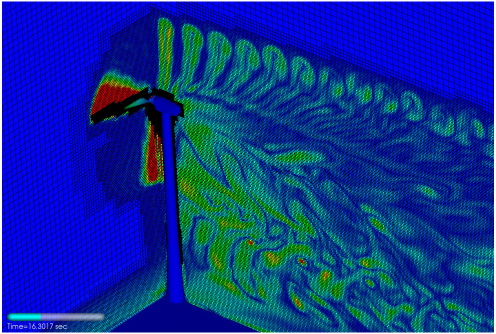
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- ~ 24 time steps for 1^o rotation

Single Vestas V27



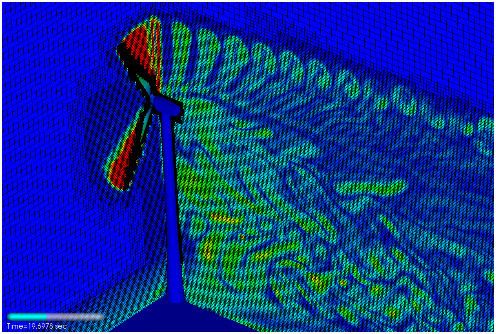
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- $\sim$ 24 time steps for 1^o rotation

Single Vestas V27



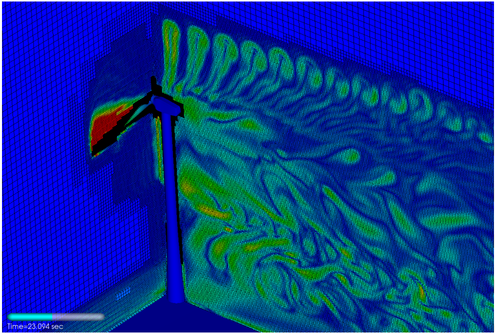
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- $\sim$ 24 time steps for 1^o rotation

Single Vestas V27



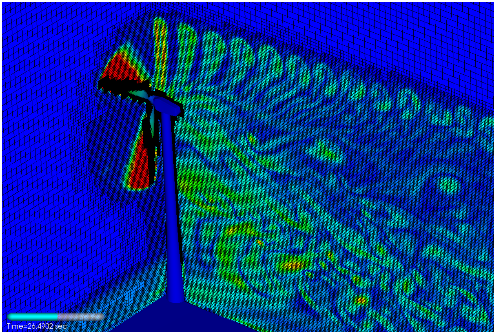
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- $\sim$ 24 time steps for 1^o rotation

Single Vestas V27



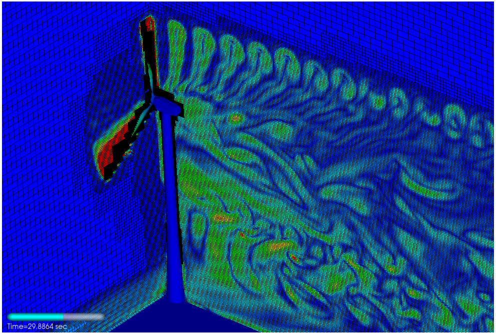
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- ~ 24 time steps for 1^o rotation

Single Vestas V27



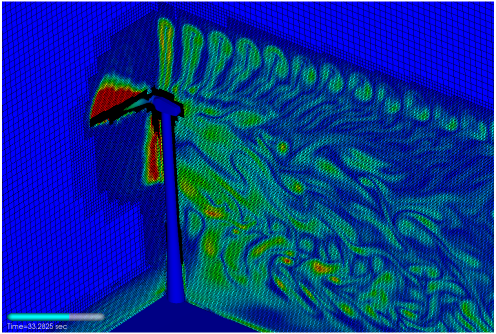
- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- ~ 24 time steps for 1^o rotation

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.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- $\sim$ 24 time steps for 1^o rotation

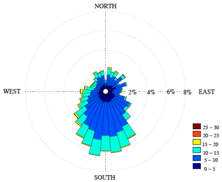
Single Vestas V27

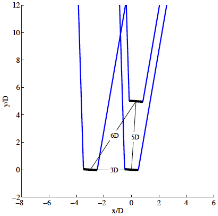


- Inflow velocity $u_\infty = 8\,\mathrm{m/s}$. Prescribed motion of rotor with $n_{\mathrm{rpm}} = 33$, $r = 14.5\,\mathrm{m}$: tip speed $46.7\,\mathrm{m/s}$, $\mathrm{Re}_r \approx 919,700$ TSR 5.84
- Simulation with three additional levels with refinement factors $2, 2, 4$.
- Refinement based on vorticity and level set.
- Sampled rotor and circular regions ($r_c = 1.5r$) every $0.034\,\mathrm{s}$ over $t = [8, 18]\,\mathrm{s}$
- Computing 84,806 highest level iterations to $t_e = 18\,\mathrm{s}$.
- ~ 24 time steps for 1^o rotation

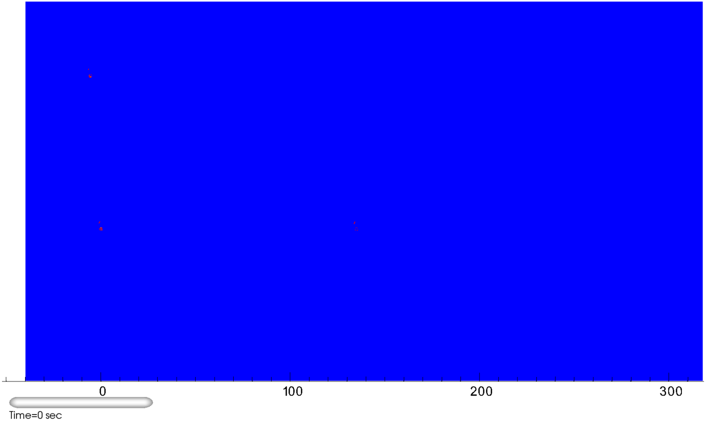
Simulation of the SWIFT array

- Three Vestas V27 turbines (geometric details prototypical). $225\,\mathrm{kW}$ power generation at wind speeds 14 to $25\,\mathrm{m/s}$ (then cut-off)
- Prescribed motion of rotor with 33 and $43\,\mathrm{rpm}$. Inflow velocity 8 and $25\,\mathrm{m/s}$
- TSR: 5.84 and 2.43, $\mathrm{Re}_r \approx 919,700$ and 1,208,000
- Simulation domain $448\,\mathrm{m} \times 240\,\mathrm{m} \times 100\,\mathrm{m}$
- Base mesh $448 \times 240 \times 100$ cells with refinement factors 2, 2,4. Resolution of rotor and tower $\Delta x = 6.25\,\mathrm{cm}$
- 94,224 highest level iterations to $t_e = 40\,\mathrm{s}$ computed, then statistics are gathered for $10\,\mathrm{s}$ [Deiterding and Wood, 2016a]



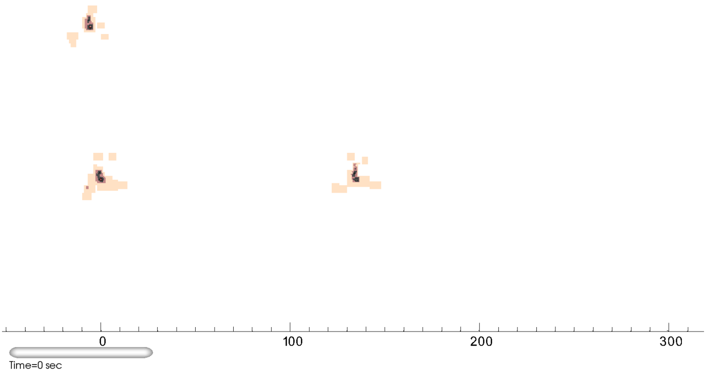


Vorticity – inflow at 30^o, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



- Top view in plane in z-direction at $30\,\mathrm{m}$ (hub height)
- Turbine hub and inflow at 30^o yaw leads to off-axis wake impact.
- 160 cores Intel-Xeon E5 2.6 GHz, $33.03\,\mathrm{h}$ wall time for interval $[50, 60]\,\mathrm{s}$ (including gathering of statistical data)
- $\sim 6.01\,\mathrm{h}$ per revolution ($961\,\mathrm{h}$ CPU) $\longrightarrow \sim 5.74\,\mathrm{h}$ CPU/1M cells/revolution

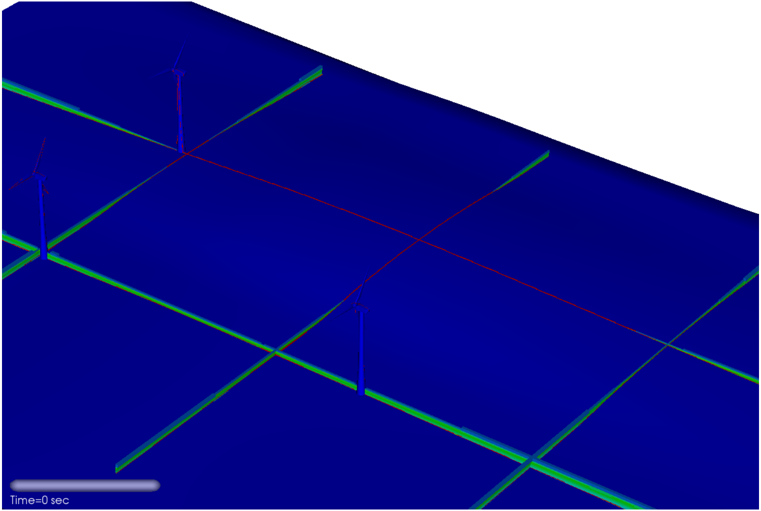
Levels – inflow at 30°, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



- 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

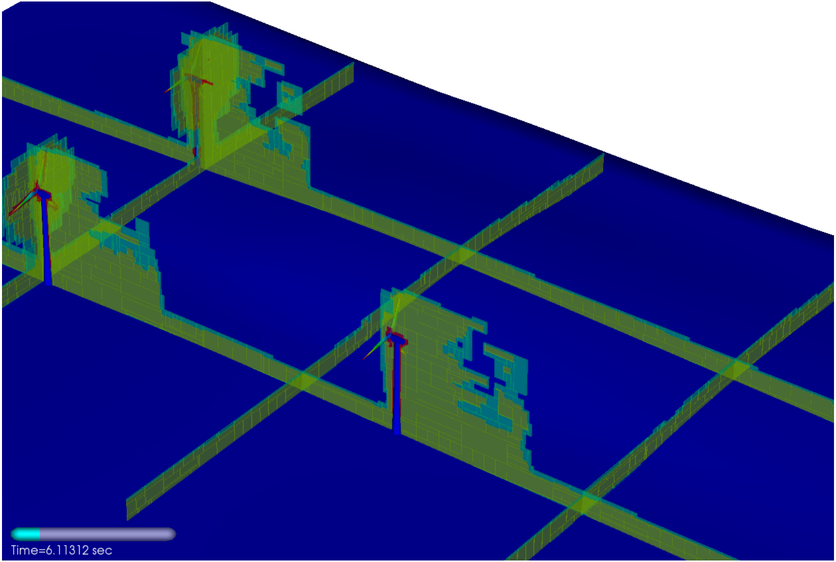
Level	Grids	Cells
0	2,463	10,752,000
1	6,464	20,674,760
2	39,473	131,018,832
3	827	4,909,632

Vorticity development – inflow at 0^o, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$

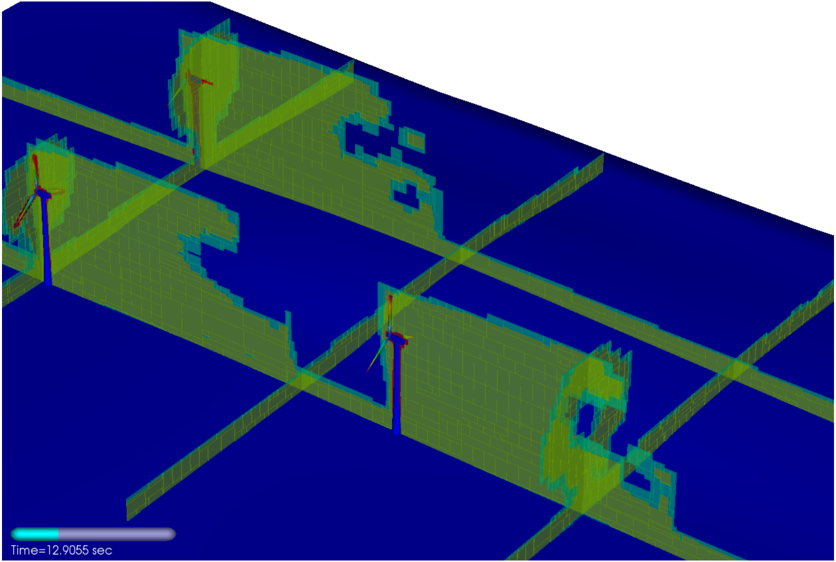


- Refinement of wake up to level 2 ($\Delta x = 25\,\mathrm{cm}$).
- Vortex break-up before 2nd turbine is reached.

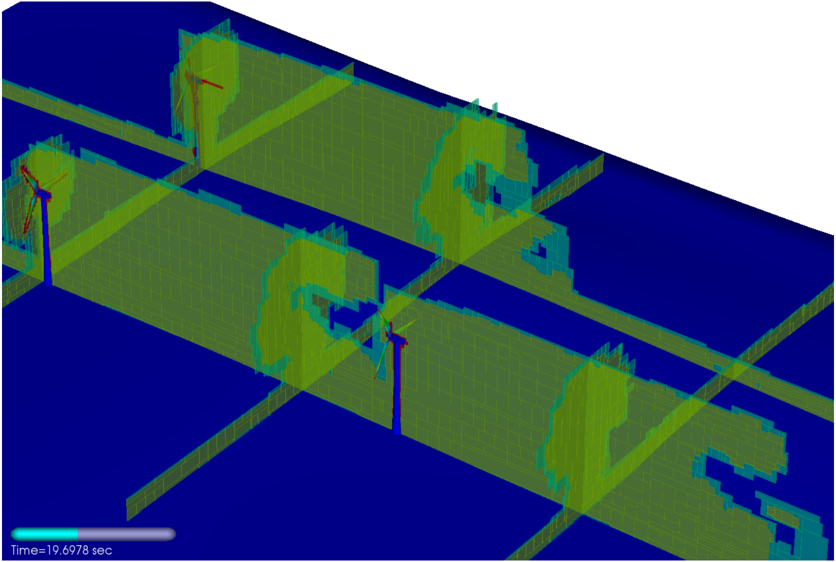
Refinement – inflow at 0^o, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



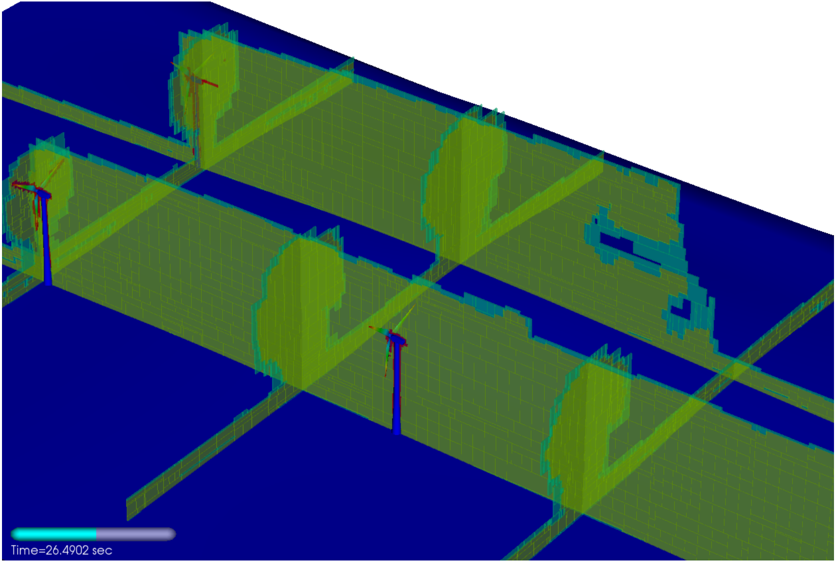
Refinement – inflow at 0^o, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



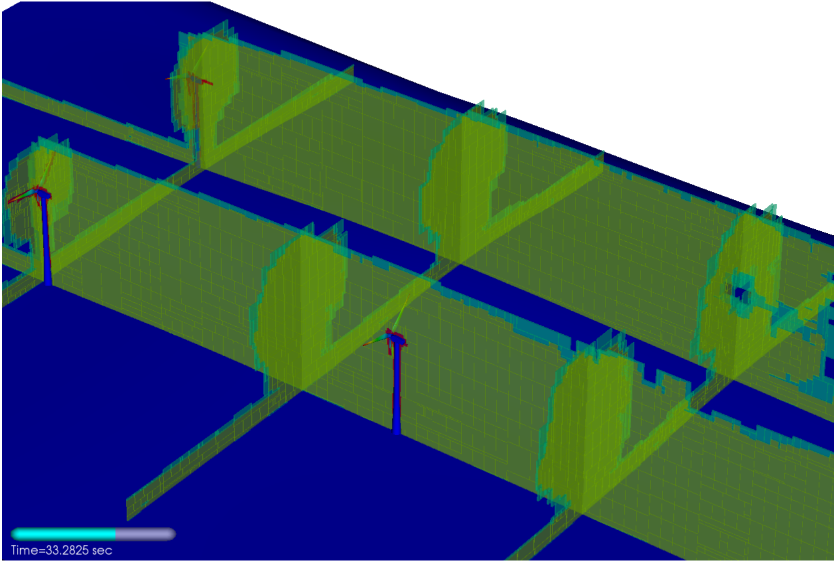
Refinement – inflow at 0^o, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



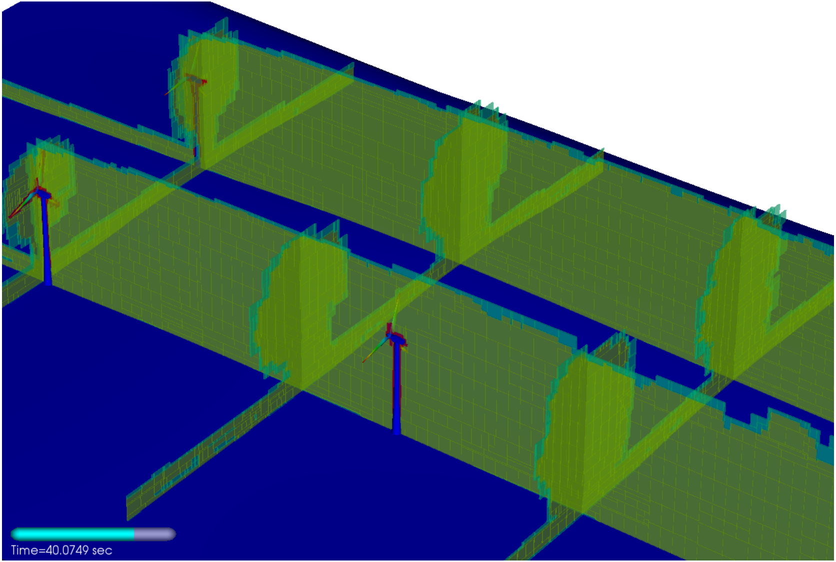
Refinement – inflow at 0^o, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



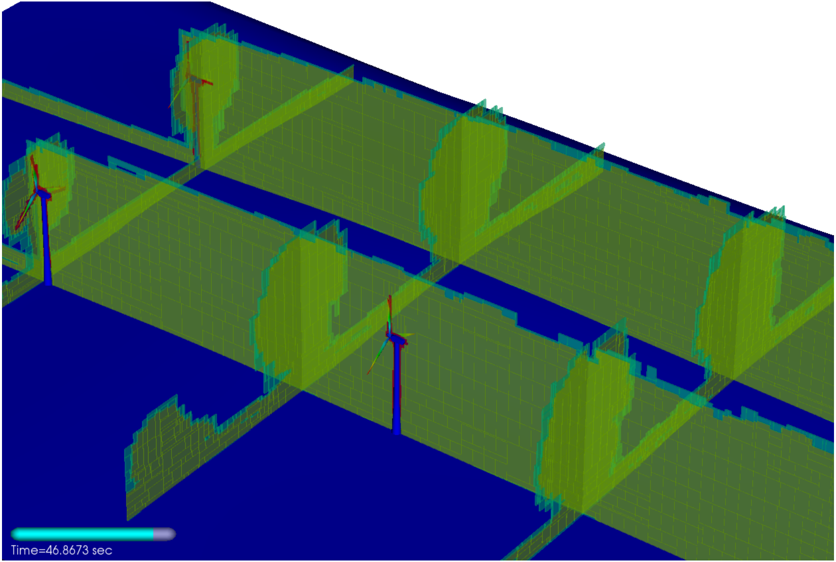
Refinement – inflow at 0^o, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



Refinement – inflow at 0^o, $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$

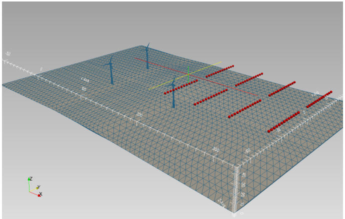


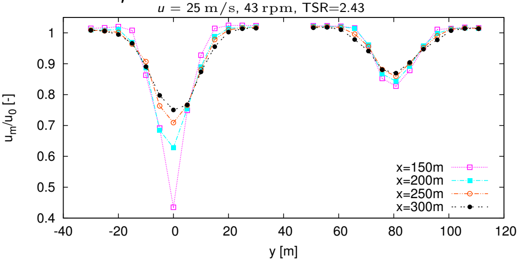
Refinement – inflow at 0°, $u = 8\,\mathrm{m/s}$, 33 rpm

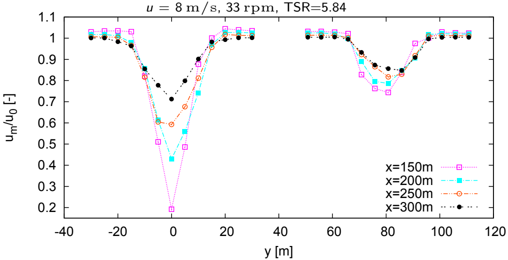


Mean point values – inflow at 0^o,

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





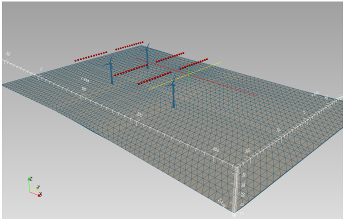


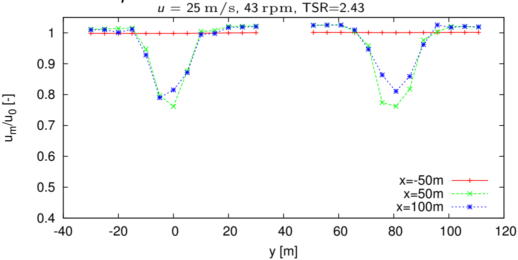
- Velocity deficits larger for higher TSR.

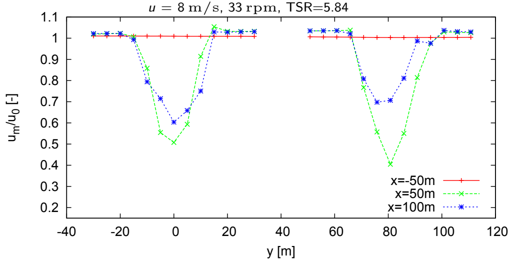
RD, S. L. Wood. New Results in Numerical and Experimental Fluid Mechanics X, pages 845-857, Springer, 2016.

Mean point values – inflow at 0^o,

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







- Velocity deficits larger for higher TSR.
- Velocity deficit before 2nd turbine more homogenous for small TSR.

RD, S. L. Wood. New Results in Numerical and Experimental Fluid Mechanics X, pages 845-857, Springer, 2016.

Conclusions and outlook

- Developed a general parallel system for implementing block-based dynamically adaptive LBMs with moving boundaries

Conclusions and outlook

- Developed a general parallel system for implementing block-based dynamically adaptive LBMs with moving boundaries
- Verification versus conventional Navier-Stokes solvers has been shown

Conclusions and outlook

- Developed a general parallel system for implementing block-based dynamically adaptive LBMs with moving boundaries
- Verification versus conventional Navier-Stokes solvers has been shown
- Validation achieved even for complex 3D testcases

Conclusions and outlook

- Developed a general parallel system for implementing block-based dynamically adaptive LBMs with moving boundaries
- Verification versus conventional Navier-Stokes solvers has been shown
- Validation achieved even for complex 3D testcases
- Adaptive LBM significantly faster than conventional CFD solvers, $\times 16$ faster than OpenFOAM [Fragner and Deiterding, 2016].

Conclusions and outlook

- Developed a general parallel system for implementing block-based dynamically adaptive LBMs with moving boundaries
- Verification versus conventional Navier-Stokes solvers has been shown
- Validation achieved even for complex 3D testcases
- Adaptive LBM significantly faster than conventional CFD solvers, $\times 16$ faster than OpenFOAM [Fragner and Deiterding, 2016].
- Thanks to the low dissipation property of the LBM, the approach is very suitable for wake prediction and direct numerical simulation (DNS). Hierachical meshes are vital for efficiency.

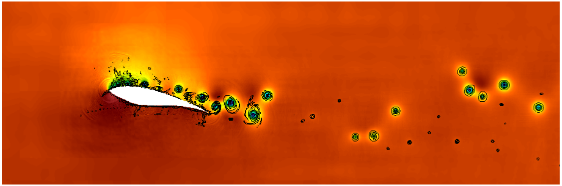
Conclusions and outlook

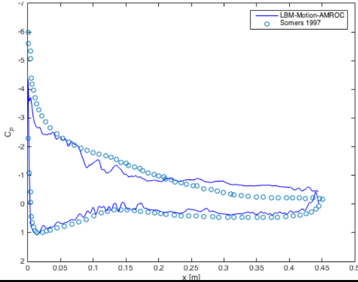
- Developed a general parallel system for implementing block-based dynamically adaptive LBMs with moving boundaries
- Verification versus conventional Navier-Stokes solvers has been shown
- Validation achieved even for complex 3D testcases
- Adaptive LBM significantly faster than conventional CFD solvers, $\times 16$ faster than OpenFOAM [Fragner and Deiterding, 2016].
- Thanks to the low dissipation property of the LBM, the approach is very suitable for wake prediction and direct numerical simulation (DNS). Hierachical meshes are vital for efficiency.
- Currently testing more complex LES turbulence models: dynamic Smagorinsky, a wall-adaptive linear eddy, and a coherent structure LES turbulence model.

Outlook

- For accurate prediction of shear flows and boundary layers, a wall-function model for high Re flows will be implemented.

S825 airfoil – $\alpha = 13.1^\circ$, $\mathrm{Re} = 2 \cdot 10^6$





References I

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

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

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

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

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

[Deiterding and Wood, 2016a] Deiterding, R. and Wood, S. L. (2016a). An adaptive lattice boltzmann method for predicting wake fields behind wind turbines. In Dillmann, A. e. a., editor, *New Results in Numerical and Experimental Fluid Mechanics X*, volume 132 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pages 845–857. Springer.

[Deiterding and Wood, 2016b] Deiterding, R. and Wood, S. L. (2016b). Predictive wind turbine simulation with an adaptive lattice boltzmann method for moving boundaries. *J. Phys. Conf. Series*, 753:082005.

[Fragner and Deiterding, 2016] Fragner, M. M. and Deiterding, R. (2016). Investigating cross-wind stability of high speed trains with large-scale parallel cfd. *Int. J. Comput. Fluid Dynamics*, 30:402–407.

[Fusegi et al., 1991] Fusegi, T., Hyun, J., Kuwahara, K., and Farouk, B. (1991). A numerical study of three-dimensional natural convection in a differentially heated cubical enclosure. *Int. J. Heat and Mass Transfer*, 34:1543–1557.

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

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

References II

[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.

[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.

[Schepers and Boorsma, 2012] Schepers, J. G. and Boorsma, K. (2012). Final report of iea task 29: Mexnext (phase 1) – Analysis of Mexico wind tunnel measurements. Technical Report ECN-E-12-004, European research Centre of the Netherlands.

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

[Sørensen et al., 2014] Sørensen, N. N., Bechmann, A., Rethore, P. E., and Zahle, F. (2014). Near wake reynolds-averaged Navier-Stokes predictions of the wake behind the MEXICO rotor in axial and yawed flow conditions. *Wind Energy*, 17:75–86.

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

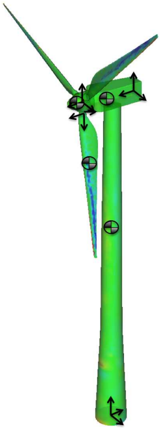
[Wood, 2016] Wood, S. L. (2016). *Lattice Boltzmann methods for wind energy analysis*. PhD thesis, University of Tennessee Knoxville.

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

Motion solver

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

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

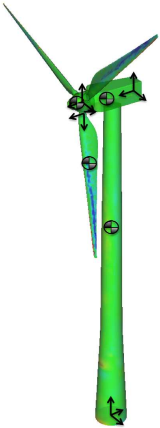


m = mass of the body, 1 = the 4×4 homogeneous identity matrix,
$\mathbf{a}_p$ = acceleration of link frame with origin at $\mathbf{p}$ in the preceding link's frame,
$\mathbf{I}_{\mathrm{cm}}$ = moment of inertia about the center of mass,
$\boldsymbol{\omega}$ = angular velocity of the body,
$\boldsymbol{\alpha}$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

Motion solver

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

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



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

Here, we additionally define the total force and torque acting on a body,

$\mathbf{F} = (\mathbf{F}_{FSI} + \mathbf{F}_{prescribed}) \cdot \boldsymbol{\mathcal{C}}_{xyz}$ and

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

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