Lecture 4
Advanced topics

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

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

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

Outline

Outline

Outline

Outline

Approximation of Boltzmann equation

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

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

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

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

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

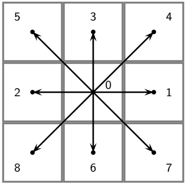
- $\mathrm{Kn} = l_f / L \ll 1$, where l_f is replaced with Δx
- Weak compressibilty and small Mach number assumed
- Assume a simplified phase space

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

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

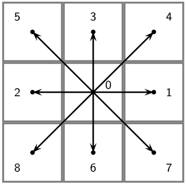
- $\mathrm{Kn} = l_f / L \ll 1$, where l_f is replaced with Δx
- Weak compressibilty and small Mach number assumed
- Assume a simplified phase space

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

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

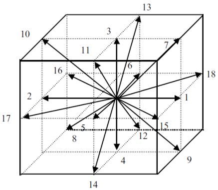
- $\mathrm{Kn} = l_f / L \ll 1$, where l_f is replaced with Δx
- Weak compressibilty and small Mach number assumed
- Assume a simplified phase space

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:

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

Approximation of thermal equilibrium

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

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

Approximation of thermal equilibrium

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

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

with equilibrium function

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

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

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

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

with equilibrium function

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

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

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

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

with equilibrium function

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

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

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

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

with equilibrium function

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

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

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

and using

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

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 $\bar{f}^{eq}_\alpha$, i.e.

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

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

Turbulence modeling

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

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

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

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

Turbulence modeling

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

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

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

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

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

Turbulence modeling

Pursue a large-eddy simulation approach with $\bar{f}_\alpha$ and $\bar{f}^{eq}_\alpha$, 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: $\nu^\star = \nu + \nu_t = \dfrac{1}{3}\left(\dfrac{\tau_L^\star}{\Delta t} - \dfrac{1}{2}\right) c\Delta x$ with $\tau_L^\star = \tau_L + \tau_t$

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

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

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

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

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

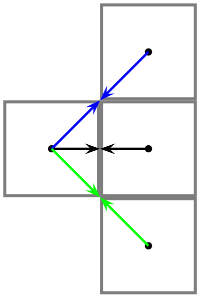
$$\tau_t = \frac{1}{2}\left(\sqrt{\tau_L^2 + 18\sqrt{2}(\rho_0 c^2)^{-1} C_{sm}^2 \Delta x \bar{S}} - \tau_L \right)$$

Initial and boundary conditions

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

Simple boundary conditions:

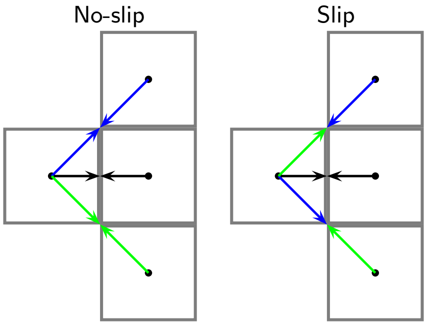
No-slip



Initial and boundary conditions

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

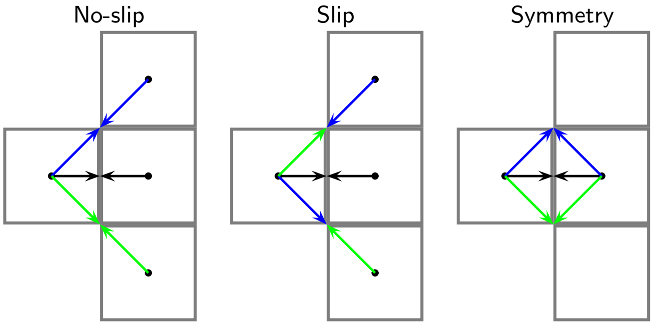
Simple boundary conditions:



Initial and boundary conditions

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

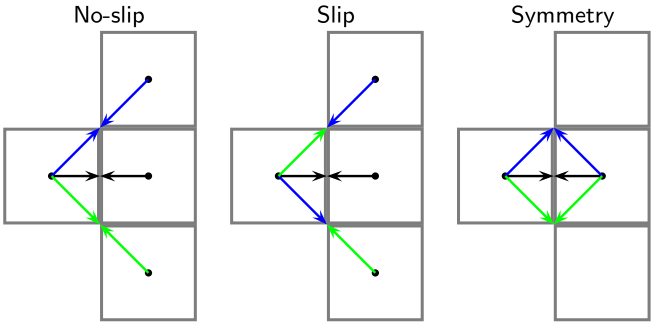
Simple boundary conditions:



Initial and boundary conditions

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

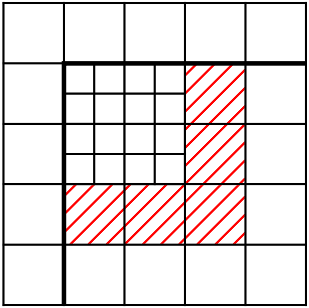
Simple boundary conditions:



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

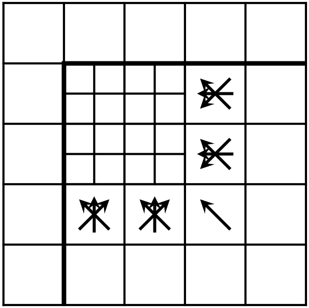
Adaptive LBM

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



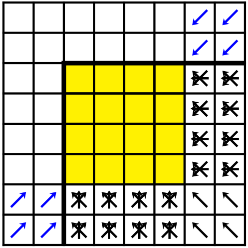
Adaptive LBM

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

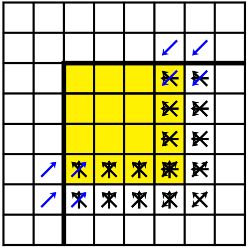
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.



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

Adaptive LBM

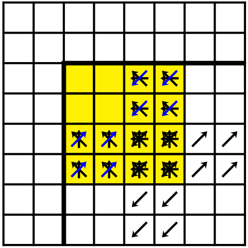
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



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

Adaptive LBM

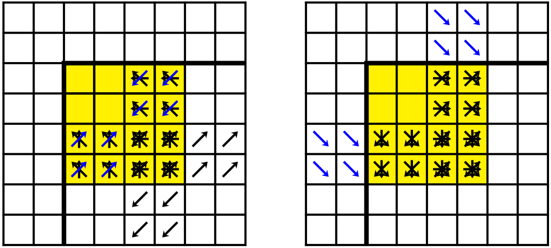
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

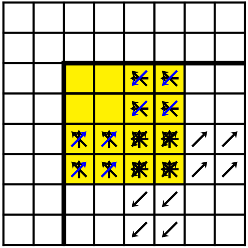
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



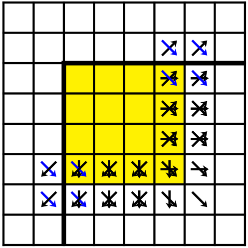
$\tilde{f}_{\alpha,in}^{f,n+1/2}$ $\qquad$ $f_{\alpha,out}^{f,n}$

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.



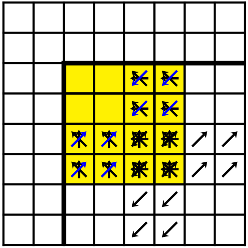
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



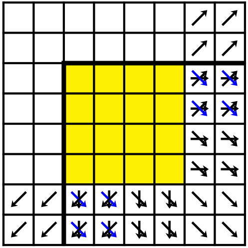
$\tilde{f}_{\alpha,out}^{f,n}$

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.



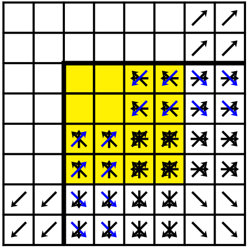
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



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

Adaptive LBM

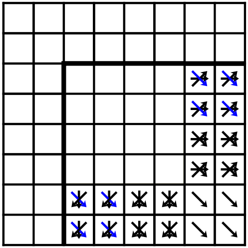
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

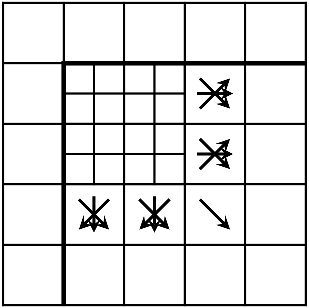


$\tilde{f}_{\alpha,out}^{f,n+1/2}$, $\tilde{f}_{\alpha,out}^{f,n}$

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

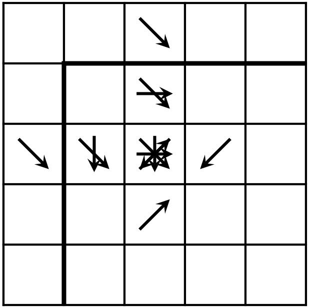
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

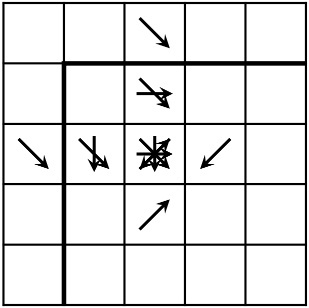
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

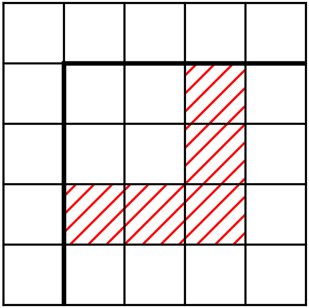
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

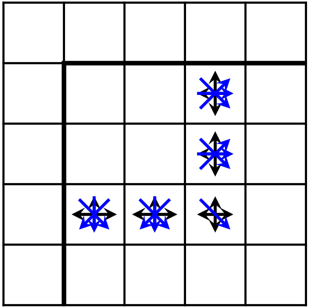
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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

Classes

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

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

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

  `code/amroc/doc/html/lbm/classLBMD3Q19Thermal.html`

- Using function within LBMD?D?, the special coarse-fine correction is implemented in **LBMFixup**<**LBMType, FixupType, dim**>

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

- **LBMIntegrator**<**LBMType, dim** >, **LBMGFMBoundary**<**LBMType, dim** >, etc. interface to the generic classes in amroc/amr/Interfaces

  `code/amroc/doc/html/amr/classSchemeGFMBoundary.html`

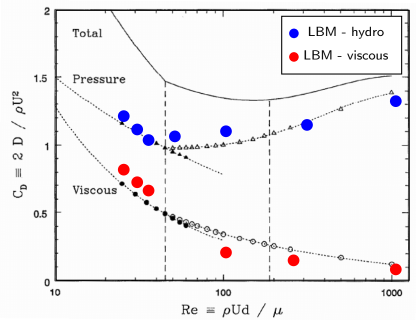
- **Problem.h**: Specific simulation is defined in Problem.h only. Predefined classes specified in **LBMStdProblem.h**, **LBMStdGFMProblem.h** and **LBMProblem.h**.

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

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

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

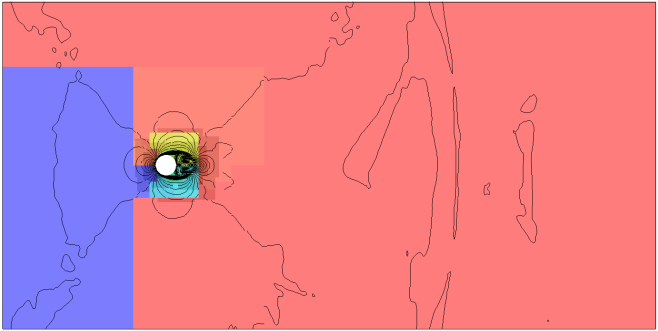
- Air with
  $\nu = 1.61 \cdot 10^{-5}\,\mathrm{m}^2/\mathrm{s}$,
  $\rho = 1.205\,\mathrm{kg}/\mathrm{m}^3$
- Domain size
  $[-8d, 24d] \times [-8d, 8d]$
- Dynamic refinement based on velocity. Last level to refine structure further.
- Inflow from left. Characteristic boundary conditions [Schlaffer,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



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

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

Isolines on refinement and distribution to processors



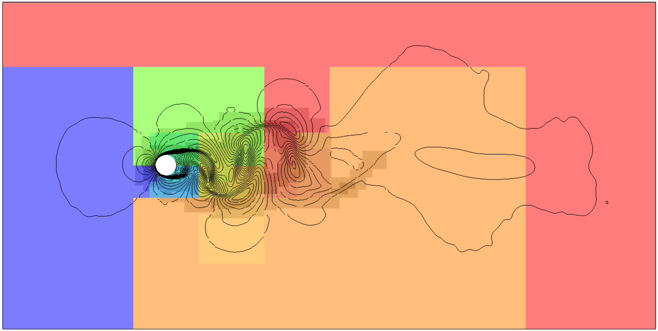
Mesh adaptation with LBM:

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

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

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

Isolines on refinement and distribution to processors



Mesh adaptation with LBM:

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

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

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

Isolines on refinement and distribution to processors

Mesh adaptation with LBM:

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

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

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

Isolines on refinement and distribution to processors

Mesh adaptation with LBM:

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

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

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

Isolines on refinement and distribution to processors

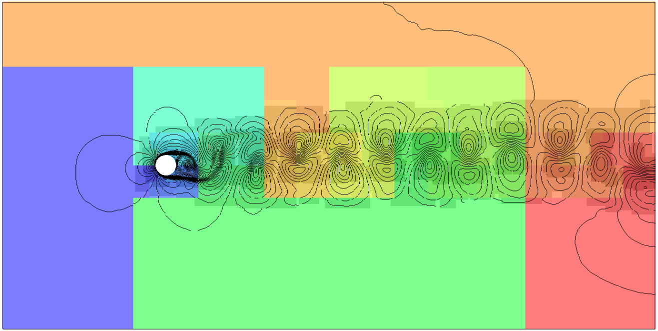
Mesh adaptation with LBM:

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

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

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

Isolines on refinement and distribution to processors



Mesh adaptation with LBM:

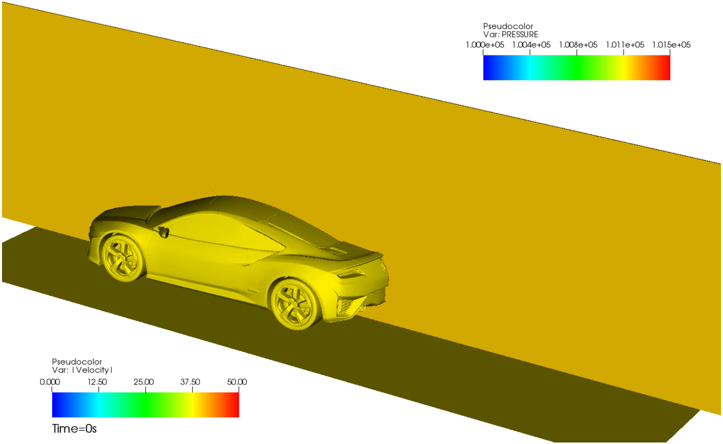
1. Level-wise evaluation of $\omega_L^l = \frac{c_s^2}{\nu + \Delta t_l c_s^2/2}$
2. Exchange of distributions streaming across refinement interfaces

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

Outline

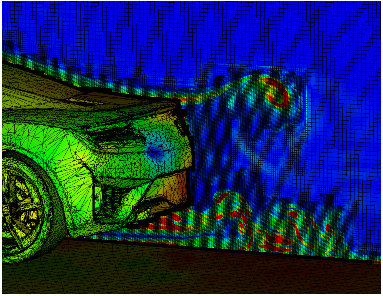
Wind tunnel simulation of a prototype car

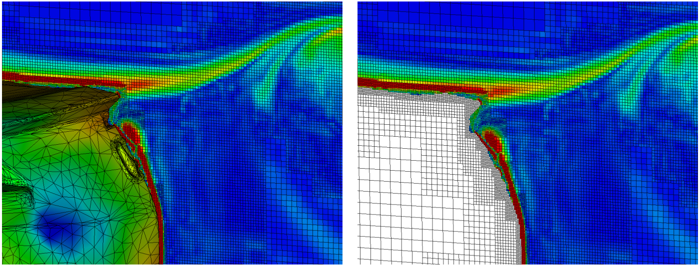
Fluid velocity and pressure on vehicle



- 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 $\sim 37\,\mathrm{h}$ on 200 cores (7389 h CPU). Channel: $15\,\mathrm{m} \times 5\,\mathrm{m} \times 3.3\,\mathrm{m}$

Mesh adaptation

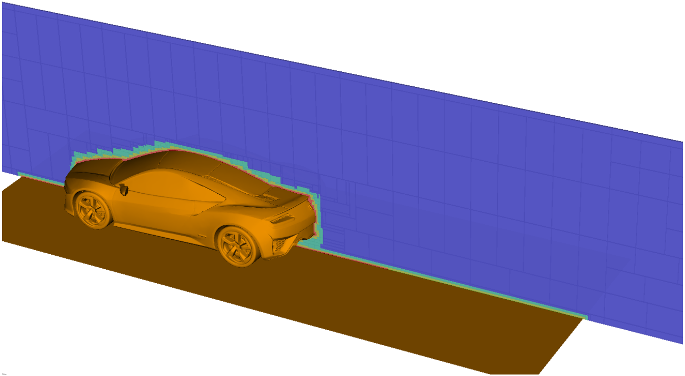




code/amroc/doc/html/apps/lbm_2applications_2Navier-Stokes_23d_2VehicleOnGround_2src_2Problem_8h_source.html

Mesh adaptation

Used refinement blocks and levels (indicated by color)



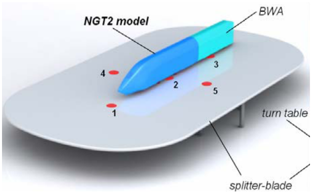
- SAMR base grid 600 × 200 × 132 cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 3.125\,\mathrm{mm}$
- Adaptation based on level set and scaled gradient of magnitude of vorticity vector
- 236M cells vs. 8.1 billion (uniform) at $t = 0.4075\,\mathrm{s}$

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

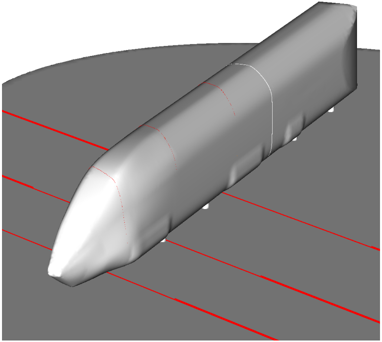
Level	Grids	Cells
0	11,605	15,840,000
1	11,513	23,646,984
2	31,382	144,447,872
3	21,221	52,388,336

code/amroc/doc/html/apps/lbm_2applications_2Navier-Stokes_23d_2VehicleOnGround_2src_2Problem_8h_source.html

Next Generation Train (NGT)



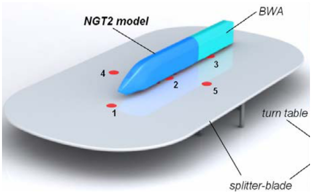
- 1:25 train model of 74,670 triangles
- Wind tunnel: air at room temperature with $33.48\,\mathrm{m/s}$, $\mathrm{Re} = 250,000$, yaw angle 30^o
- Comparison between LBM (fluid air) and incompressible OpenFOAM solvers

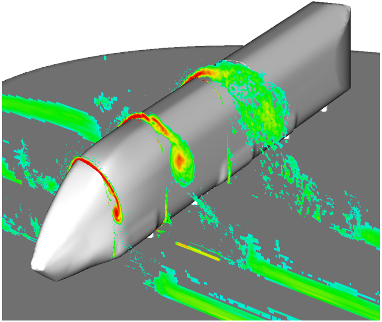


code/amroc/doc/html/apps/lbm_2applications_2Navier-Stokes_23d_2NGT2_2src_2Problem_8h_source.html

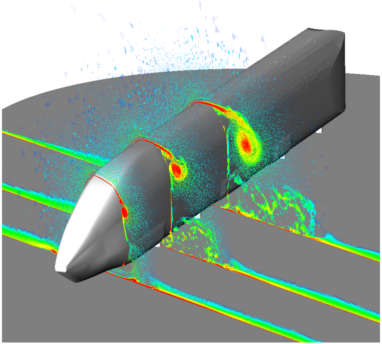
Next Generation Train (NGT)

- 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



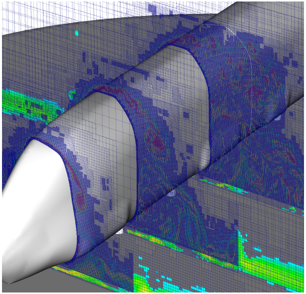
Averaged vorticity OpenFOAM-LES

code/amroc/doc/html/apps/lbm_2applications_2Navier-Stokes_23d_2NGT2_2src_2Problem_8h_source.html

NGT model

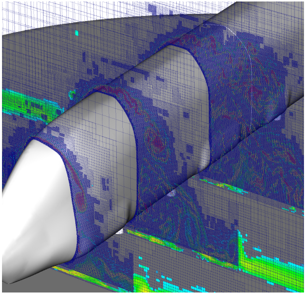
- LBM-AMR computation with 5 additional levels, factor 2 refinement (uniform: 120.4e9 cells)
- Dynamic AMR until $T_c = 34$, then static for $\sim 12T_C$ to obtain average coefficients
- OpenFOAM simulations by M. Fragner (DLR)

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



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)

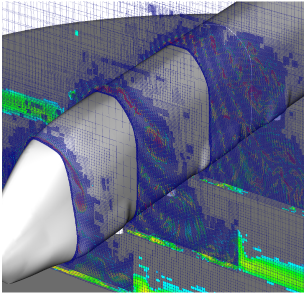


Simulation	Mesh	CFX	CFY	CMX
Wind tunnel	–	-0.06	-5.28	-3.46
DDES	low	-0.40	-5.45	-3.61
Σ only	low	0.10	-0.04	-0.05
LES	high	-0.45	-6.07	-4.14
DDES	high	-0.43	-5.72	-3.77
LBM - p only	–	-0.30	-5.09	-3.46

	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 $\sim 12 T_C$ to obtain average coefficients
- OpenFOAM simulations by M. Fragner (DLR)



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

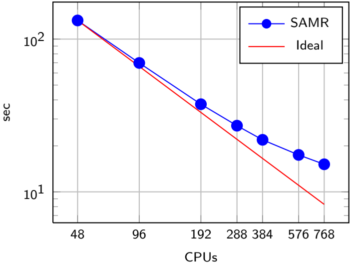
	LBM	DDES(l)	LES	DDES(h)
Cells	147M	34.1M	219M	219M
y^+	43	3.2	1.7	1.7
x^+, z^+	43	313	140	140
Δx wake [mm]	0.936	3.0	1.5	1.5
Runtime [T_C]	34	35.7	10.3	9.2
Processors	200	80	280	280
CPU [h]	34,680	49,732	194,483	164,472
$T_C/\Delta t$	1790	1325	1695	1695
CPU [h]/T_C/1M cells	5.61	39.75	86.4	81.36

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

Strong scalability test (1:25 train)

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

Time per coarse level step



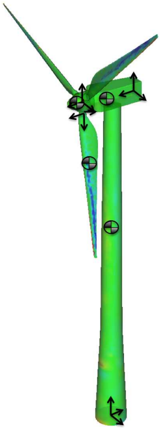
Time in % spent in main operations

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

Motion solver

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

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

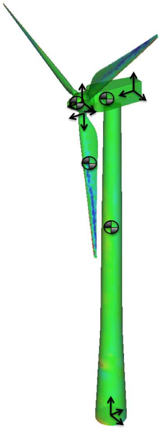


m = mass of the body, 1 = the 4×4 homogeneous identity matrix,
$\mathbf{a}_p$ = acceleration of link frame with origin at $\mathbf{p}$ in the preceding link's frame,
$\mathbf{I}_{\mathrm{cm}}$ = moment of inertia about the center of mass,
$\boldsymbol{\omega}$ = angular velocity of the body,
$\boldsymbol{\alpha}$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

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.

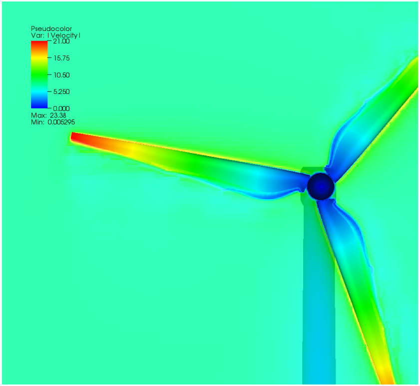
Here, we additionally define the total force and torque acting on a body,
$\mathbf{F} = (\mathbf{F}_{FSI} + \mathbf{F}_{prescribed}) \cdot \mathcal{C}_{xyz}$ and

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

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

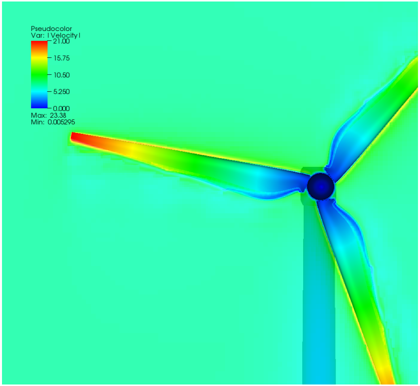
Simulation of a single turbine

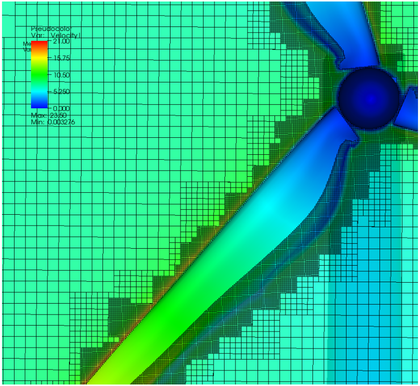
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim 35\,\mathrm{m}$. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh $400 \times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



Simulation of a single turbine

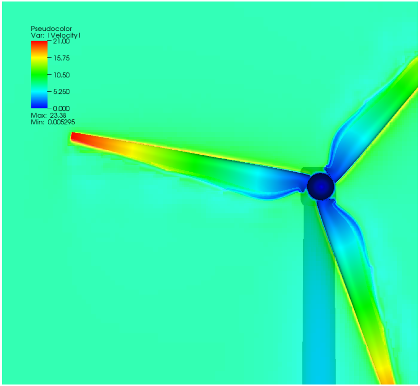
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh $400 \times 200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

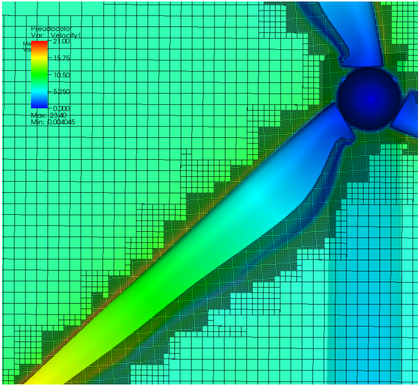




Simulation of a single turbine

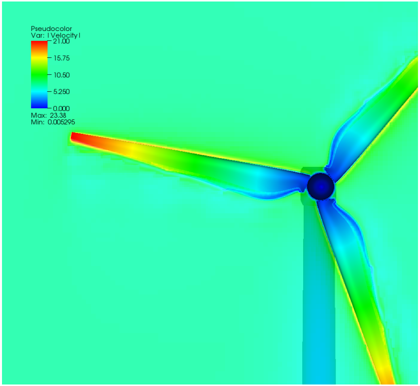
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh $400 \times 200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

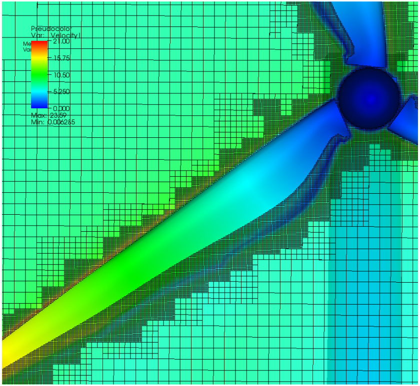




Simulation of a single turbine

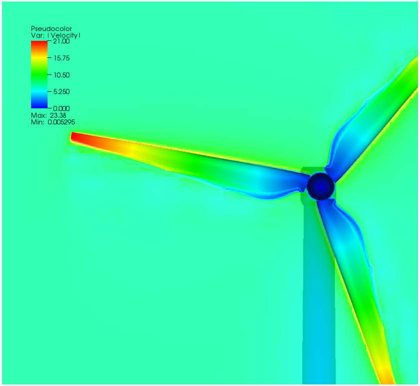
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh $400 \times 200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

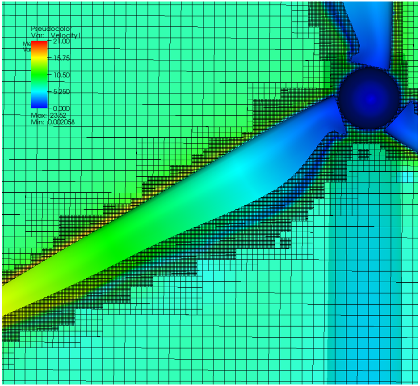




Simulation of a single turbine

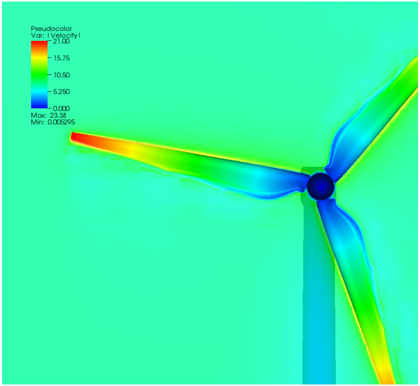
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh $400 \times 200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

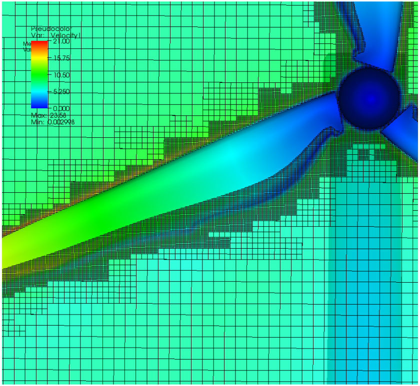




Simulation of a single turbine

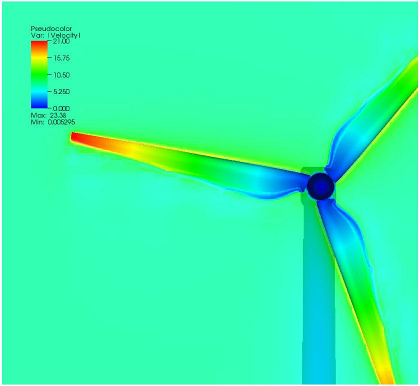
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim 35\,\mathrm{m}$. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh $400 \times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

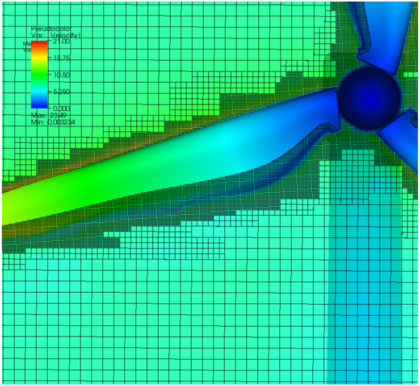




Simulation of a single turbine

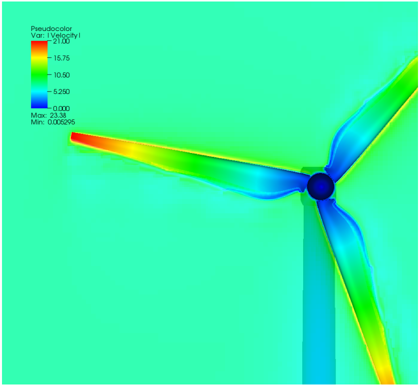
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh $400 \times 200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

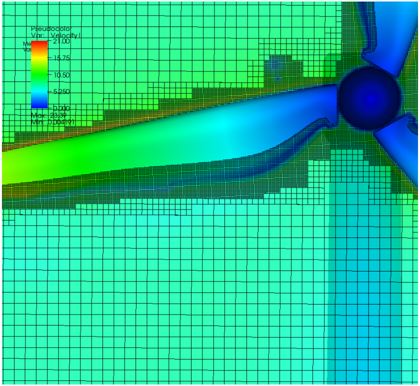




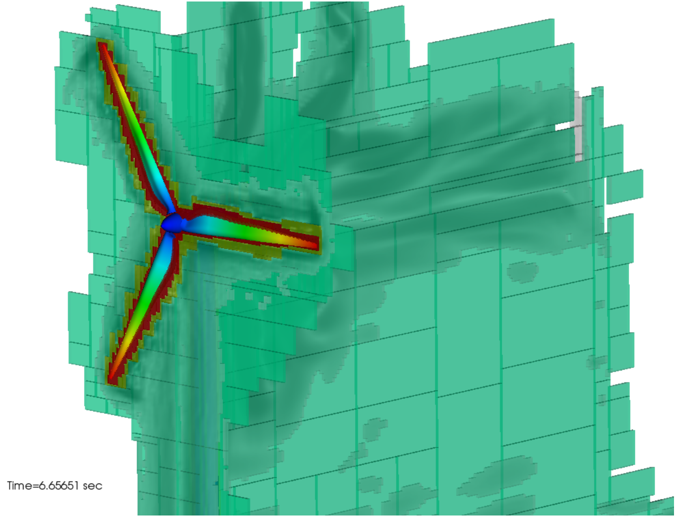
Simulation of a single turbine

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





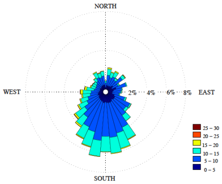
Adaptive refinement

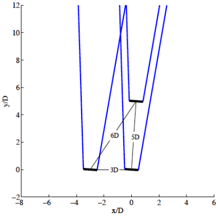


Dynamic evolution of refinement blocks (indicated by color).
code/doc/html/capps/motion-amroc_2WindTurbine__Terrain_2src_2FluidProblem_8h_source.html,
code/doc/html/capps/motion-amroc_2WindTurbine__Terrain_2src_2SolidProblem_8h_source.html,
code/doc/html/capps/Terrain_2src_2Terrain_8h_source.html

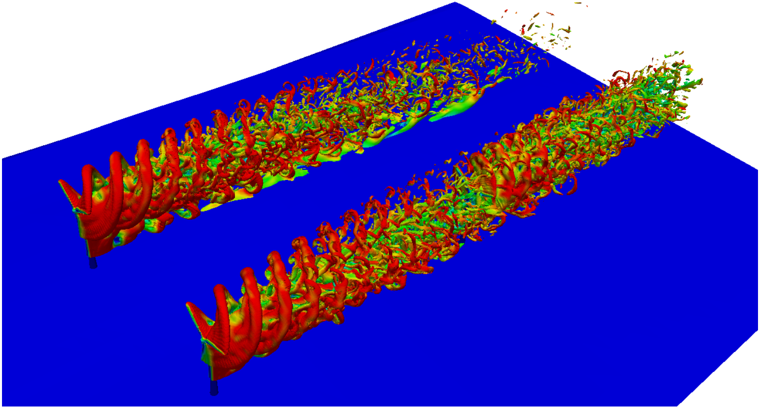
Simulation of the SWIFT array

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





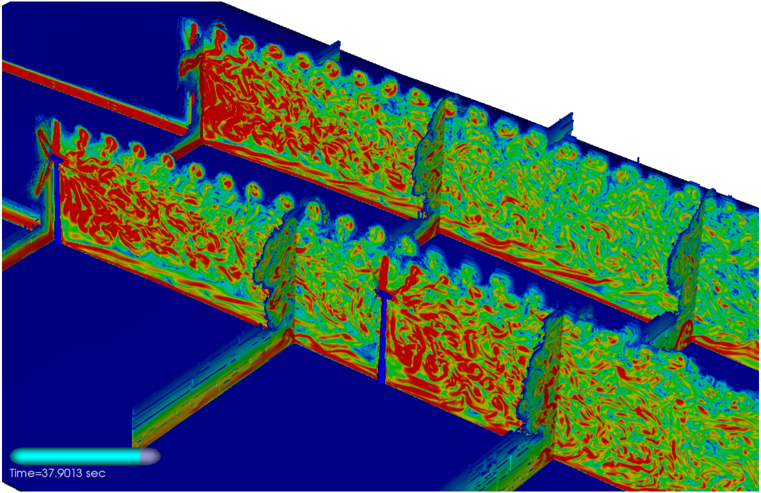
Wake propagation through array – 25 $\mathrm{m/s}$, 43 rpm



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

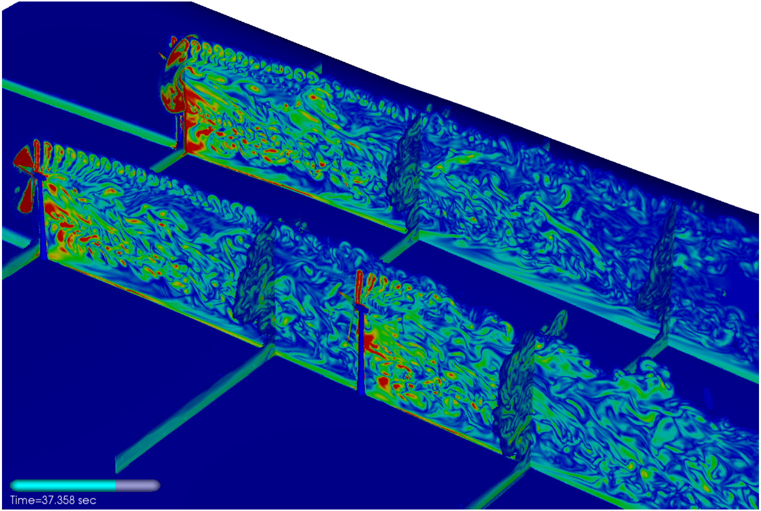
Level	Grids	Cells
0	3,234	10,752,000
1	11,921	21,020,256
2	66,974	102,918,568
3	896	5,116,992

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



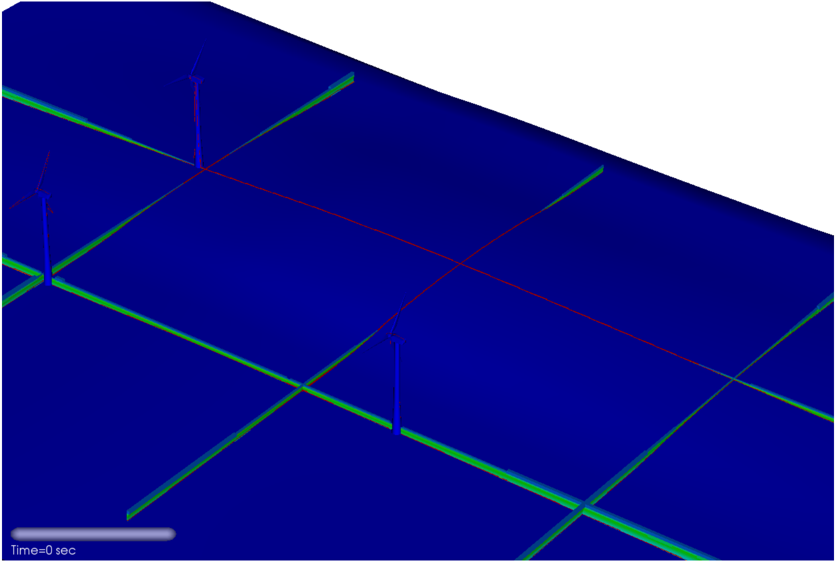
- Refinement of wake up to level 2 ($\Delta x = 25\,\mathrm{cm}$).

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

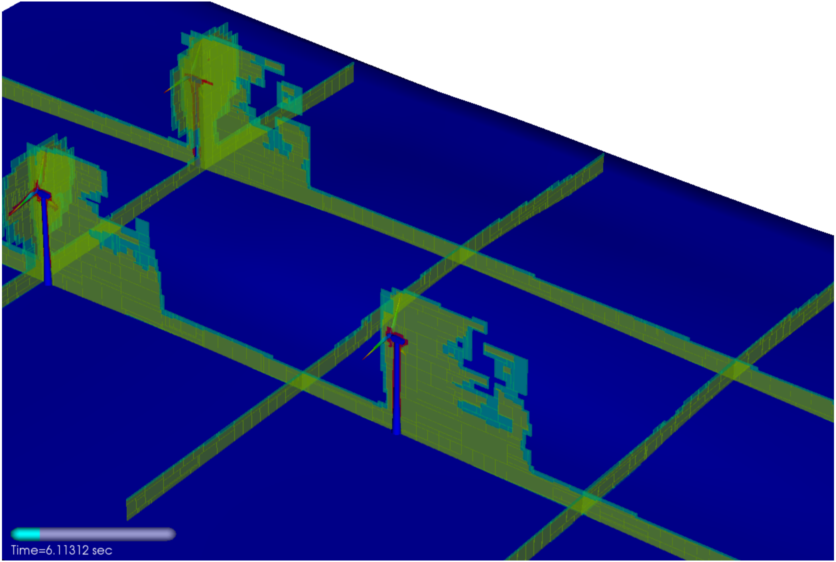


- Refinement of wake up to level 2 ($\Delta x = 25\,\mathrm{cm}$).
- Vortex break-up before 2nd turbine is reached.

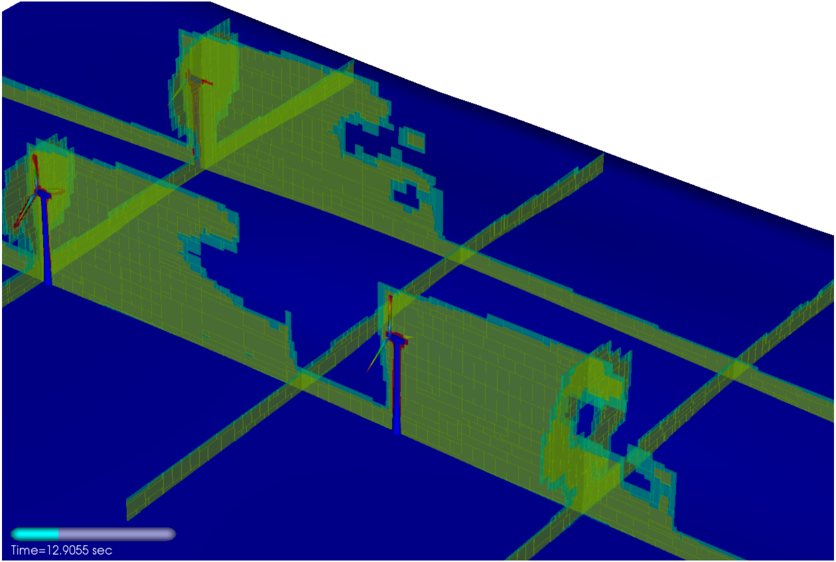
Vorticity development - $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



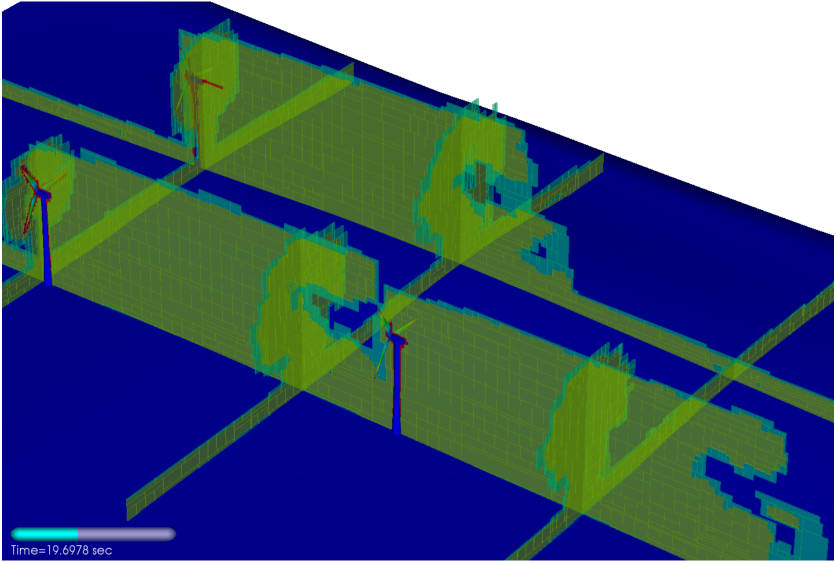
Refinement $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



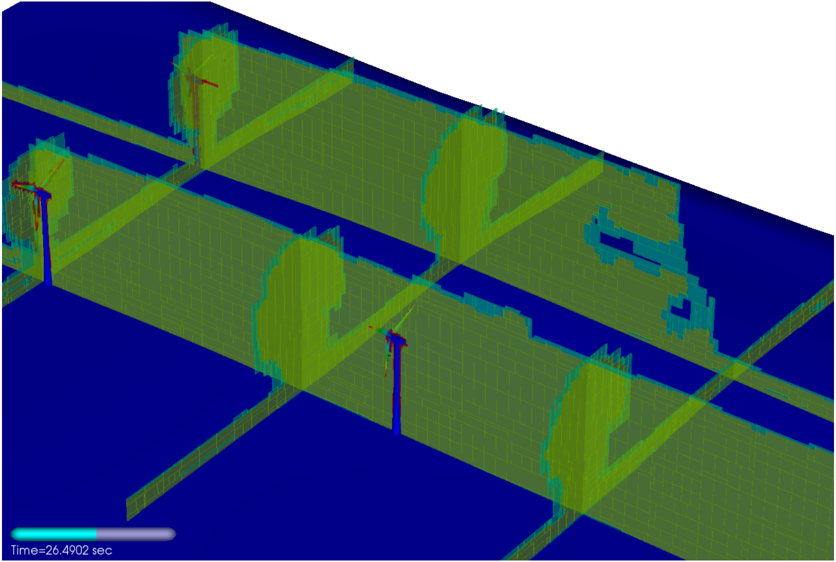
Refinement $u = 8\,\mathrm{m/s},\ 33\,\mathrm{rpm}$



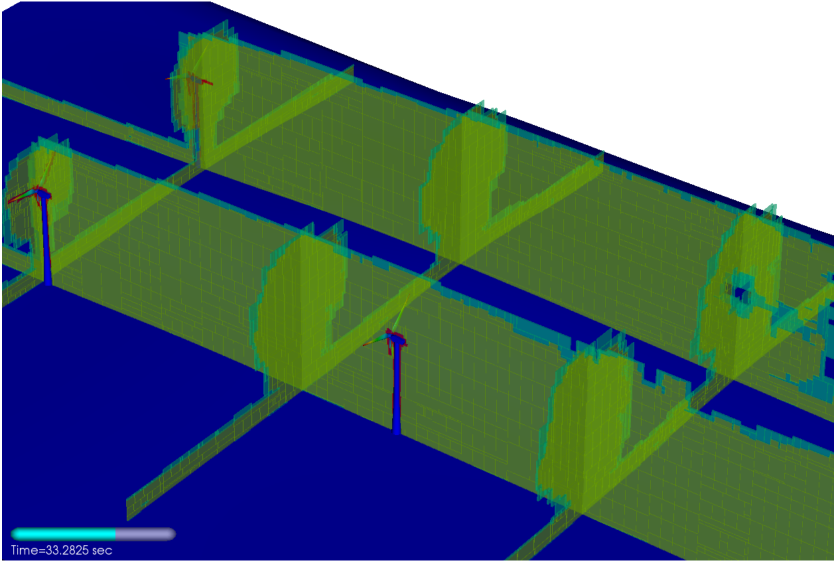
Refinement $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



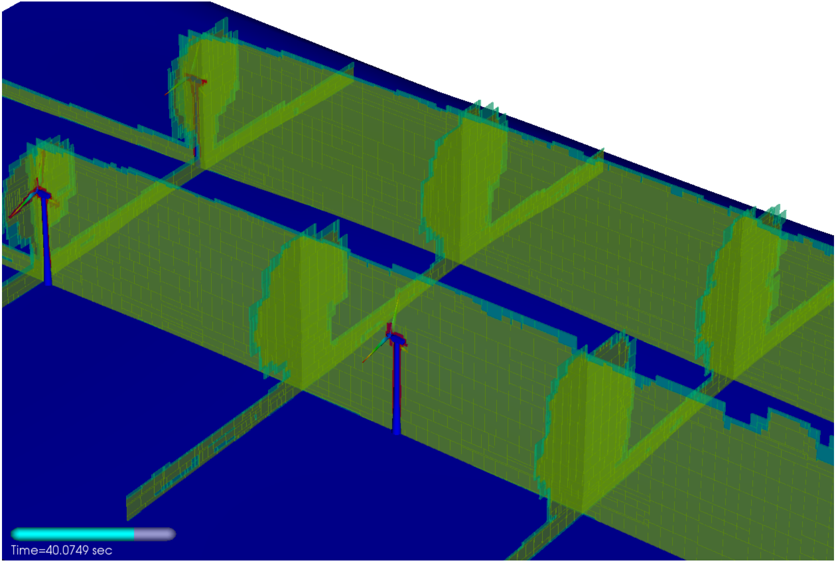
Refinement $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



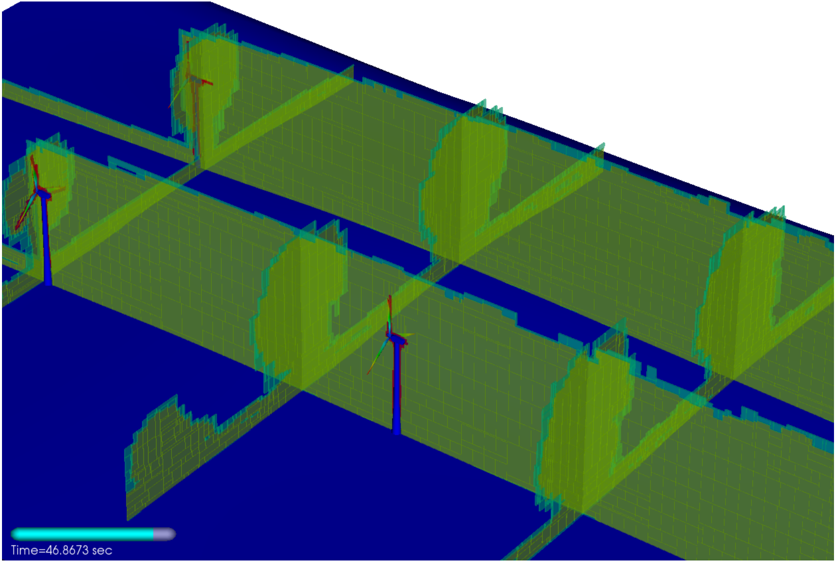
Refinement $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



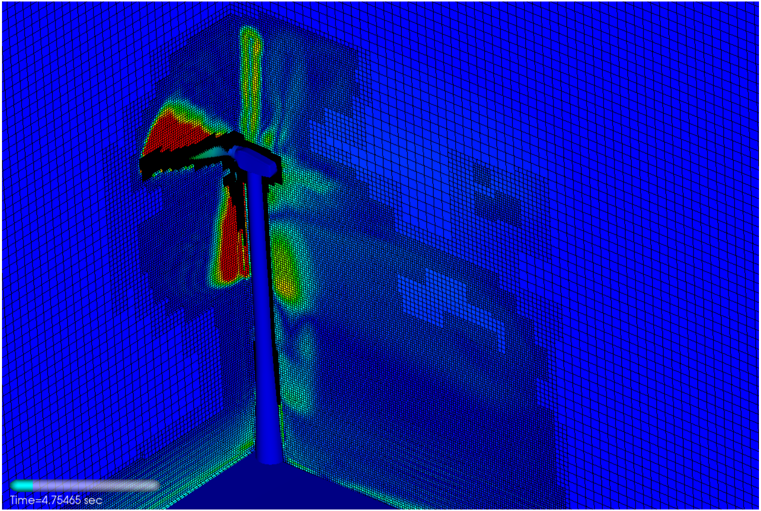
Refinement $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



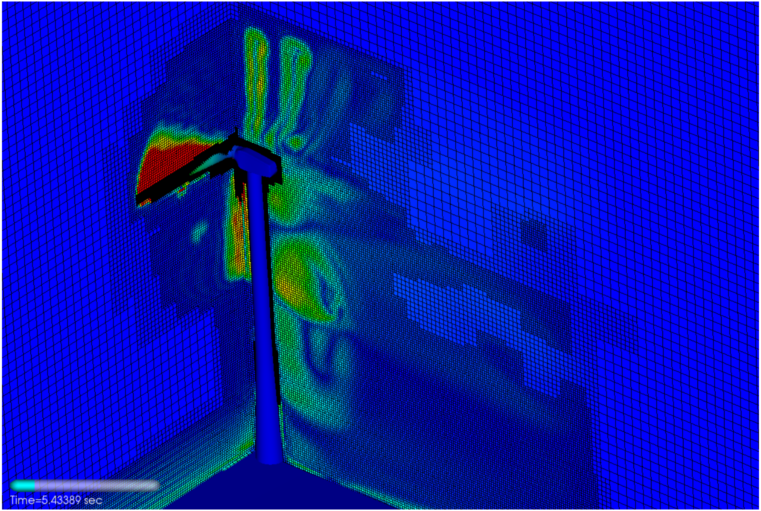
Refinement $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$



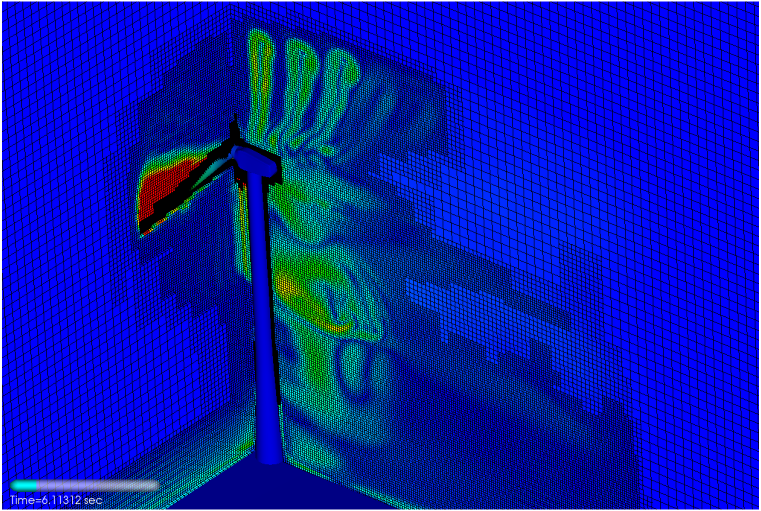
Wake refinement behind a leading turbine



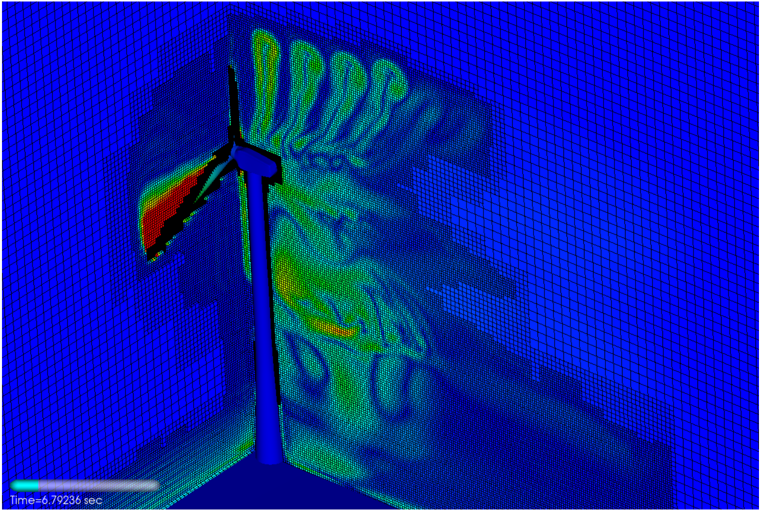
Wake refinement behind a leading turbine



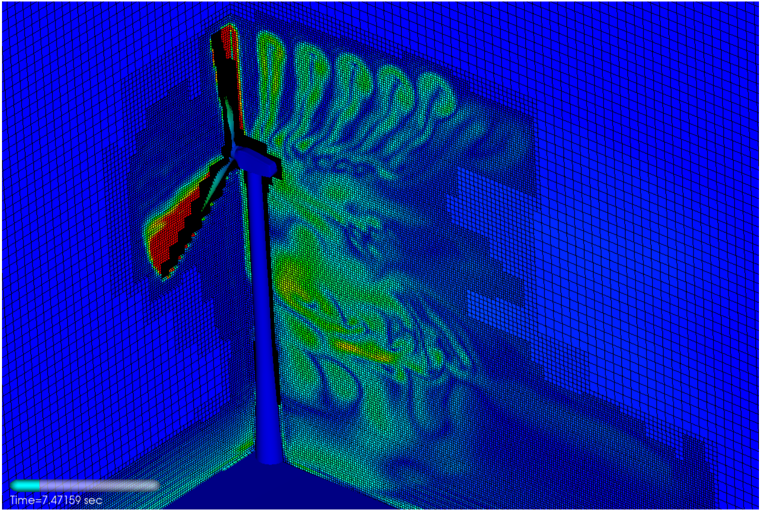
Wake refinement behind a leading turbine



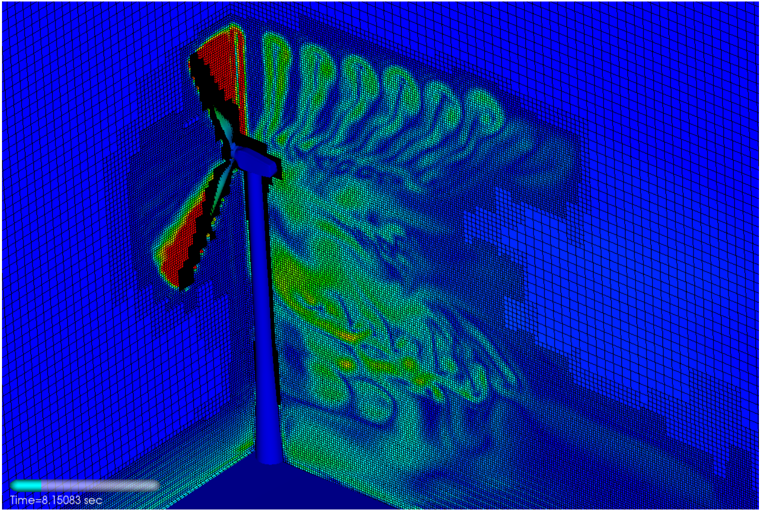
Wake refinement behind a leading turbine



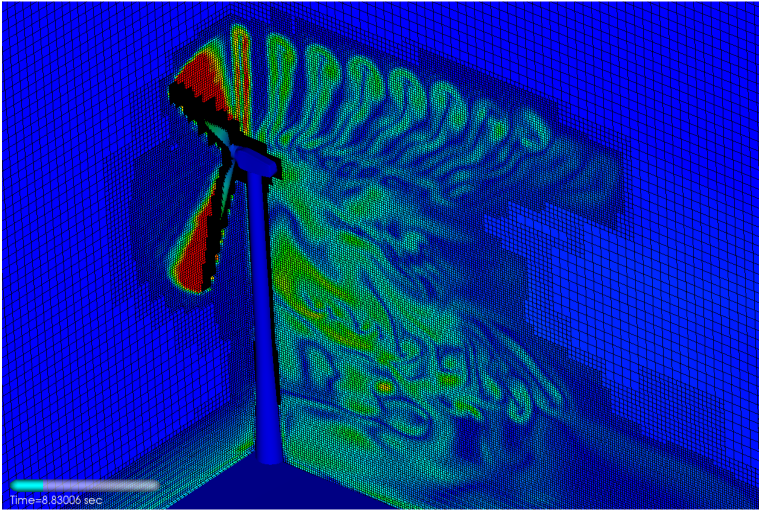
Wake refinement behind a leading turbine



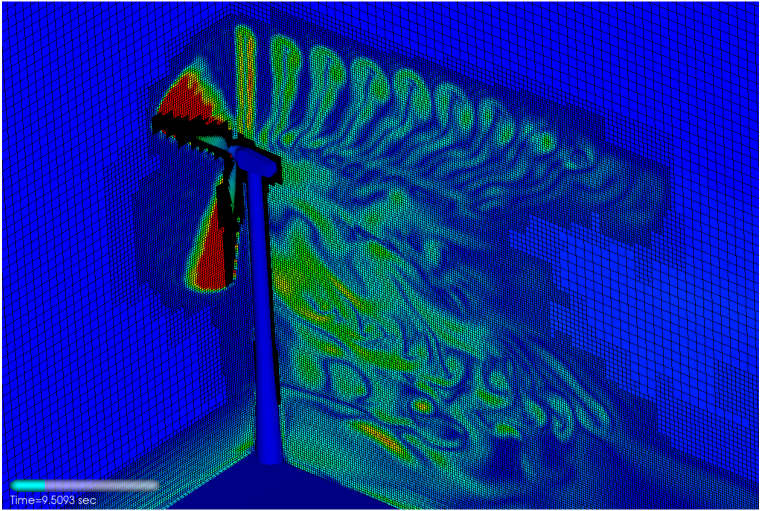
Wake refinement behind a leading turbine



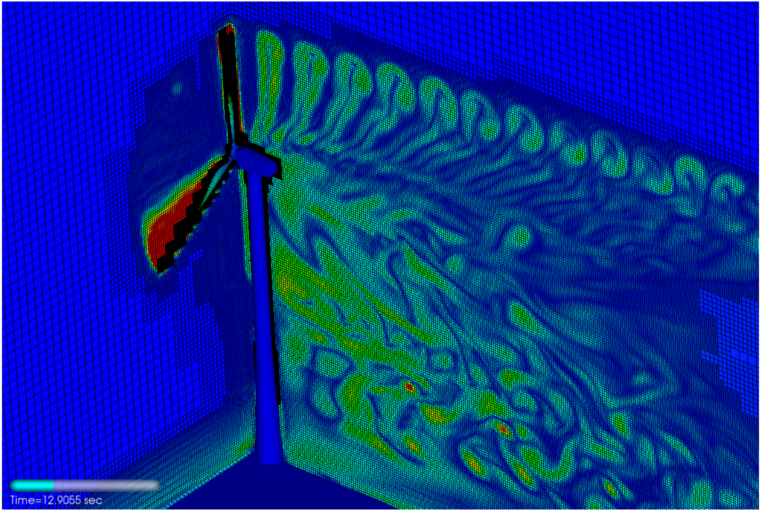
Wake refinement behind a leading turbine



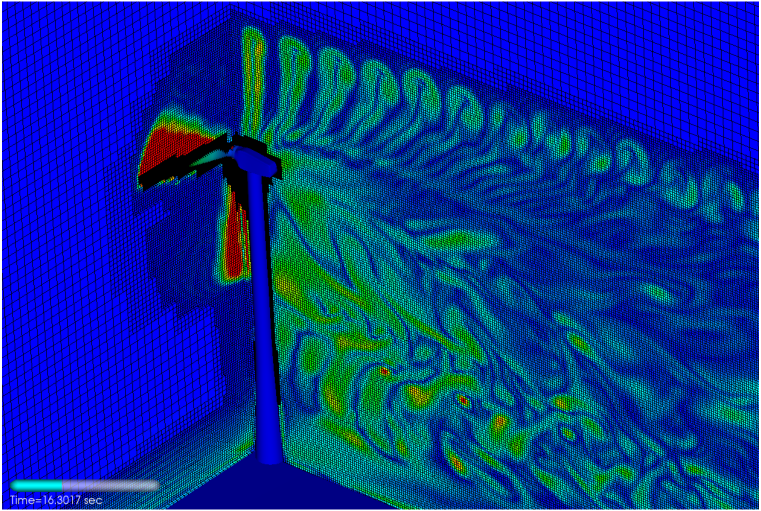
Wake refinement behind a leading turbine



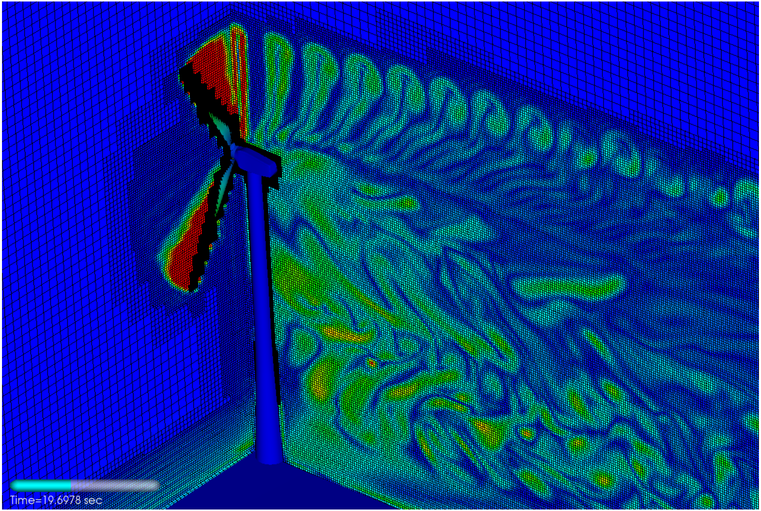
Wake refinement behind a leading turbine



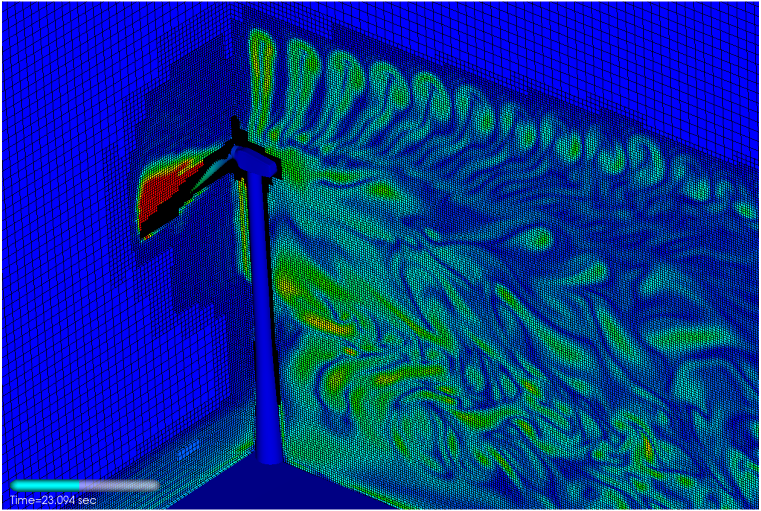
Wake refinement behind a leading turbine



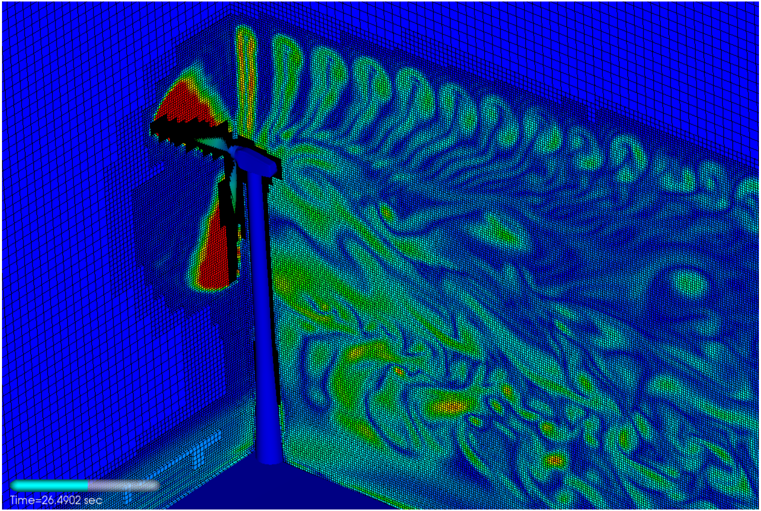
Wake refinement behind a leading turbine



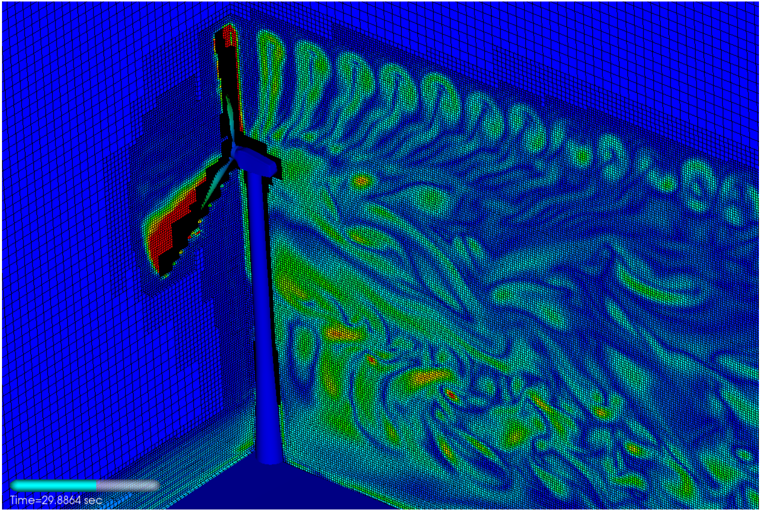
Wake refinement behind a leading turbine



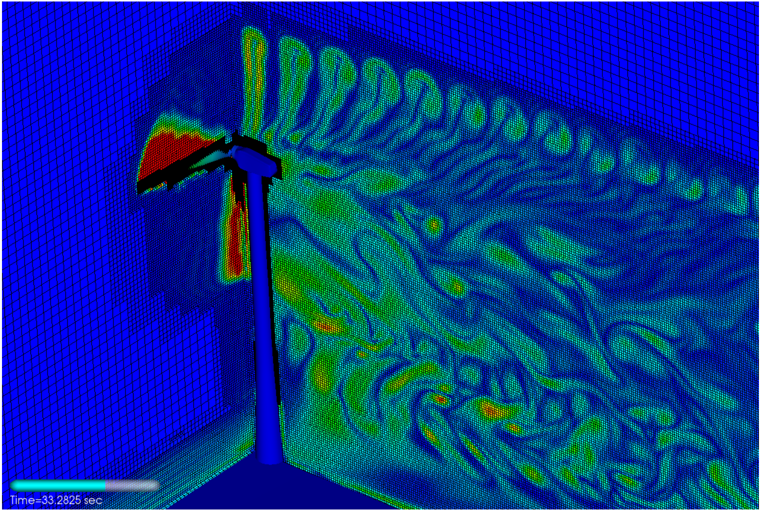
Wake refinement behind a leading turbine



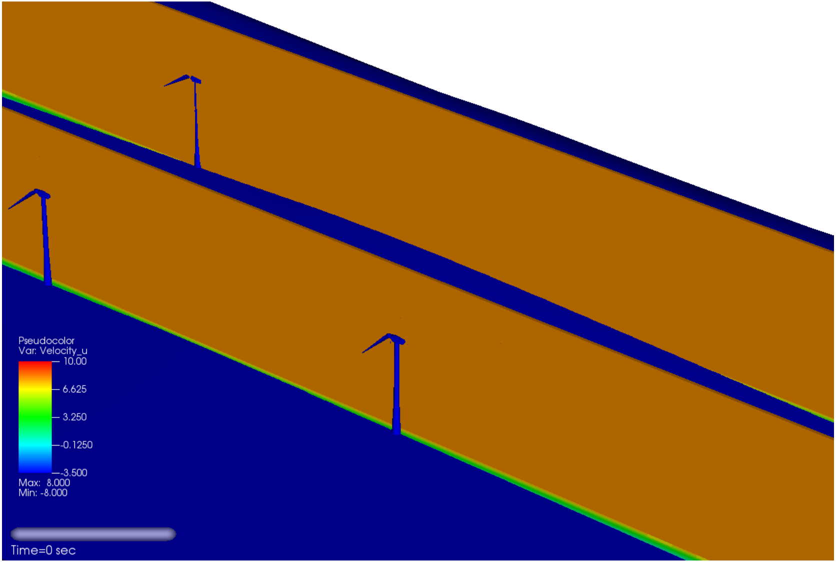
Wake refinement behind a leading turbine



Wake refinement behind a leading turbine

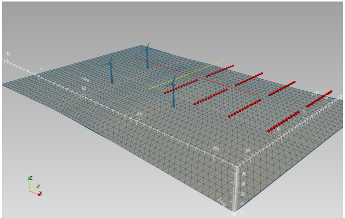


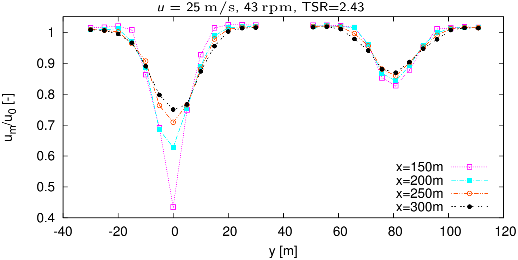
Axial velocity - $u = 8\,\mathrm{m/s}$, $33\,\mathrm{rpm}$

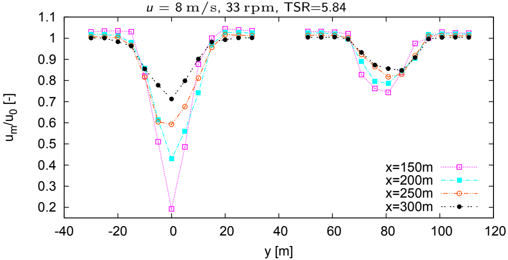


Mean point values

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



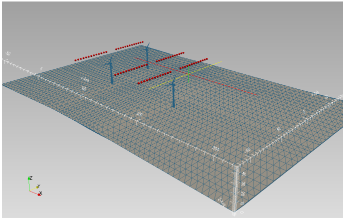


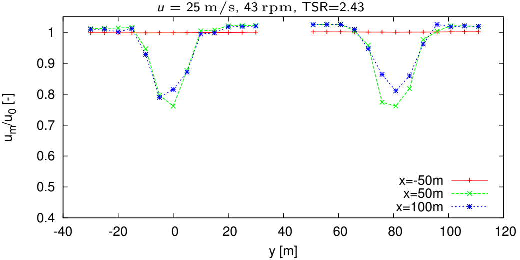


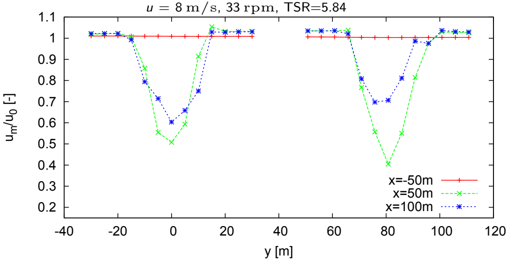
- Velocity deficits larger for higher TSR.

Mean point values

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



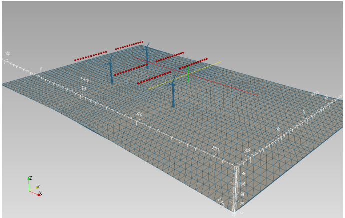


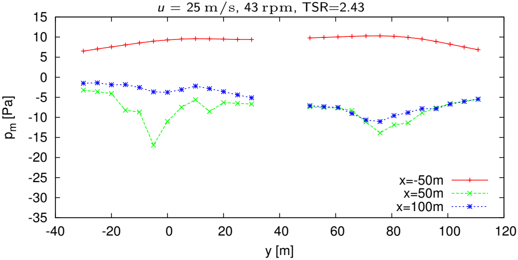


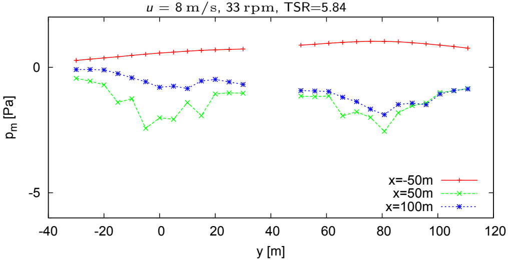
- Velocity deficits larger for higher TSR.
- Velocity deficit before 2nd turbine more homogenous.

Mean point values

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







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

Outline

Poisson equation

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

Poisson equation

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

Discrete Poisson equation in 2D:

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

Poisson equation

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

Discrete Poisson equation in 2D:

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

Operator

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

Poisson equation

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

Discrete Poisson equation in 2D:

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

Operator

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

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

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

Iterative methods

Jacobi iteration

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

Iterative methods

Jacobi iteration

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

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

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

Iterative methods

Jacobi iteration

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

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

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

Efficient parallelization / patch-wise application not possible!

Iterative methods

Jacobi iteration

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

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

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

Efficient parallelization / patch-wise application not possible!

Checker-board or Red-Black Gauss Seidel iteration

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

Iterative methods

Jacobi iteration

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

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

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

Efficient parallelization / patch-wise application not possible!

Checker-board or Red-Black Gauss Seidel iteration

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

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

Smoothing vs. solving

ν iterations with iterative linear solver

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

Smoothing vs. solving

ν iterations with iterative linear solver

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

Defect after m iterations

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

Smoothing vs. solving

ν iterations with iterative linear solver

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

Defect after m iterations

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

Defect after $m + \nu$ iterations

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

with correction

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

Smoothing vs. solving

ν iterations with iterative linear solver

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

Defect after m iterations

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

Defect after $m+\nu$ iterations

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

with correction

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

Neglecting the sub-iterations in the smoother we write

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

Smoothing vs. solving

ν iterations with iterative linear solver

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

Defect after m iterations

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

Defect after $m+\nu$ iterations

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

with correction

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

Neglecting the sub-iterations in the smoother we write

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

Observation: Oscillations are damped faster on coarser grid.

Smoothing vs. solving

ν iterations with iterative linear solver

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

Defect after m iterations

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

Defect after $m + \nu$ iterations

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

with correction

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

Neglecting the sub-iterations in the smoother we write

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

Observation: Oscillations are damped faster on coarser grid.

Coarse grid correction:

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

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

Two-grid correction method

Relaxation on current grid:

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

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

Two-grid correction method

Relaxation on current grid:

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

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

Algorithm:

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

$$d_c := \mathcal{R}(d)$$
$$v_c := \mathcal{S}(0, d_c, \mu)$$
$$v := \mathcal{P}(v_c)$$
$$Q^{n+1} := \bar{Q} + v$$

Two-grid correction method

Relaxation on current grid:

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

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

Algorithm:

$$\begin{aligned}
\bar{Q} &:= \mathcal{S}(Q^n, \psi, \nu) \\
d &:= \psi - \mathcal{A}(\bar{Q}) \\
d_c &:= \mathcal{R}(d) \\
v_c &:= \mathcal{S}(0, d_c, \mu) \\
v &:= \mathcal{P}(v_c) \\
Q^{n+1} &:= \bar{Q} + v
\end{aligned}$$

with smoothing:

$$\begin{aligned}
d &:= \psi - \mathcal{A}(Q) \\
v &:= \mathcal{S}(0, d, \nu) \\
r &:= d - \mathcal{A}(v) \\
d_c &:= \mathcal{R}(r) \\
v_c &:= \mathcal{S}(0, d_c, \mu) \\
v &:= v + \mathcal{P}(v_c) \\
Q^{n+1} &:= Q + v
\end{aligned}$$

Two-grid correction method

Relaxation on current grid:

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

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

Algorithm:

$$\begin{aligned} \bar{Q} &:= \mathcal{S}(Q^n, \psi, \nu) \\ d &:= \psi - \mathcal{A}(\bar{Q}) \\ \\ d_c &:= \mathcal{R}(d) \\ v_c &:= \mathcal{S}(0, d_c, \mu) \\ v &:= \mathcal{P}(v_c) \\ Q^{n+1} &:= \bar{Q} + v \end{aligned}$$

with smoothing:

$$\begin{aligned} d &:= \psi - \mathcal{A}(Q) \\ v &:= \mathcal{S}(0, d, \nu) \\ r &:= d - \mathcal{A}(v) \\ d_c &:= \mathcal{R}(r) \\ v_c &:= \mathcal{S}(0, d_c, \mu) \\ v &:= v + \mathcal{P}(v_c) \\ Q^{n+1} &:= Q + v \end{aligned}$$

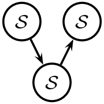
with pre- and post-iteration:

$$\begin{aligned} d &:= \psi - \mathcal{A}(Q) \\ v &:= \mathcal{S}(0, d, \nu_1) \\ r &:= d - \mathcal{A}(v) \\ d_c &:= \mathcal{R}(r) \\ v_c &:= \mathcal{S}(0, d_c, \mu) \\ v &:= v + \mathcal{P}(v_c) \\ d &:= d - \mathcal{A}(v) \\ r &:= \mathcal{S}(0, d, \nu_2) \\ Q^{n+1} &:= Q + v + r \end{aligned}$$

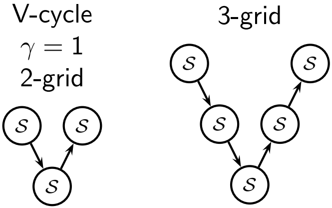
[Hackbusch, 1985]

Multi-level methods and cycles

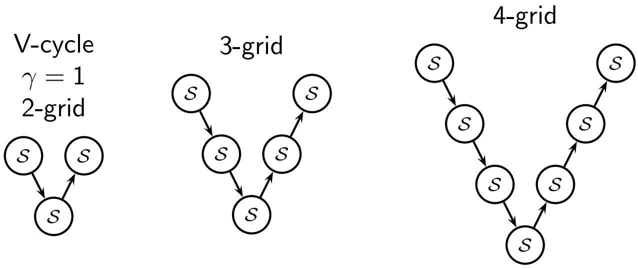
V-cycle
$\gamma = 1$
2-grid



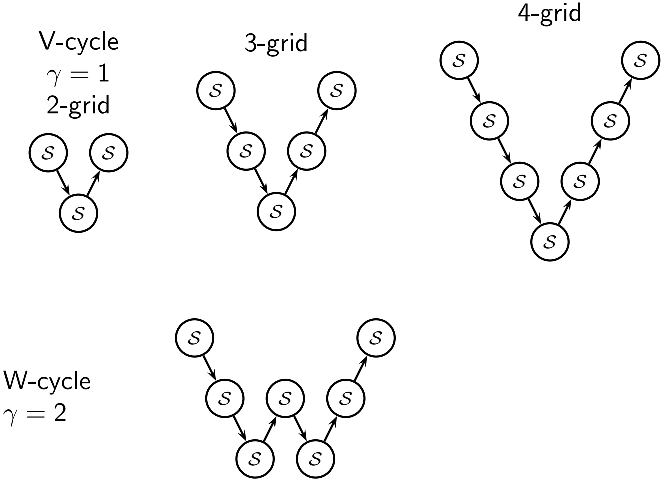
Multi-level methods and cycles



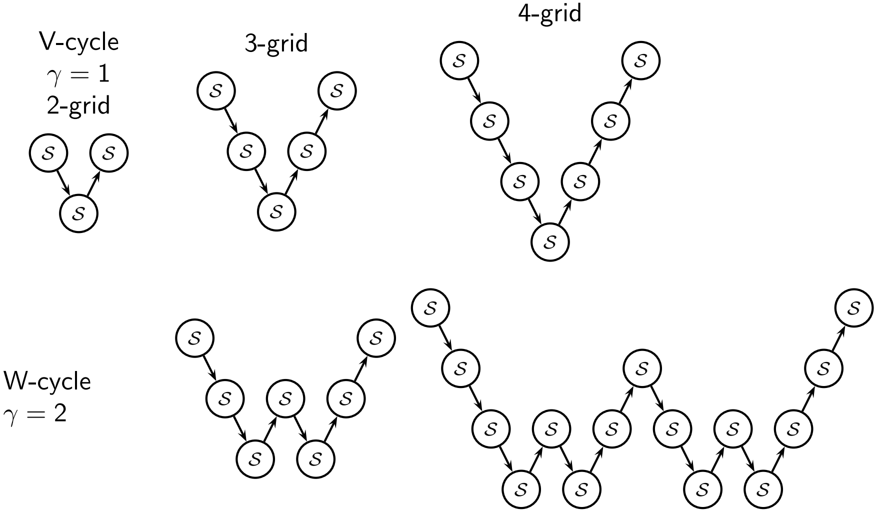
Multi-level methods and cycles



Multi-level methods and cycles



Multi-level methods and cycles



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

Stencil modification at coarse-fine boundaries in 1D

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

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

Stencil modification at coarse-fine boundaries in 1D

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

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

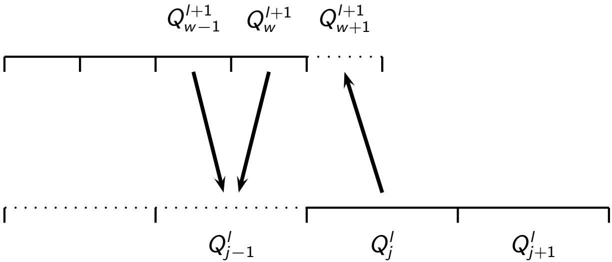
H is approximation to *derivative* of Q^l.

Stencil modification at coarse-fine boundaries in 1D

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

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

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

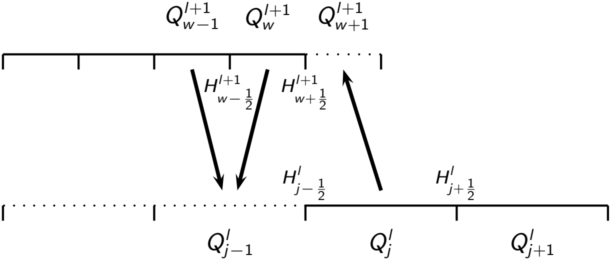


Stencil modification at coarse-fine boundaries in 1D

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

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

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

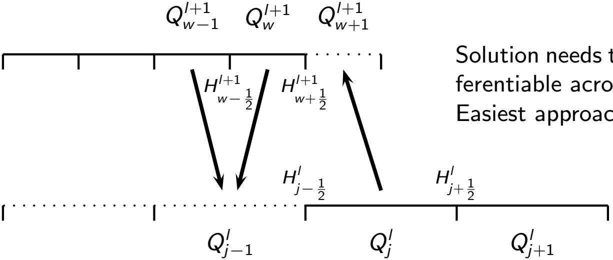


Stencil modification at coarse-fine boundaries in 1D

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

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

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



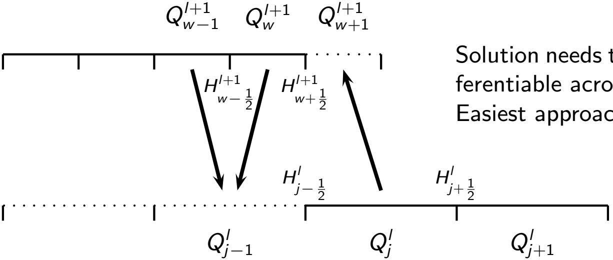
Solution needs to be continuously differentiable across interface.
Easiest approach: $H_{w+\frac{1}{2}}^{l+1} \equiv H_{j-\frac{1}{2}}^l$

Stencil modification at coarse-fine boundaries in 1D

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

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

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



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

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

Stencil modification at coarse-fine boundaries in 1D II

Set $H^{l+1}_{w+\frac{1}{2}} = H_{\mathcal{I}}$.

Stencil modification at coarse-fine boundaries in 1D II

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

$$\frac{Q^{l+1}_{w+1} - Q^{l+1}_{w}}{\Delta x_{l+1}} = \frac{Q^{l}_{j} - Q^{l+1}_{w}}{\frac{3}{2}\Delta x_{l+1}}$$

Stencil modification at coarse-fine boundaries in 1D II

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

$$\frac{Q^{l+1}_{w+1} - Q^{l+1}_{w}}{\Delta x_{l+1}} = \frac{Q^{l}_{j} - Q^{l+1}_{w}}{\frac{3}{2}\Delta x_{l+1}}$$

from which we readily derive

$$Q^{l+1}_{w+1} = \frac{2}{3} Q^{l}_{j} + \frac{1}{3} Q^{l+1}_{w}$$

for the boundary cell on $l+1$.

Stencil modification at coarse-fine boundaries in 1D II

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

$$\frac{Q^{l+1}_{w+1} - Q^{l+1}_{w}}{\Delta x_{l+1}} = \frac{Q^{l}_{j} - Q^{l+1}_{w}}{\frac{3}{2}\Delta x_{l+1}}$$

from which we readily derive

$$Q^{l+1}_{w+1} = \frac{2}{3} Q^{l}_{j} + \frac{1}{3} Q^{l+1}_{w}$$

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

Stencil modification at coarse-fine boundaries in 1D II

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

$$\frac{Q^{l+1}_{w+1} - Q^{l+1}_{w}}{\Delta x_{l+1}} = \frac{Q^{l}_{j} - Q^{l+1}_{w}}{\frac{3}{2}\Delta x_{l+1}}$$

from which we readily derive

$$Q^{l+1}_{w+1} = \frac{2}{3}Q^{l}_{j} + \frac{1}{3}Q^{l+1}_{w}$$

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

Correction pass [Martin, 1998]

1. $\delta H^{l+1}_{j-\frac{1}{2}} := -H^{l}_{j-\frac{1}{2}}$

Stencil modification at coarse-fine boundaries in 1D II

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

$$\frac{Q^{l+1}_{w+1} - Q^{l+1}_{w}}{\Delta x_{l+1}} = \frac{Q^{l}_{j} - Q^{l+1}_{w}}{\frac{3}{2}\Delta x_{l+1}}$$

from which we readily derive

$$Q^{l+1}_{w+1} = \frac{2}{3} Q^{l}_{j} + \frac{1}{3} Q^{l+1}_{w}$$

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

Correction pass [Martin, 1998]

1. $\delta H^{l+1}_{j-\frac{1}{2}} := -H^{l}_{j-\frac{1}{2}}$
2. $\delta H^{l+1}_{j-\frac{1}{2}} := \delta H^{l+1}_{j-\frac{1}{2}} + H^{l+1}_{w+\frac{1}{2}} = -H^{l}_{j-\frac{1}{2}} + (Q^{l}_{j} - Q^{l+1}_{w}) / \frac{3}{2}\Delta x_{l+1}$

Stencil modification at coarse-fine boundaries in 1D II

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

$$\frac{Q^{l+1}_{w+1} - Q^{l+1}_{w}}{\Delta x_{l+1}} = \frac{Q^{l}_{j} - Q^{l+1}_{w}}{\frac{3}{2}\Delta x_{l+1}}$$

from which we readily derive

$$Q^{l+1}_{w+1} = \frac{2}{3} Q^{l}_{j} + \frac{1}{3} Q^{l+1}_{w}$$

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

Correction pass [Martin, 1998]

1. $\delta H^{l+1}_{j-\frac{1}{2}} := -H^{l}_{j-\frac{1}{2}}$
2. $\delta H^{l+1}_{j-\frac{1}{2}} := \delta H^{l+1}_{j-\frac{1}{2}} + H^{l+1}_{w+\frac{1}{2}} = -H^{l}_{j-\frac{1}{2}} + (Q^{l}_{j} - Q^{l+1}_{w}) / \frac{3}{2}\Delta x_{l+1}$
3. $\check{d}^{l}_{j} := d^{l}_{j} + \frac{1}{\Delta x_l} \delta H^{l+1}_{j-\frac{1}{2}}$

Stencil modification at coarse-fine boundaries in 1D II

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

$$\frac{Q^{l+1}_{w+1} - Q^{l+1}_{w}}{\Delta x_{l+1}} = \frac{Q^{l}_{j} - Q^{l+1}_{w}}{\frac{3}{2}\Delta x_{l+1}}$$

from which we readily derive

$$Q^{l+1}_{w+1} = \frac{2}{3} Q^{l}_{j} + \frac{1}{3} Q^{l+1}_{w}$$

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

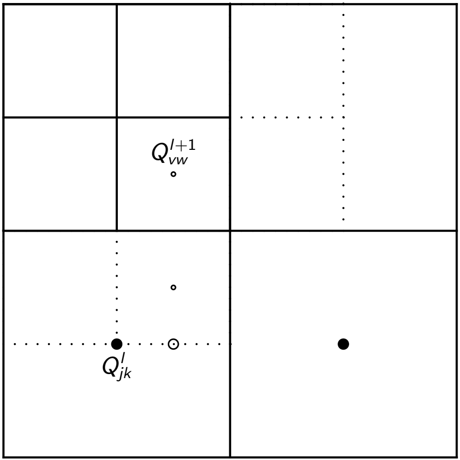
Correction pass [Martin, 1998]

1. $\delta H^{l+1}_{j-\frac{1}{2}} := -H^{l}_{j-\frac{1}{2}}$
2. $\delta H^{l+1}_{j-\frac{1}{2}} := \delta H^{l+1}_{j-\frac{1}{2}} + H^{l+1}_{w+\frac{1}{2}} = -H^{l}_{j-\frac{1}{2}} + (Q^{l}_{j} - Q^{l+1}_{w})/\frac{3}{2}\Delta x_{l+1}$
3. $\check{d}^{l}_{j} := d^{l}_{j} + \frac{1}{\Delta x_l}\delta H^{l+1}_{j-\frac{1}{2}}$

yields

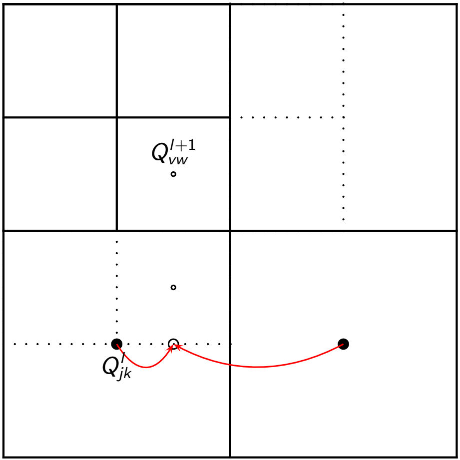
$$\check{d}^{l}_{j} = \psi_j - \frac{1}{\Delta x_l}\left(\frac{1}{\Delta x_l}(Q^{l}_{j+1} - Q^{l}_{j}) - \frac{2}{3\Delta x_{l+1}}(Q^{l}_{j} - Q^{l+1}_{w})\right)$$

Stencil modification at coarse-fine boundaries: 2D



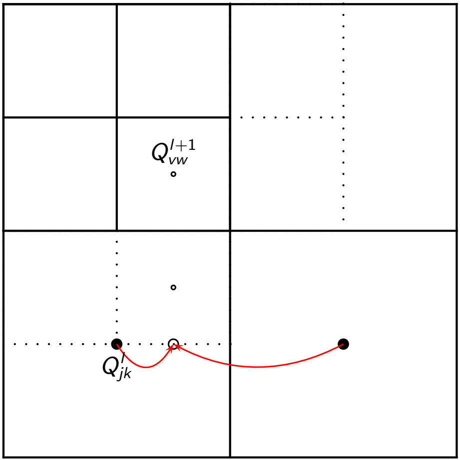
$$Q_{v,w-1}^{l+1} = \qquad +$$

Stencil modification at coarse-fine boundaries: 2D



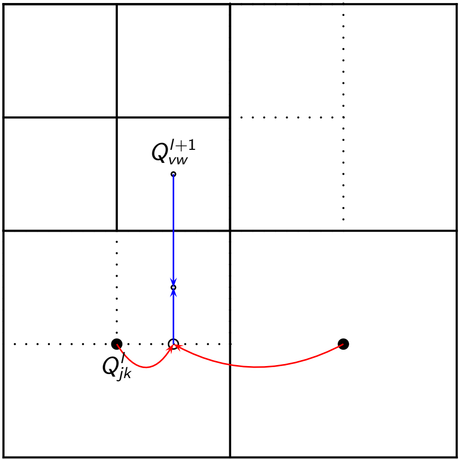
$$Q^{l+1}_{v,w-1} = \qquad +$$

Stencil modification at coarse-fine boundaries: 2D



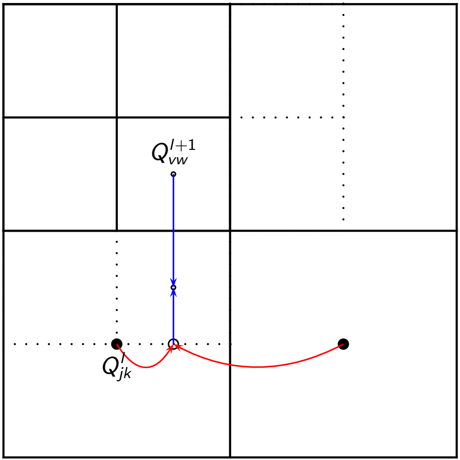
$$Q_{v,w-1}^{l+1} = \qquad + \left(\frac{3}{4}Q_{jk}^{l} + \frac{1}{4}Q_{j+1,k}^{l}\right)$$

Stencil modification at coarse-fine boundaries: 2D



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

Stencil modification at coarse-fine boundaries: 2D



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

In general:

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

with

$$f = \frac{x^v_{1,l+1} - x^j_{1,l}}{\Delta x_{1,l}}$$

Components of an SAMR multigrid method

- Stencil operators

Components of an SAMR multigrid method

- Stencil operators
  - Application of defect $d^l = \psi^l - \mathcal{A}(Q^l)$ on each grid $G_{l,m}$ of level l

Components of an SAMR multigrid method

- Stencil operators
  - Application of defect $d^l = \psi^l - \mathcal{A}(Q^l)$ on each grid $G_{l,m}$ of level l
  - Computation of correction $v^l = \mathcal{S}(0, d^l, \nu)$ on each grid of level l

Components of an SAMR multigrid method

- Stencil operators
  - Application of defect $d^l = \psi^l - \mathcal{A}(Q^l)$ on each grid $G_{l,m}$ of level l
  - Computation of correction $v^l = \mathcal{S}(0, d^l, \nu)$ on each grid of level l
- Boundary (ghost cell) operators

Components of an SAMR multigrid method

- Stencil operators
  - Application of defect $d^l = \psi^l - \mathcal{A}(Q^l)$ on each grid $G_{l,m}$ of level l
  - Computation of correction $v^l = \mathcal{S}(0, d^l, \nu)$ on each grid of level l
- Boundary (ghost cell) operators
  - Synchronization of Q^l and v^l on $\tilde{S}_l^1$

Components of an SAMR multigrid method

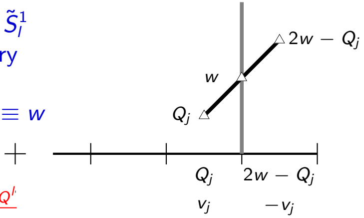
- Stencil operators
  - Application of defect $d^l = \psi^l - \mathcal{A}(Q^l)$ on each grid $G_{l,m}$ of level l
  - Computation of correction $v^l = \mathcal{S}(0, d^l, \nu)$ on each grid of level l
- Boundary (ghost cell) operators
  - Synchronization of Q^l and v^l on $\tilde{S}_l^1$
  - Specification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

Components of an SAMR multigrid method

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

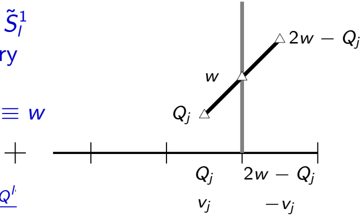
Components of an SAMR multigrid method

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



Components of an SAMR multigrid method

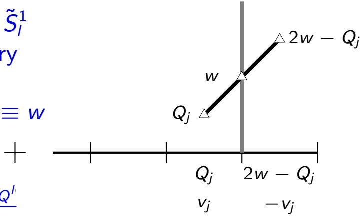
- Stencil operators
  - Application of defect $d^l = \psi^l - \mathcal{A}(Q^l)$ on each grid $G_{l,m}$ of level l
  - Computation of correction $v^l = \mathcal{S}(0, d^l, \nu)$ on each grid of level l
- Boundary (ghost cell) operators
  - Synchronization of Q^l and v^l on $\tilde{S}_l^1$
  - Specification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



- Coarse-fine boundary flux accumulation and application of δH^{l+1} on defect d^l

Components of an SAMR multigrid method

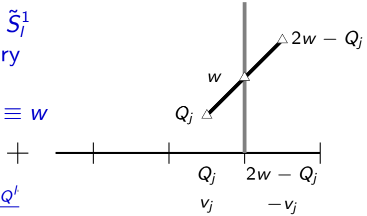
- Stencil operators
  - Application of defect $d^l = \psi^l - \mathcal{A}(Q^l)$ on each grid $G_{l,m}$ of level l
  - Computation of correction $v^l = \mathcal{S}(0, d^l, \nu)$ on each grid of level l
- Boundary (ghost cell) operators
  - Synchronization of Q^l and v^l on $\tilde{S}_l^1$
  - Specification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



- Coarse-fine boundary flux accumulation and application of δH^{l+1} on defect d^l
- Standard prolongation and restriction on grids between adjacent levels

Components of an SAMR multigrid method

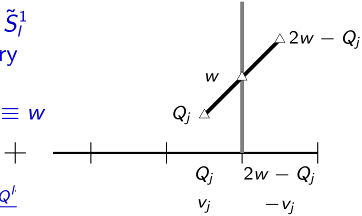
- Stencil operators
  - Application of defect $d^l = \psi^l - \mathcal{A}(Q^l)$ on each grid $G_{l,m}$ of level l
  - Computation of correction $v^l = \mathcal{S}(0, d^l, \nu)$ on each grid of level l
- Boundary (ghost cell) operators
  - Synchronization of Q^l and v^l on $\tilde{S}_l^1$
  - Specification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



- Coarse-fine boundary flux accumulation and application of δH^{l+1} on defect d^l
- Standard prolongation and restriction on grids between adjacent levels
- Adaptation criteria

Components of an SAMR multigrid method

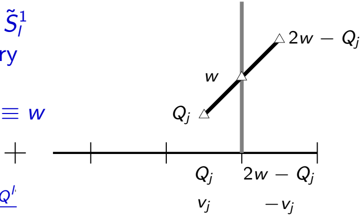
- Stencil operators
  - Application of defect $d^l = \psi^l - \mathcal{A}(Q^l)$ on each grid $G_{l,m}$ of level l
  - Computation of correction $v^l = \mathcal{S}(0, d^l, \nu)$ on each grid of level l
- Boundary (ghost cell) operators
  - Synchronization of Q^l and v^l on $\tilde{S}_l^1$
  - Specification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



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

Components of an SAMR multigrid method

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



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

Additive geometric multigrid algorithm

AdvanceLevelMG(l) - Correction Scheme

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

Additive Geometric Multiplicative Multigrid Algorithm

Start - Start iteration on level l_{max}

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

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

Example

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

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

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

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

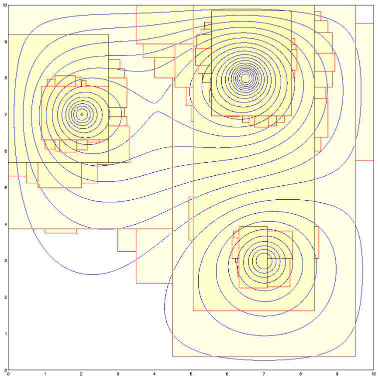
Example

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

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

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

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



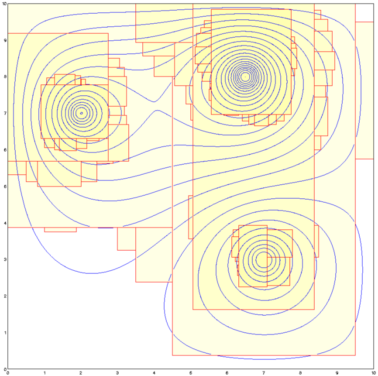
Example

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

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

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

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



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

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

Some comments on parabolic problems

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

References I

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

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

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

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

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

References II

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

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

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

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

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

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

References III

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

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

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

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

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

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