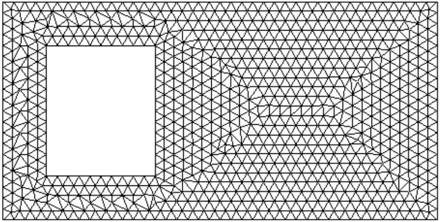
Ein massiv paralleles, dynamisch adaptives Lattice-Boltzmann Verfahren für Fluid-Struktur-Kopplung

Ralf Deiterding

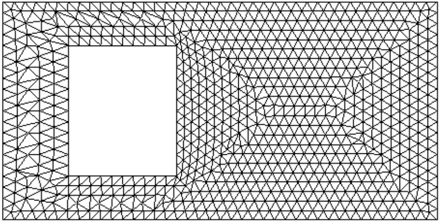
Deutsches Zentrum für Luft- und Raumfahrt
Bunsenstr. 10, Göttingen, Germany
E-mail: ralf.deiterding@dlr.de

May 12, 2014

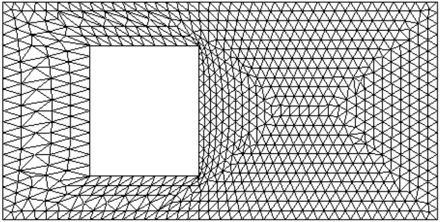
FSI simulation with unstructured grids



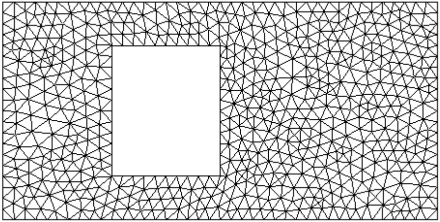
FSI simulation with unstructured grids



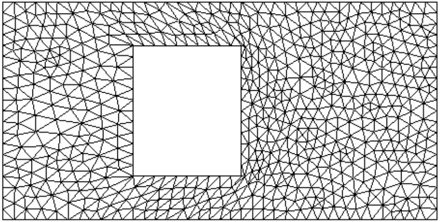
FSI simulation with unstructured grids



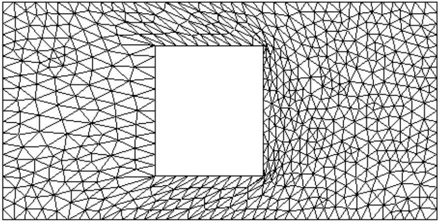
FSI simulation with unstructured grids



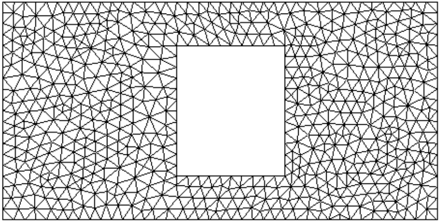
FSI simulation with unstructured grids



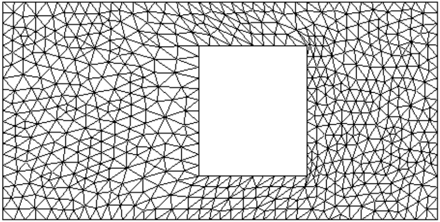
FSI simulation with unstructured grids



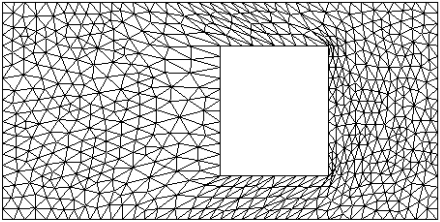
FSI simulation with unstructured grids



FSI simulation with unstructured grids

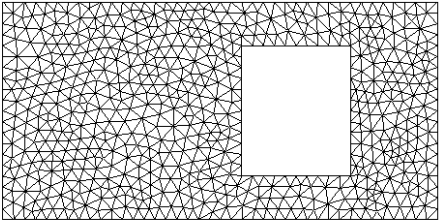


FSI simulation with unstructured grids



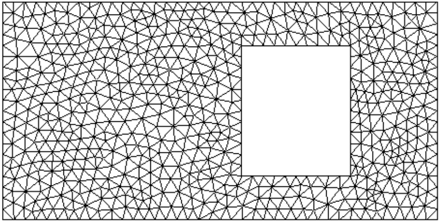
FSI simulation with unstructured grids

- Global grid (re-)generation is part of the simulation (major parallelization and scalability obstacle)
- Sophisticated data remapping required when grid topology changes



FSI simulation with unstructured grids

- Global grid (re-)generation is part of the simulation (major parallelization and scalability obstacle)
- Sophisticated data remapping required when grid topology changes
- Mesh deformation/motion requires arbitrary Lagrangian (re-)formulation of equations

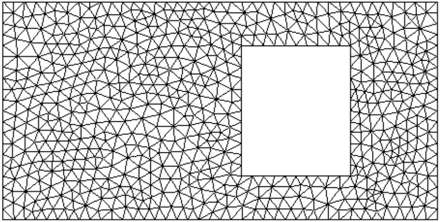


FSI simulation with unstructured grids

- Global grid (re-)generation is part of the simulation (major parallelization and scalability obstacle)
- Sophisticated data remapping required when grid topology changes
- Mesh deformation/motion requires arbitrary Lagrangian (re-)formulation of equations

► Some generalities about unstructured grids:

+ Hanging nodes can be avoided
- Higher order difficult to achieve
- High computational performance challenging



FSI simulation with unstructured grids

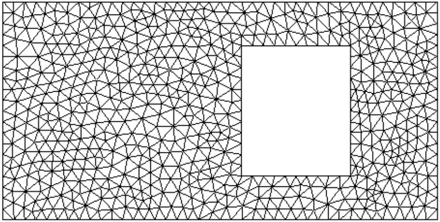
- Global grid (re-)generation is part of the simulation (major parallelization and scalability obstacle)
- Sophisticated data remapping required when grid topology changes
- Mesh deformation/motion requires arbitrary Lagrangian (re-)formulation of equations

▶ Some generalities about unstructured grids:

\+ Hanging nodes can be avoided

- Higher order difficult to achieve
- High computational performance challenging

$\longrightarrow$ Alternative: Adaptive Cartesian methods with embedded boundaries



Outline

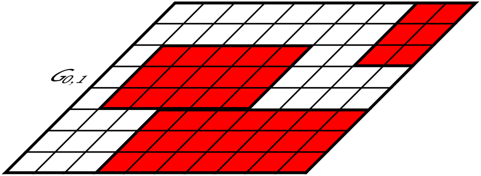
Collaboration with

- Sean Mauch and Daniel Meiron (Computational and Applied Mathematics, California Institute of Technology)
- Stuart Laurence (University of Maryland, College Park)
- Stephen Wood (University of Tennessee Knoxville, Oak Ridge National Laboratory)

Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(\mathbf{x},t) + \nabla \cdot \mathbf{f}(\mathbf{q}(\mathbf{x},t)) = 0$

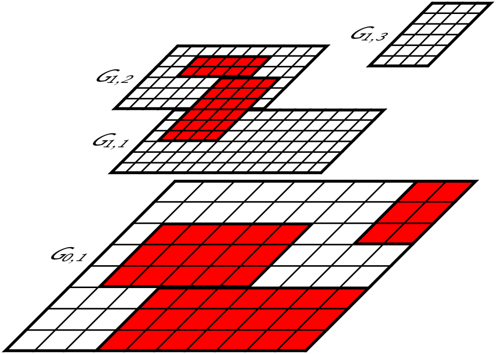
- Refined blocks overlay coarser ones



Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(\mathbf{x},t) + \nabla \cdot \mathbf{f}(\mathbf{q}(\mathbf{x},t)) = 0$

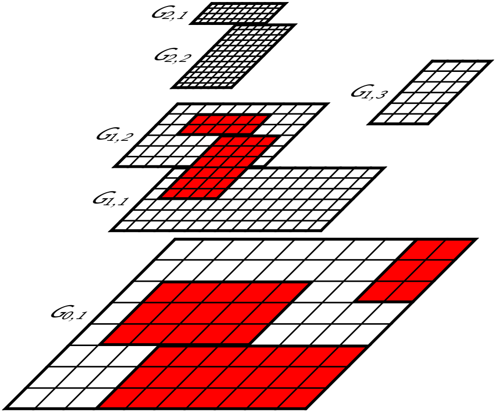
- Refined blocks overlay coarser ones



Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(\mathbf{x},t) + \nabla \cdot \mathbf{f}(\mathbf{q}(\mathbf{x},t)) = 0$

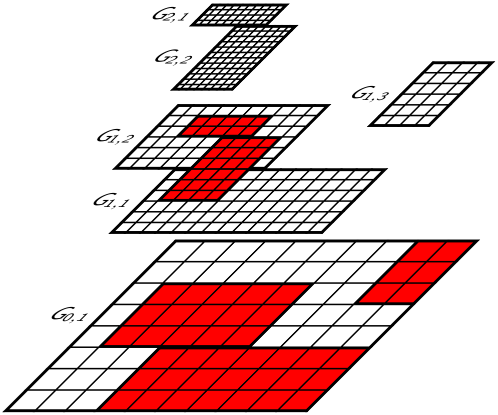
- Refined blocks overlay coarser ones



Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(\mathbf{x}, t) + \nabla \cdot \mathbf{f}(\mathbf{q}(\mathbf{x}, t)) = 0$

- Refined blocks overlay coarser ones
- Refinement in space *and time* by factor r_l



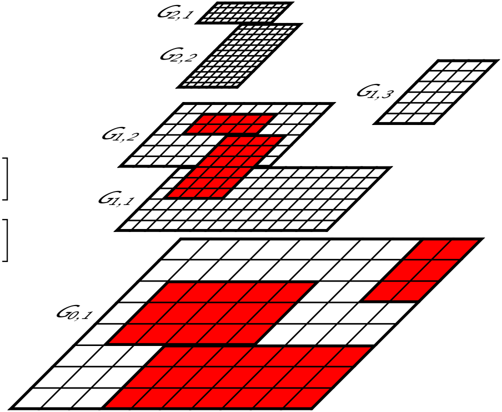
Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(\mathbf{x},t) + \nabla \cdot \mathbf{f}(\mathbf{q}(\mathbf{x},t)) = 0$

- Refined blocks overlay coarser ones
- Refinement in space *and time* by factor r_l
- Block (aka patch) based data structures
- Numerical scheme

$$\mathbf{Q}_{jk}^{n+1} = \mathbf{Q}_{jk}^{n} - \frac{\Delta t}{\Delta x_1}\left[\mathbf{F}^1_{j+\frac{1}{2},k} - \mathbf{F}^1_{j-\frac{1}{2},k}\right] - \frac{\Delta t}{\Delta x_2}\left[\mathbf{F}^2_{j,k+\frac{1}{2}} - \mathbf{F}^2_{j,k-\frac{1}{2}}\right]$$

only for single patch necessary



Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(\mathbf{x},t) + \nabla \cdot \mathbf{f}(\mathbf{q}(\mathbf{x},t)) = 0$

- Refined blocks overlay coarser ones
- Refinement in space *and time* by factor r_l
- Block (aka patch) based data structures

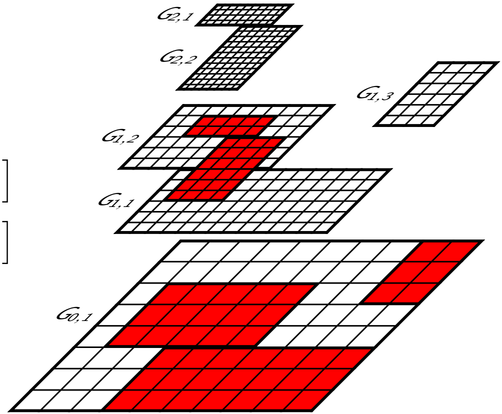
\+ Numerical scheme

$$\mathbf{Q}_{jk}^{n+1} = \mathbf{Q}_{jk}^{n} - \frac{\Delta t}{\Delta x_1}\left[\mathbf{F}^1_{j+\frac{1}{2},k} - \mathbf{F}^1_{j-\frac{1}{2},k}\right] - \frac{\Delta t}{\Delta x_2}\left[\mathbf{F}^2_{j,k+\frac{1}{2}} - \mathbf{F}^2_{j,k-\frac{1}{2}}\right]$$

only for single patch necessary

\+ Efficient cache-reuse / vectorization possible

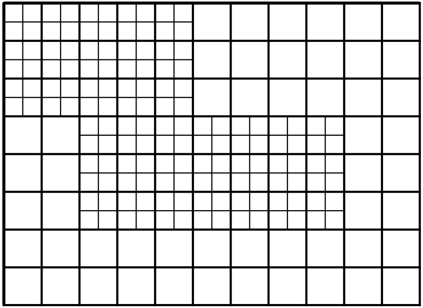
\- Cluster-algorithm necessary



Level transfer / setting of ghost cells

Conservative averaging (restriction):

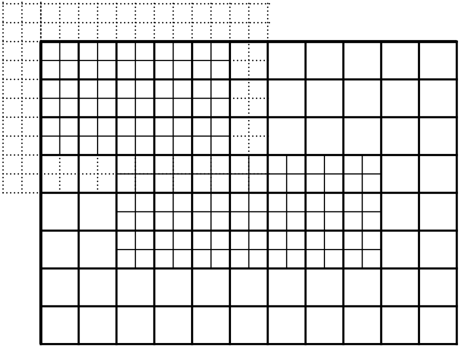
$$\hat{\mathbf{Q}}^l_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$



Level transfer / setting of ghost cells

Conservative averaging (restriction):

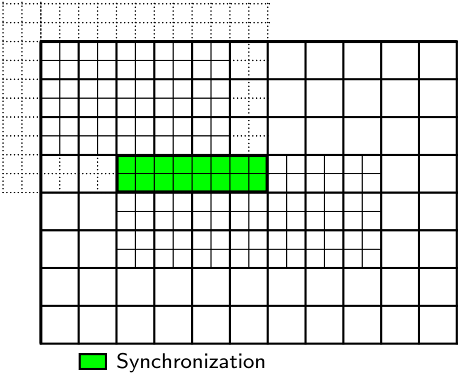
$$\hat{\mathbf{Q}}^l_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$



Level transfer / setting of ghost cells

Conservative averaging (restriction):

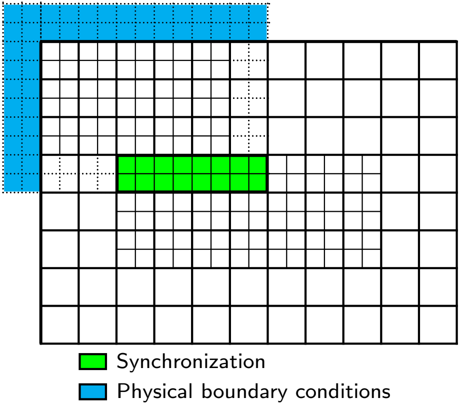
$$\hat{\mathbf{Q}}^{l}_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$



Level transfer / setting of ghost cells

Conservative averaging (restriction):

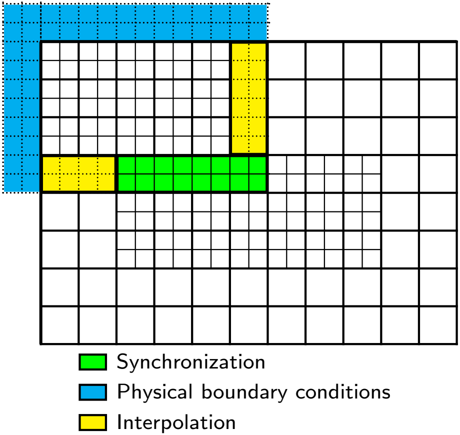
$$\hat{\mathbf{Q}}^{l}_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$



Level transfer / setting of ghost cells

Conservative averaging (restriction):

$$\hat{\mathbf{Q}}^{l}_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$



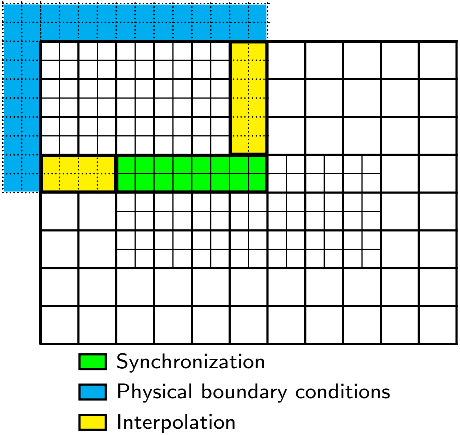
Level transfer / setting of ghost cells

Conservative averaging (restriction):

$$\hat{\mathbf{Q}}^{l}_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$

Bilinear interpolation (prolongation):

$$\check{\mathbf{Q}}^{l+1}_{vw} := (1-f_1)(1-f_2)\,\mathbf{Q}^{l}_{j-1,k-1} + f_1(1-f_2)\,\mathbf{Q}^{l}_{j,k-1} + (1-f_1)f_2\,\mathbf{Q}^{l}_{j-1,k} + f_1 f_2\,\mathbf{Q}^{l}_{jk}$$



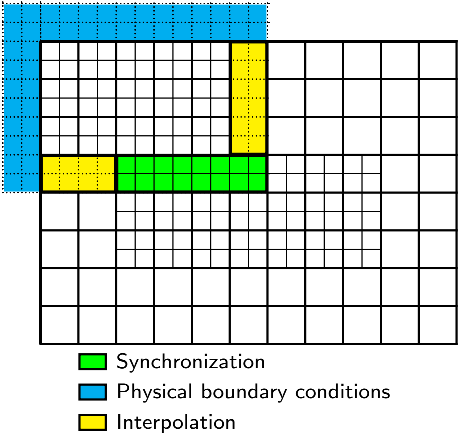
Level transfer / setting of ghost cells

Conservative averaging (restriction):

$$\hat{\mathbf{Q}}^l_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$

Bilinear interpolation (prolongation):

$$\check{\mathbf{Q}}^{l+1}_{vw} := (1-f_1)(1-f_2)\,\mathbf{Q}^l_{j-1,k-1} + f_1(1-f_2)\,\mathbf{Q}^l_{j,k-1} + (1-f_1)f_2\,\mathbf{Q}^l_{j-1,k} + f_1 f_2\,\mathbf{Q}^l_{jk}$$



For boundary conditions: linear time interpolation

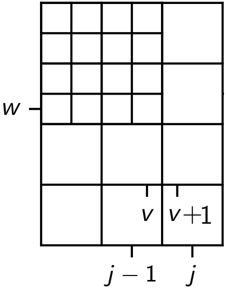
$$\tilde{\mathbf{Q}}^{l+1}(t+\kappa\Delta t_{l+1}) := \left(1-\frac{\kappa}{r_{l+1}}\right)\check{\mathbf{Q}}^{l+1}(t)+\frac{\kappa}{r_{l+1}}\,\check{\mathbf{Q}}^{l+1}(t+\Delta t_l) \quad \text{for } \kappa=0,\dots r_{l+1}$$

Conservative flux correction

Example: Cell j, k

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

Correction pass:



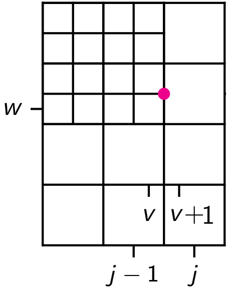
Conservative flux correction

Example: Cell j, k

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

Correction pass:

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



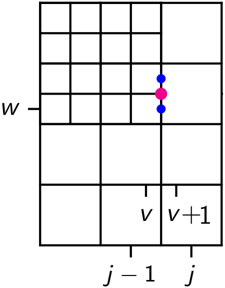
Conservative flux correction

Example: Cell j, k

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

Correction pass:

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



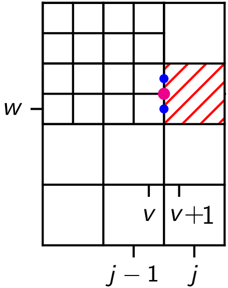
Conservative flux correction

Example: Cell j, k

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

Correction pass:

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



The basic recursive algorithm

```
AdvanceLevel(l)

  Repeat r_l times
      Set ghost cells of Q^l(t)


      UpdateLevel(l)






      t := t + Δt_l
```

The basic recursive algorithm

AdvanceLevel(l)

Repeat r_l times
    Set ghost cells of $\mathbf{Q}^l(t)$

    UpdateLevel(l)
    If level $l+1$ exists?
        Set ghost cells of $\mathbf{Q}^l(t+\Delta t_l)$
        AdvanceLevel($l+1$)

    $t := t + \Delta t_l$

- Recursion

The basic recursive algorithm

AdvanceLevel(l)

  Repeat r_l times
    Set ghost cells of $\mathbf{Q}^l(t)$

    UpdateLevel(l)
    If level $l+1$ exists?
      Set ghost cells of $\mathbf{Q}^l(t+\Delta t_l)$
      AdvanceLevel($l+1$)
      Average $\mathbf{Q}^{l+1}(t+\Delta t_l)$ onto $\mathbf{Q}^l(t+\Delta t_l)$
      Correct $\mathbf{Q}^l(t+\Delta t_l)$ with $\delta\mathbf{F}^{l+1}$
    $t := t + \Delta t_l$

- Recursion
- Restriction and flux correction

The basic recursive algorithm

AdvanceLevel(l)

Repeat r_l times
    Set ghost cells of $\mathbf{Q}^l(t)$
    If time to regrid?
        Regrid(l)
    UpdateLevel(l)
    If level $l+1$ exists?
        Set ghost cells of $\mathbf{Q}^l(t+\Delta t_l)$
        AdvanceLevel($l+1$)
        Average $\mathbf{Q}^{l+1}(t+\Delta t_l)$ onto $\mathbf{Q}^l(t+\Delta t_l)$
        Correct $\mathbf{Q}^l(t+\Delta t_l)$ with $\delta\mathbf{F}^{l+1}$
    $t := t+\Delta t_l$

- Recursion
- Restriction and flux correction
- Re-organization of hierarchical data

The basic recursive algorithm

```
AdvanceLevel(l)

  Repeat r_l times
      Set ghost cells of Q^l(t)
      If time to regrid?
          Regrid(l)
      UpdateLevel(l)
      If level l+1 exists?
          Set ghost cells of Q^l(t+Δt_l)
          AdvanceLevel(l+1)
          Average Q^{l+1}(t+Δt_l) onto Q^l(t+Δt_l)
          Correct Q^l(t+Δt_l) with δF^{l+1}
      t := t+Δt_l

Start - Start integration on level 0

     l = 0, r_0 = 1
     AdvanceLevel(l)
```

- Recursion
- Restriction and flux correction
- Re-organization of hierarchical data

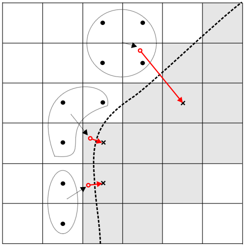
The basic recursive algorithm

```
AdvanceLevel(l)

  Repeat r_l times
      Set ghost cells of Q^l(t)
      If time to regrid?
          Regrid(l)
      UpdateLevel(l)
      If level l+1 exists?
          Set ghost cells of Q^l(t+Δt_l)
          AdvanceLevel(l+1)
          Average Q^(l+1)(t+Δt_l) onto Q^l(t+Δt_l)
          Correct Q^l(t+Δt_l) with δF^(l+1)
      t := t+Δt_l

Start - Start integration on level 0

    l = 0, r_0 = 1
    AdvanceLevel(l)
```

- Recursion
- Restriction and flux correction
- Re-organization of hierarchical data

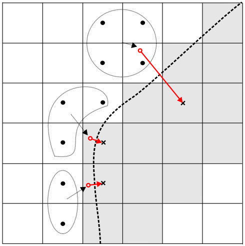
[Berger and Colella, 1988][Berger and Oliger, 1984]

Level-set method for boundary embedding



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

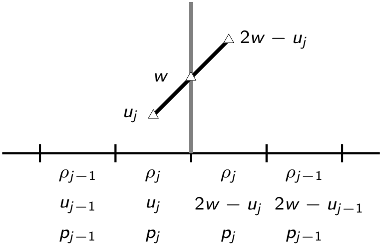
Level-set method for boundary embedding



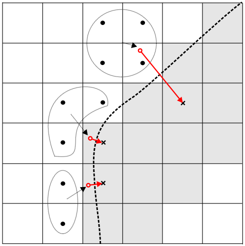
- Implicit boundary representation via distance function φ, normal $\mathbf{n} = \nabla\varphi / |\nabla\varphi|$
- Complex boundary moving with local velocity $\mathbf{w}$, treat interface as moving rigid wall
- Construction of values in embedded boundary cells by interpolation / extrapolation

Interpolate / constant value extrapolate values at

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



Level-set method for boundary embedding



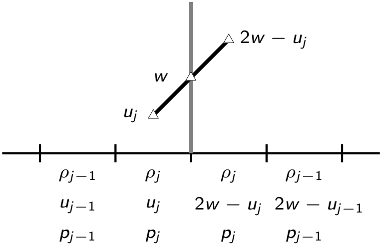
- Implicit boundary representation via distance function φ, normal $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$
- Complex boundary moving with local velocity $\mathbf{w}$, treat interface as moving rigid wall
- Construction of values in embedded boundary cells by interpolation / extrapolation

Interpolate / constant value extrapolate values at

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

Velocity in ghost cells

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

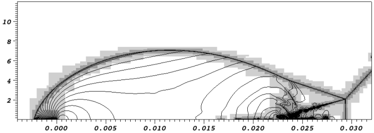


Verification: shock reflection

- Reflection of a Mach 2.38 shock in nitrogen at 43^o wedge
- 2nd order MUSCL scheme with Roe solver, 2nd order multidimensional wave propagation method

Verification: shock reflection

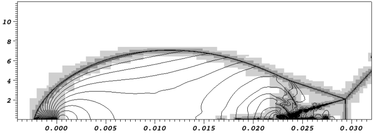
- Reflection of a Mach 2.38 shock in nitrogen at 43^o wedge
- 2nd order MUSCL scheme with Roe solver, 2nd order multidimensional wave propagation method



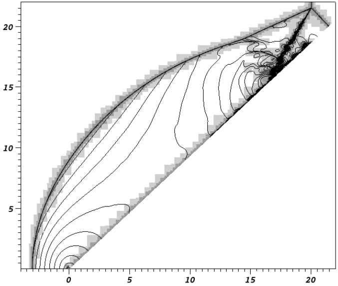
Cartesian base grid 360 $\times$ 160 cells on domain of 36 mm $\times$ 16 mm with up to 3 refinement levels with $r_l = 2, 4, 4$ and $\Delta x_{1,2} = 3.125\mu m$, 38 h CPU

Verification: shock reflection

- Reflection of a Mach 2.38 shock in nitrogen at 43^o wedge
- 2nd order MUSCL scheme with Roe solver, 2nd order multidimensional wave propagation method

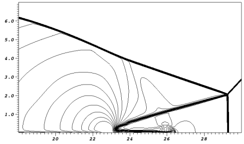


Cartesian base grid 360 $\times$ 160 cells on domain of $36\,\mathrm{mm} \times 16\,\mathrm{mm}$ with up to 3 refinement levels with $r_l = 2, 4, 4$ and $\Delta x_{1,2} = 3.125\mu m$, $38\,\mathrm{h}$ CPU

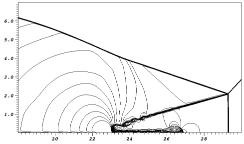


GFM base grid 390 $\times$ 330 cells on domain of $26\,\mathrm{mm} \times 22\,\mathrm{mm}$ with up to 3 refinement levels with $r_l = 2, 4, 4$ and $\Delta x_{e,1,2} = 2.849\mu m$, $200\,\mathrm{h}$ CPU

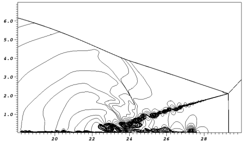
Verification: Shock reflection for Euler equations



$\Delta x = 25\,\mathrm{mm}$

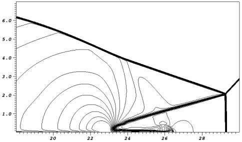


$\Delta x = 12.5\,\mathrm{mm}$

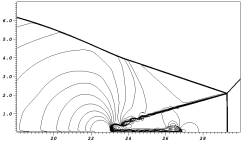


$\Delta x = 3.125\,\mathrm{mm}$

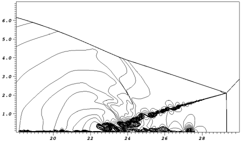
Verification: Shock reflection for Euler equations



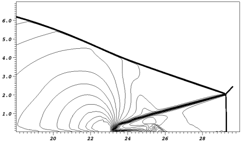
$\Delta x = 25\,\mathrm{mm}$



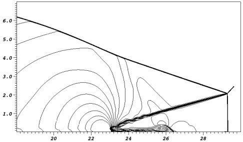
$\Delta x = 12.5\,\mathrm{mm}$



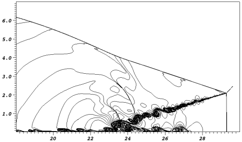
$\Delta x = 3.125\,\mathrm{mm}$



$\Delta x_e = 22.8\,\mathrm{mm}$

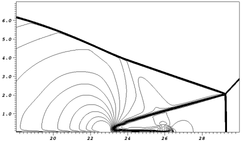


$\Delta x_e = 11.4\,\mathrm{mm}$

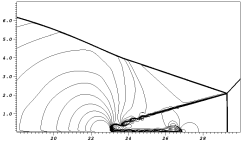


$\Delta x_e = 2.849\,\mathrm{mm}$

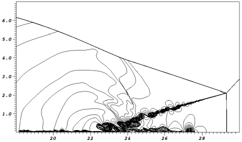
Verification: Shock reflection for Euler equations



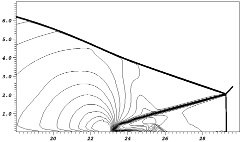
$\Delta x = 25\,\mathrm{mm}$



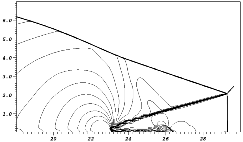
$\Delta x = 12.5\,\mathrm{mm}$



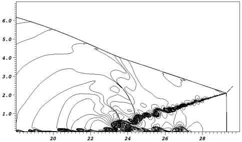
$\Delta x = 3.125\,\mathrm{mm}$



$\Delta x_e = 22.8\,\mathrm{mm}$

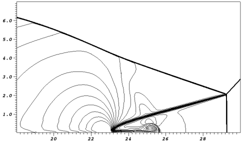


$\Delta x_e = 11.4\,\mathrm{mm}$

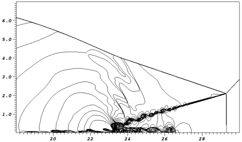


$\Delta x_e = 2.849\,\mathrm{mm}$

2nd order MUSCL scheme with Van Leer FVS, dimensional splitting



$\Delta x = 12.5\,\mathrm{mm}$



$\Delta x = 3.125\,\mathrm{mm}$

Parallelization strategies

Decomposition of the hierarchical data

- Distribution of each grid

Parallelization strategies

Decomposition of the hierarchical data

- Distribution of each grid
- Separate distribution of each level, cf. [Rendleman et al., 2000]
- Rigorous domain decomposition

Parallelization strategies

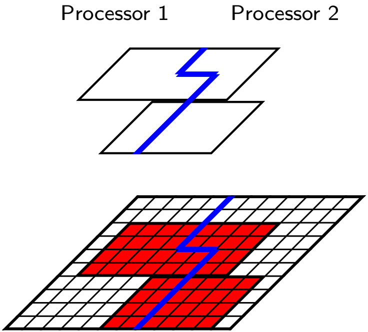
Decomposition of the hierarchical data

- Distribution of each grid
- Separate distribution of each level, cf. [Rendleman et al., 2000]
- Rigorous domain decomposition
  - Data of all levels resides on same node
  - Grid hierarchy defines unique "floor-plan"
  - Redistribution of data blocks during reorganization of hierarchical data
  - Synchronization when setting ghost cells

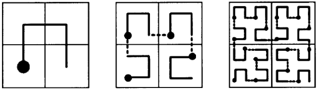
Parallelization strategies

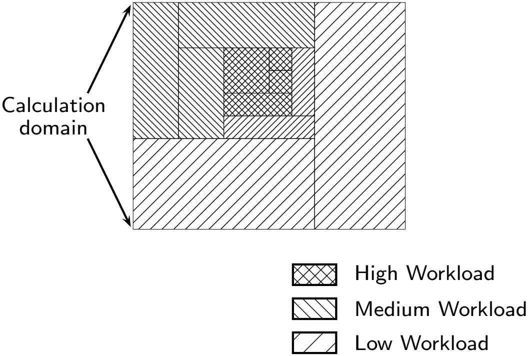
Decomposition of the hierarchical data

- Distribution of each grid
- Separate distribution of each level, cf. [Rendleman et al., 2000]
- Rigorous domain decomposition
  - Data of all levels resides on same node
  - Grid hierarchy defines unique "floor-plan"
  - Redistribution of data blocks during reorganization of hierarchical data
  - Synchronization when setting ghost cells

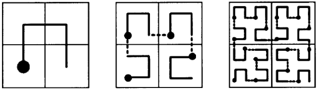


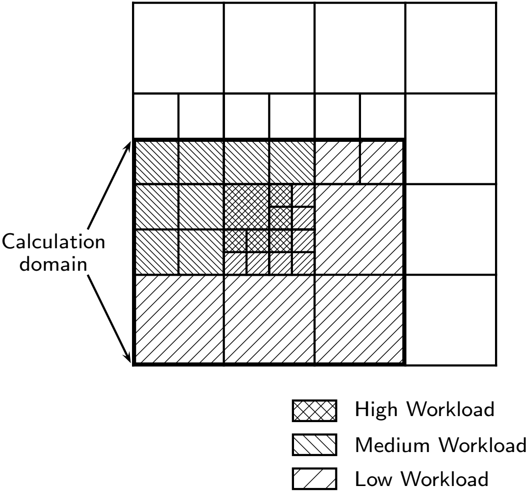
Space-filling curve algorithm



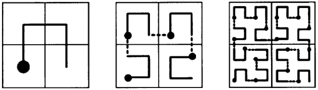


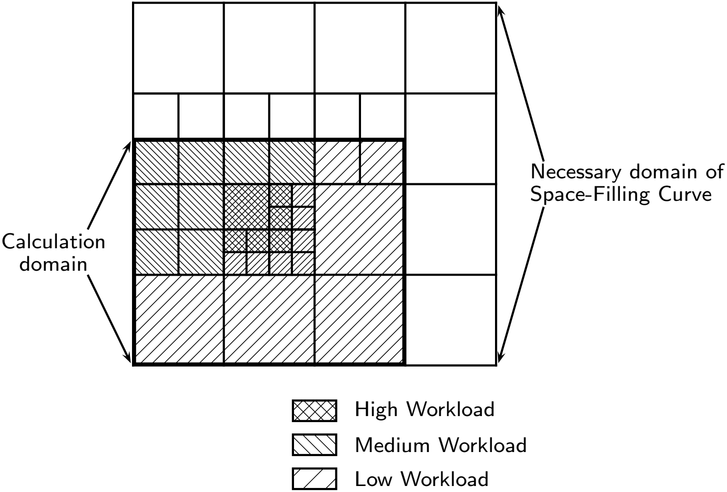
Space-filling curve algorithm



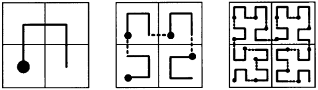


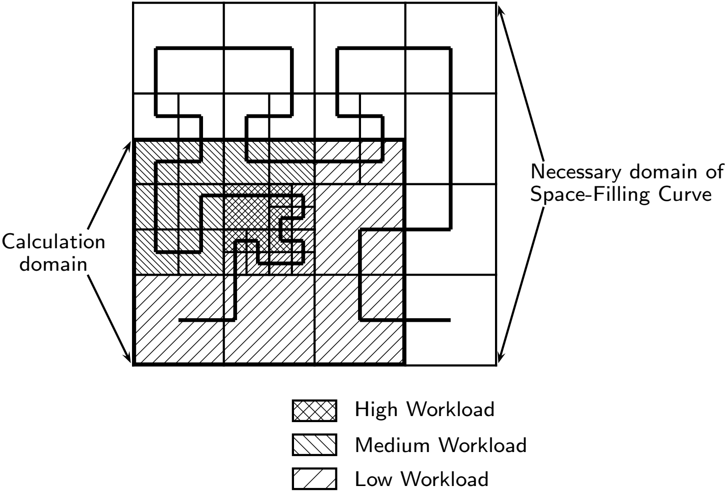
Space-filling curve algorithm



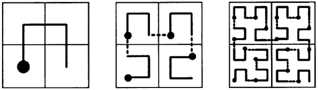


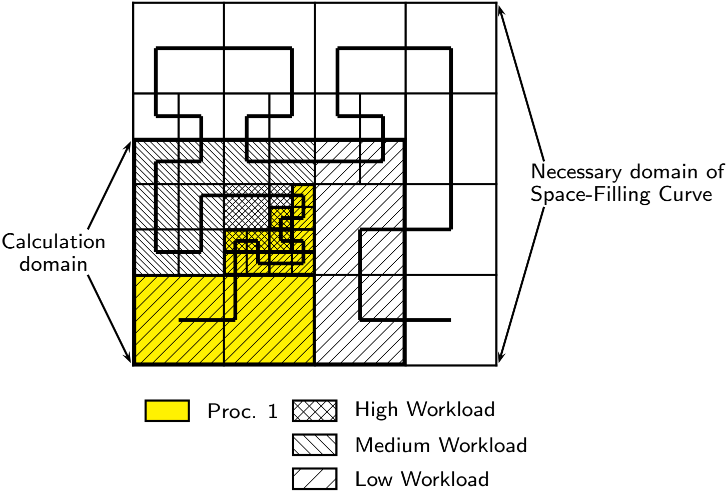
Space-filling curve algorithm



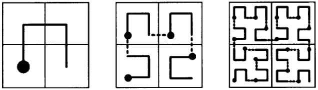


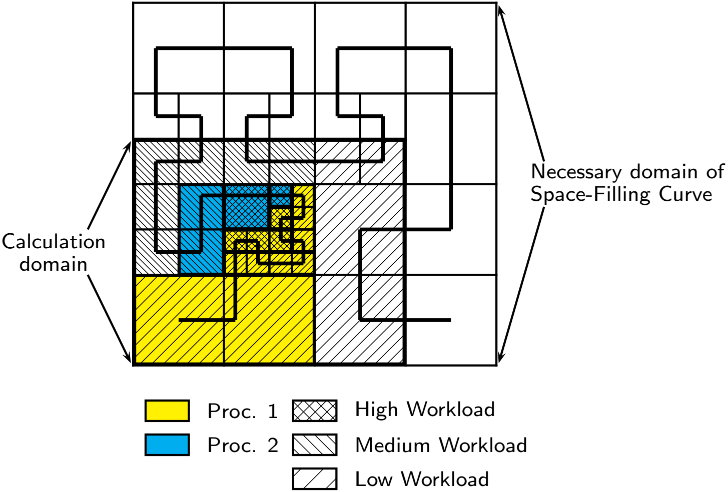
Space-filling curve algorithm



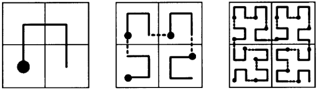


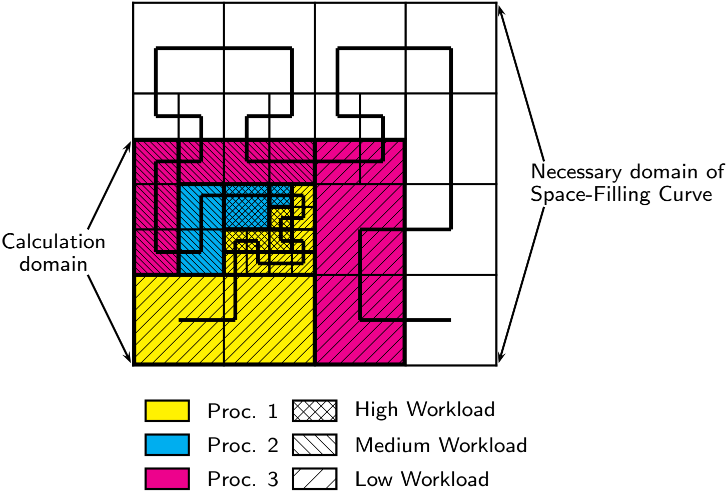
Space-filling curve algorithm





Space-filling curve algorithm





AMROC scalability tests

Basic test configuration

- Spherical blast wave, Euler equations, 3D wave propagation method
- AMR base grid 32^3 with $r_{1,2} = 2, 4$. 5 time steps on coarsest level
- Uniform grid $256^3 = 16.8 \cdot 10^6$ cells, 19 time steps
- Flux correction deactivated
- No volume I/O operations
- Tests run IBM BG/P (mode VN)

AMROC scalability tests

Basic test configuration

- Spherical blast wave, Euler equations, 3D wave propagation method
- AMR base grid 32^3 with $r_{1,2} = 2, 4$. 5 time steps on coarsest level
- Uniform grid $256^3 = 16.8 \cdot 10^6$ cells, 19 time steps
- Flux correction deactivated
- No volume I/O operations
- Tests run IBM BG/P (mode VN)

Weak scalability test

- Reproduction of configuration each 64 CPUs
- On 1024 CPUs: $128 \times 64 \times 64$ base grid, $> 33,500$ Grids, $\sim 61 \cdot 10^6$ cells, uniform $1024 \times 512 \times 512 = 268 \cdot 10^6$ cells

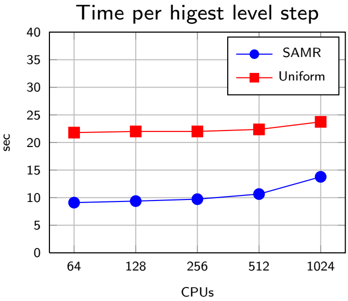
Level	Grids	Cells
0	606	32,768
1	575	135,312
2	910	3,639,040

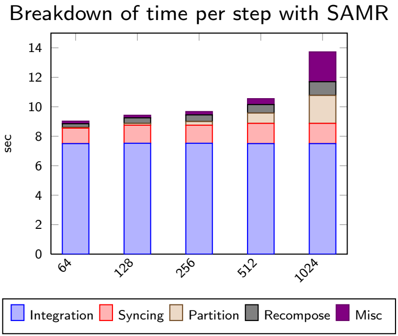
Strong scalability test

- $64 \times 32 \times 32$ base grid, uniform $512 \times 256 \times 256 = 33.6 \cdot 10^6$ cells

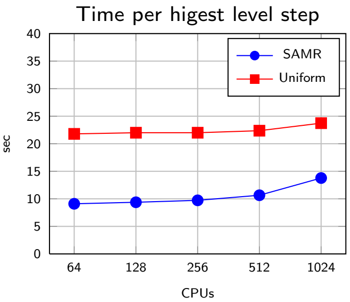
Level	Grids	Cells
0	1709	65,536
1	1735	271,048
2	2210	7,190,208

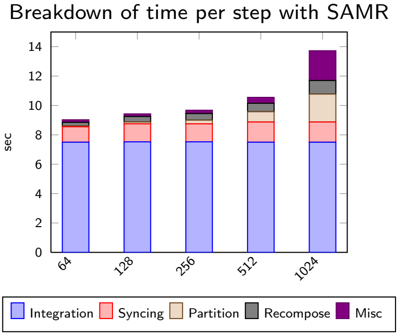
Weak scalability test





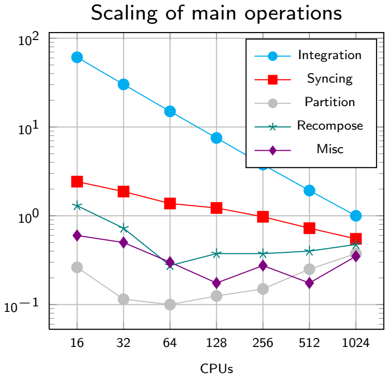
Weak scalability test

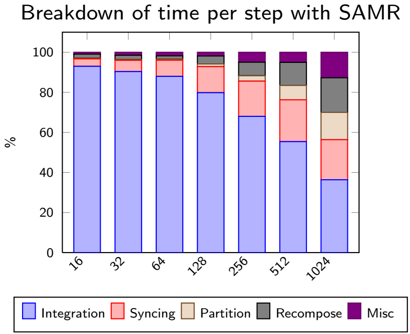




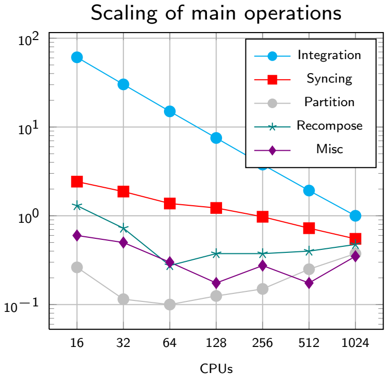
- Costs for Syncing basically constant
- Partitioning, Recompose, Misc increase
- 1024 required usage of -DUAL option due to usage of global lists data structures in Partition and Recompose

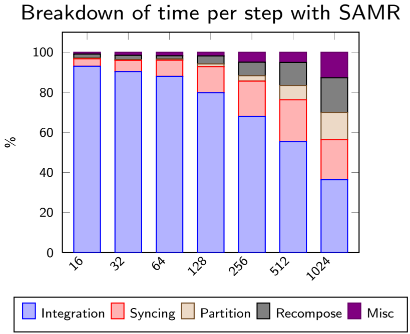
Strong scalability test





Strong scalability test





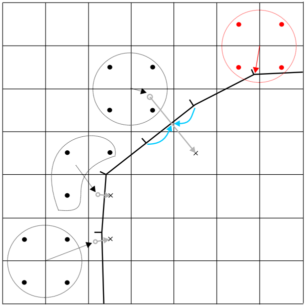
- SAMR visibly looses efficiency for > 512 CPU, or $15,000$ finite volume cells per CPU
- Perfect scaling of Integration, reasonable scaling of Syncing
- Strong scalability of Partition needs to be addressed (eliminate global lists)

Construction of coupling data

- Moving boundary/interface is treated as a moving contact discontinuity and represented by level set [Fedkiw, 2002][Arienti et al., 2003]
- Efficient construction of level set from triangulated surface data with closest-point-transform (CPT) algorithm [Mauch, 2003]

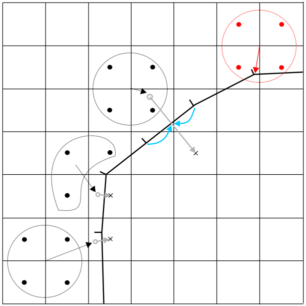
Construction of coupling data

- Moving boundary/interface is treated as a moving contact discontinuity and represented by level set [Fedkiw, 2002][Arienti et al., 2003]
- Efficient construction of level set from triangulated surface data with closest-point-transform (CPT) algorithm [Mauch, 2003]
- One-sided construction of mirrored ghost cell and new FEM nodal point values



Construction of coupling data

- Moving boundary/interface is treated as a moving contact discontinuity and represented by level set [Fedkiw, 2002][Arienti et al., 2003]
- Efficient construction of level set from triangulated surface data with closest-point-transform (CPT) algorithm [Mauch, 2003]
- One-sided construction of mirrored ghost cell and new FEM nodal point values
- FEM ansatz-function interpolation to obtain intermediate surface values



Coupling conditions on interface

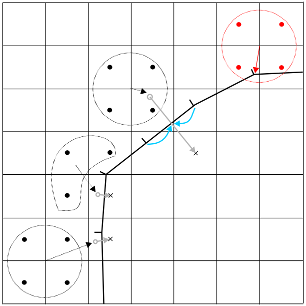
$$\left.\begin{array}{rcl} u_n^S & = & u_n^F \\ \sigma_{nn}^S & = & -p^F \\ \sigma_{nm}^S & = & 0 \end{array}\right|_{\mathcal{I}}$$

Construction of coupling data

- Moving boundary/interface is treated as a moving contact discontinuity and represented by level set [Fedkiw, 2002][Arienti et al., 2003]

- Efficient construction of level set from triangulated surface data with closest-point-transform (CPT) algorithm [Mauch, 2003]
- One-sided construction of mirrored ghost cell and new FEM nodal point values
- FEM ansatz-function interpolation to obtain intermediate surface values
- Explicit coupling possible if geometry and velocities are prescribed for the more compressible medium [Specht, 2000]

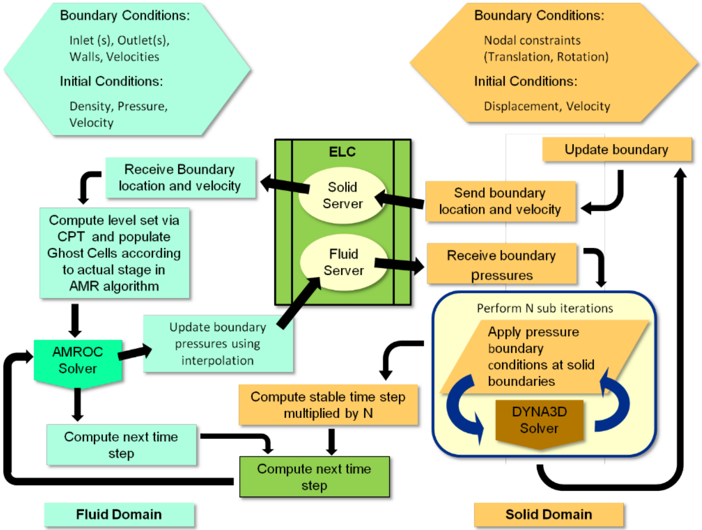
$$
\begin{array}{l}
u_n^F := u_n^S(t)|_{\mathcal{I}} \\
\texttt{UpdateFluid(}\,\Delta t\,\texttt{)} \\
\sigma_{nn}^S := -p^F(t+\Delta t)|_{\mathcal{I}} \\
\texttt{UpdateSolid(}\,\Delta t\,\texttt{)} \\
t := t + \Delta t
\end{array}
$$



Coupling conditions on interface

$$
\left.\begin{array}{rcl}
u_n^S & = & u_n^F \\
\sigma_{nn}^S & = & -p^F \\
\sigma_{nm}^S & = & 0
\end{array}\right|_{\mathcal{I}}
$$

Coupling elements



Usage of SAMR

- Eulerian SAMR + non-adaptive Lagrangian FEM scheme
- Exploit SAMR time step refinement for effective coupling to solid solver

Usage of SAMR

- Eulerian SAMR + non-adaptive Lagrangian FEM scheme
- Exploit SAMR time step refinement for effective coupling to solid solver
  - Lagrangian simulation is called only at level $l_{\mathrm{fsi}} \le l_{\mathrm{max}}$
  - SAMR refines solid boundary at least at level l_{fsi}
  - Additional levels can be used resolve geometric ambiguities
- Nevertheless: Inserting sub-steps accommodates for time step reduction from the solid solver within an SAMR cycle

Usage of SAMR

- Eulerian SAMR + non-adaptive Lagrangian FEM scheme
- Exploit SAMR time step refinement for effective coupling to solid solver
  - Lagrangian simulation is called only at level $l_{\rm fsi} \le l_{\max}$
  - SAMR refines solid boundary at least at level $l_{\rm fsi}$
  - Additional levels can be used resolve geometric ambiguities
- Nevertheless: Inserting sub-steps accommodates for time step reduction from the solid solver within an SAMR cycle
- Basic communication strategy:
  - Updated boundary info from solid solver must be received before regridding operation
  - Boundary data is sent to solid when highest level available

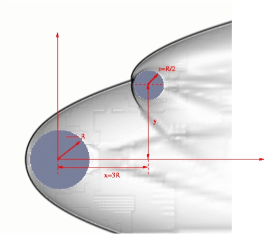
Proximal bodies in hypersonic flow

Flow modeled by Euler equations for a single polytropic gas with $p = (\gamma - 1)\,\rho e$

$$\partial_t \rho + \partial_{x_n}(\rho u_n) = 0 \;, \quad \partial_t(\rho u_k) + \partial_{x_n}(\rho u_k u_n + \delta_{kn} p) = 0 \;, \quad \partial_t(\rho E) + \partial_{x_n}(u_n(\rho E + p)) = 0$$

Numerical approximation with

- Finite volume flux-vector splitting scheme with MUSCL reconstruction, dimensional splitting



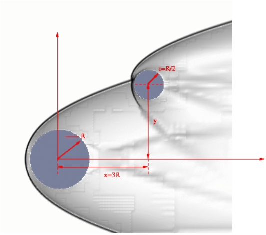
Proximal bodies in hypersonic flow

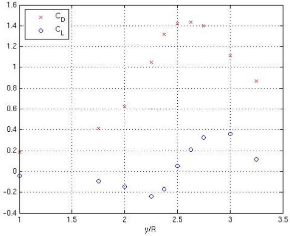
Flow modeled by Euler equations for a single polytropic gas with $p = (\gamma - 1)\,\rho e$

$$\partial_t \rho + \partial_{x_n}(\rho u_n) = 0 \,, \quad \partial_t(\rho u_k) + \partial_{x_n}(\rho u_k u_n + \delta_{kn} p) = 0 \,, \quad \partial_t(\rho E) + \partial_{x_n}(u_n(\rho E + p)) = 0$$

Numerical approximation with

- Finite volume flux-vector splitting scheme with MUSCL reconstruction, dimensional splitting
- Spherical bodies, force computation with overlaid lattitude-longitude mesh to obtain drag and lift coefficients $C_{D,L} = \dfrac{2F_{D,L}}{\rho v^2 \pi r^2}$
- inflow $M = 10$, C_D and C_L on secondary sphere, lateral position varied, no motion





Verification and validation

Static force measurements, $M = 10$:

S. Laurence, RD, H. Hornung. *J. Fluid Mech.* 590:209-237, 2007.

- Refinement study: $40 \times 40 \times 32$ base grid , up to without AMR up to $\sim 209.7 \cdot 10^6$ cells, largest run $\sim 35,000\,\mathrm{h}$ CPU

$l_{\max}$	C_D	ΔC_D	C_L	ΔC_L
1	1.264		-0.176	
2	1.442	0.178	-0.019	0.157
3	1.423	-0.019	0.052	0.071
4	1.408	-0.015	0.087	0.035

Verification and validation

Static force measurements, $M = 10$:

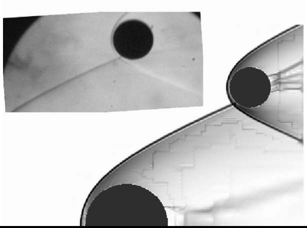
S. Laurence, RD, H. Hornung. *J. Fluid Mech.* 590:209-237, 2007.

- Refinement study: $40 \times 40 \times 32$ base grid , up to without AMR up to $\sim 209.7 \cdot 10^6$ cells, largest run $\sim 35,000\,\mathrm{h}$ CPU

$l_{\max}$	C_D	ΔC_D	C_L	ΔC_L
1	1.264		-0.176	
2	1.442	0.178	-0.019	0.157
3	1.423	-0.019	0.052	0.071
4	1.408	-0.015	0.087	0.035

- Comparison with experimental results: 3 additional levels, $\sim 2000\,\mathrm{h}$ CPU

	Experimental	Computational
C_D	1.11 ± 0.08	1.01
C_L	0.29 ± 0.05	0.28



Verification and validation

Static force measurements, $M = 10$:

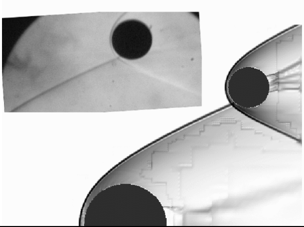
S. Laurence, RD, H. Hornung. *J. Fluid Mech.* 590:209-237, 2007.

- Refinement study: $40 \times 40 \times 32$ base grid , up to without AMR up to $\sim 209.7 \cdot 10^6$ cells, largest run $\sim 35,000\,\mathrm{h}$ CPU

$l_{\max}$	C_D	ΔC_D	C_L	ΔC_L
1	1.264		-0.176	
2	1.442	0.178	-0.019	0.157
3	1.423	-0.019	0.052	0.071
4	1.408	-0.015	0.087	0.035

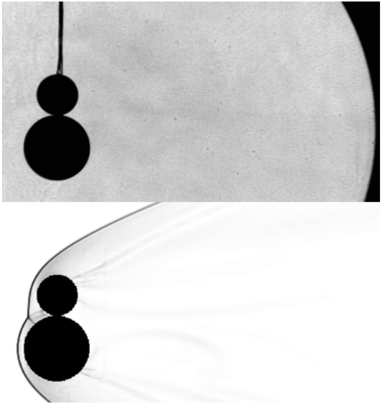
- Comparison with experimental results: 3 additional levels, $\sim 2000\,\mathrm{h}$ CPU

	Experimental	Computational
C_D	1.11 ± 0.08	1.01
C_L	0.29 ± 0.05	0.28



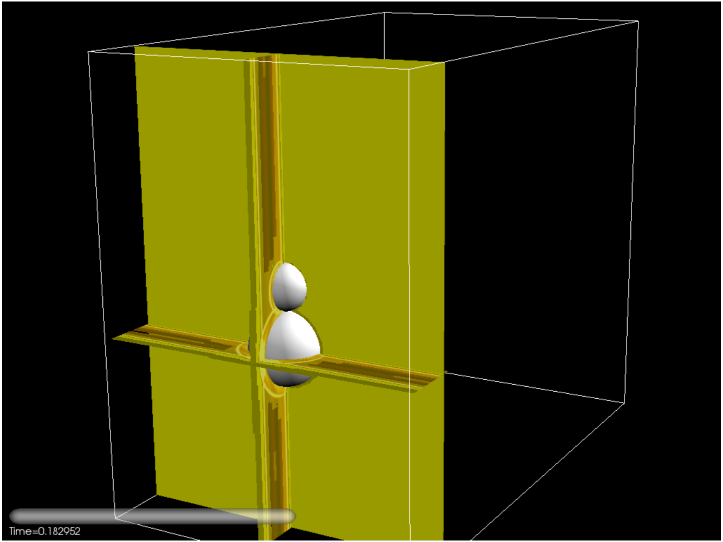
Dynamic motion, $M = 4$:

- Base grid $150 \times 125 \times 90$, two additional levels with $r_{1,2} = 2$
- 24,704 time steps, $36,808\,\mathrm{h}$ CPU on 256 cores IBM BG/P



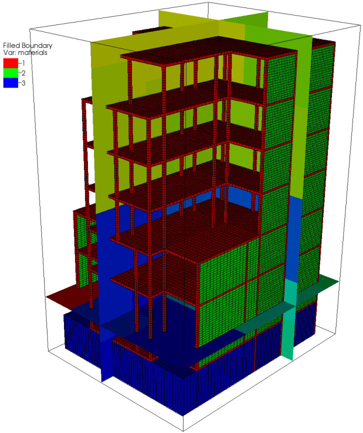
S. Laurence, RD. *J. Fluid Mech.* 676: 396-431, 2011.

Schlieren graphics on refinement regions



Blast explosion in a multistory building

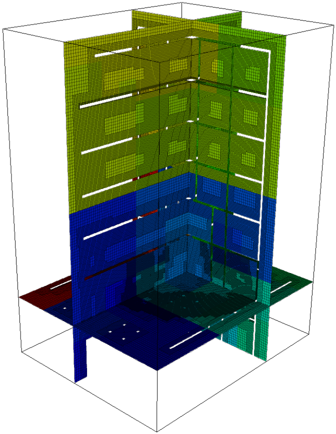
- $20\,\mathrm{m} \times 40\,\mathrm{m} \times 25\,\mathrm{m}$ seven-story building similar to [Luccioni et al., 2004]
- Spherical energy deposition $\equiv 400\,\mathrm{kg}$ TNT, $r = 0.5\,\mathrm{m}$ in lobby of building
- SAMR: $80 \times 120 \times 90$ base level, three additional levels $r_{1,2} = 2$, $l_{\mathrm{fsi}} = 1$, $k = 1$
- Simulation with ground: $1,070$ coupled time steps, $830\,\mathrm{h}$ CPU ($\sim 25.9\,\mathrm{h}$ wall time) on 31+1 cores
- $\sim 8,000,000$ cells instead of $55,296,000$ (uniform)
- $69,709$ hexahedral elements and with material parameters



	ρ_s [kg/m^3]	σ_0 [MPa]	E_T [GPa]	β	K [GPa]	G [GPa]	$\bar{\epsilon}^p$	p_f [MPa]
Columns	2010	50	11.2	1.0	21.72	4.67	0.02	-30
Walls	2010	25	11.2	1.0	6.22	4.67	0.01	-15

Blast explosion in a multistory building

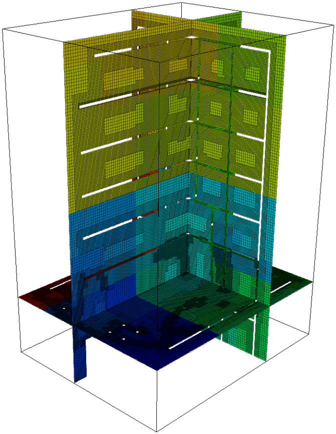
- $20\,\mathrm{m} \times 40\,\mathrm{m} \times 25\,\mathrm{m}$ seven-story building similar to [Luccioni et al., 2004]
- Spherical energy deposition $\equiv 400\,\mathrm{kg}$ TNT, $r = 0.5\,\mathrm{m}$ in lobby of building
- SAMR: $80 \times 120 \times 90$ base level, three additional levels $r_{1,2} = 2$, $l_{\mathrm{fsi}} = 1$, $k = 1$
- Simulation with ground: $1,070$ coupled time steps, $830\,\mathrm{h}$ CPU ($\sim 25.9\,\mathrm{h}$ wall time) on 31+1 cores
- $\sim 8,000,000$ cells instead of $55,296,000$ (uniform)
- $69,709$ hexahedral elements and with material parameters



	ρ_s [kg/m^3]	σ_0 [MPa]	E_T [GPa]	β	K [GPa]	G [GPa]	$\bar{\epsilon}^p$	p_f [MPa]
Columns	2010	50	11.2	1.0	21.72	4.67	0.02	-30
Walls	2010	25	11.2	1.0	6.22	4.67	0.01	-15

Blast explosion in a multistory building

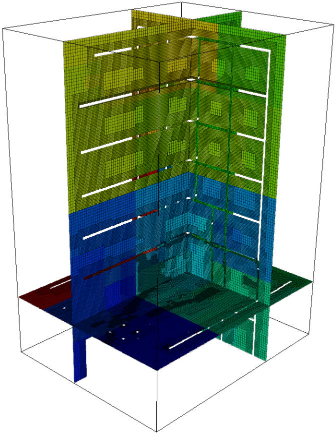
- $20\,\mathrm{m} \times 40\,\mathrm{m} \times 25\,\mathrm{m}$ seven-story building similar to [Luccioni et al., 2004]
- Spherical energy deposition $\equiv 400\,\mathrm{kg}$ TNT, $r = 0.5\,\mathrm{m}$ in lobby of building
- SAMR: $80 \times 120 \times 90$ base level, three additional levels $r_{1,2} = 2$, $l_{\mathrm{fsi}} = 1$, $k = 1$
- Simulation with ground: $1,070$ coupled time steps, $830\,\mathrm{h}$ CPU ($\sim 25.9\,\mathrm{h}$ wall time) on 31+1 cores
- $\sim 8,000,000$ cells instead of $55,296,000$ (uniform)
- $69,709$ hexahedral elements and with material parameters



	ρ_s [kg/m^3]	σ_0 [MPa]	E_T [GPa]	β	K [GPa]	G [GPa]	$\bar{\epsilon}^p$	p_f [MPa]
Columns	2010	50	11.2	1.0	21.72	4.67	0.02	-30
Walls	2010	25	11.2	1.0	6.22	4.67	0.01	-15

Blast explosion in a multistory building

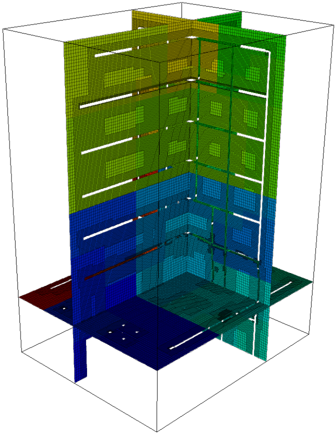
- $20\,\mathrm{m} \times 40\,\mathrm{m} \times 25\,\mathrm{m}$ seven-story building similar to [Luccioni et al., 2004]
- Spherical energy deposition $\equiv 400\,\mathrm{kg}$ TNT, $r = 0.5\,\mathrm{m}$ in lobby of building
- SAMR: $80 \times 120 \times 90$ base level, three additional levels $r_{1,2} = 2$, $l_{\mathrm{fsi}} = 1$, $k = 1$
- Simulation with ground: $1,070$ coupled time steps, $830\,\mathrm{h}$ CPU ($\sim 25.9\,\mathrm{h}$ wall time) on 31+1 cores
- $\sim 8,000,000$ cells instead of $55,296,000$ (uniform)
- $69,709$ hexahedral elements and with material parameters



	ρ_s [kg/m^3]	σ_0 [MPa]	E_T [GPa]	β	K [GPa]	G [GPa]	$\bar{\epsilon}^p$	p_f [MPa]
Columns	2010	50	11.2	1.0	21.72	4.67	0.02	-30
Walls	2010	25	11.2	1.0	6.22	4.67	0.01	-15

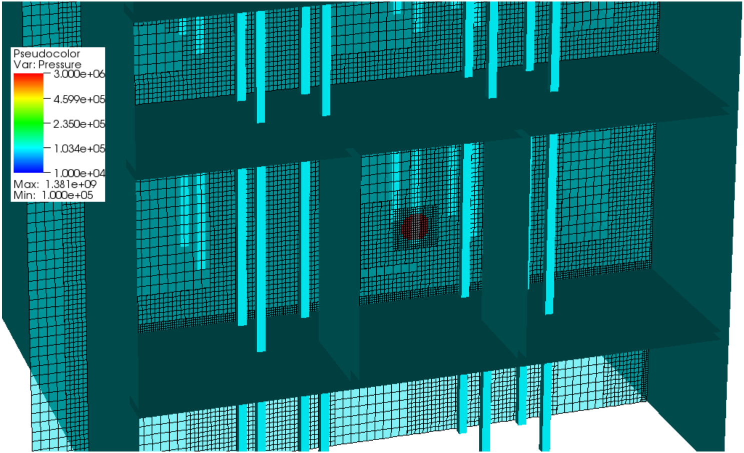
Blast explosion in a multistory building

- $20\,\mathrm{m} \times 40\,\mathrm{m} \times 25\,\mathrm{m}$ seven-story building similar to [Luccioni et al., 2004]
- Spherical energy deposition $\equiv 400\,\mathrm{kg}$ TNT, $r = 0.5\,\mathrm{m}$ in lobby of building
- SAMR: $80 \times 120 \times 90$ base level, three additional levels $r_{1,2} = 2$, $l_{\mathrm{fsi}} = 1$, $k = 1$
- Simulation with ground: $1,070$ coupled time steps, $830\,\mathrm{h}$ CPU ($\sim 25.9\,\mathrm{h}$ wall time) on 31+1 cores
- $\sim 8,000,000$ cells instead of $55,296,000$ (uniform)
- $69,709$ hexahedral elements and with material parameters



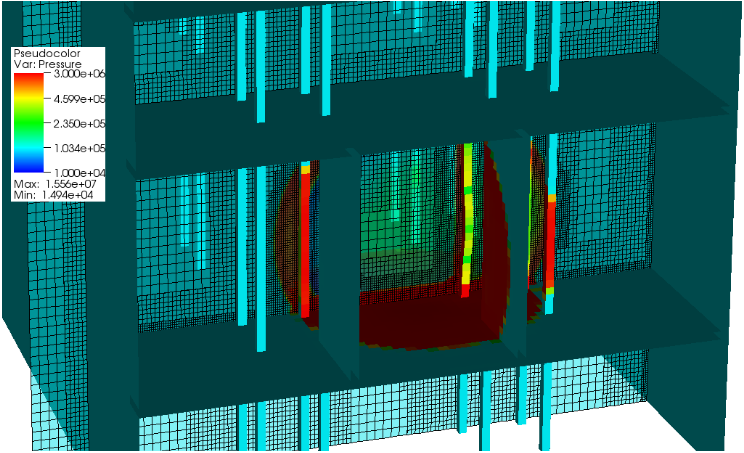
	ρ_s [kg/m^3]	σ_0 [MPa]	E_T [GPa]	β	K [GPa]	G [GPa]	$\bar{\epsilon}^p$	p_f [MPa]
Columns	2010	50	11.2	1.0	21.72	4.67	0.02	-30
Walls	2010	25	11.2	1.0	6.22	4.67	0.01	-15

Blast explosion in a multistory building – II

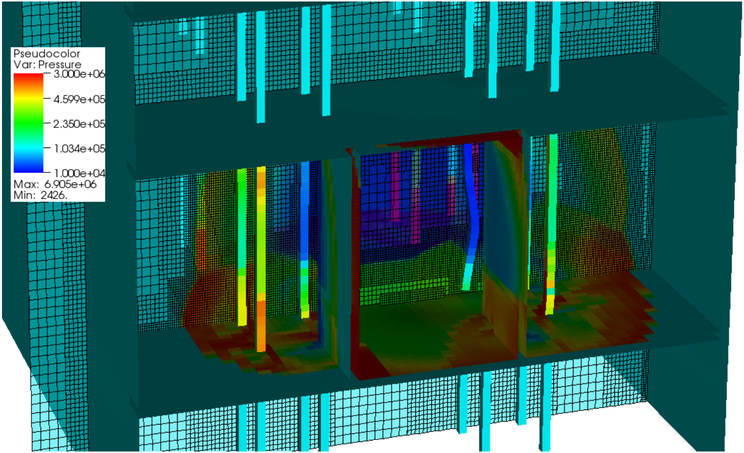


$t = 0$

Blast explosion in a multistory building – II

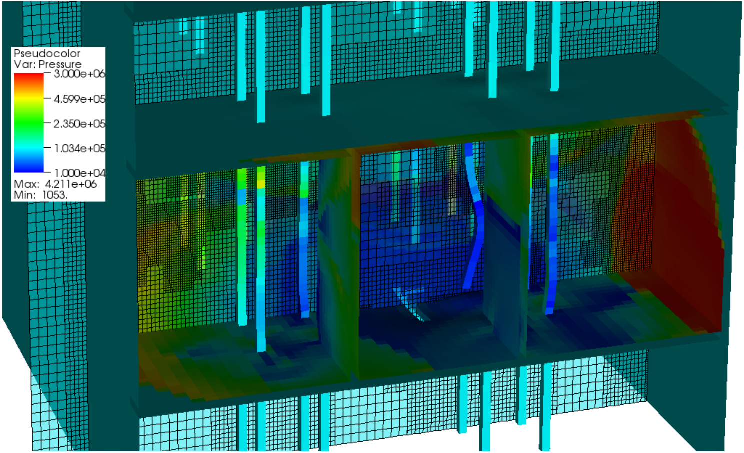


Blast explosion in a multistory building – II

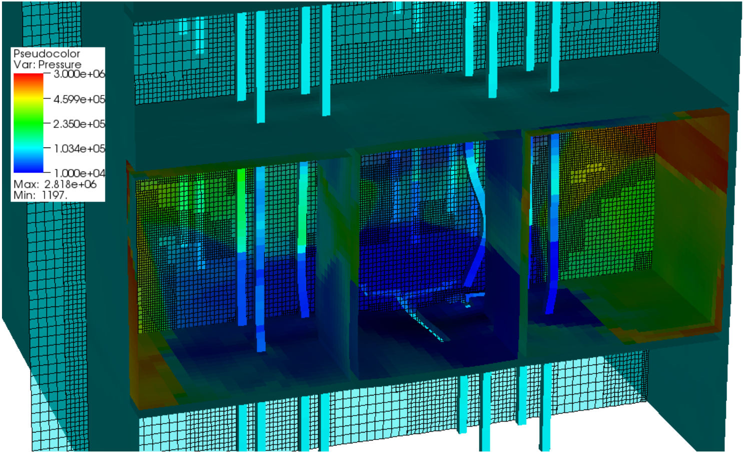


$t = 6.1\,\mathrm{ms}$

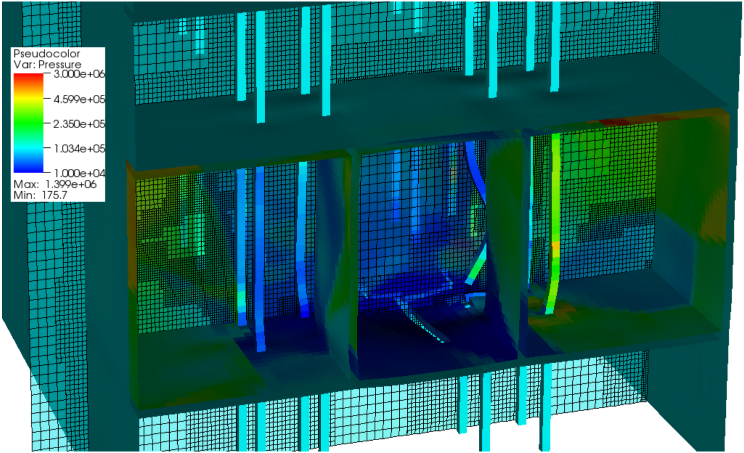
Blast explosion in a multistory building – II



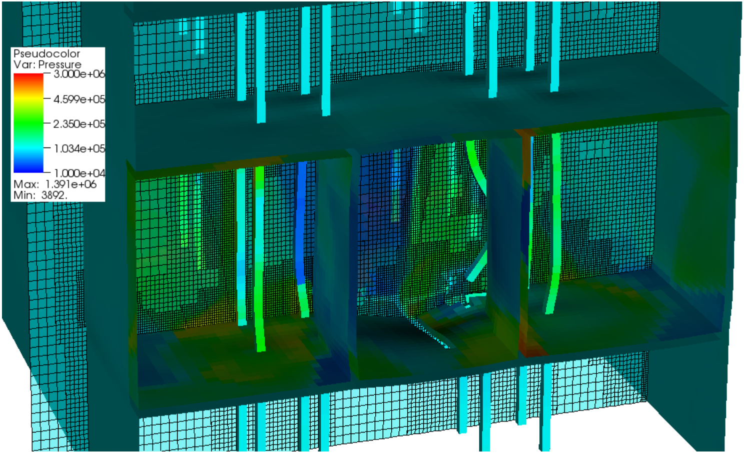
Blast explosion in a multistory building – II



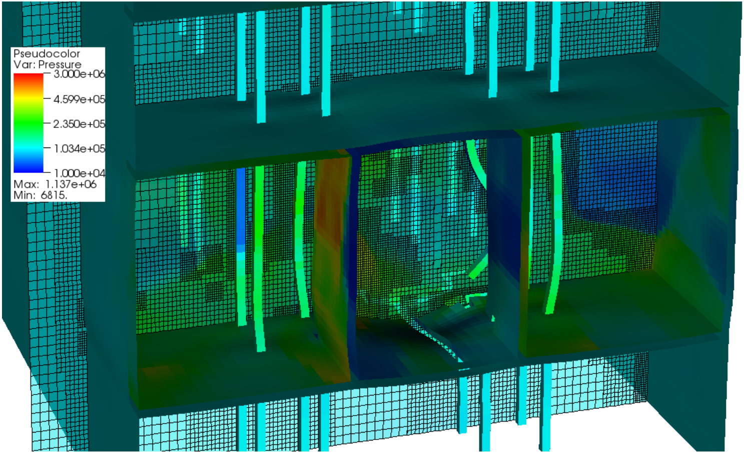
Blast explosion in a multistory building – II



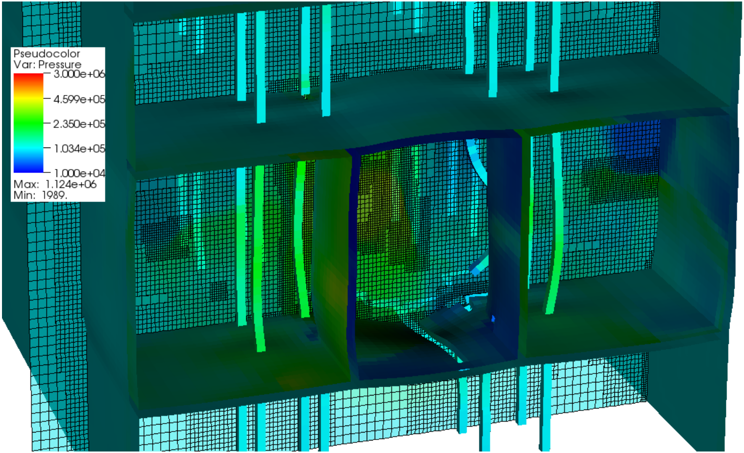
Blast explosion in a multistory building – II



Blast explosion in a multistory building – II

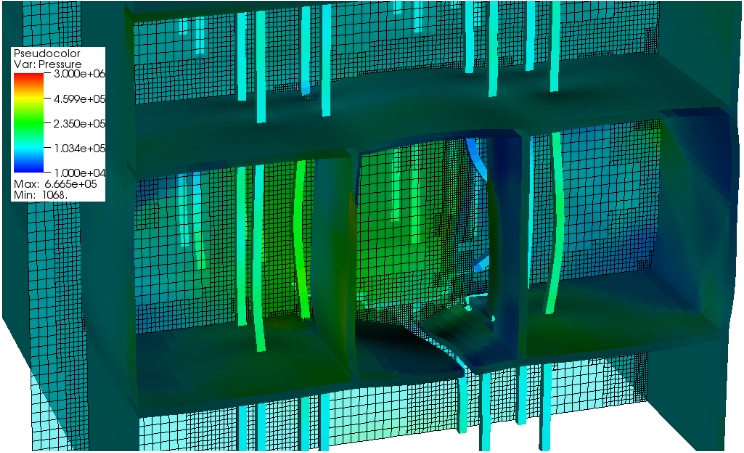


Blast explosion in a multistory building – II

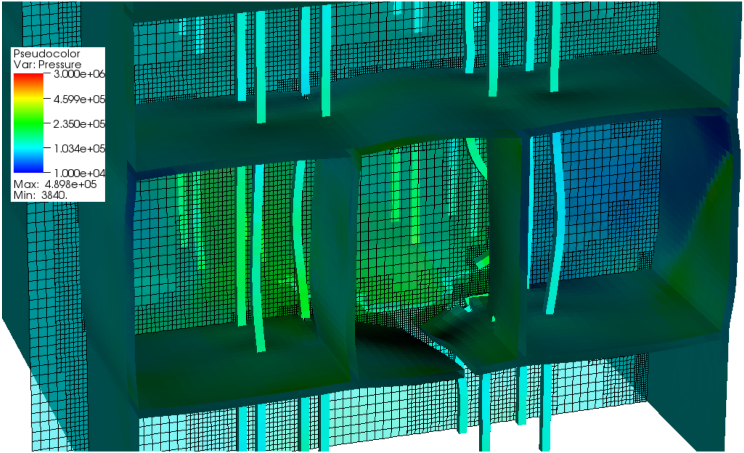


$t = 29.2\,\mathrm{ms}$

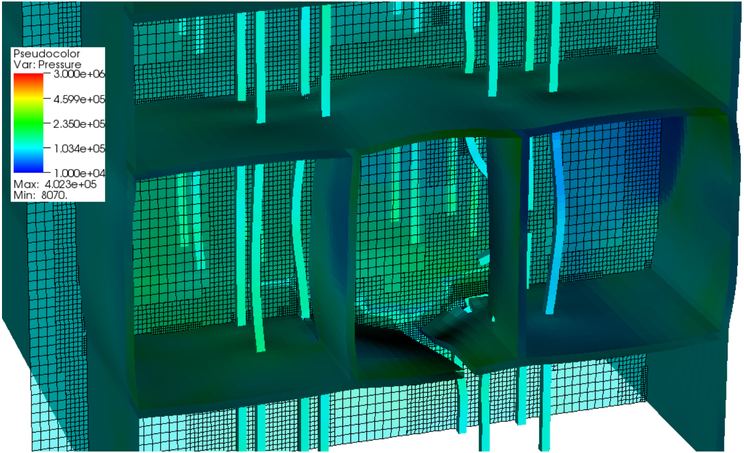
Blast explosion in a multistory building – II



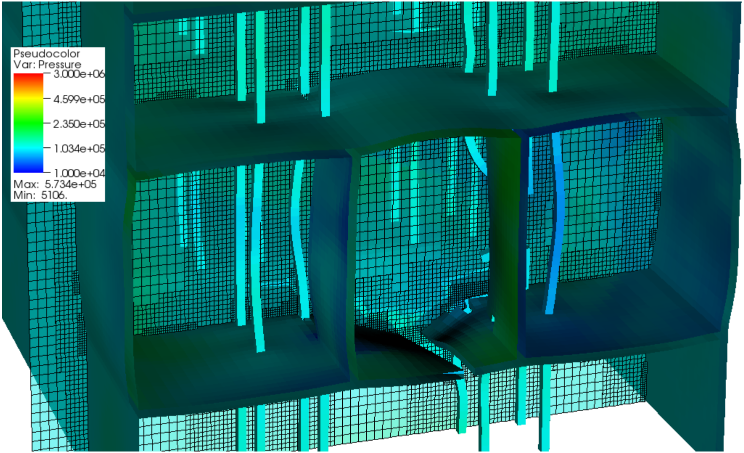
Blast explosion in a multistory building – II



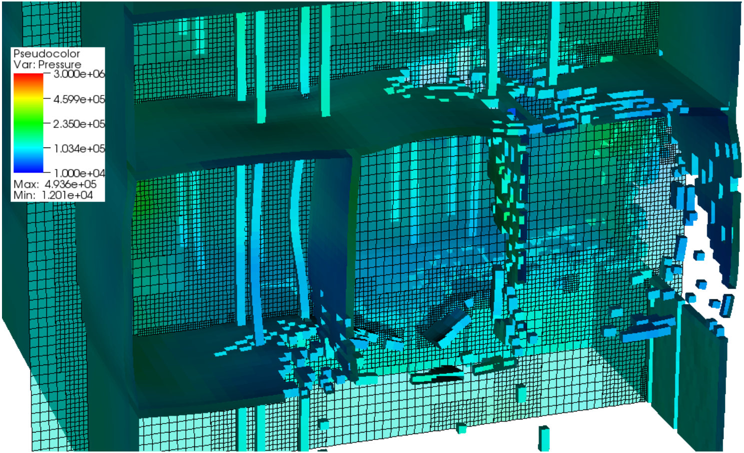
Blast explosion in a multistory building – II



Blast explosion in a multistory building – II



Blast explosion in a multistory building – II



$t = 48.7\,\mathrm{ms}$

AMROC

- Implements all described algorithms plus scalar multigrid methods
- Many shock-capturing methods (MUSCL, (hybrid) WENO, etc.) implemented for complex flux functions.

AMROC

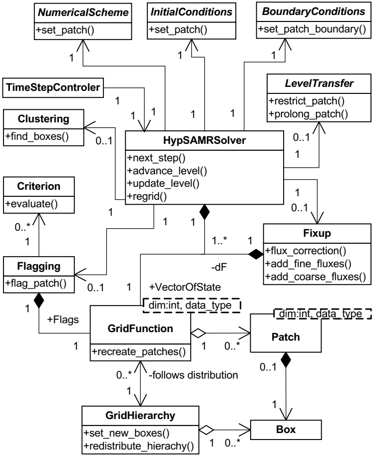
- Implements all described algorithms plus scalar multigrid methods
- Many shock-capturing methods (MUSCL, (hybrid) WENO, etc.) implemented for complex flux functions.
- Drives Virtual Test Facility (VTF) FSI software.
- Targets strongly driven problems (shocks, blast, detonations)
- Geometry embedding via ghost fluid techniques and level set functions. Distance computation with CPT algorithm [Mauch, 2000].

AMROC

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

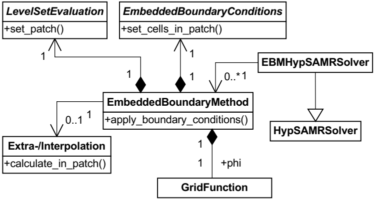
UML design of AMROC

- Classical framework approach with generic main program in C++
- Customization / modification in Problem.h include file by derivation from base classes and redefining virtual interface functions
- Predefined, scheme-specific classes (F77 interfaces or C++) provided for standard simulations
- Expert usage (algorithm modification, advanced output, etc.) in C++



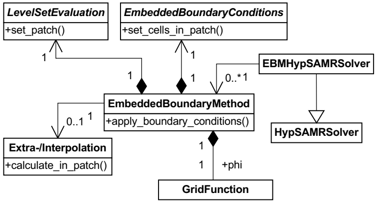
Embedded boundary method / FSI coupling

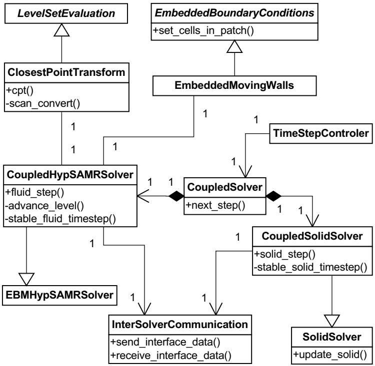
- Multiple independent EmbeddedBoundaryMethod objects possible
- Specialization of GFM boundary conditions, level set description in scheme-specific F77 interface classes



Embedded boundary method / FSI coupling

- Multiple independent EmbeddedBoundaryMethod objects possible
- Specialization of GFM boundary conditions, level set description in scheme-specific F77 interface classes

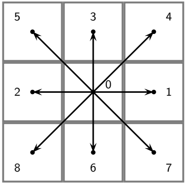




- Coupling algorithm implemented in further derived HypSAMRSolver class
- Level set evaluation always with CPT algorithm
- Parallel communication through efficient non-blocking communication module
- Time step selection for both solvers through CoupledSolver class

Lattice Boltzmann method

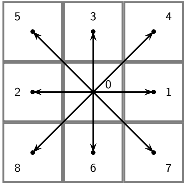
Boltzmann equation: $\partial_t f + \mathbf{u} \cdot \nabla f = \omega(f^{eq} - f)$
Two-dimensional LBM for weakly compressible flows
Formulated on FV grids! ($\rightarrow$ boundary conditions!)



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

Lattice Boltzmann method

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

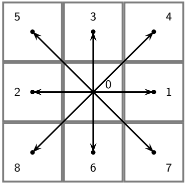


$$\rho(\mathbf{x},t) = \sum_{\alpha=0}^{8} f_\alpha(\mathbf{x},t), \quad \rho(\mathbf{x},t) u_i(\mathbf{x},t) = \sum_{\alpha=0}^{8} \mathbf{e}_{\alpha i} f_\alpha(\mathbf{x},t)$$

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

Lattice Boltzmann method

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



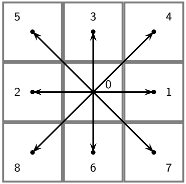
$$\rho(\mathbf{x},t) = \sum_{\alpha=0}^{8} f_\alpha(\mathbf{x},t), \quad \rho(\mathbf{x},t) u_i(\mathbf{x},t) = \sum_{\alpha=0}^{8} \mathbf{e}_{\alpha i} f_\alpha(\mathbf{x},t)$$

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

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

Lattice Boltzmann method

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



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

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

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

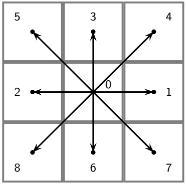
with equilibrium function

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

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

Lattice Boltzmann method

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



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

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

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

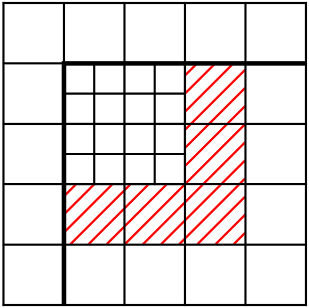
with equilibrium function

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

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

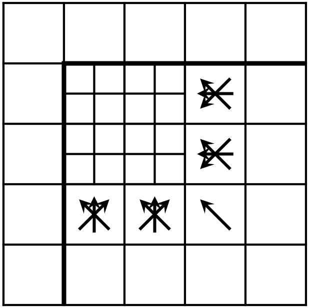
Adaptive LBM

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



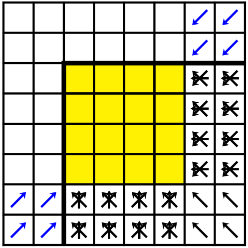
Adaptive LBM

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

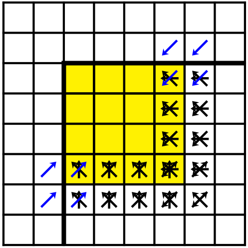
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.



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

Adaptive LBM

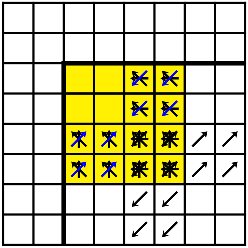
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



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

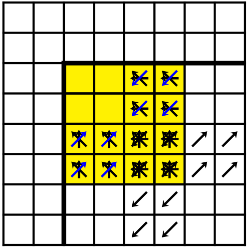
Adaptive LBM

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.

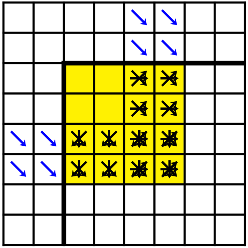


Adaptive LBM

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



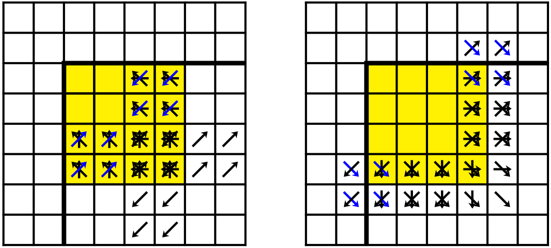
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



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

Adaptive LBM

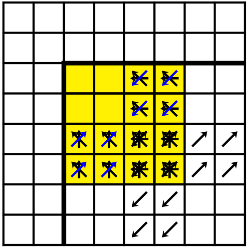
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



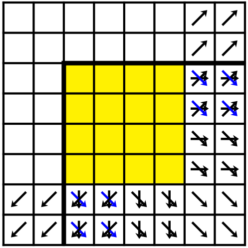
$\tilde{f}_{\alpha,in}^{f,n+1/2}$ $\tilde{f}_{\alpha,out}^{f,n}$

Adaptive LBM

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



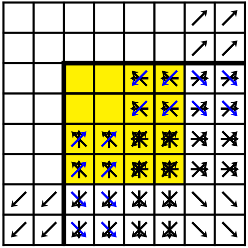
$\tilde{f}_{\alpha,in}^{f,n+1/2}$



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

Adaptive LBM

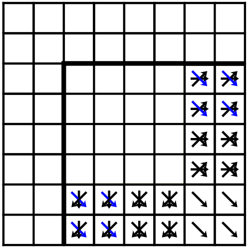
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



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

Adaptive LBM

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

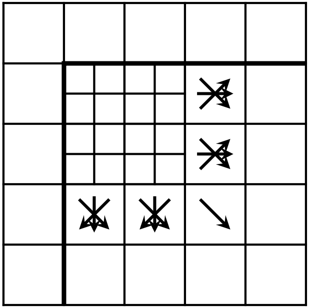


$\tilde{f}_{\alpha,out}^{f,n+1/2}$, $\tilde{f}_{\alpha,out}^{f,n}$

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

Adaptive LBM

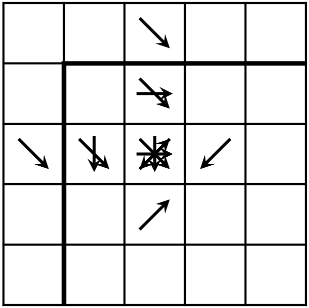
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



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

Adaptive LBM

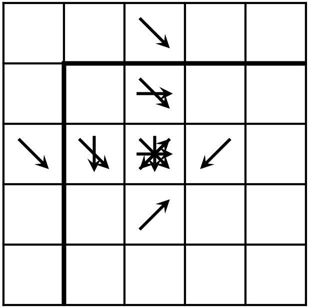
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



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

Adaptive LBM

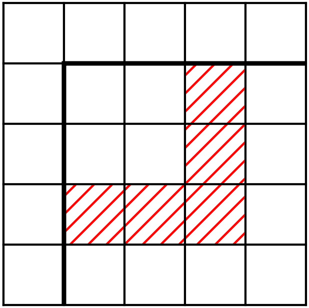
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



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

Adaptive LBM

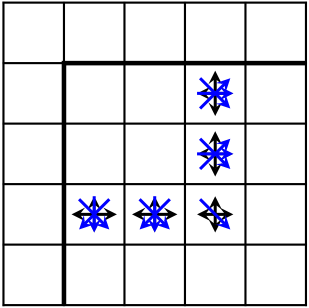
1. Complete update on coarse grid: $f_\alpha^{C,n+1} := \mathcal{CT}(f_\alpha^{C,n})$
2. Interpolate $f_{\alpha,in}^{C,n}$ onto $f_{\alpha,in}^{f,n}$ to fill fine halos. Set physical boundary conditions.
3. $\tilde{f}_\alpha^{f,n} := \mathcal{T}(f_\alpha^{f,n})$ on whole fine mesh. $f_\alpha^{f,n+1/2}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



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

Adaptive LBM

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

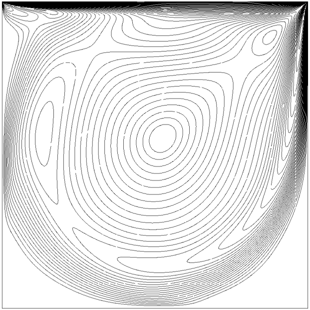


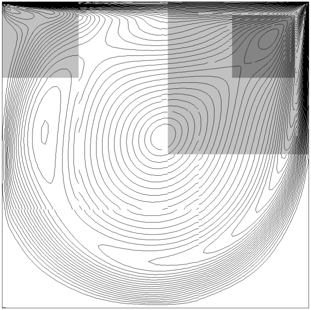
5. Average $\tilde{f}_{\alpha,out}^{f,n+1/2}$ (inner halo layer), $\tilde{f}_{\alpha,out}^{f,n}$ (outer halo layer) to obtain $\tilde{f}_{\alpha,out}^{C,n}$.
6. Revert transport into halos: $\bar{f}_{\alpha,out}^{C,n} := \mathcal{T}^{-1}(\tilde{f}_{\alpha,out}^{C,n})$
7. Parallel synchronization of $f_\alpha^{C,n},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

Algorithm equivalent to [Chen et al., 2006].

Verification - driven cavity

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

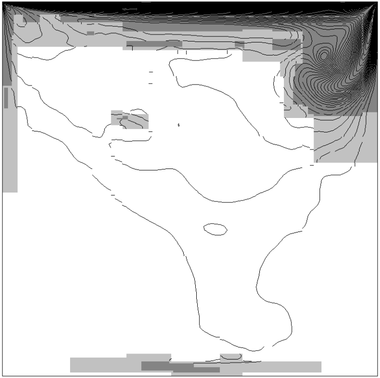


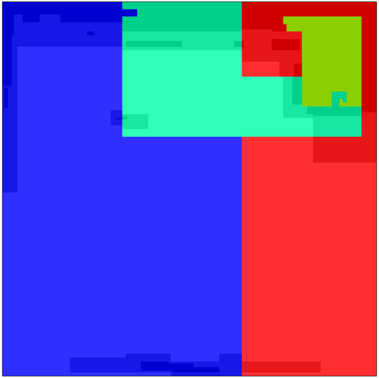


Isolines of density. Left: reference, right on refinement at t_e.

Driven cavity - dynamic refinement

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

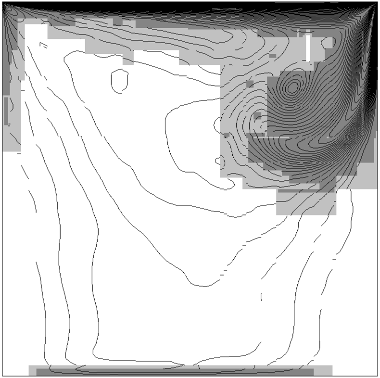




Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

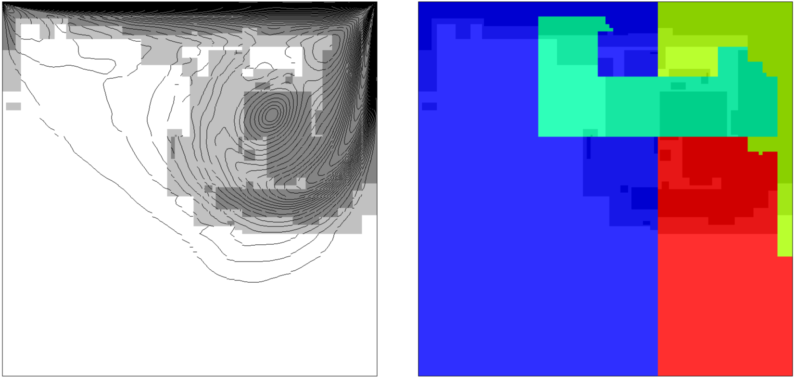
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



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

Driven cavity - dynamic refinement

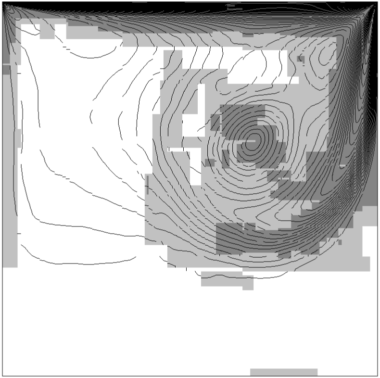
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



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

Driven cavity - dynamic refinement

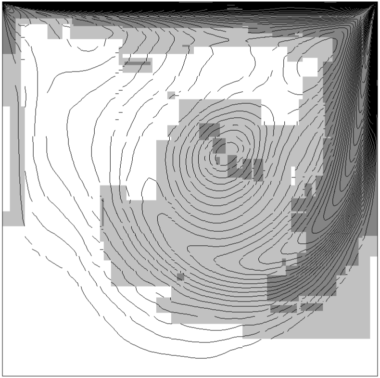
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution

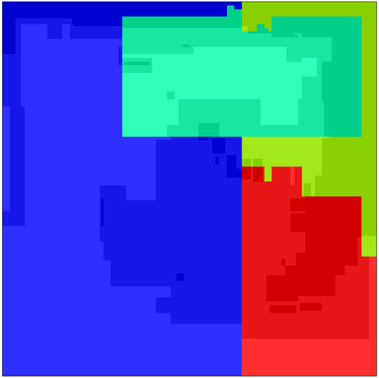


Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

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

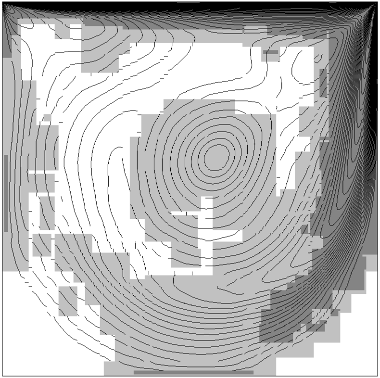


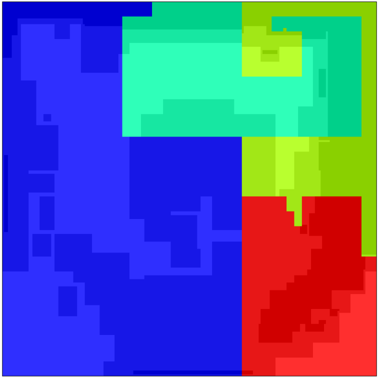


Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

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

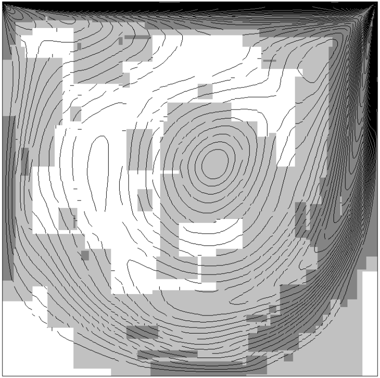


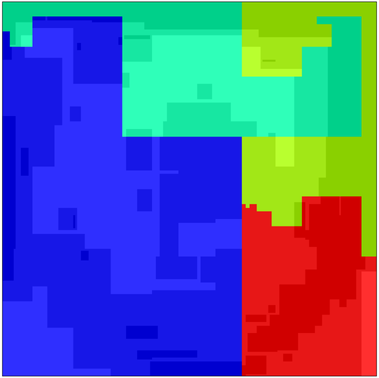


Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

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

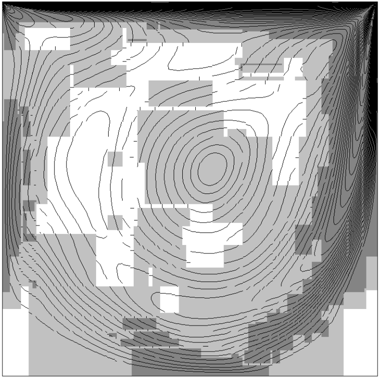




Isolines of density on refinement (left), distribution to 4 processors (right).

Driven cavity - dynamic refinement

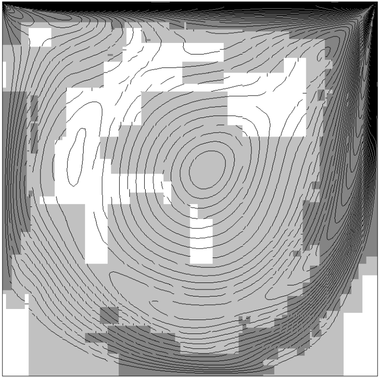
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



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

Driven cavity - dynamic refinement

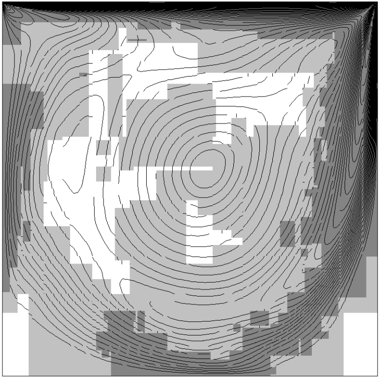
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



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

Driven cavity - dynamic refinement

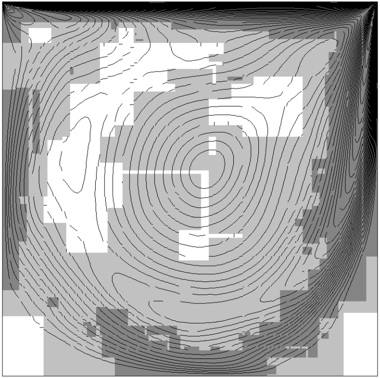
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



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

Driven cavity - dynamic refinement

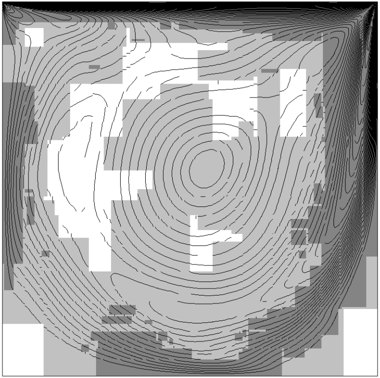
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



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

Driven cavity - dynamic refinement

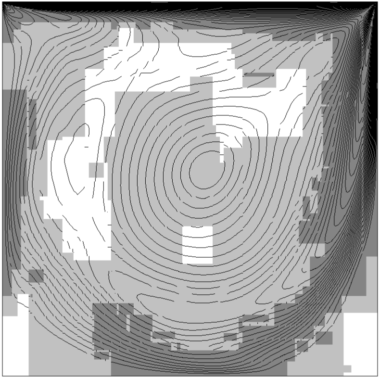
- Dynamic refinement based on heuristic error estimation of $|\mathbf{u}|$
- Threshold intentionally chosen to show refinement evolution



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

Driven cavity - dynamic refinement

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



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

Driven cavity - 3d cavity

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

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

Grid and cells used on 24 cores

Cores	6	12	24	48	96
Time per step	1.82s	0.94s	0.50s	0.28s	0.16s
Par. Efficiency	100.00%	96.47%	90.00%	81.68%	73.04%
LBM Update	44.97%	42.83%	39.64%	35.37%	31.10%
Error Estimation	1.37%	1.30%	1.20%	1.07%	0.94%
Regridding	14.59%	14.79%	15.60%	16.75%	19.14%
Fixup	4.18%	3.96%	3.74%	3.42%	3.07%
Interp. Boundaries	9.34%	9.15%	8.30%	7.17%	6.13%
Interp. Regridding	3.53%	3.23%	3.02%	2.73%	2.44%
Sync Boundaries	8.69%	11.20%	14.28%	18.26%	21.07%
Sync Fixup	2.41%	3.41%	4.70%	6.50%	7.99%
Sync Regridding	0.77%	0.72%	0.74%	0.83%	0.99%
Phys. Boundaries	0.69%	0.68%	0.63%	0.56%	0.49%
Clustering	0.55%	0.48%	0.44%	0.40%	0.36%
Misc	8.90%	8.25%	7.72%	6.95%	6.26%

Driven cavity - 3d cavity

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

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

- AMR with 4 ghost cells

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

Driven cavity - 3d cavity

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

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

- AMR with 4 ghost cells

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

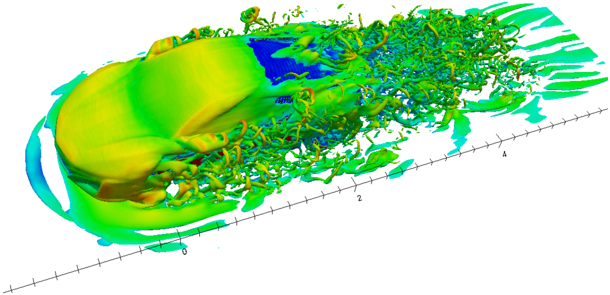
- Expense for boundary is increased compared to FV methods because the algorithm uses few floating point operations but a large state vector!

Wind tunnel simulation of a prototype car

- Use level set implementation basically as before to enforce no-slip boundary conditions $\mathbf{u}' = 2\mathbf{w} - \mathbf{u}$
- Construction of macro-values in embedded boundary cells by interpolation / extrapolation.
- Then use $f_\alpha^{eq}(\rho', \mathbf{u}')$ to construct distributions in embedded ghost cells.
- 2nd order improvements possible, cf. [Peng and Luo, 2008].

Wind tunnel simulation of a prototype car

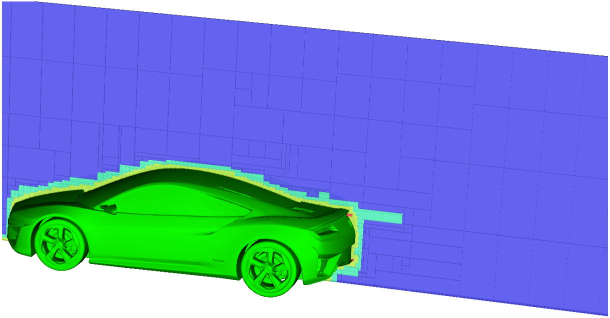
- Use level set implementation basically as before to enforce no-slip boundary conditions $\mathbf{u}' = 2\mathbf{w} - \mathbf{u}$
- Construction of macro-values in embedded boundary cells by interpolation / extrapolation.
- Then use $f_\alpha^{eq}(\rho', \mathbf{u}')$ to construct distributions in embedded ghost cells.
- 2nd order improvements possible, cf. [Peng and Luo, 2008].



- Inflow velocity: $40\,\mathrm{m/s}$, domain $13\,\mathrm{m} \times 5\,\mathrm{m} \times 3\,\mathrm{m}$

Wind tunnel simulation of a prototype car

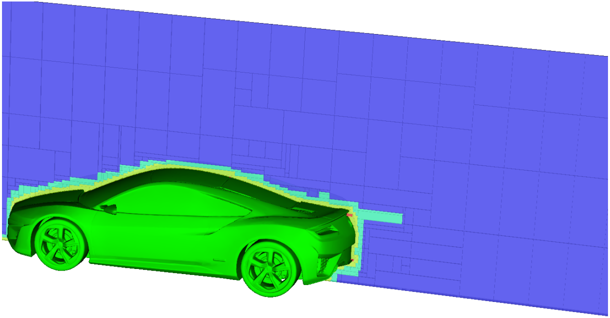
Used refinement blocks and levels (indicated by color)



- SAMR base grid 520 × 200 × 120 cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 3.125\,\mathrm{mm}$
- From $t = 0.1\,\mathrm{s}$ to $t = 0.455\,\mathrm{s}$ ($\sim$ 3 characteristic lengths) with 22,360 time steps on finest level in $48\,\mathrm{h}$ on 144 cores ($6912\,\mathrm{h}$ CPU)
- Adaptation based on level set and scaled gradient of magnitude of vorticity vector

Wind tunnel simulation of a prototype car

Used refinement blocks and levels (indicated by color)



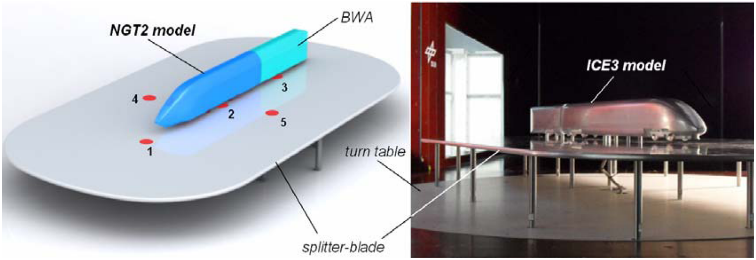
- SAMR base grid 520 × 200 × 120 cells, $r_{1,2,3} = 2$ yielding finest resolution of $\Delta x = 3.125\,\mathrm{mm}$
- From $t = 0.1\,\mathrm{s}$ to $t = 0.455\,\mathrm{s}$ ($\sim$ 3 characteristic lengths) with 22,360 time steps on finest level in $48\,\mathrm{h}$ on 144 cores ($6912\,\mathrm{h}$ CPU)
- Adaptation based on level set and scaled gradient of magnitude of vorticity vector
- 186M cells vs. 6.4 billion (uniform) at $t = 0.455\,\mathrm{s}$

Refinement at $t = 0.455\,\mathrm{s}$

Level	Grids	Cells
0	14,963	12,480,000
1	12,535	20,873,472
2	29,247	110,924,480
3	19,048	42,094,064

Aero-dynamic investigation of train models

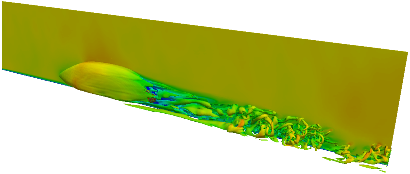
- 1:25 train model represented with 74,670 triangles (41,226 front body, 12,398 back body, 21,006 blade)
- Wind tunnel conditions: air at room temperature with 60.25 $\mathrm{m/s}$ ($M = 0.18$), $\mathrm{Re} = 450,000$
- Purpose: systematic side wind investigation with $0 \geq \beta \geq 30^o$ to obtain lift, drag and roll moment coefficients

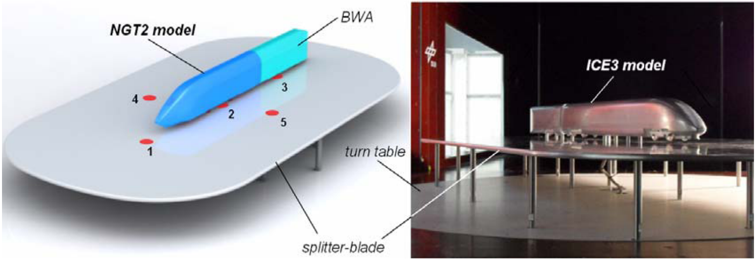


Aero-dynamic investigation of train models

- 1:25 train model represented with 74,670 triangles (41,226 front body, 12,398 back body, 21,006 blade)
- Wind tunnel conditions: air at room temperature with 60.25 $\mathrm{m/s}$ ($M = 0.18$), $\mathrm{Re} = 450,000$
- Purpose: systematic side wind investigation with $0 \geq \beta \geq 30^o$ to obtain lift, drag and roll moment coefficients

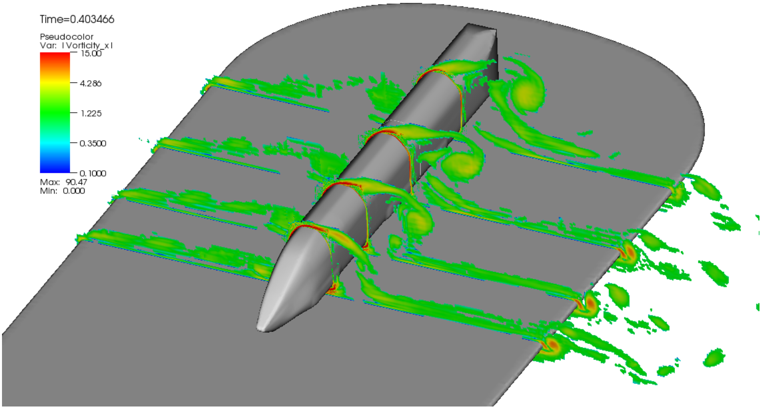
Test: Vorticity and velocity behind mirrored train head at 30 $\mathrm{m/s}$





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

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

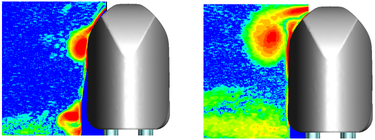


Vorticity vector component perpendicular to middle axis.

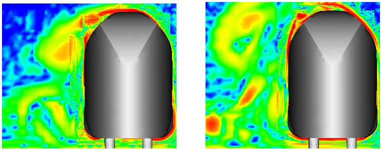
Flow prediction, $\mathrm{Re} = 450,000$, $\beta = 30^o$

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

Experiment (time-averaged)



AMROC-LBM Simulation (instantaneous snapshots)

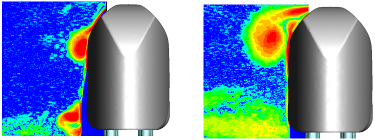


Vorticity component (seen from behind) in axial direction $80\,\mathrm{mm}$ and $290\,\mathrm{mm}$ away from model tip.

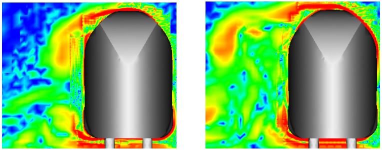
Flow prediction, $\mathrm{Re} = 450,000$, $\beta = 30^o$

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

Experiment (time-averaged)



AMROC-LBM Simulation (instantaneous snapshots)

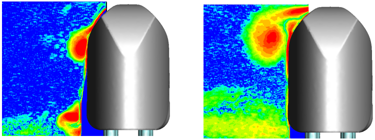


Vorticity component (seen from behind) in axial direction $80\,\mathrm{mm}$ and $290\,\mathrm{mm}$ away from model tip.

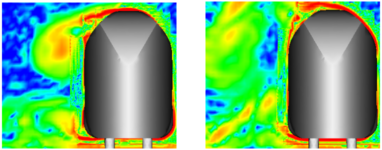
Flow prediction, $\mathrm{Re} = 450,000$, $\beta = 30^o$

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

Experiment (time-averaged)



AMROC-LBM Simulation (instantaneous snapshots)

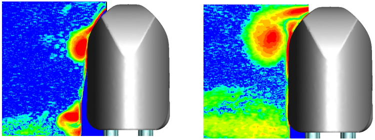


Vorticity component (seen from behind) in axial direction 80 mm and 290 mm away from model tip.

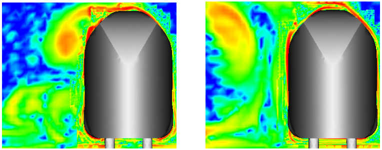
Flow prediction, $\mathrm{Re} = 450,000$, $\beta = 30^o$

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

Experiment (time-averaged)



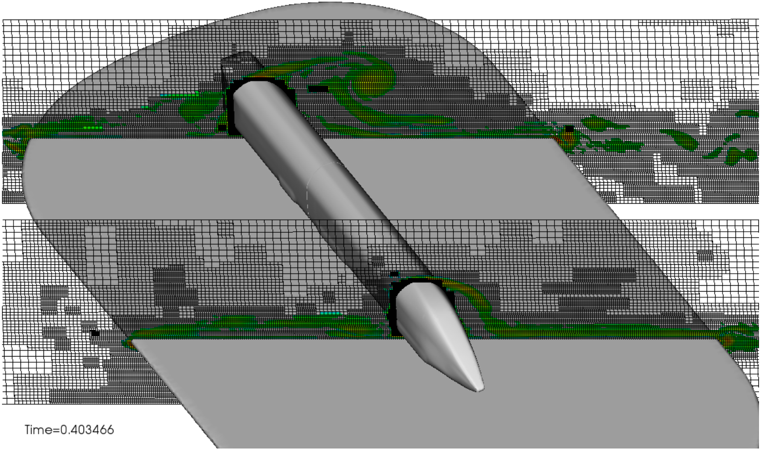
AMROC-LBM Simulation (instantaneous snapshots)



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

Dynamic mesh adaptation

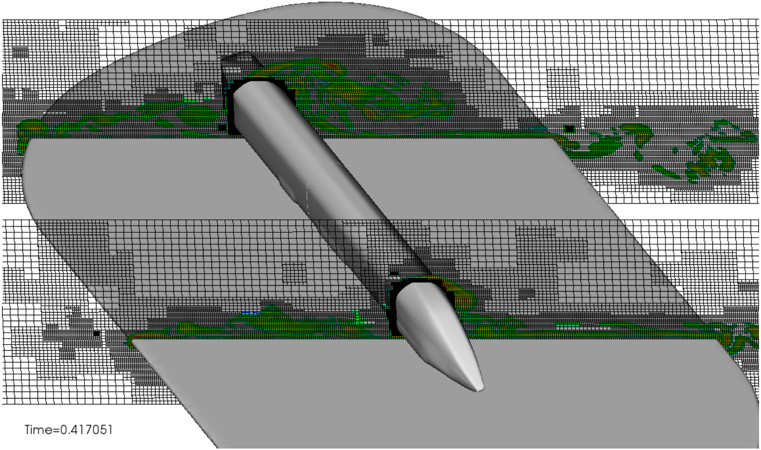
- Base mesh 500 $\times$ 120 $\times$ 80 cells, refinement factors 2,2,4.
- Refinement based on error estimation of $|\mathbf{u}|$ up to second highest level.
- Highest level reserved to geometry refinement with $\Delta x = 1.25\,\mathrm{mm}$.



Dynamically adapting mesh. View in wind direction.

Dynamic mesh adaptation

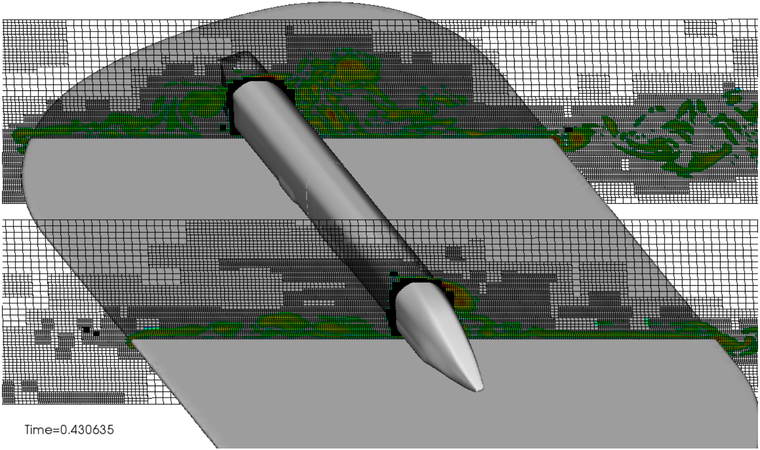
- Base mesh $500 \times 120 \times 80$ cells, refinement factors 2,2,4.
- Refinement based on error estimation of $|\mathbf{u}|$ up to second highest level.
- Highest level reserved to geometry refinement with $\Delta x = 1.25\,\mathrm{mm}$.



Dynamically adapting mesh. View in wind direction.

Dynamic mesh adaptation

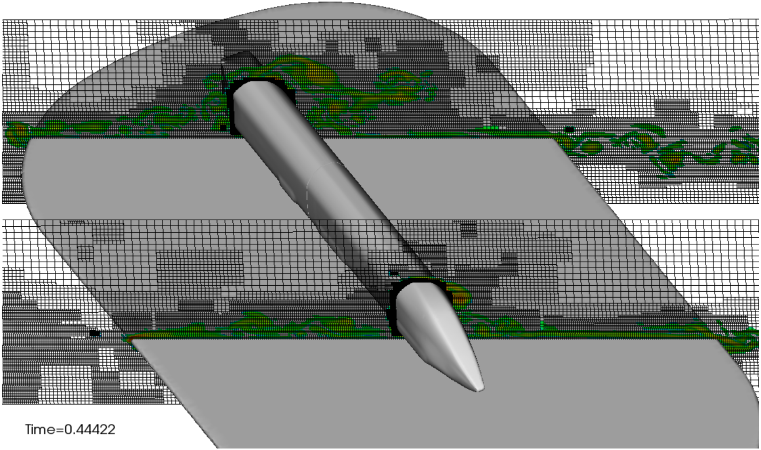
- Base mesh $500 \times 120 \times 80$ cells, refinement factors 2,2,4.
- Refinement based on error estimation of $|\mathbf{u}|$ up to second highest level.
- Highest level reserved to geometry refinement with $\Delta x = 1.25\,\mathrm{mm}$.



Dynamically adapting mesh. View in wind direction.

Dynamic mesh adaptation

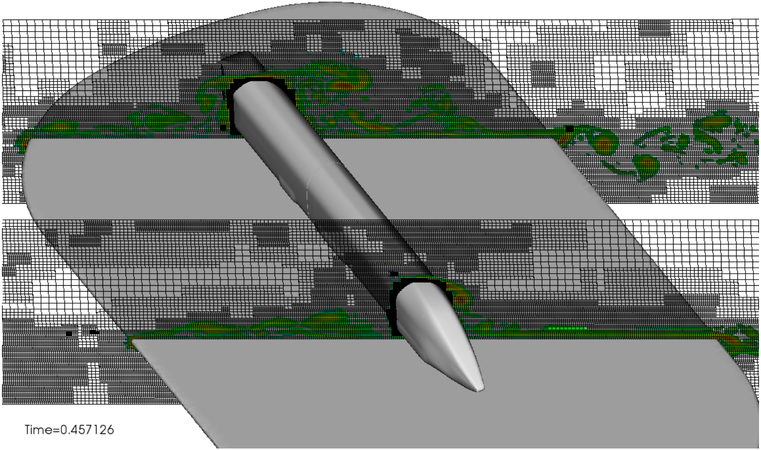
- Base mesh 500 × 120 × 80 cells, refinement factors 2,2,4.
- Refinement based on error estimation of $|\mathbf{u}|$ up to second highest level.
- Highest level reserved to geometry refinement with $\Delta x = 1.25\,\mathrm{mm}$.



Dynamically adapting mesh. View in wind direction.

Dynamic mesh adaptation

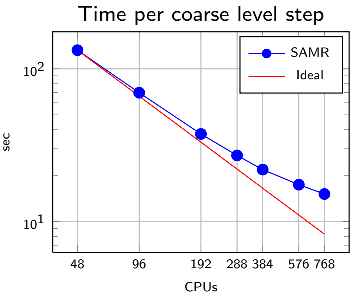
- Base mesh $500 \times 120 \times 80$ cells, refinement factors 2,2,4.
- Refinement based on error estimation of $|\mathbf{u}|$ up to second highest level.
- Highest level reserved to geometry refinement with $\Delta x = 1.25\,\mathrm{mm}$.



Dynamically adapting mesh. View in wind direction.

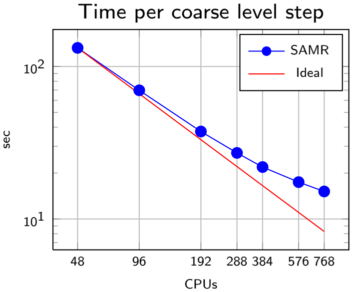
Strong scalability test

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



Strong scalability test

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

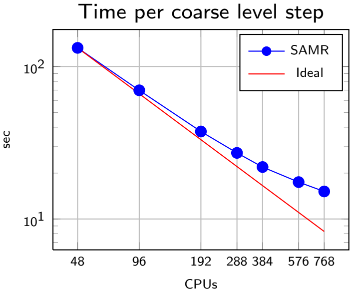


Time in % spent in main operations

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

Strong scalability test

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

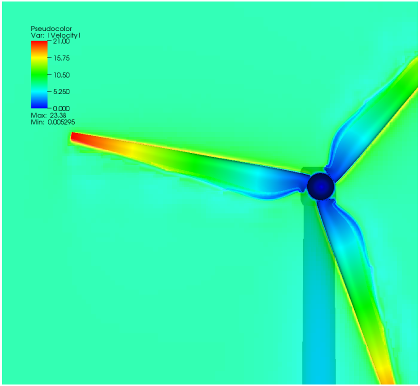


Time in % spent in main operations

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

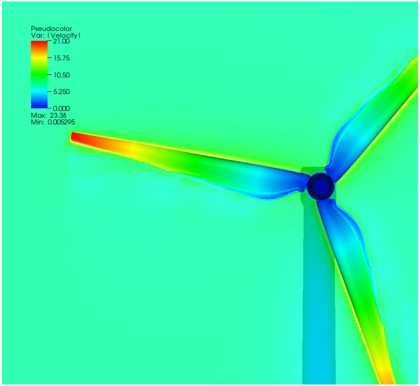
Simulation of a single turbine

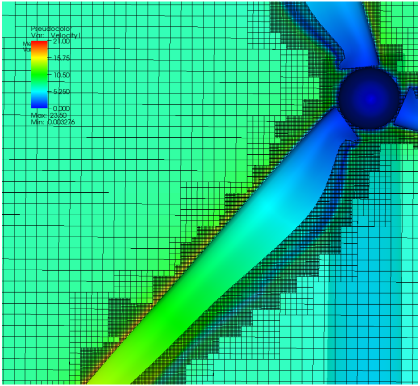
- Geometry from realistic Vestas V27 turbine. Rotor diameter $27\,\mathrm{m}$, tower height $\sim 35\,\mathrm{m}$. Ground considered.
- Prescribed motion of rotor with $15\,\mathrm{rpm}$. Inflow velocity $7\,\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.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



Simulation of a single turbine

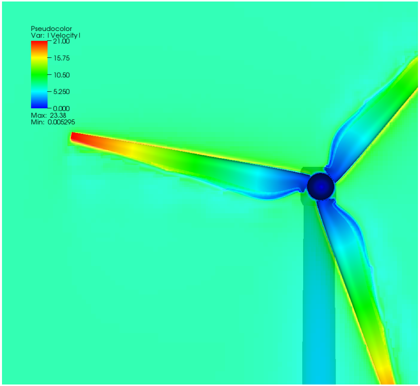
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$.
- Simulation domain 200 $\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh 400 $\times$ 200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

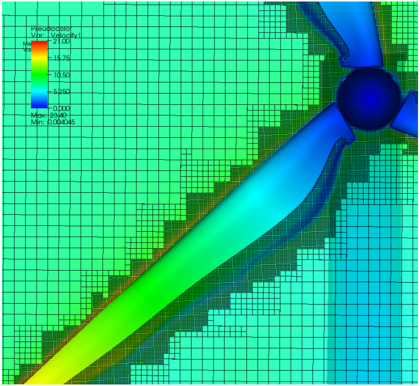




Simulation of a single turbine

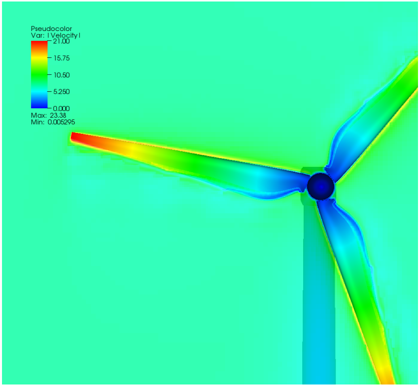
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 m/s.
- Simulation domain 200 m $\times$ 100 m $\times$ 100 m.
- Base mesh 400 $\times$ 200 $\times$ 200 cells with refinement factors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

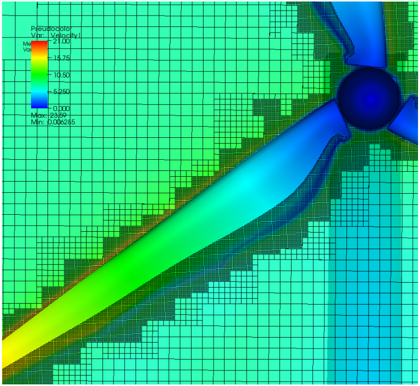




Simulation of a single turbine

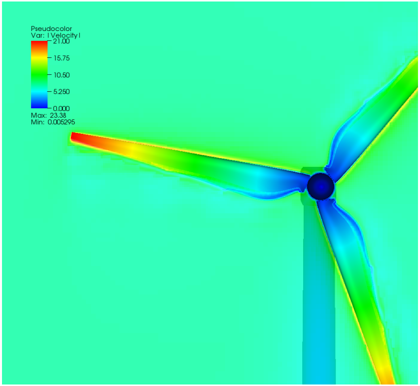
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh 400 $\times$ 200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

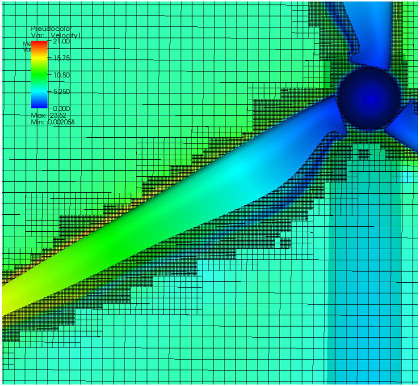




Simulation of a single turbine

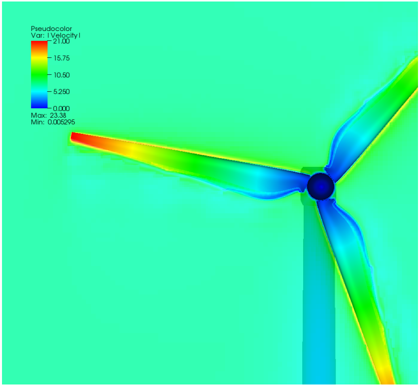
- Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
- Prescribed motion of rotor with 15 rpm. Inflow velocity 7 $\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh $400 \times 200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

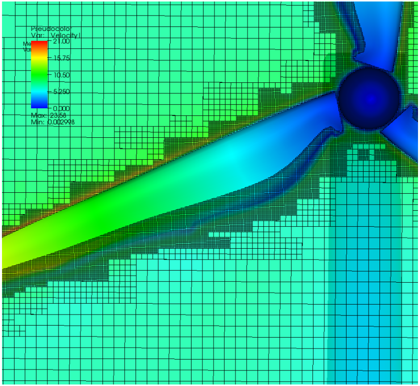




Simulation of a single turbine

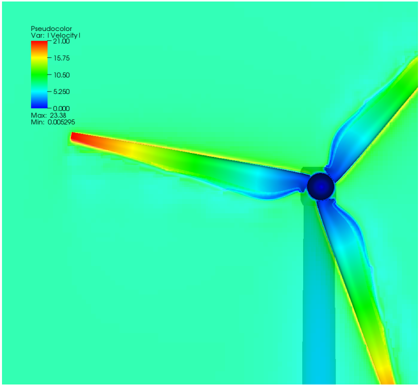
- Geometry from realistic Vestas V27 turbine. Rotor diameter $27\,\mathrm{m}$, tower height $\sim 35\,\mathrm{m}$. Ground considered.
- Prescribed motion of rotor with $15\,\mathrm{rpm}$. Inflow velocity $7\,\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.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

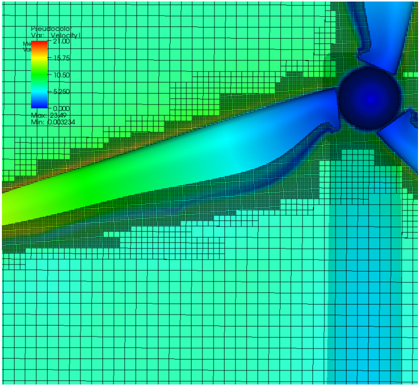




Simulation of a single turbine

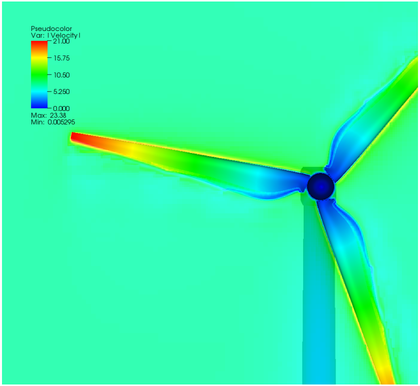
- Geometry from realistic Vestas V27 turbine. Rotor diameter $27\,\mathrm{m}$, tower height $\sim 35\,\mathrm{m}$. Ground considered.
- Prescribed motion of rotor with $15\,\mathrm{rpm}$. Inflow velocity $7\,\mathrm{m/s}$.
- Simulation domain $200\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$.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

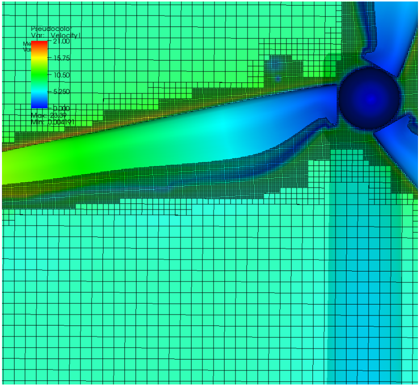




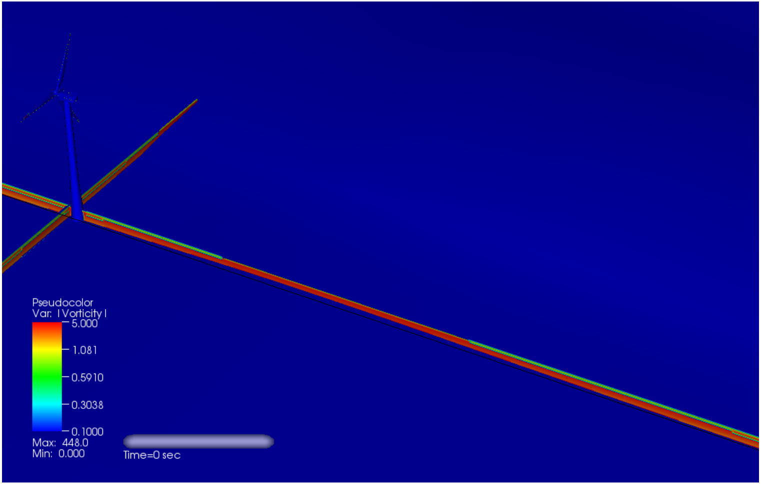
Simulation of a single turbine

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



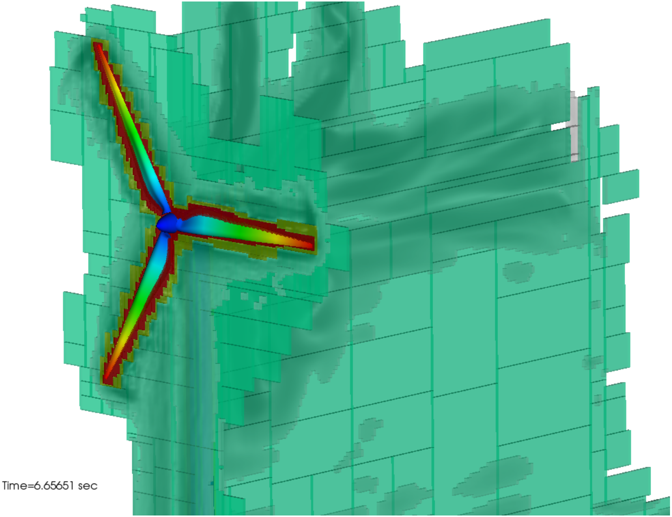


Wake field behind turbine



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

Adaptive refinement

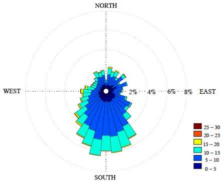


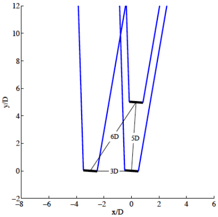
Dynamic evolution of refinement blocks (indicated by color).

Simulation of the SWIFT array

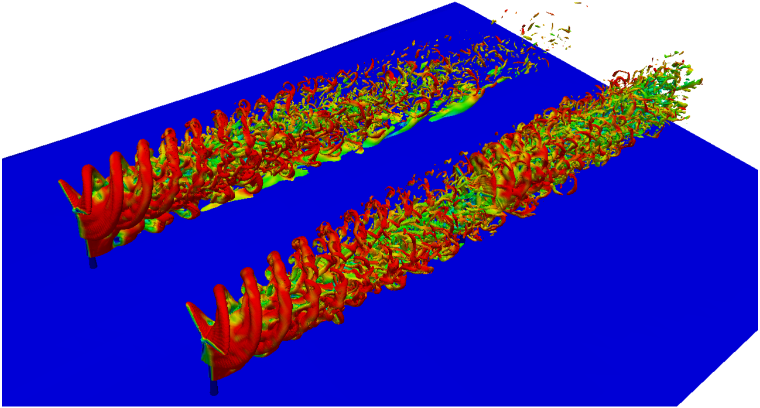
- Three Vestas V27 turbines. 225 kW power generation at wind speeds 14 to 25 $\mathrm{m/s}$ (then cut-off).
- Prescribed motion of rotor with 15 rpm and 43 rpm. Inflow velocity 7 $\mathrm{m/s}$ (power generation 52.5 kW) and 25 $\mathrm{m/s}$.

- Simulation domain 488 $\mathrm{m} \times$ 240 $\mathrm{m} \times 100\,\mathrm{m}$.
- Base mesh 448 $\times$ 240 $\times$ 100 cells with refinement factors 2,2,4. Resolution of rotor and tower $\Delta x = 6.25\,\mathrm{cm}$.
- 94,224 highest level iterations to $t_e = 40\,\mathrm{s}$ computed.





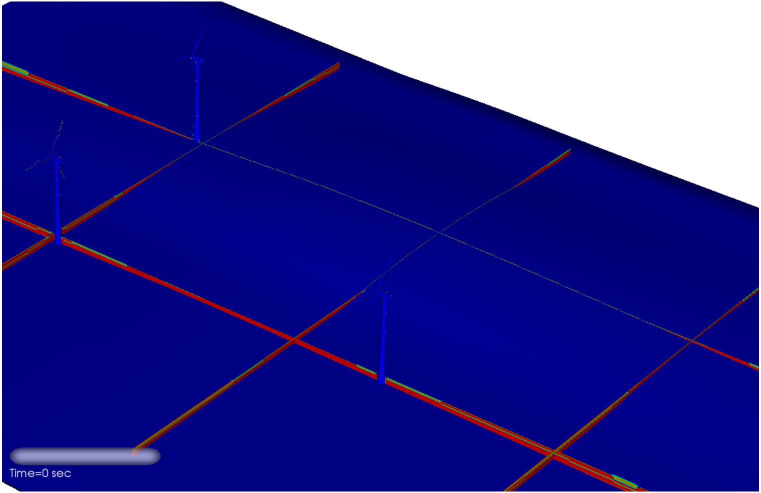
Wake propagation through array – 25 m/s, 43 rpm



- Simulation on 288 cores Intel Xeon-Westmere
- Refinement based on level set and error estimation in $|u|$
- At t_e approximately 140M cells used vs. 44 billion (factor 315)
- Only levels 0 and 1 used for iso-surface visualization

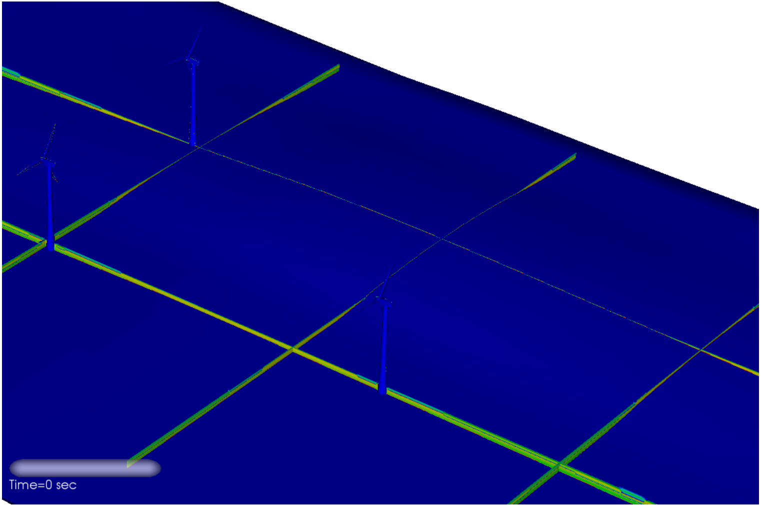
Level	Grids	Cells
0	3,234	10,752,000
1	11,921	21,020,256
2	66,974	102,918,568
3	896	5,116,992

Vorticity generation – 25 m/s, 43 rpm



- Refinement of wake up to level 2 ($\Delta x = 25\,\mathrm{cm}$).
- Rotation starts at $t = 4\,\mathrm{s}$, full refinement at $t = 8\,\mathrm{s}$ to avoid refining initial acoustic waves.

Vorticity generation – 7 $\mathrm{m/s}$, 15 rpm



- Number of wakes between turbines increased from ~ 12 to ~ 15 but vorticity production visibly reduced

Conclusions

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

Conclusions

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

Conclusions

- Developed and demonstrated a first version of a block-based dynamically adaptive LBM for real-world CFD with moving boundaries.
- Reuse of templatized AMROC classes from previous finite volume methods already provides robust real-world capabilities.
- Performance for moderate core count is reasonable, some improvements for larger core count still desirable.
  - Reduce communication width to a single halo layer.
  - Consider workload due to embedded boundary method in partitioning algorithm.
  - Allow other than rigorous domain decomposition.
- Currently developing dynamic Smagorinsky turbulence model (very promising first results)
- Need to develop wall-function model for high Re flows

Conclusions

- Developed and demonstrated a first version of a block-based dynamically adaptive LBM for real-world CFD with moving boundaries.
- Reuse of templatized AMROC classes from previous finite volume methods already provides robust real-world capabilities.
- Performance for moderate core count is reasonable, some improvements for larger core count still desirable.
  - Reduce communication width to a single halo layer.
  - Consider workload due to embedded boundary method in partitioning algorithm.
  - Allow other than rigorous domain decomposition.
- Currently developing dynamic Smagorinsky turbulence model (very promising first results)
- Need to develop wall-function model for high Re flows
- Some more developments for post-processing (drag, lift and friction coefficients), time-averaging of fields on-the-fly under way
- Characteristic boundary conditions for eliminating pressure reflections from small domain boundaries

Conclusions

- Developed and demonstrated a first version of a block-based dynamically adaptive LBM for real-world CFD with moving boundaries.
- Reuse of templatized AMROC classes from previous finite volume methods already provides robust real-world capabilities.
- Performance for moderate core count is reasonable, some improvements for larger core count still desirable.
  - Reduce communication width to a single halo layer.
  - Consider workload due to embedded boundary method in partitioning algorithm.
  - Allow other than rigorous domain decomposition.
- Currently developing dynamic Smagorinsky turbulence model (very promising first results)
- Need to develop wall-function model for high Re flows
- Some more developments for post-processing (drag, lift and friction coefficients), time-averaging of fields on-the-fly under way
- Characteristic boundary conditions for eliminating pressure reflections from small domain boundaries
- Realistic wind turbine model with dynamic pitch angle, nacelle rotation, etc. under development by S. Wood (UT Knoxville)

References I

[Agrawal and Yi, 2009] Agrawal, A. K. and Yi, Z. (2009). Blast load effects on highway bridges. Technical report, University Transportation Research Center, City College of New York.

[Arienti et al., 2003] Arienti, M., Hung, P., Morano, E., and Shepherd, J. E. (2003). A level set approach to Eulerian-Lagrangian coupling. *J. Comput. Phys.*, 185:213–251.

[Bell et al., 1994] Bell, J., Berger, M., Saltzman, J., and Welcome, M. (1994). Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comp.*, 15(1):127–138.

[Berger, 1986] Berger, M. (1986). Data structures for adaptive grid generation. *SIAM J. Sci. Stat. Comput.*, 7(3):904–916.

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

[Berger and Oliger, 1984] Berger, M. and Oliger, J. (1984). Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512.

References II

[Berger and Rigoutsos, 1991] Berger, M. and Rigoutsos, I. (1991). An algorithm for point clustering and grid generation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1278–1286.

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

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

[Deiterding et al., 2009] Deiterding, R., Cirak, F., and Mauch, S. P. (2009). Efficient fluid-structure interaction simulation of viscoplastic and fracturing thin-shells subjected to underwater shock loading. In Hartmann, S., Meister, A., Schäfer, M., and Turek, S., editors, *Int. Workshop on Fluid-Structure Interaction. Theory, Numerics and Applications, Herrsching am Ammersee 2008*, pages 65–80. kassel university press GmbH.

References III

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

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

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

[Fedkiw, 2002] Fedkiw, R. P. (2002). Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method. *J. Comput. Phys.*, 175:200–224.

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

[Luccioni et al., 2004] Luccioni, B. M., Ambrosini, R. D., and Danesi, R. F. (2004). Analysis of building collapse under blast loads. *Engineering & Structures*, 26:63–71.

References IV

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

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

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

[Rendleman et al., 2000] Rendleman, C. A., Beckner, V. E., Lijewski, M., Crutchfield, W., and Bell, J. B. (2000). Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, 3:147–157.

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

[Specht, 2000] Specht, U. (2000). *Numerische Simulation mechanischer Wellen an Fluid-Festkörper-Mediengrenzen*. Number 398 in VDI Reihe 7. VDU Verlag, Düsseldorf.

Clustering by signatures

				x	x	x	x	x	x	6
				x	x	x	x	x	x	6
			x	x	x					3
	x	x	x							3
	x	x								2
	x	x								2
	x	x								2
										0
	x	x								2
	x	x								2
Υ	6	6	2	3	2	2	2	2	2	

Υ Flagged cells per row/column

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

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

Clustering by signatures

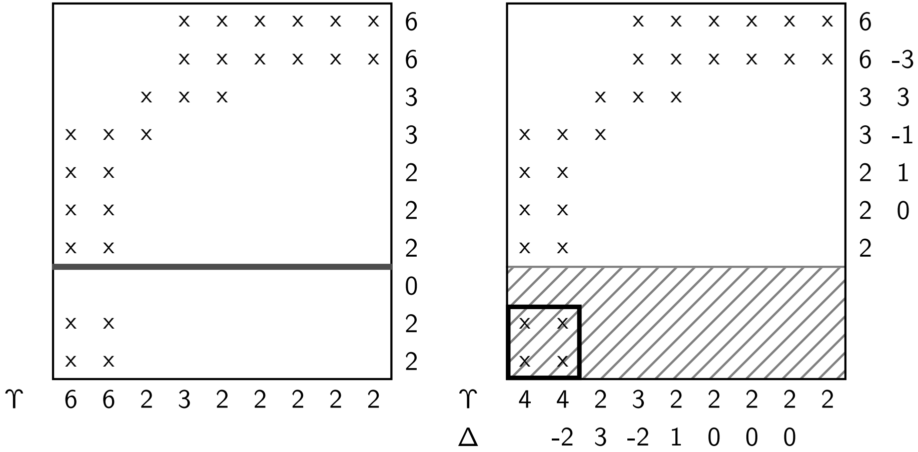
				x	x	x	x	x	x	6
				x	x	x	x	x	x	6
			x	x	x					3
	x	x	x							3
	x	x								2
	x	x								2
	x	x								2
										0
	x	x								2
	x	x								2
Υ	6	6	2	3	2	2	2	2	2	

Υ Flagged cells per row/column

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

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

Clustering by signatures

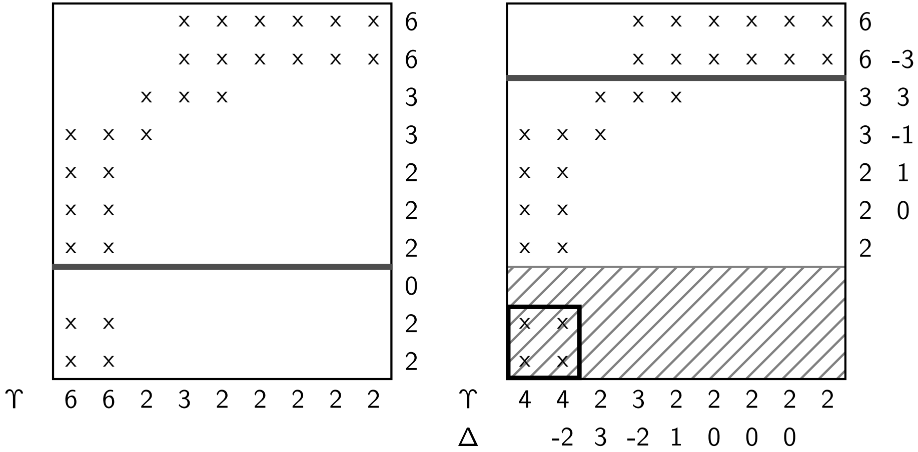


Υ Flagged cells per row/column

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

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

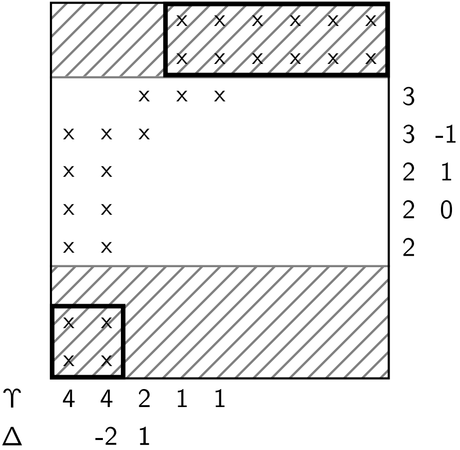
Clustering by signatures



Υ Flagged cells per row/column

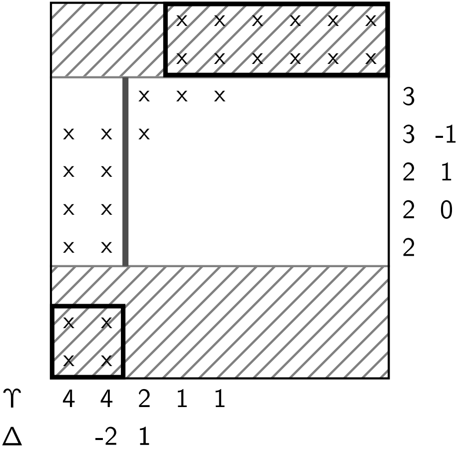
Δ Second derivative of Υ, $\Delta = \Upsilon_{\nu+1} - 2\,\Upsilon_{\nu} + \Upsilon_{\nu-1}$

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



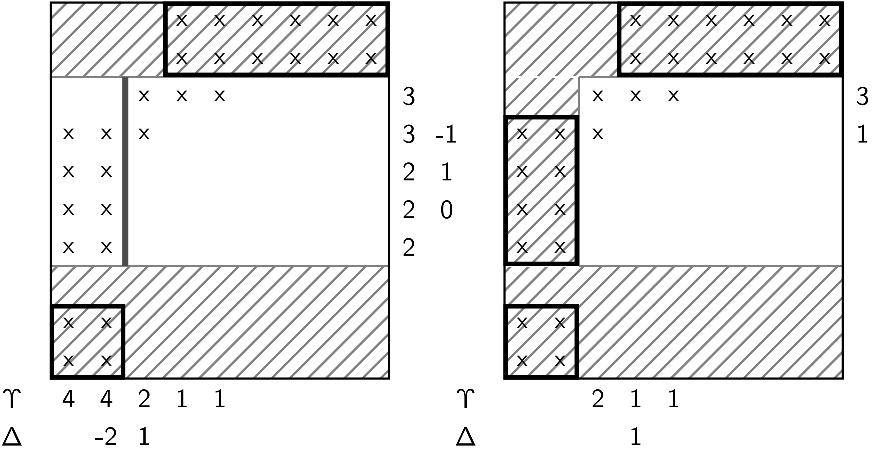
Recursive generation of $\breve{G}_{l,m}$

1. 0 in Υ
2. Largest difference in Δ
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$



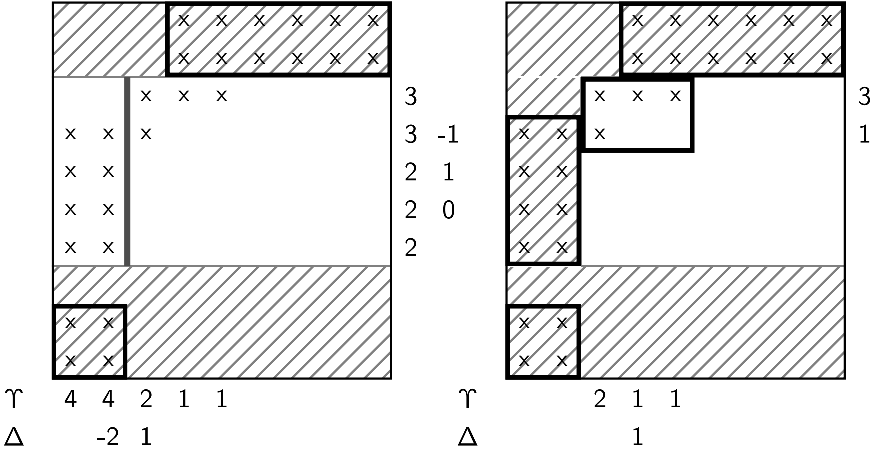
Recursive generation of $\breve{G}_{l,m}$

1. 0 in Υ
2. Largest difference in Δ
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$



Recursive generation of $\breve{G}_{l,m}$

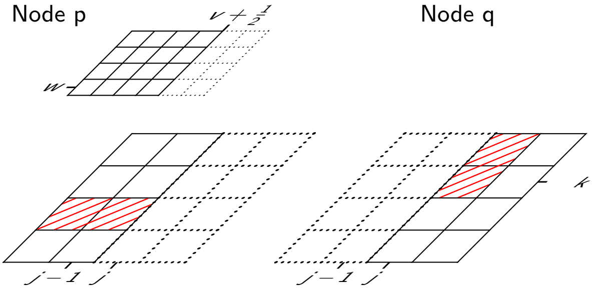
1. 0 in Υ
2. Largest difference in Δ
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$



Recursive generation of $\breve{G}_{l,m}$

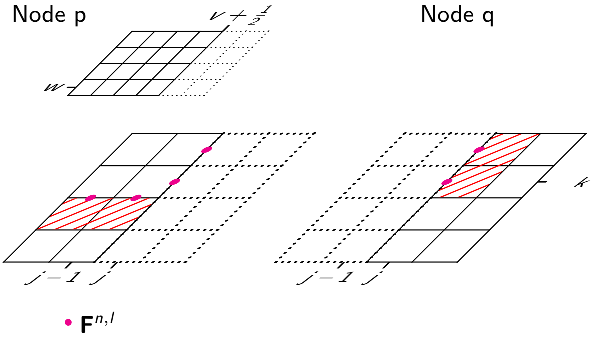
1. 0 in Υ
2. Largest difference in Δ
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$

Parallel flux correction



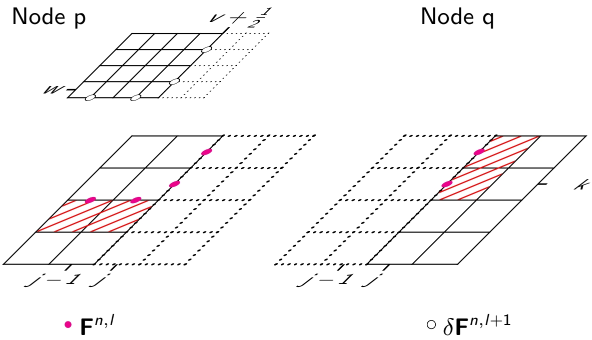
Parallel flux correction

1. Strictly local: Init $\delta \mathbf{F}^{n,l+1}$ with $\mathbf{F}^n(\bar{G}_{l,m} \cap \partial G_{l+1}, t)$



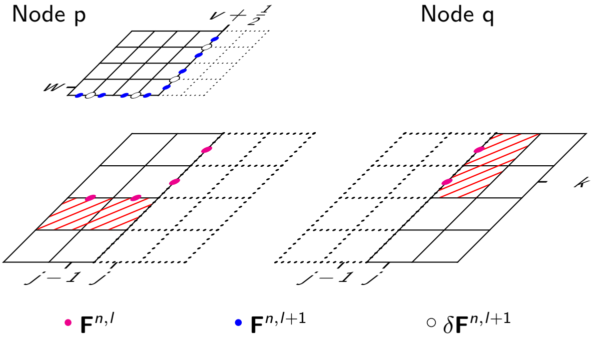
Parallel flux correction

1. Strictly local: Init $\delta\mathbf{F}^{n,l+1}$ with $\mathbf{F}^n(\bar{G}_{l,m} \cap \partial G_{l+1}, t)$



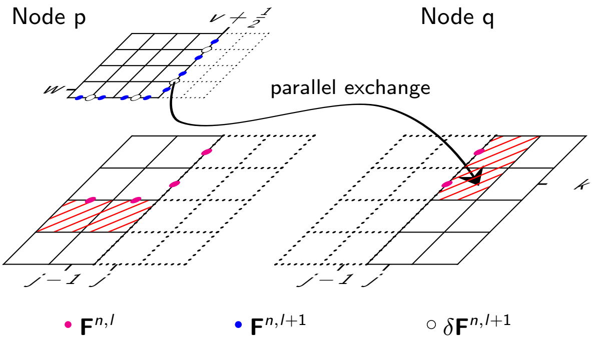
Parallel flux correction

1. Strictly local: Init $\delta\mathbf{F}^{n,l+1}$ with $\mathbf{F}^n(\bar{G}_{l,m} \cap \partial G_{l+1}, t)$
2. Strictly local: Add $\mathbf{F}^n(\partial G_{l,m}, t)$ to $\delta\mathbf{F}^{n,l}$



Parallel flux correction

1. Strictly local: Init $\delta\mathbf{F}^{n,l+1}$ with $\mathbf{F}^n(\bar{G}_{l,m} \cap \partial G_{l+1}, t)$
2. Strictly local: Add $\mathbf{F}^n(\partial G_{l,m}, t)$ to $\delta\mathbf{F}^{n,l}$
3. Parallel communication: Correct $\mathbf{Q}^l(t + \Delta t_l)$ with $\delta\mathbf{F}^{l+1}$



Closest point transform algorithm

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

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

Solution smooth but non-diferentiable across characteristics.

Closest point transform algorithm

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

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

Solution smooth but non-diferentiable across characteristics.

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

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

Closest point transform algorithm

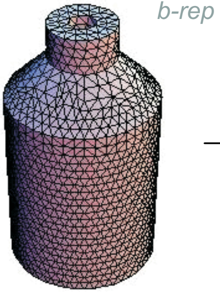
The signed distance φ to a surface $\mathcal{I}$ satisfies the eikonal equation [Sethian, 1999]

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

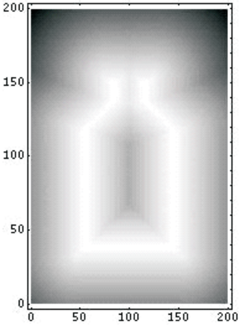
Solution smooth but non-diferentiable across characteristics.

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

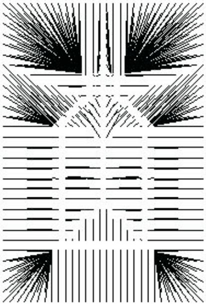
- Geometric solution approach with plosest-point-transform algorithm [Mauch, 2003]



Surface mesh $\mathcal{I}$



Distance φ

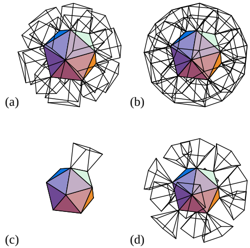


Normal to closest point

The characteristic / scan conversion algorithm

1. Build the characteristic polyhedrons for the surface mesh

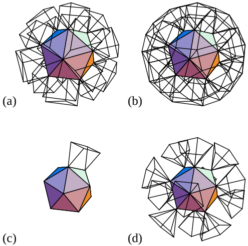
Characteristic polyhedra for faces, edges, and vertices



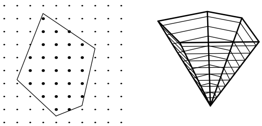
The characteristic / scan conversion algorithm

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

Characteristic polyhedra for faces, edges, and vertices



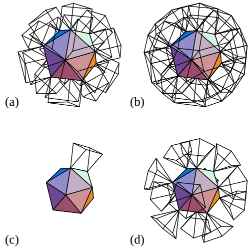
Slicing and scan conversion of apolygon



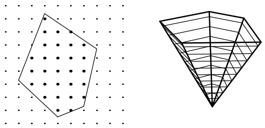
The characteristic / scan conversion algorithm

1. Build the characteristic polyhedrons for the surface mesh
2. For each face/edge/vertex
   2.1 Scan convert the polyhedron.
   2.2 Compute distance to that primitive for the scan converted points

Characteristic polyhedra for faces, edges, and vertices



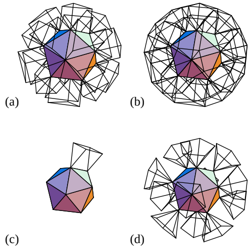
Slicing and scan conversion of apolygon



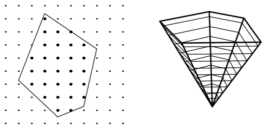
The characteristic / scan conversion algorithm

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

Characteristic polyhedra for faces, edges, and vertices



Slicing and scan conversion of apolygon

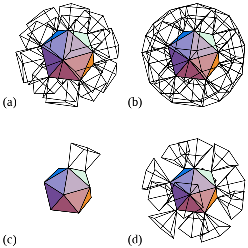


The characteristic / scan conversion algorithm

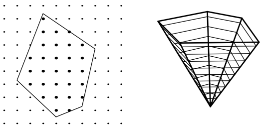
1. Build the characteristic polyhedrons for the surface mesh
2. For each face/edge/vertex
   2.1 Scan convert the polyhedron.
   2.2 Compute distance to that primitive for the scan converted points
3. Computational complexity.
   - $O(m)$ to build the b-rep and the polyhedra.
   - $O(n)$ to scan convert the polyhedra and compute the distance, etc.
4. Problem reduction by evaluation only within specified max. distance

[Mauch, 2003], see also [Deiterding et al., 2006]

Characteristic polyhedra for faces, edges, and vertices

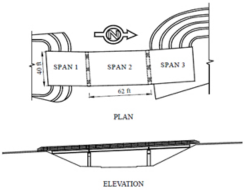


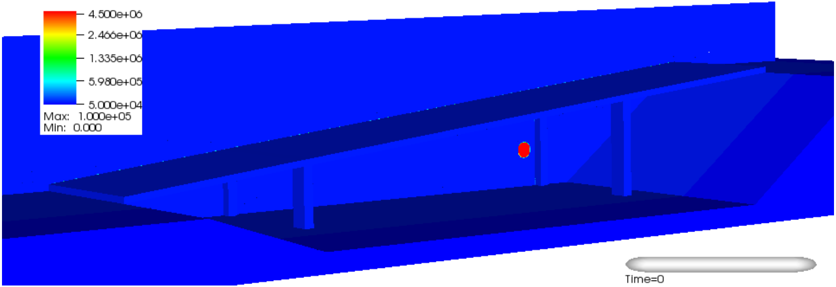
Slicing and scan conversion of apolygon



Blast under a highway bridge

- Case follows [Agrawal and Yi, 2009]: blast explosion $0.5\,\mathrm{m}$ in front of the high middle column, $2\,\mathrm{m}$ above the ground
- *Elastic* material model with $\rho_s = 2010\,\mathrm{kg/m^3}$, $E = 21.72\,\mathrm{GPa}$, $\nu = 0.2$, 3365 solid hexahedron elements
- SAMR: $240 \times 40 \times 80$ base level, three additional levels $r_{1,2,3} = 2$, $l_{\mathrm{fsi}} = 2$, $k = 1$
- $487\,\mathrm{h}$ CPU on 63+1 CPU 3.4 GHz Intel-Xeon, 1504 coupled time steps to $t_{\mathrm{end}} = 20\,\mathrm{ms}$





Blast under highway bridge - initial conditions and meshing

Uniform pressure in sphere

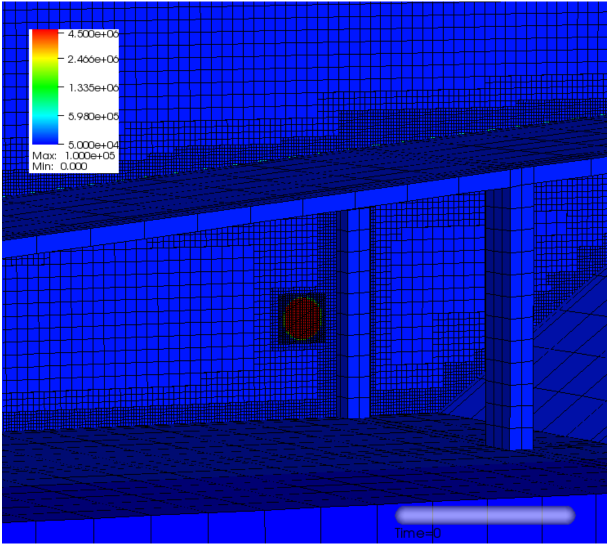
$$p = (\gamma - 1) e_i \left(\frac{4}{3} \pi r^3 \right)^{-1}$$

with energy $e_{\mathrm{TNT}} = 4,520,000\,\mathrm{J/kg}$ and density from

$$\rho = (pW)/(\mathcal{R}T)$$

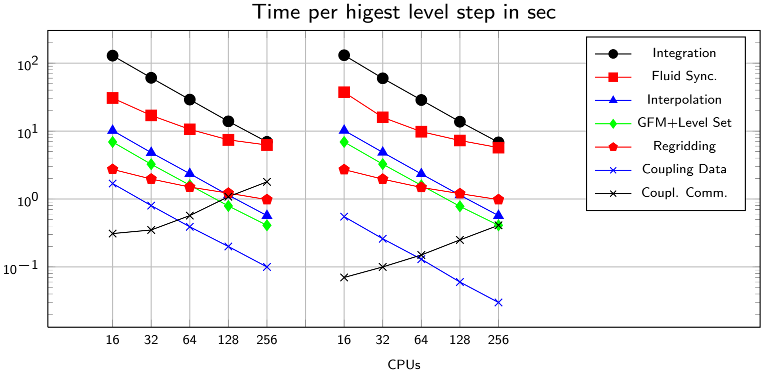
with $T = 1465\,\mathrm{K}$ in air, otherwise at atmospheric conditions at $T_0 = 293\,\mathrm{K}$

- Here: $750\,\mathrm{kg}$ TNT, $r = 0.4\,\mathrm{m}$



Blast under a highway bridge – strong scalability

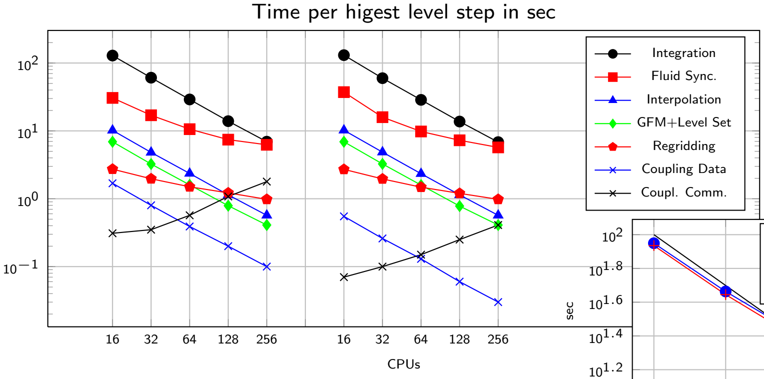
- SAMR: $240 \times 40 \times 80$, two levels: $r_1 = 2$, $r_2 = 4$; coupling: $l_{\mathrm{fsi}} = 2$, $k = 1$
- Timing done on fluid side for 24 steps on finest level
- $\sim 56,500,000$ cells instead $393,216,000$

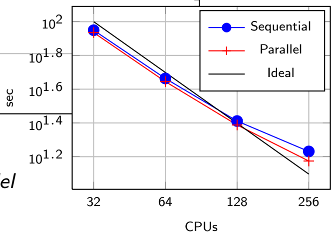


Left: *update_type=sequential*, right: *update_type=parallel*

Blast under a highway bridge – strong scalability

- SAMR: $240 \times 40 \times 80$, two levels: $r_1 = 2$, $r_2 = 4$; coupling: $l_{\mathrm{fsi}} = 2$, $k = 1$
- Timing done on fluid side for 24 steps on finest level
- $\sim 56,500,000$ cells instead $393,216,000$





Left: *update_type=sequential*, right: *update_type=parallel*