Lecture 2

The SAMR method for hyperbolic problems

Course *Block-structured Adaptive Mesh Refinement Methods for Conservation Laws*
Theory, Implementation and Application

Ralf Deiterding
Computer Science and Mathematics Division
Oak Ridge National Laboratory
P.O. Box 2008 MS6367, Oak Ridge, TN 37831, USA

E-mail: deiterdingr@ornl.gov

Outline

Outline

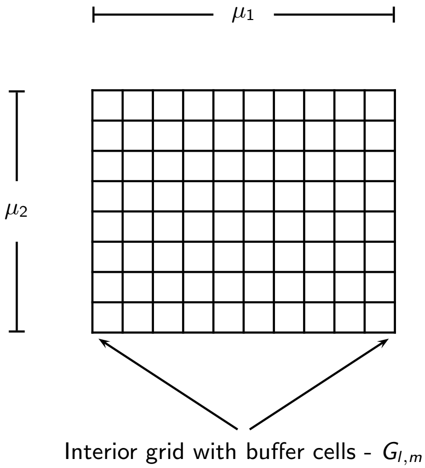
Outline

Outline

The mth refinement grid on level l

Notations:

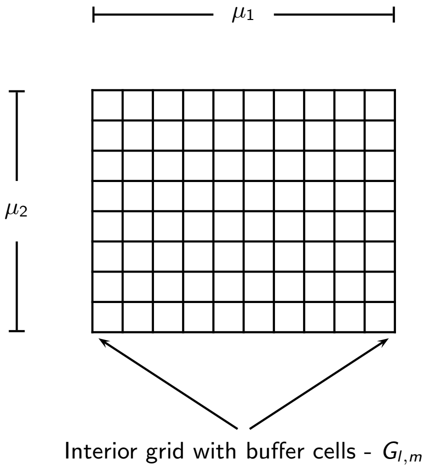
- Boundary: $\partial G_{l,m}$



The mth refinement grid on level l

Notations:

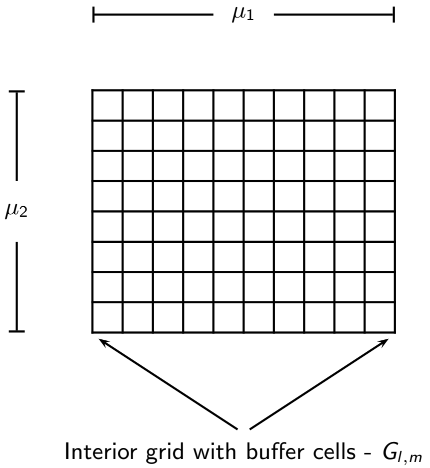
- Boundary: $\partial G_{l,m}$
- Hull: $\bar{G}_{l,m} = G_{l,m} \cup \partial G_{l,m}$



The *m*th refinement grid on level *l*

Notations:

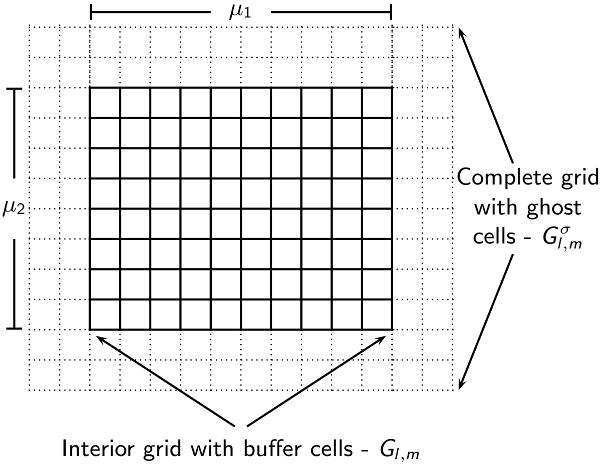
- Boundary: $\partial G_{l,m}$
- Hull:
  $\bar{G}_{l,m} = G_{l,m} \cup \partial G_{l,m}$



The mth refinement grid on level l

Notations:

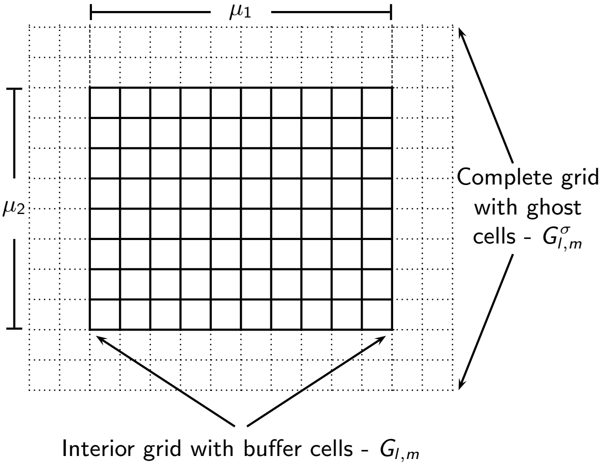
- Boundary: $\partial G_{l,m}$
- Hull:
  $\bar{G}_{l,m} = G_{l,m} \cup \partial G_{l,m}$



The mth refinement grid on level l

Notations:

- Boundary: $\partial G_{l,m}$
- Hull: $\bar{G}_{l,m} = G_{l,m} \cup \partial G_{l,m}$
- Ghost cell region: $\tilde{G}^{\sigma}_{l,m} = G^{\sigma}_{l,m} \backslash \bar{G}_{l,m}$



Refinement data

- Resolution: $\Delta t_l := \dfrac{\Delta t_{l-1}}{r_l}$ and $\Delta x_{n,l} := \dfrac{\Delta x_{n,l-1}}{r_l}$

Refinement data

- Resolution: $\Delta t_l := \dfrac{\Delta t_{l-1}}{r_l}$ and $\Delta x_{n,l} := \dfrac{\Delta x_{n,l-1}}{r_l}$
- Refinement factor: $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$

Refinement data

- Resolution: $\Delta t_l := \dfrac{\Delta t_{l-1}}{r_l}$ and $\Delta x_{n,l} := \dfrac{\Delta x_{n,l-1}}{r_l}$
- Refinement factor: $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$
- Integer coordinate system for internal organization [Bell et al., 1994]:
  $\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$

Refinement data

- Resolution: $\Delta t_l := \dfrac{\Delta t_{l-1}}{r_l}$ and $\Delta x_{n,l} := \dfrac{\Delta x_{n,l-1}}{r_l}$
- Refinement factor: $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$
- Integer coordinate system for internal organization [Bell et al., 1994]:

  $$\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$$

- Computational Domain: $G_0 = \bigcup_{m=1}^{M_0} G_{0,m}$

Refinement data

- Resolution: $\Delta t_l := \dfrac{\Delta t_{l-1}}{r_l}$ and $\Delta x_{n,l} := \dfrac{\Delta x_{n,l-1}}{r_l}$
- Refinement factor: $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$
- Integer coordinate system for internal organization [Bell et al., 1994]:
$$\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$$
- Computational Domain: $G_0 = \bigcup_{m=1}^{M_0} G_{0,m}$
- Domain of level l: $G_l := \bigcup_{m=1}^{M_l} G_{l,m}$ with $G_{l,m} \cap G_{l,n} = \emptyset$ for $m \neq n$

Refinement data

- Resolution: $\Delta t_l := \dfrac{\Delta t_{l-1}}{r_l}$ and $\Delta x_{n,l} := \dfrac{\Delta x_{n,l-1}}{r_l}$
- Refinement factor: $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$
- Integer coordinate system for internal organization [Bell et al., 1994]:
$$\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$$
- Computational Domain: $G_0 = \bigcup_{m=1}^{M_0} G_{0,m}$
- Domain of level l: $G_l := \bigcup_{m=1}^{M_l} G_{l,m}$ with $G_{l,m} \cap G_{l,n} = \emptyset$ for $m \neq n$
- Refinements are properly nested: $G_l^1 \subset G_{l-1}$

Refinement data

- Resolution: $\Delta t_l := \frac{\Delta t_{l-1}}{r_l}$ and $\Delta x_{n,l} := \frac{\Delta x_{n,l-1}}{r_l}$
- Refinement factor: $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$
- Integer coordinate system for internal organization [Bell et al., 1994]:
  $\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$
- Computational Domain: $G_0 = \bigcup_{m=1}^{M_0} G_{0,m}$
- Domain of level l: $G_l := \bigcup_{m=1}^{M_l} G_{l,m}$ with $G_{l,m} \cap G_{l,n} = \emptyset$ for $m \neq n$
- Refinements are properly nested: $G_l^1 \subset G_{l-1}$
- Assume a FD scheme with stencil radius s. Necessary data:

Refinement data

- Resolution: $\Delta t_l := \dfrac{\Delta t_{l-1}}{r_l}$ and $\Delta x_{n,l} := \dfrac{\Delta x_{n,l-1}}{r_l}$
- Refinement factor: $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$
- Integer coordinate system for internal organization [Bell et al., 1994]:
  $$\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$$
- Computational Domain: $G_0 = \bigcup_{m=1}^{M_0} G_{0,m}$
- Domain of level l: $G_l := \bigcup_{m=1}^{M_l} G_{l,m}$ with $G_{l,m} \cap G_{l,n} = \emptyset$ for $m \neq n$
- Refinements are properly nested: $G_l^1 \subset G_{l-1}$
- Assume a FD scheme with stencil radius s. Necessary data:
  - Vector of state: $\mathbf{Q}^l := \bigcup_m \mathbf{Q}(G_{l,m}^s)$

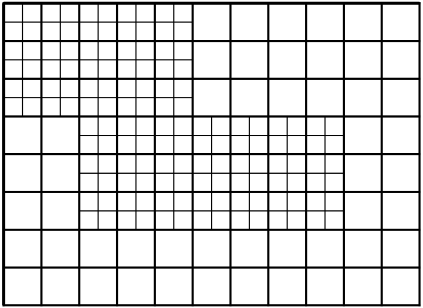
Refinement data

- Resolution: $\Delta t_l := \dfrac{\Delta t_{l-1}}{r_l}$ and $\Delta x_{n,l} := \dfrac{\Delta x_{n,l-1}}{r_l}$
- Refinement factor: $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$
- Integer coordinate system for internal organization [Bell et al., 1994]:
  $$\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$$
- Computational Domain: $G_0 = \bigcup_{m=1}^{M_0} G_{0,m}$
- Domain of level l: $G_l := \bigcup_{m=1}^{M_l} G_{l,m}$ with $G_{l,m} \cap G_{l,n} = \emptyset$ for $m \neq n$
- Refinements are properly nested: $G_l^1 \subset G_{l-1}$
- Assume a FD scheme with stencil radius s. Necessary data:
  - Vector of state: $\mathbf{Q}^l := \bigcup_m \mathbf{Q}(G_{l,m}^s)$
  - Numerical fluxes: $\mathbf{F}^{n,l} := \bigcup_m \mathbf{F}^n(\bar{G}_{l,m})$

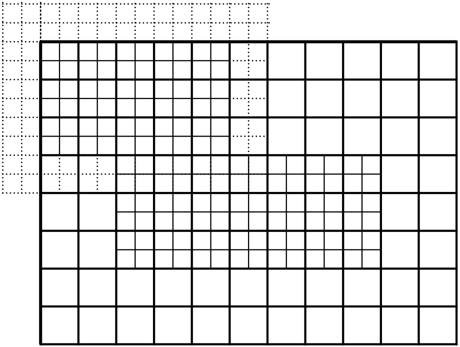
Refinement data

- Resolution: $\Delta t_l := \dfrac{\Delta t_{l-1}}{r_l}$ and $\Delta x_{n,l} := \dfrac{\Delta x_{n,l-1}}{r_l}$
- Refinement factor: $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$
- Integer coordinate system for internal organization [Bell et al., 1994]:
  $$\Delta x_{n,l} \cong \prod_{\kappa=l+1}^{l_{\max}} r_\kappa$$
- Computational Domain: $G_0 = \bigcup_{m=1}^{M_0} G_{0,m}$
- Domain of level l: $G_l := \bigcup_{m=1}^{M_l} G_{l,m}$ with $G_{l,m} \cap G_{l,n} = \emptyset$ for $m \neq n$
- Refinements are properly nested: $G_l^1 \subset G_{l-1}$
- Assume a FD scheme with stencil radius s. Necessary data:
  - Vector of state: $\mathbf{Q}^l := \bigcup_m \mathbf{Q}(G^s_{l,m})$
  - Numerical fluxes: $\mathbf{F}^{n,l} := \bigcup_m \mathbf{F}^n(\bar{G}_{l,m})$
  - Flux corrections: $\delta\mathbf{F}^{n,l} := \bigcup_m \delta\mathbf{F}^n(\partial G_{l,m})$

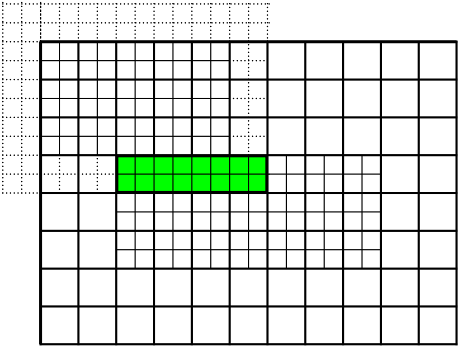
Setting of ghost cells



Setting of ghost cells

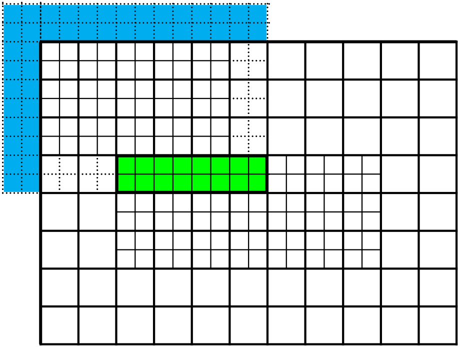


Setting of ghost cells



Synchronization with G_l - $\tilde{S}^s_{l,m} = \tilde{G}^s_{l,m} \cap G_l$

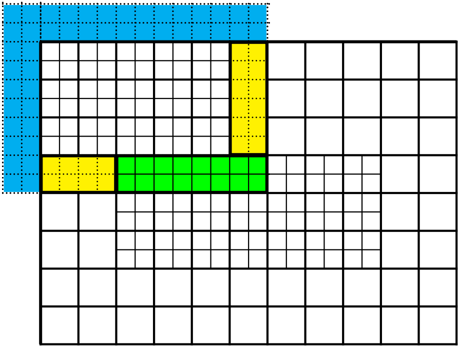
Setting of ghost cells



Synchronization with G_l - $\tilde{S}^s_{l,m} = \tilde{G}^s_{l,m} \cap G_l$

Physical boundary conditions - $\tilde{P}^s_{l,m} = \tilde{G}^s_{l,m} \backslash G_0$

Setting of ghost cells



- Synchronization with G_l - $\tilde{S}^s_{l,m} = \tilde{G}^s_{l,m} \cap G_l$
- Physical boundary conditions - $\tilde{P}^s_{l,m} = \tilde{G}^s_{l,m} \backslash G_0$
- Interpolation from G_{l-1} - $\tilde{I}^s_{l,m} = \tilde{G}^s_{l,m} \backslash (\tilde{S}^s_{l,m} \cup \tilde{P}^s_{l,m})$

Numerical update

Time-explicit conservative finite volume scheme

$$\mathcal{H}^{(\Delta t)}:\ \mathbf{Q}_{jk}(t+\Delta t) = \mathbf{Q}_{jk}(t) - \frac{\Delta t}{\Delta x_1}\left(\mathbf{F}^1_{j+\frac{1}{2},k} - \mathbf{F}^1_{j-\frac{1}{2},k}\right) - \frac{\Delta t}{\Delta x_2}\left(\mathbf{F}^2_{j,k+\frac{1}{2}} - \mathbf{F}^2_{j,k-\frac{1}{2}}\right)$$

Numerical update

Time-explicit conservative finite volume scheme

$$\mathcal{H}^{(\Delta t)}: \ \mathbf{Q}_{jk}(t+\Delta t) = \mathbf{Q}_{jk}(t) - \frac{\Delta t}{\Delta x_1}\left(\mathbf{F}^1_{j+\frac{1}{2},k} - \mathbf{F}^1_{j-\frac{1}{2},k}\right) - \frac{\Delta t}{\Delta x_2}\left(\mathbf{F}^2_{j,k+\frac{1}{2}} - \mathbf{F}^2_{j,k-\frac{1}{2}}\right)$$

`UpdateLevel(`l`)`

`For all` $m = 1$ `To` M_l `Do`

$$\mathbf{Q}(G^s_{l,m}, t) \xrightarrow{\mathcal{H}^{(\Delta t_l)}} \mathbf{Q}(G_{l,m}, t + \Delta t_l)\ , \mathbf{F}^n(\bar{G}_{l,m}, t)$$

Numerical update

Time-explicit conservative finite volume scheme

$$\mathcal{H}^{(\Delta t)}: \ \mathbf{Q}_{jk}(t+\Delta t) = \mathbf{Q}_{jk}(t) - \frac{\Delta t}{\Delta x_1}\left(\mathbf{F}^1_{j+\frac{1}{2},k} - \mathbf{F}^1_{j-\frac{1}{2},k}\right) - \frac{\Delta t}{\Delta x_2}\left(\mathbf{F}^2_{j,k+\frac{1}{2}} - \mathbf{F}^2_{j,k-\frac{1}{2}}\right)$$

`UpdateLevel(`l`)`

  `For all` $m = 1$ `To` M_l `Do`

$$\mathbf{Q}(G^s_{l,m}, t) \stackrel{\mathcal{H}^{(\Delta t_l)}}{\longrightarrow} \mathbf{Q}(G_{l,m}, t+\Delta t_l)\ , \mathbf{F}^n(\bar{G}_{l,m}, t)$$

    `If level` $l+1$ `exists`

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

Numerical update

Time-explicit conservative finite volume scheme

$$\mathcal{H}^{(\Delta t)}: \; \mathbf{Q}_{jk}(t+\Delta t) = \mathbf{Q}_{jk}(t) - \frac{\Delta t}{\Delta x_1}\left(\mathbf{F}^1_{j+\frac{1}{2},k} - \mathbf{F}^1_{j-\frac{1}{2},k}\right) - \frac{\Delta t}{\Delta x_2}\left(\mathbf{F}^2_{j,k+\frac{1}{2}} - \mathbf{F}^2_{j,k-\frac{1}{2}}\right)$$

`UpdateLevel(`l`)`

For all $m = 1$ To M_l Do

$\mathbf{Q}(G^s_{l,m}, t) \stackrel{\mathcal{H}^{(\Delta t_l)}}{\longrightarrow} \mathbf{Q}(G_{l,m}, t+\Delta t_l)$, $\mathbf{F}^n(\bar{G}_{l,m}, t)$

If level $l > 0$

Add $\mathbf{F}^n(\partial G_{l,m}, t)$ to $\delta \mathbf{F}^{n,l}$

If level $l+1$ exists

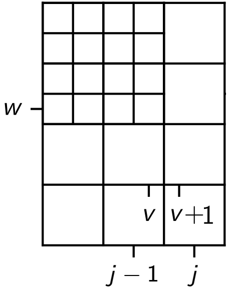
Init $\delta \mathbf{F}^{n,l+1}$ with $\mathbf{F}^n(\bar{G}_{l,m} \cap \partial G_{l+1}, t)$

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:



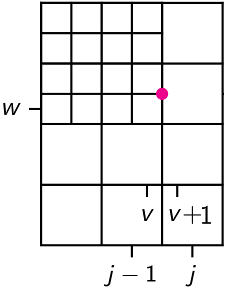
Conservative flux correction

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



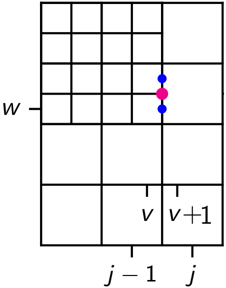
Conservative flux correction

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



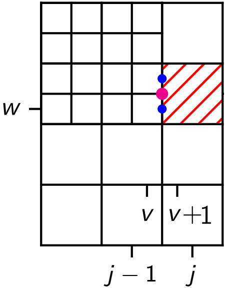
Conservative flux correction

Example: Cell j, k

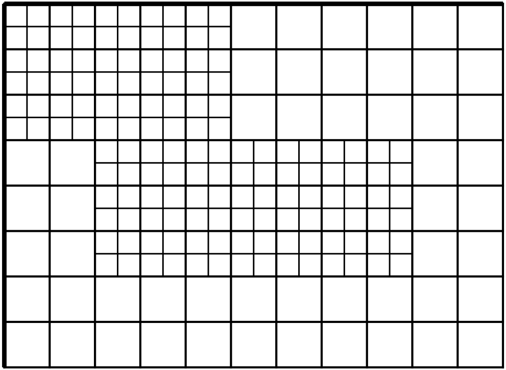
$$\check{\mathbf{Q}}^l_{jk}(t+\Delta t_l) = \mathbf{Q}^l_{jk}(t) - \frac{\Delta t_l}{\Delta x_{1,l}} \left(\mathbf{F}^{1,l}_{j+\frac{1}{2},k} - \frac{1}{r^2_{l+1}} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{F}^{1,l+1}_{v+\frac{1}{2},w+\iota}(t+\kappa \Delta t_{l+1}) \right) - \frac{\Delta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

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. $\check{\mathbf{Q}}^l_{jk}(t+\Delta t_l) := \mathbf{Q}^l_{jk}(t+\Delta t_l) + \frac{\Delta t_l}{\Delta x_{1,l}} \delta \mathbf{F}^{1,l+1}_{j-\frac{1}{2},k}$

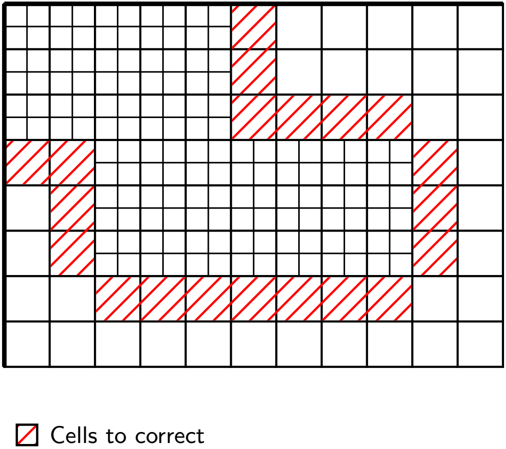


Conservative flux correction II



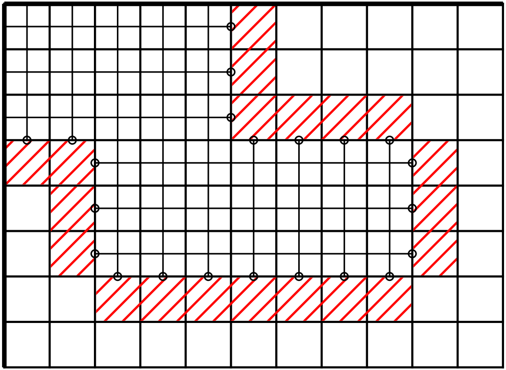
Conservative flux correction II

- Level l cells needing correction $(G_{l+1}^{r_{l+1}} \setminus G_{l+1}) \cap G_l$



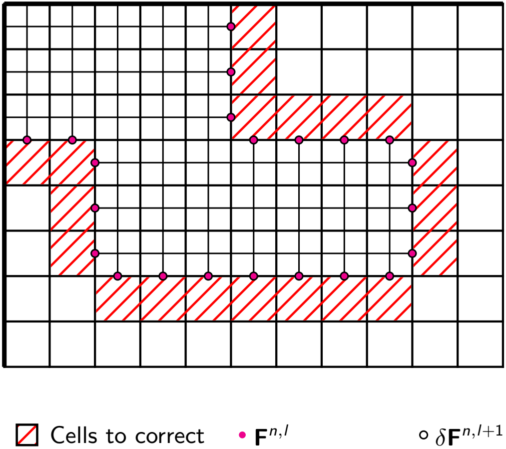
Conservative flux correction II

- Level l cells needing correction $(G_{l+1}^{r_{l+1}} \backslash G_{l+1}) \cap G_l$
- Corrections $\delta \mathbf{F}^{n,l+1}$ stored on level $l+1$ along ∂G_{l+1} (lower-dimensional data coarsened by r_{l+1})



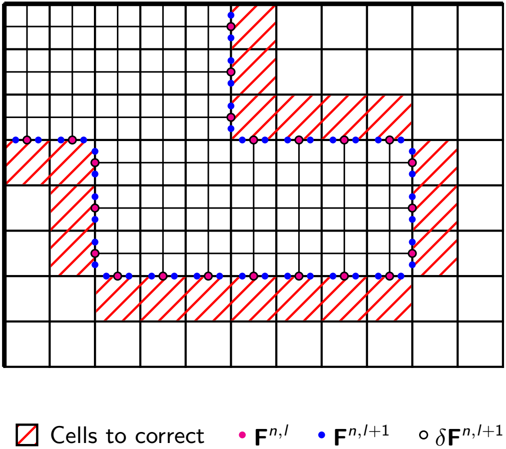
Conservative flux correction II

- Level l cells needing correction $(G_{l+1}^{r_{l+1}} \backslash G_{l+1}) \cap G_l$
- Corrections $\delta \mathbf{F}^{n,l+1}$ stored on level $l+1$ along ∂G_{l+1} (lower-dimensional data coarsened by r_{l+1})
- Init $\delta \mathbf{F}^{n,l+1}$ with level l fluxes $\mathbf{F}^{n,l}(\bar{G}_l \cap \partial G_{l+1})$



Conservative flux correction II

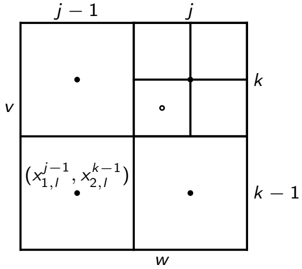
- Level l cells needing correction $(G_{l+1}^{r_{l+1}} \backslash G_{l+1}) \cap G_l$
- Corrections $\delta\mathbf{F}^{n,l+1}$ stored on level $l+1$ along ∂G_{l+1} (lower-dimensional data coarsened by r_{l+1})
- Init $\delta\mathbf{F}^{n,l+1}$ with level l fluxes $\mathbf{F}^{n,l}(\bar{G}_l \cap \partial G_{l+1})$
- Add level $l+1$ fluxes $\mathbf{F}^{n,l+1}(\partial G_{l+1})$ to $\delta\mathbf{F}^{n,l}$



Level transfer operators

Conservative averaging (restriction):
Replace cells on level l covered by level $l+1$, i.e. $G_l \cap G_{l+1}$, by

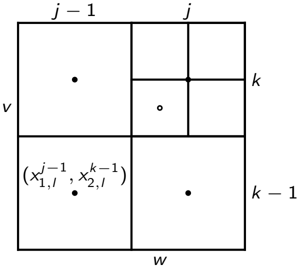
$$\hat{\mathbf{Q}}^l_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$



Level transfer operators

Conservative averaging (restriction):
Replace cells on level l covered by level $l+1$, i.e. $G_l \cap G_{l+1}$, by

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



Bilinear interpolation (prolongation):

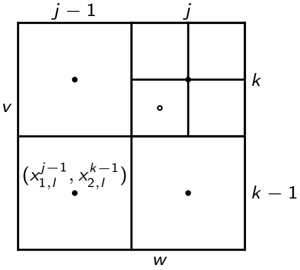
$$\check{\mathbf{Q}}^{l+1}_{vw} := (1-f_1)(1-f_2)\,\mathbf{Q}^l_{j-1,k-1} + f_1(1-f_2)\,\mathbf{Q}^l_{j,k-1} + (1-f_1)f_2\,\mathbf{Q}^l_{j-1,k} + f_1 f_2\,\mathbf{Q}^l_{jk}$$

with factors $f_1 := \dfrac{x^v_{1,l+1} - x^{j-1}_{1,l}}{\Delta x_{1,l}}$, $f_2 := \dfrac{x^w_{2,l+1} - x^{k-1}_{2,l}}{\Delta x_{2,l}}$ derived from the spatial coordinates of the cell centers $(x^{j-1}_{1,l}, x^{k-1}_{2,l})$ and $(x^v_{1,l+1}, x^w_{2,l+1})$.

Level transfer operators

Conservative averaging (restriction):
Replace cells on level l covered by level $l+1$, i.e. $G_l \cap G_{l+1}$, by

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



Bilinear interpolation (prolongation):

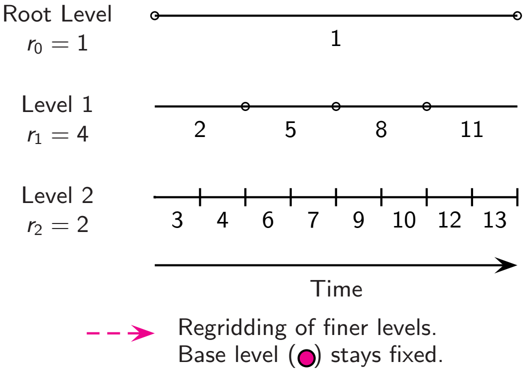
$$\check{\mathbf{Q}}^{l+1}_{vw} := (1-f_1)(1-f_2)\,\mathbf{Q}^l_{j-1,k-1} + f_1(1-f_2)\,\mathbf{Q}^l_{j,k-1} + (1-f_1)f_2\,\mathbf{Q}^l_{j-1,k} + f_1 f_2\,\mathbf{Q}^l_{jk}$$

with factors $f_1 := \dfrac{x^v_{1,l+1} - x^{j-1}_{1,l}}{\Delta x_{1,l}}$, $\quad f_2 := \dfrac{x^w_{2,l+1} - x^{k-1}_{2,l}}{\Delta x_{2,l}}$ derived from the spatial coordinates of the cell centers $(x^{j-1}_{1,l}, x^{k-1}_{2,l})$ and $(x^v_{1,l+1}, x^w_{2,l+1})$.

For boundary conditions on $\tilde{I}^s_l$: linear time interpolation

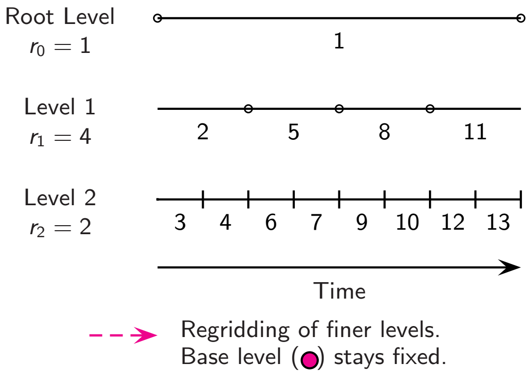
$$\tilde{\mathbf{Q}}^{l+1}(t+\kappa\Delta t_{l+1}) := \left(1-\frac{\kappa}{r_{l+1}}\right)\check{\mathbf{Q}}^{l+1}(t) + \frac{\kappa}{r_{l+1}}\check{\mathbf{Q}}^{l+1}(t+\Delta t_l) \quad \text{for } \kappa = 0,\dots r_{l+1}$$

Recursive integration order



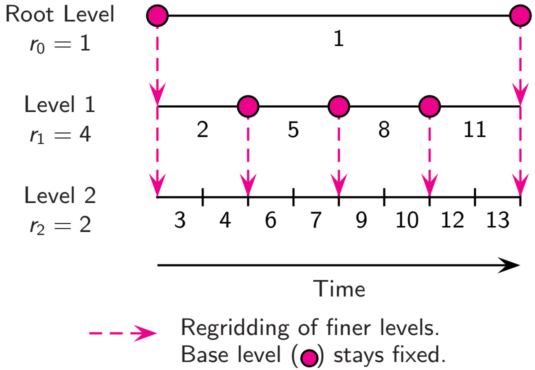
Recursive integration order

- Space-time interpolation of coarse data to set I_l^s, $l > 0$



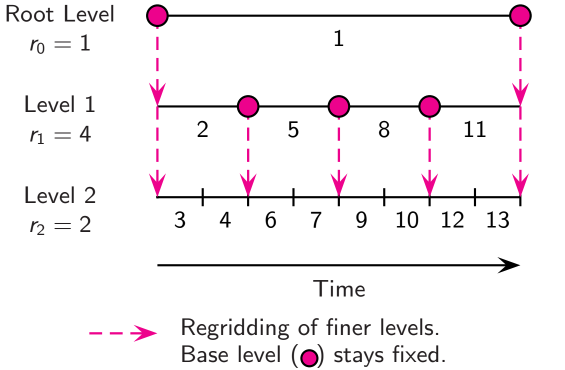
Recursive integration order

- Space-time interpolation of coarse data to set I_l^s, $l > 0$
- Regridding:
  - Creation of new grids, copy existing cells on level $l > 0$



Recursive integration order

- Space-time interpolation of coarse data to set I_l^s, $l > 0$
- Regridding:
  - Creation of new grids, copy existing cells on level $l > 0$
  - Spatial interpolation to initialize new cells on level $l > 0$



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 Q^l(t)


      UpdateLevel(l)
      If level l+1 exists?
          Set ghost cells of Q^l(t+Δt_l)
          AdvanceLevel(l+1)


      t := t + Δt_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 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
```

- 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)
```

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

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

Regridding algorithm

```
Regrid(l) - Regrid all levels ι > l

  For ι = l_f Downto l Do
      Flag N^ι according to Q^ι(t)
```

Regridding algorithm

```
Regrid(l) - Regrid all levels ι > l

   For ι = l_f Downto l Do
       Flag N^ι according to Q^ι(t)
```

- Refinement flags: $N^l := \bigcup_m N(\partial G_{l,m})$

Regridding algorithm

```
Regrid(l) - Regrid all levels ι > l

   For ι = l_f Downto l Do
       Flag N^ι according to Q^ι(t)
       If level ι+1 exists?
           Flag N^ι below Ğ^{ι+2}
```

- Refinement flags: $N^l := \bigcup_m N(\partial G_{l,m})$
- Activate flags below higher levels

Regridding algorithm

```
Regrid(l) - Regrid all levels ι > l

  For ι = l_f Downto l Do
      Flag N^ι according to Q^ι(t)
      If level ι+1 exists?
          Flag N^ι below Ğ^{ι+2}
      Flag buffer zone on N^ι
```

- Refinement flags: $N^l := \bigcup_m N(\partial G_{l,m})$
- Activate flags below higher levels
- Flag buffer cells of $b > \kappa_r$ cells, κ_r steps between calls of `Regrid(l)`

Regridding algorithm

```
Regrid(l) - Regrid all levels ι > l

  For ι = l_f Downto l Do
      Flag N^ι according to Q^ι(t)
      If level ι+1 exists?
          Flag N^ι below Ğ^(ι+2)
      Flag buffer zone on N^ι
      Generate Ğ^(ι+1) from N^ι
```

- Refinement flags: $N^l := \bigcup_m N(\partial G_{l,m})$
- Activate flags below higher levels
- Flag buffer cells of $b > \kappa_r$ cells, κ_r steps between calls of `Regrid(l)`
- Special cluster algorithm

Regridding algorithm

```
Regrid(l) - Regrid all levels ι > l

For ι = l_f Downto l Do
    Flag N^ι according to Q^ι(t)
    If level ι+1 exists?
        Flag N^ι below Ğ^{ι+2}
    Flag buffer zone on N^ι
    Generate Ğ^{ι+1} from N^ι
Ğ_l := G_l
For ι = l To l_f Do
    CĞ_ι := G_0 \ Ğ_ι
    Ğ_{ι+1} := Ğ_{ι+1} \ CĞ_ι^1
```

- Refinement flags: $N^l := \bigcup_m N(\partial G_{l,m})$
- Activate flags below higher levels
- Flag buffer cells of $b > \kappa_r$ cells, κ_r steps between calls of Regrid(l)
- Special cluster algorithm
- Use complement operation to ensure proper nesting condition

Regridding algorithm

```
Regrid(l) - Regrid all levels ι > l

  For ι = l_f Downto l Do
      Flag N^ι according to Q^ι(t)
      If level ι+1 exists?
          Flag N^ι below Ğ^{ι+2}
      Flag buffer zone on N^ι
      Generate Ğ^{ι+1} from N^ι
  Ğ_l := G_l
  For ι = l To l_f Do
      CĞ_ι := G_0 \ Ğ_ι
      Ğ_{ι+1} := Ğ_{ι+1} \ CĞ_ι^1

  Recompose(l)
```

- Refinement flags: $N^l := \bigcup_m N(\partial G_{l,m})$
- Activate flags below higher levels
- Flag buffer cells of $b > \kappa_r$ cells, κ_r steps between calls of `Regrid(l)`
- Special cluster algorithm
- Use complement operation to ensure proper nesting condition

Recomposition of data

`Recompose(`l`) - Reorganize all levels` $\iota > l$

`For` $\iota = l + 1$ `To` $l_f + 1$ `Do`

- Creates max. 1 level above l_f, but can remove multiple level if $\check{G}_\iota$ empty (no coarsening!)

Recomposition of data

`Recompose(`l`) - Reorganize all levels` $\iota > l$

`For` $\iota = l+1$ `To` $l_f + 1$ `Do`
`Interpolate` $\mathbf{Q}^{\iota-1}(t)$ `onto` $\check{\mathbf{Q}}^{\iota}(t)$

- Creates max. 1 level above l_f, but can remove multiple level if $\check{G}_\iota$ empty (no coarsening!)
- Use spatial interpolation on entire data $\check{\mathbf{Q}}^{\iota}(t)$

Recomposition of data

```
Recompose(l) - Reorganize all levels ι > l

  For ι = l + 1 To l_f + 1 Do
      Interpolate Q^{ι−1}(t) onto Q̌^ι(t)
      Copy Q^ι(t) onto Q̌^ι(t)
```

- Creates max. 1 level above l_f, but can remove multiple level if $\check{G}_\iota$ empty (no coarsening!)
- Use spatial interpolation on entire data $\check{\mathbf{Q}}^\iota(t)$
- Overwrite where old data exists

Recomposition of data

`Recompose`(l) `- Reorganize all levels` $\iota > l$

`For` $\iota = l + 1$ `To` $l_f + 1$ `Do`
  `Interpolate` $\mathbf{Q}^{\iota-1}(t)$ `onto` $\check{\mathbf{Q}}^{\iota}(t)$
  `Copy` $\mathbf{Q}^{\iota}(t)$ `onto` $\check{\mathbf{Q}}^{\iota}(t)$
  `Set ghost cells of` $\check{\mathbf{Q}}^{\iota}(t)$

- Creates max. 1 level above l_f, but can remove multiple level if $\check{G}_\iota$ empty (no coarsening!)
- Use spatial interpolation on entire data $\check{\mathbf{Q}}^{\iota}(t)$
- Overwrite where old data exists
- Synchronization and physical boundary conditions

Recomposition of data

Recompose(l) - Reorganize all levels $\iota > l$

For $\iota = l+1$ To $l_f + 1$ Do
  Interpolate $\mathbf{Q}^{\iota-1}(t)$ onto $\check{\mathbf{Q}}^{\iota}(t)$
  Copy $\mathbf{Q}^{\iota}(t)$ onto $\check{\mathbf{Q}}^{\iota}(t)$
  Set ghost cells of $\check{\mathbf{Q}}^{\iota}(t)$
  $\mathbf{Q}^{\iota}(t) := \check{\mathbf{Q}}^{\iota}(t)$, $G_\iota := \check{G}_\iota$

- Creates max. 1 level above l_f, but can remove multiple level if $\check{G}_\iota$ empty (no coarsening!)
- Use spatial interpolation on entire data $\check{\mathbf{Q}}^{\iota}(t)$
- Overwrite where old data exists
- Synchronization and physical boundary conditions

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

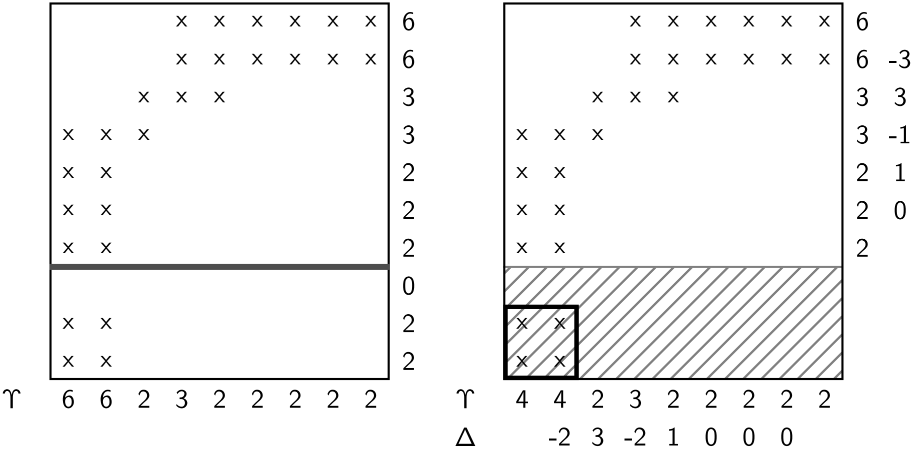
				x	x	x	x	x	x	6
				x	x	x	x	x	x	6
			x	x	x					3
	x	x	x							3
	x	x								2
	x	x								2
	x	x								2
										0
	x	x								2
	x	x								2
Υ	6	6	2	3	2	2	2	2	2	

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

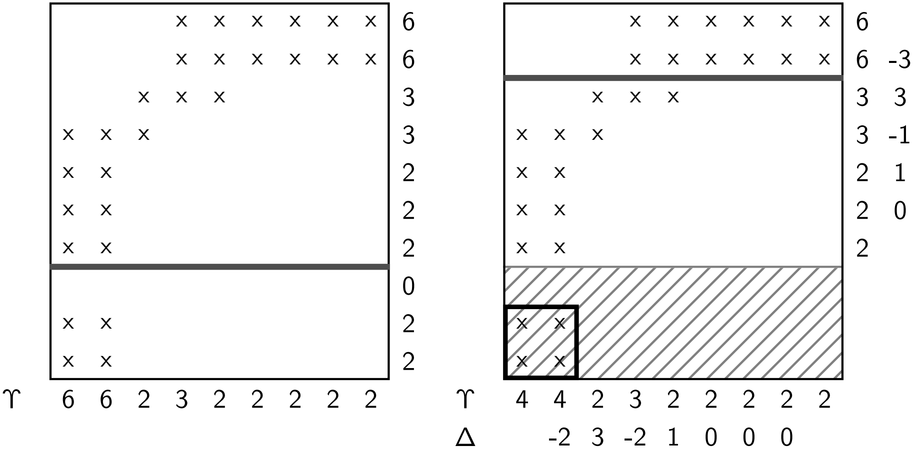


Υ Flagged cells per row/column

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]

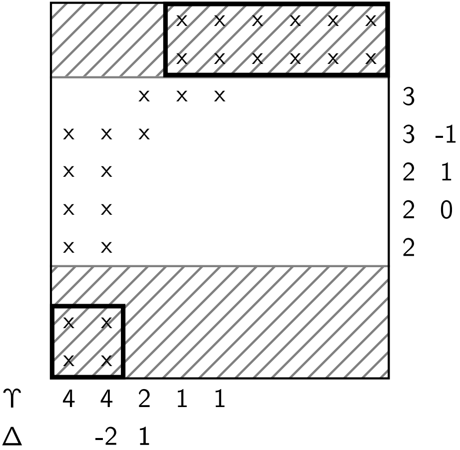
Clustering by signatures



Υ Flagged cells per row/column

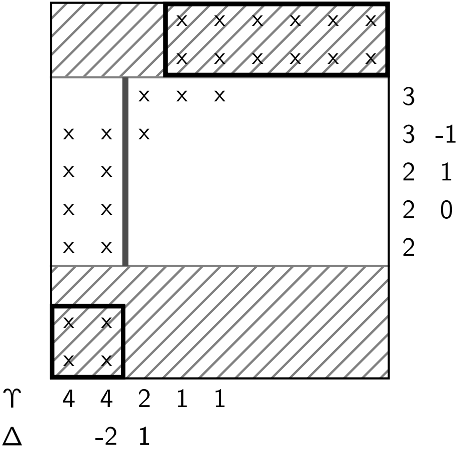
Δ Second derivative of Υ, $\Delta = \Upsilon_{\nu+1} - 2\,\Upsilon_{\nu} + \Upsilon_{\nu-1}$

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



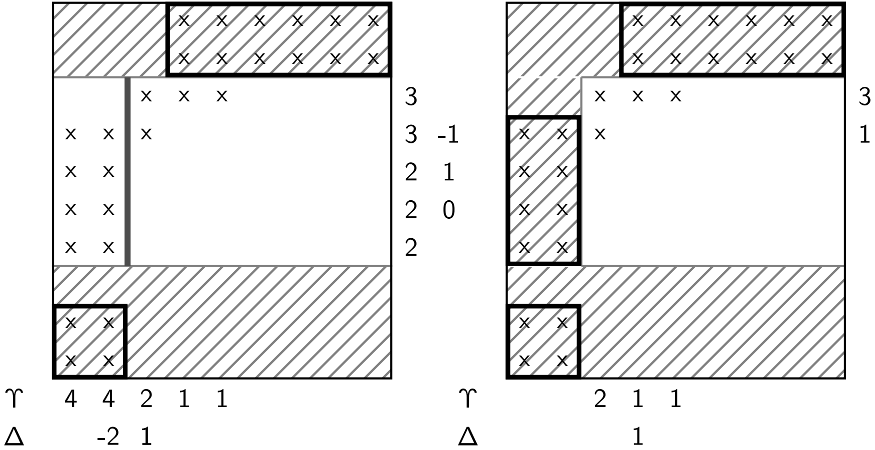
Recursive generation of $\breve{G}_{l,m}$

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



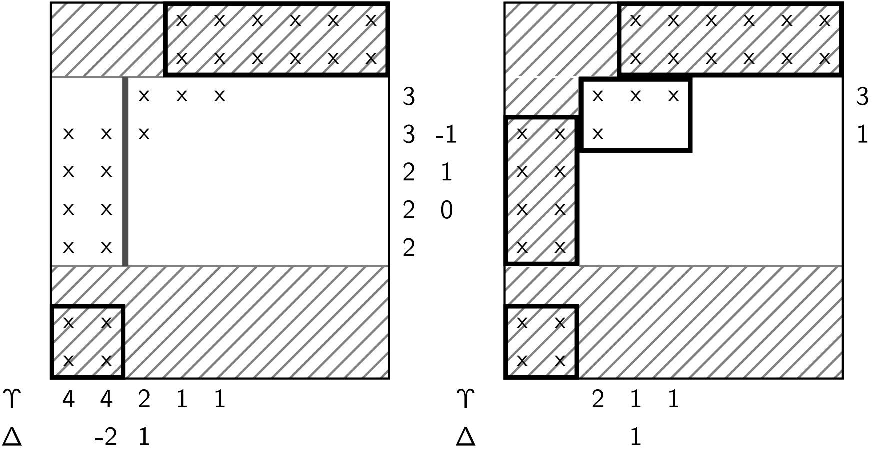
Recursive generation of $\breve{G}_{l,m}$

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 $> \eta_{tol}$

Refinement criteria

Scaled gradient of scalar quantity w

$$|w(\mathbf{Q}_{j+1,k})-w(\mathbf{Q}_{jk})| > \epsilon_w\ ,\ |w(\mathbf{Q}_{j,k+1})-w(\mathbf{Q}_{jk})| > \epsilon_w\ ,\ |w(\mathbf{Q}_{j+1,k+1})-w(\mathbf{Q}_{jk})| > \epsilon_w$$

Refinement criteria

Scaled gradient of scalar quantity w

$$|w(\mathbf{Q}_{j+1,k})-w(\mathbf{Q}_{jk})| > \epsilon_w \,,\ |w(\mathbf{Q}_{j,k+1})-w(\mathbf{Q}_{jk})| > \epsilon_w \,,\ |w(\mathbf{Q}_{j+1,k+1})-w(\mathbf{Q}_{jk})| > \epsilon_w$$

Heuristic error estimation [Berger, 1982]:
Local truncation error of scheme of order o

$$\mathbf{q}(\mathbf{x}, t+\Delta t) - \mathcal{H}^{(\Delta t)}(\mathbf{q}(\cdot, t)) = \mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

Refinement criteria

Scaled gradient of scalar quantity w

$$|w(\mathbf{Q}_{j+1,k})-w(\mathbf{Q}_{jk})| > \epsilon_w\,,\ |w(\mathbf{Q}_{j,k+1})-w(\mathbf{Q}_{jk})| > \epsilon_w\,,\ |w(\mathbf{Q}_{j+1,k+1})-w(\mathbf{Q}_{jk})| > \epsilon_w$$

Heuristic error estimation [Berger, 1982]:
Local truncation error of scheme of order o

$$\mathbf{q}(\mathbf{x}, t+\Delta t) - \mathcal{H}^{(\Delta t)}(\mathbf{q}(\cdot, t)) = \mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

For $\mathbf{q}$ smooth after 2 steps Δt

$$\mathbf{q}(\mathbf{x}, t+\Delta t) - \mathcal{H}_2^{(\Delta t)}(\mathbf{q}(\cdot, t-\Delta t)) = 2\,\mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

Refinement criteria

Scaled gradient of scalar quantity w

$$|w(\mathbf{Q}_{j+1,k})-w(\mathbf{Q}_{jk})| > \epsilon_w\ ,\ |w(\mathbf{Q}_{j,k+1})-w(\mathbf{Q}_{jk})| > \epsilon_w\ ,\ |w(\mathbf{Q}_{j+1,k+1})-w(\mathbf{Q}_{jk})| > \epsilon_w$$

Heuristic error estimation [Berger, 1982]:
Local truncation error of scheme of order o

$$\mathbf{q}(\mathbf{x}, t+\Delta t) - \mathcal{H}^{(\Delta t)}(\mathbf{q}(\cdot, t)) = \mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

For $\mathbf{q}$ smooth after 2 steps Δt

$$\mathbf{q}(\mathbf{x}, t+\Delta t) - \mathcal{H}_2^{(\Delta t)}(\mathbf{q}(\cdot, t-\Delta t)) = 2\,\mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

and after 1 step with $2\Delta t$

$$\mathbf{q}(\mathbf{x}, t+\Delta t) - \mathcal{H}^{(2\Delta t)}(\mathbf{q}(\cdot, t-\Delta t)) = 2^{o+1}\mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

Refinement criteria

Scaled gradient of scalar quantity w

$$|w(\mathbf{Q}_{j+1,k})-w(\mathbf{Q}_{jk})| > \epsilon_w \,, \; |w(\mathbf{Q}_{j,k+1})-w(\mathbf{Q}_{jk})| > \epsilon_w \,, \; |w(\mathbf{Q}_{j+1,k+1})-w(\mathbf{Q}_{jk})| > \epsilon_w$$

Heuristic error estimation [Berger, 1982]:
Local truncation error of scheme of order o

$$\mathbf{q}(\mathbf{x}, t+\Delta t) - \mathcal{H}^{(\Delta t)}(\mathbf{q}(\cdot, t)) = \mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

For $\mathbf{q}$ smooth after 2 steps Δt

$$\mathbf{q}(\mathbf{x}, t+\Delta t) - \mathcal{H}_2^{(\Delta t)}(\mathbf{q}(\cdot, t-\Delta t)) = 2\,\mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

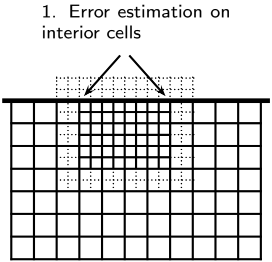
and after 1 step with $2\Delta t$

$$\mathbf{q}(\mathbf{x}, t+\Delta t) - \mathcal{H}^{(2\Delta t)}(\mathbf{q}(\cdot, t-\Delta t)) = 2^{o+1}\mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

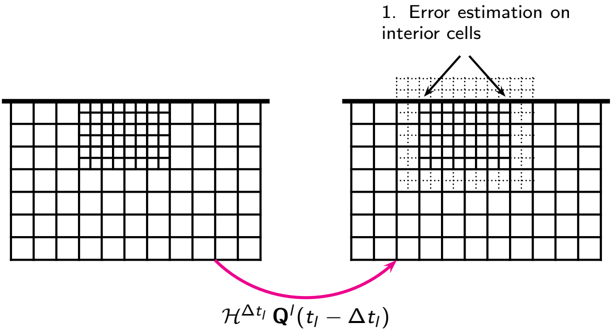
Gives

$$\mathcal{H}_2^{(\Delta t)}(\mathbf{q}(\cdot, t-\Delta t)) - \mathcal{H}^{(2\Delta t)}(\mathbf{q}(\cdot, t-\Delta t)) = (2^{o+1}-2)\mathbf{C}\Delta t^{o+1} + O(\Delta t^{o+2})$$

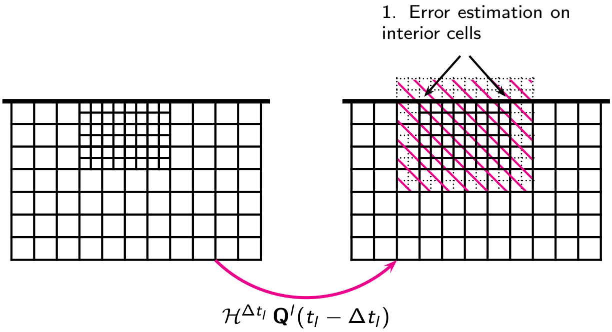
Heuristic error estimation for FV methods



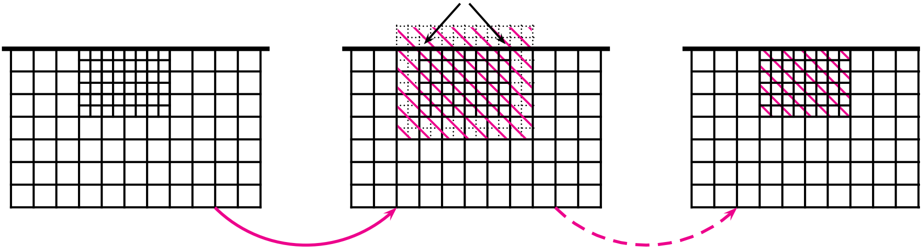
Heuristic error estimation for FV methods



Heuristic error estimation for FV methods



Heuristic error estimation for FV methods



$$\mathcal{H}^{\Delta t_l}\,\mathbf{Q}^l(t_l-\Delta t_l)$$

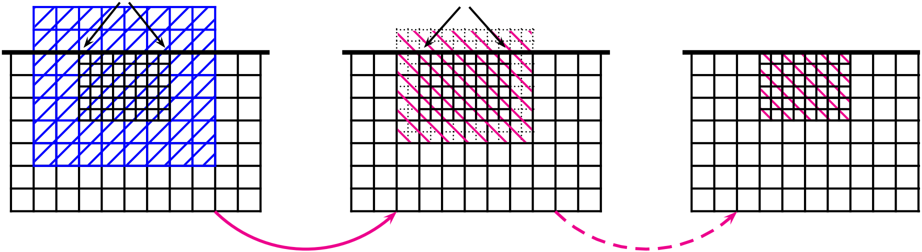
$$\begin{aligned} & \mathcal{H}^{\Delta t_l}(\mathcal{H}^{\Delta t_l}\,\mathbf{Q}^l(t_l-\Delta t_l)) \\ = \; & \mathcal{H}_2^{\Delta t_l}\,\mathbf{Q}^l(t_l-\Delta t_l) \end{aligned}$$

Heuristic error estimation for FV methods

2. Create temporary Grid coarsened by factor 2

   Initialize with fine-grid-values of preceding time step

1. Error estimation on interior cells



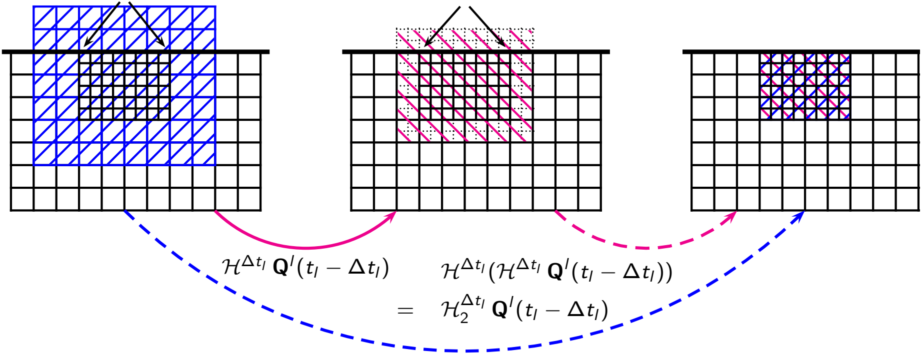
$$\mathcal{H}^{\Delta t_l}\,\mathbf{Q}^l(t_l-\Delta t_l)$$

$$\begin{aligned}&\mathcal{H}^{\Delta t_l}(\mathcal{H}^{\Delta t_l}\,\mathbf{Q}^l(t_l-\Delta t_l))\\ =\;&\mathcal{H}_2^{\Delta t_l}\,\mathbf{Q}^l(t_l-\Delta t_l)\end{aligned}$$

Heuristic error estimation for FV methods

2. Create temporary Grid coarsened by factor 2
   Initialize with fine-grid-values of preceding time step

1. Error estimation on interior cells



$$\mathcal{H}^{\Delta t_l}\,\mathbf{Q}^l(t_l-\Delta t_l)$$

$$\mathcal{H}^{\Delta t_l}(\mathcal{H}^{\Delta t_l}\,\mathbf{Q}^l(t_l-\Delta t_l)) = \mathcal{H}_2^{\Delta t_l}\,\mathbf{Q}^l(t_l-\Delta t_l)$$

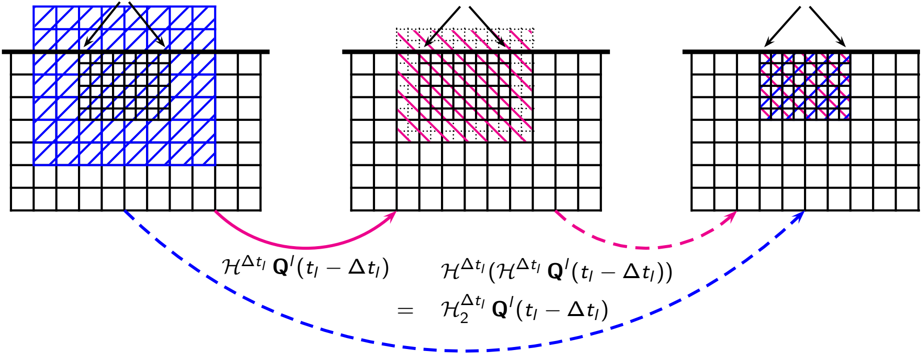
$$\mathcal{H}^{2\Delta t_l}\,\bar{\mathbf{Q}}^l(t_l-\Delta t_l)$$

Heuristic error estimation for FV methods

2. Create temporary Grid coarsened by factor 2
   Initialize with fine-grid-values of preceding time step

1. Error estimation on interior cells

3. Compare temporary solutions



$$\mathcal{H}^{\Delta t_l}\, \mathbf{Q}^l(t_l - \Delta t_l)$$

$$\mathcal{H}^{\Delta t_l}(\mathcal{H}^{\Delta t_l}\, \mathbf{Q}^l(t_l - \Delta t_l)) = \mathcal{H}_2^{\Delta t_l}\, \mathbf{Q}^l(t_l - \Delta t_l)$$

$$\mathcal{H}^{2\Delta t_l}\, \bar{\mathbf{Q}}^l(t_l - \Delta t_l)$$

Usage of heuristic error estimation

Current solution integrated tentatively 1 step with Δt_l and coarsened

$$\bar{\mathcal{Q}}(t_l + \Delta t_l) := \text{Restrict}\left(\mathcal{H}_2^{\Delta t_l}\,\mathbf{Q}^l(t_l - \Delta t_l)\right)$$

Previous solution coarsened and integrated 1 step with $2\Delta t_l$

$$\mathcal{Q}(t_l + \Delta t_l) := \mathcal{H}^{2\Delta t_l}\,\text{Restrict}\left(\mathbf{Q}^l(t_l - \Delta t_l)\right)$$

Usage of heuristic error estimation

Current solution integrated tentatively 1 step with Δt_l and coarsened

$$\bar{\mathcal{Q}}(t_l + \Delta t_l) := \text{Restrict}\left(\mathcal{H}_2^{\Delta t_l}\,\mathbf{Q}^l(t_l - \Delta t_l)\right)$$

Previous solution coarsened and integrated 1 step with $2\Delta t_l$

$$\mathcal{Q}(t_l + \Delta t_l) := \mathcal{H}^{2\Delta t_l}\,\text{Restrict}\left(\mathbf{Q}^l(t_l - \Delta t_l)\right)$$

Local error estimation of scalar quantity w

$$\tau_{jk}^w := \frac{|w(\bar{\mathcal{Q}}_{jk}(t + \Delta t)) - w(\mathcal{Q}_{jk}(t + \Delta t))|}{2^{o+1} - 2}$$

Usage of heuristic error estimation

Current solution integrated tentatively 1 step with Δt_l and coarsened

$$\bar{\mathcal{Q}}(t_l + \Delta t_l) := \text{Restrict}\left(\mathcal{H}_2^{\Delta t_l}\, \mathbf{Q}^l(t_l - \Delta t_l)\right)$$

Previous solution coarsened and integrated 1 step with $2\Delta t_l$

$$\mathcal{Q}(t_l + \Delta t_l) := \mathcal{H}^{2\Delta t_l}\, \text{Restrict}\left(\mathbf{Q}^l(t_l - \Delta t_l)\right)$$

Local error estimation of scalar quantity w

$$\tau_{jk}^w := \frac{|w(\bar{\mathcal{Q}}_{jk}(t + \Delta t)) - w(\mathcal{Q}_{jk}(t + \Delta t))|}{2^{o+1} - 2}$$

In practice [Deiterding, 2003] use

$$\frac{\tau_{jk}^w}{\max(|w(\mathcal{Q}_{jk}(t + \Delta t))|, S_w)} > \eta_w^r$$

Outline

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]

Parallelization strategies

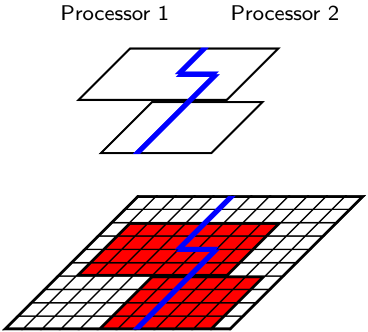
Decomposition of the hierarchical data

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

Parallelization strategies

Decomposition of the hierarchical data

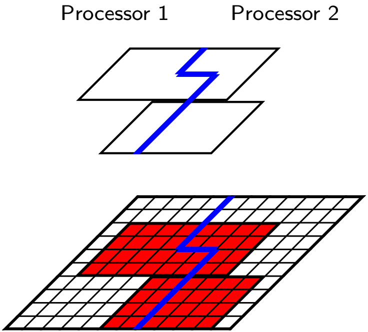
- Distribution of each grid
- Separate distribution of each level, cf. [Rendleman et al., 2000]
- Rigorous domain decomposition
  - Data of all levels resides on same node



Parallelization strategies

Decomposition of the hierarchical data

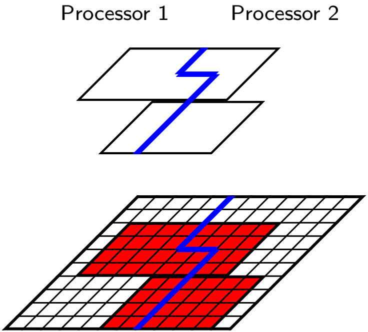
- Distribution of each grid
- Separate distribution of each level, cf. [Rendleman et al., 2000]
- Rigorous domain decomposition
  - Data of all levels resides on same node
  - Grid hierarchy defines unique "floor-plan"



Parallelization strategies

Decomposition of the hierarchical data

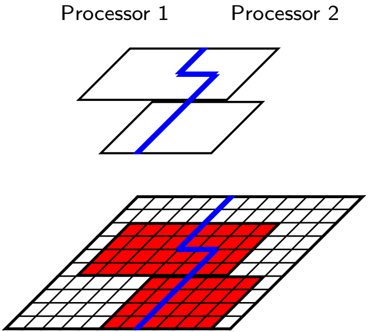
- Distribution of each grid
- Separate distribution of each level, cf. [Rendleman et al., 2000]
- Rigorous domain decomposition
  - Data of all levels resides on same node
  - Grid hierarchy defines unique "floor-plan"
  - Redistribution of data blocks during reorganization of hierarchical data



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



Rigorous domain decomposition formalized

Parallel machine with P identical nodes. P non-overlapping portions G_0^p, $p = 1, \ldots, P$ as

$$G_0 = \bigcup_{p=1}^{P} G_0^p \quad \text{with} \quad G_0^p \cap G_0^q = \emptyset \;\; \text{for } p \neq q$$

Rigorous domain decomposition formalized

Parallel machine with P identical nodes. P non-overlapping portions G_0^p, $p = 1, \ldots, P$ as

$$G_0 = \bigcup_{p=1}^{P} G_0^p \quad \text{with} \quad G_0^p \cap G_0^q = \emptyset \;\text{ for } p \neq q$$

Higher level domains G_l follow decomposition of root level

$$G_l^p := G_l \cap G_0^p$$

Rigorous domain decomposition formalized

Parallel machine with P identical nodes. P non-overlapping portions G_0^p, $p = 1, \ldots, P$ as

$$G_0 = \bigcup_{p=1}^{P} G_0^p \quad \text{with} \quad G_0^p \cap G_0^q = \emptyset \;\text{ for } p \neq q$$

Higher level domains G_l follow decomposition of root level

$$G_l^p := G_l \cap G_0^p$$

With $\mathcal{N}_l(\cdot)$ denoting number of cells, we estimate the workload as

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right]$$

Rigorous domain decomposition formalized

Parallel machine with P identical nodes. P non-overlapping portions G_0^p, $p = 1, \dots, P$ as

$$G_0 = \bigcup_{p=1}^{P} G_0^p \quad \text{with} \quad G_0^p \cap G_0^q = \emptyset \ \text{ for } p \neq q$$

Higher level domains G_l follow decomposition of root level

$$G_l^p := G_l \cap G_0^p$$

With $\mathcal{N}_l(\cdot)$ denoting number of cells, we estimate the workload as

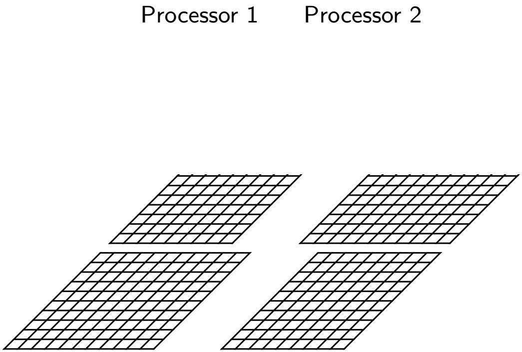
$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right]$$

Equal work distribution necessitates

$$\mathcal{L}^p := \frac{P \cdot \mathcal{W}(G_0^p)}{\mathcal{W}(G_0)} \approx 1 \quad \text{for all } p = 1, \dots, P$$

[Deiterding, 2005]

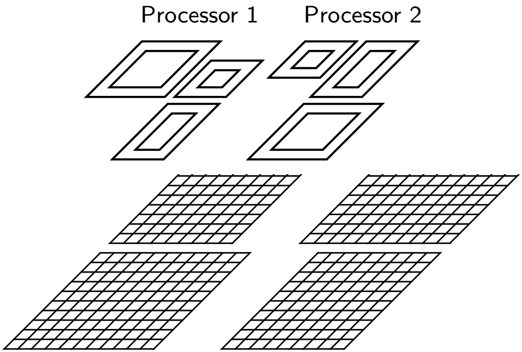
Ghost cell setting



Ghost cell values:

- Interpolation
- Local synchronization
- Parallel synchronization
- Physical boundary

Ghost cell setting



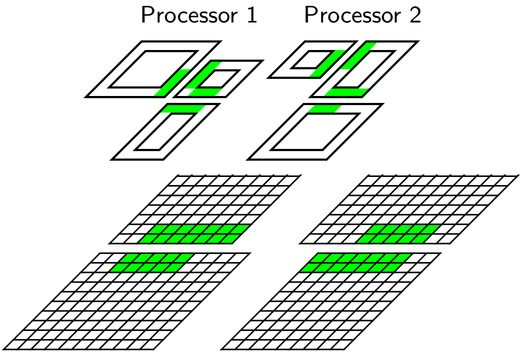
Ghost cell values:

Interpolation	Parallel synchronization
Local synchronization	Physical boundary

Ghost cell setting

Local synchronization

$$\tilde{S}^{s,p}_{l,m} = \tilde{G}^{s,p}_{l,m} \cap G^p_l$$



Ghost cell values:

- Interpolation
- Local synchronization
- Parallel synchronization
- Physical boundary

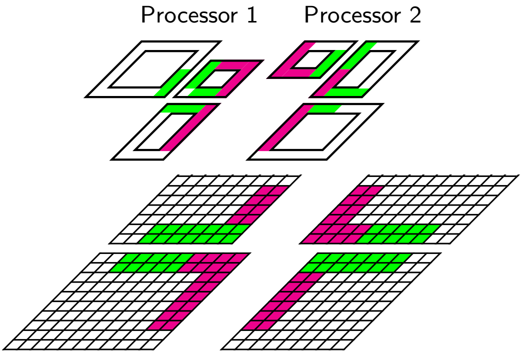
Ghost cell setting

Local synchronization

$$\tilde{S}^{s,p}_{l,m} = \tilde{G}^{s,p}_{l,m} \cap G^p_l$$

Parallel synchronization

$$\tilde{S}^{s,q}_{l,m} = \tilde{G}^{s,p}_{l,m} \cap G^q_l, q \neq p$$



Ghost cell values:

- Interpolation
- Local synchronization
- Parallel synchronization
- Physical boundary

Ghost cell setting

Local synchronization

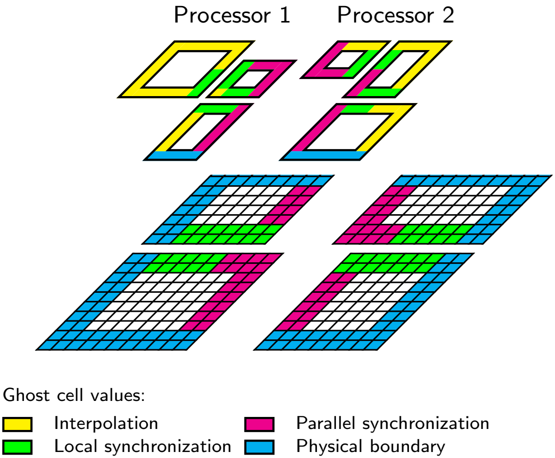
$$\tilde{S}^{s,p}_{l,m} = \tilde{G}^{s,p}_{l,m} \cap G^p_l$$

Parallel synchronization

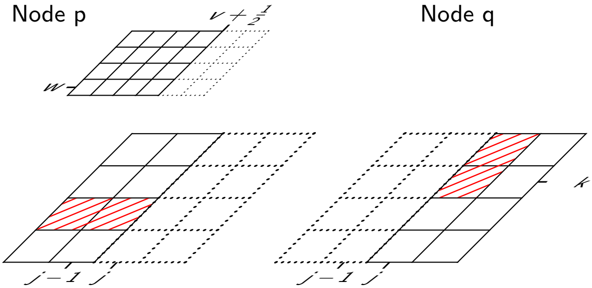
$$\tilde{S}^{s,q}_{l,m} = \tilde{G}^{s,p}_{l,m} \cap G^q_l, \, q \neq p$$

Interpolation and physical boundary conditions remain strictly local

- Scheme $\mathcal{H}^{(\Delta t_l)}$ evaluated locally
- Restriction and propolongation local

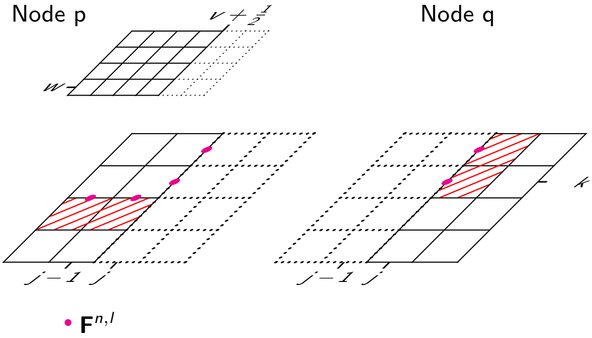


Parallel flux correction



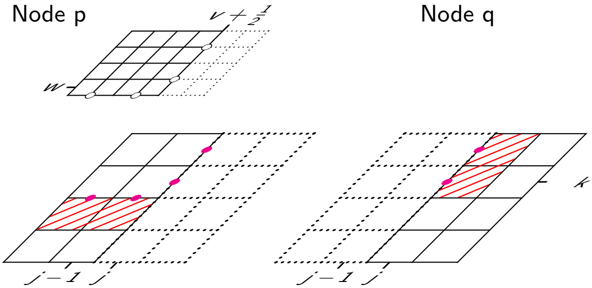
Parallel flux correction

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

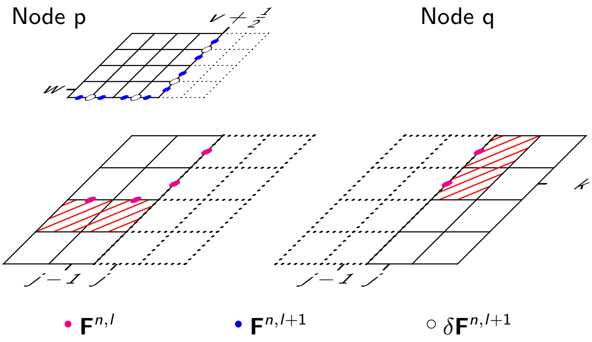
1. Strictly local: Init $\delta\mathbf{F}^{n,l+1}$ with $\mathbf{F}^{n}(\bar{G}_{l,m} \cap \partial G_{l+1}, t)$



• $\mathbf{F}^{n,l}$ ○ $\delta\mathbf{F}^{n,l+1}$

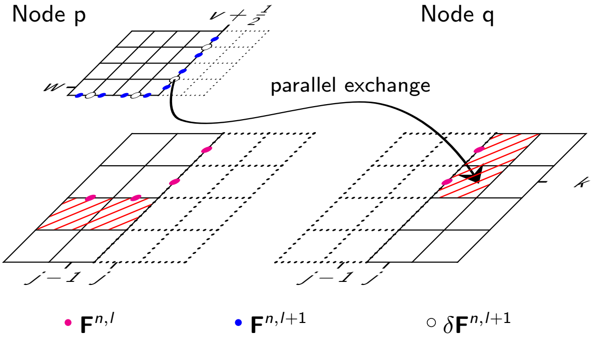
Parallel flux correction

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 $\delta\mathbf{F}^{l+1}$



The recursive algorithm in parallel

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 $\delta\mathbf{F}^{l+1}$
  $t := t+\Delta t_l$

UpdateLevel(l)

For all $m = 1$ To M_l Do
  $\mathbf{Q}(G^s_{l,m},t) \stackrel{\mathcal{H}^{(\Delta t_l)}}{\longrightarrow} \mathbf{Q}(G_{l,m},t+\Delta t_l)\;, \mathbf{F}^n(\bar{G}_{l,m},t)$
  If level $l>0$
    Add $\mathbf{F}^n(\partial G_{l,m},t)$ to $\delta\mathbf{F}^{n,l}$
  If level $l+1$ exists
    Init $\delta\mathbf{F}^{n,l+1}$ with $\mathbf{F}^n(\bar{G}_{l,m}\cap\partial G_{l+1},t)$

The recursive algorithm in parallel

```
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
```

```
UpdateLevel(l)

  For all m = 1 To M_l Do
      Q(G^s_{l,m}, t) --H^(Δt_l)--> Q(G_{l,m}, t + Δt_l), F^n(Ḡ_{l,m}, t)
      If level l > 0
          Add F^n(∂G_{l,m}, t) to δF^{n,l}
      If level l+1 exists
          Init δF^{n,l+1} with F^n(Ḡ_{l,m} ∩ ∂G_{l+1}, t)
```

- Numerical update strictly local

The recursive algorithm in parallel

```
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
```

```
UpdateLevel(l)

  For all m = 1 To M_l Do
      Q(G^s_{l,m}, t) --H^(Δt_l)--> Q(G_{l,m}, t+Δt_l) , F^n(Ḡ_{l,m}, t)
      If level l > 0
          Add F^n(∂G_{l,m}, t) to δF^{n,l}
      If level l+1 exists
          Init δF^{n,l+1} with F^n(Ḡ_{l,m} ∩ ∂G_{l+1}, t)
```

- Numerical update strictly local
- Inter-level transfer local

The recursive algorithm in parallel

```
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

UpdateLevel(l)

  For all m = 1 To M_l Do
      Q(G^s_{l,m}, t) --H^{(Δt_l)}--> Q(G_{l,m}, t + Δt_l) , F^n(Ḡ_{l,m}, t)
      If level l > 0
          Add F^n(∂G_{l,m}, t) to δF^{n,l}
      If level l+1 exists
          Init δF^{n,l+1} with F^n(Ḡ_{l,m} ∩ ∂G_{l+1}, t)
```

- Numerical update strictly local
- Inter-level transfer local
- Parallel synchronization

The recursive algorithm in parallel

```
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

UpdateLevel(l)

  For all m = 1 To M_l Do
      Q(G^s_{l,m}, t) --H^(Δt_l)--> Q(G_{l,m}, t+Δt_l), F^n(Ḡ_{l,m}, t)
      If level l > 0
          Add F^n(∂G_{l,m}, t) to δF^{n,l}
      If level l+1 exists
          Init δF^{n,l+1} with F^n(Ḡ_{l,m} ∩ ∂G_{l+1}, t)
```

- Numerical update strictly local
- Inter-level transfer local
- Parallel synchronization
- Application of $\delta\mathbf{F}^{l+1}$ on ∂G_l^q

The recursive algorithm in parallel

```
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

UpdateLevel(l)

  For all m = 1 To M_l Do
      Q(G^s_{l,m}, t) --H^{(Δt_l)}--> Q(G_{l,m}, t+Δt_l) , F^n(Ḡ_{l,m}, t)
      If level l > 0
          Add F^n(∂G_{l,m}, t) to δF^{n,l}
      If level l+1 exists
          Init δF^{n,l+1} with F^n(Ḡ_{l,m} ∩ ∂G_{l+1}, t)
```

- Numerical update strictly local
- Inter-level transfer local
- Parallel synchronization
- Application of $\delta\mathbf{F}^{l+1}$ on ∂G_l^q

Regridding algorithm in parallel

Regrid(l) - Regrid all levels $\iota > l$

For $\iota = l_f$ Downto l Do
    Flag N^ι according to $\mathbf{Q}^\iota(t)$
    If level $\iota + 1$ exists?
        Flag N^ι below $\breve{G}^{\iota+2}$
    Flag buffer zone on N^ι
    Generate $\breve{G}^{\iota+1}$ from N^ι
$\breve{G}_l := G_l$
For $\iota = l$ To l_f Do
    $C\breve{G}_\iota := G_0 \backslash \breve{G}_\iota$
    $\breve{G}_{\iota+1} := \breve{G}_{\iota+1} \backslash C\breve{G}_\iota^1$
Recompose(l)

Regridding algorithm in parallel

```
Regrid(l) - Regrid all levels ι > l

  For ι = l_f Downto l Do
      Flag N^ι according to Q^ι(t)
      If level ι+1 exists?
          Flag N^ι below Ğ^(ι+2)
      Flag buffer zone on N^ι
      Generate Ğ^(ι+1) from N^ι
  Ğ_l := G_l
  For ι = l To l_f Do
      CĞ_ι := G_0 \ Ğ_ι
      Ğ_(ι+1) := Ğ_(ι+1) \ CĞ_ι^1

  Recompose(l)
```

$\texttt{Regrid}(l)$ - Regrid all levels $\iota > l$

For $\iota = l_f$ Downto l Do
- Flag N^ι according to $\mathbf{Q}^\iota(t)$
- If level $\iota + 1$ exists?
  - Flag N^ι below $\breve{G}^{\iota+2}$
- Flag buffer zone on N^ι
- Generate $\breve{G}^{\iota+1}$ from N^ι

$\breve{G}_l := G_l$

For $\iota = l$ To l_f Do
- $C\breve{G}_\iota := G_0 \backslash \breve{G}_\iota$
- $\breve{G}_{\iota+1} := \breve{G}_{\iota+1} \backslash C\breve{G}_\iota^1$

Recompose(l)

Regridding algorithm in parallel

```
Regrid(l) - Regrid all levels ι > l

  For ι = l_f Downto l Do
      Flag N^ι according to Q^ι(t)
      If level ι+1 exists?
          Flag N^ι below Ğ^{ι+2}
      Flag buffer zone on N^ι
      Generate Ğ^{ι+1} from N^ι
  Ğ_l := G_l
  For ι = l To l_f Do
      CĞ_ι := G_0 \ Ğ_ι
      Ğ_{ι+1} := Ğ_{ι+1} \ CĞ_ι^1

  Recompose(l)
```

- Need a ghost cell overlap of b cells to ensure correct setting of refinement flags in parallel

Regridding algorithm in parallel

```
Regrid(l) - Regrid all levels ι > l

  For ι = l_f Downto l Do
      Flag N^ι according to Q^ι(t)
      If level ι + 1 exists?
          Flag N^ι below Ğ^{ι+2}
      Flag buffer zone on N^ι
      Generate Ğ^{ι+1} from N^ι
  Ğ_l := G_l
  For ι = l To l_f Do
      CĞ_ι := G_0 \ Ğ_ι
      Ğ_{ι+1} := Ğ_{ι+1} \ CĞ^1_ι

  Recompose(l)
```

- Need a ghost cell overlap of b cells to ensure correct setting of refinement flags in parallel
- Two options exist (we choose the latter):
  - Global clustering algorithm
  - Local clustering algorithm and concatenation of new lists $\breve{G}^{\iota+1}$

Regridding algorithm in parallel

```
Regrid(l) - Regrid all levels ι > l

  For ι = l_f Downto l Do
      Flag N^ι according to Q^ι(t)
      If level ι+1 exists?
          Flag N^ι below Ğ^{ι+2}
      Flag buffer zone on N^ι
      Generate Ğ^{ι+1} from N^ι
  Ğ_l := G_l
  For ι = l To l_f Do
      CĞ_ι := G_0 \ Ğ_ι
      Ğ_{ι+1} := Ğ_{ι+1} \ CĞ_ι^1

  Recompose(l)
```

- Need a ghost cell overlap of b cells to ensure correct setting of refinement flags in parallel
- Two options exist (we choose the latter):
  - Global clustering algorithm
  - Local clustering algorithm and concatenation of new lists $\breve{G}^{\iota+1}$

Regridding algorithm in parallel

```
Regrid(l) - Regrid all levels ι > l

    For ι = l_f Downto l Do
        Flag N^ι according to Q^ι(t)
        If level ι + 1 exists?
            Flag N^ι below Ğ^{ι+2}
        Flag buffer zone on N^ι
        Generate Ğ^{ι+1} from N^ι
    Ğ_l := G_l
    For ι = l To l_f Do
        CĞ_ι := G_0 \ Ğ_ι
        Ğ_{ι+1} := Ğ_{ι+1} \ CĞ_ι^1

    Recompose(l)
```

- Need a ghost cell overlap of b cells to ensure correct setting of refinement flags in parallel
- Two options exist (we choose the latter):
  - Global clustering algorithm
  - Local clustering algorithm and concatenation of new lists $\breve{G}^{\iota+1}$

Recomposition algorithm in parallel

`Recompose`(l) - `Reorganize all levels`

`For` $\iota = l + 1$ `To` $l_f + 1$ `Do`

`Interpolate` $\mathbf{Q}^{\iota-1}(t)$ `onto` $\check{\mathbf{Q}}^{\iota}(t)$

`Copy` $\mathbf{Q}^{\iota}(t)$ `onto` $\check{\mathbf{Q}}^{\iota}(t)$
`Set ghost cells of` $\check{\mathbf{Q}}^{\iota}(t)$
$\mathbf{Q}^{\iota}(t) := \check{\mathbf{Q}}^{\iota}(t)$
$G_\iota := \check{G}_\iota$

Recomposition algorithm in parallel

`Recompose(`l`) - Reorganize all levels`

`Generate` G_0^p `from` $\{G_0, ..., G_l, \breve{G}_{l+1}, ..., \breve{G}_{l_f+1}\}$
`For` $\iota = 0$ `To` $l_f + 1$ `Do`

`Interpolate` $\mathbf{Q}^{\iota-1}(t)$ `onto` $\breve{\mathbf{Q}}^{\iota}(t)$

`Copy` $\mathbf{Q}^{\iota}(t)$ `onto` $\breve{\mathbf{Q}}^{\iota}(t)$
`Set ghost cells of` $\breve{\mathbf{Q}}^{\iota}(t)$
$\mathbf{Q}^{\iota}(t) := \breve{\mathbf{Q}}^{\iota}(t)$
$G_\iota^p := \breve{G}_\iota^p$, $G_\iota := \bigcup_p G_\iota^p$

- Global redistribution can also be required when regridding higher levels and $G_0, ..., G_l$ do not change (drawback of domain decomposition)

Recomposition algorithm in parallel

```
Recompose(l) - Reorganize all levels

  Generate G_0^p from {G_0, ..., G_l, Ğ_{l+1}, ..., Ğ_{l_f+1}}
  For ι = 0 To l_f + 1 Do
      If ι > l
          Ğ_ι^p := Ğ_ι ∩ G_0^p
          Interpolate Q^{ι-1}(t) onto Q̌^ι(t)

          Copy Q^ι(t) onto Q̌^ι(t)
          Set ghost cells of Q̌^ι(t)
          Q^ι(t) := Q̌^ι(t)
      G_ι^p := Ğ_ι^p,  G_ι := ⋃_p G_ι^p
```

- Global redistribution can also be required when regridding higher levels and $G_0, \ldots, G_l$ do not change (drawback of domain decomposition)
- When $\iota > l$ do nothing special
- For $\iota \le l$, redistribute additionally

Recomposition algorithm in parallel

Recompose(l) - Reorganize all levels

    Generate G_0^p from $\{G_0, \ldots, G_l, \breve{G}_{l+1}, \ldots, \breve{G}_{l_f+1}\}$
    For $\iota = 0$ To $l_f + 1$ Do
        If $\iota > l$
            $\breve{G}_\iota^p := \breve{G}_\iota \cap G_0^p$
            Interpolate $\mathbf{Q}^{\iota-1}(t)$ onto $\breve{\mathbf{Q}}^\iota(t)$
        else
            $\breve{G}_\iota^p := G_\iota \cap G_0^p$
            If $\iota > 0$
                Copy $\delta\mathbf{F}^{n,\iota}$ onto $\delta\breve{\mathbf{F}}^{n,\iota}$
                $\delta\mathbf{F}^{n,\iota} := \delta\breve{\mathbf{F}}^{n,\iota}$
            Copy $\mathbf{Q}^\iota(t)$ onto $\breve{\mathbf{Q}}^\iota(t)$
            Set ghost cells of $\breve{\mathbf{Q}}^\iota(t)$
            $\mathbf{Q}^\iota(t) := \breve{\mathbf{Q}}^\iota(t)$
        $G_\iota^p := \breve{G}_\iota^p$, $G_\iota := \bigcup_p G_\iota^p$

- Global redistribution can also be required when regridding higher levels and $G_0, \ldots, G_l$ do not change (drawback of domain decomposition)
- When $\iota > l$ do nothing special
- For $\iota \le l$, redistribute additionally
    - Flux corrections $\delta\mathbf{F}^{n,\iota}$

Recomposition algorithm in parallel

```
Recompose(l) - Reorganize all levels

    Generate G_0^p from {G_0, ..., G_l, Ğ_{l+1}, ..., Ğ_{l_f+1}}
    For ι = 0 To l_f + 1 Do
        If ι > l
            Ğ_ι^p := Ğ_ι ∩ G_0^p
            Interpolate Q^{ι-1}(t) onto Q̌^ι(t)
        else
            Ğ_ι^p := G_ι ∩ G_0^p
            If ι > 0
                Copy δF^{n,ι} onto δF̌^{n,ι}
                δF^{n,ι} := δF̌^{n,ι}
        If ι ≥ l then κ_ι = 0 else κ_ι = 1
        For κ = 0 To κ_ι Do
            Copy Q^ι(t + κΔt_ι) onto Q̌^ι(t + κΔt_ι)
            Set ghost cells of Q̌^ι(t + κΔt_ι)
            Q^ι(t + κΔt_ι) := Q̌^ι(t + κΔt_ι)
        G_ι^p := Ğ_ι^p ,  G_ι := ⋃_p G_ι^p
```

- Global redistribution can also be required when regridding higher levels and $G_0, \ldots, G_l$ do not change (drawback of domain decomposition)
- When $\iota > l$ do nothing special
- For $\iota \le l$, redistribute additionally
  - Flux corrections $\delta \mathbf{F}^{n,\iota}$
  - Already updated time level $\mathbf{Q}^\iota(t + \kappa \Delta t_\iota)$

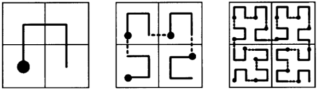
Recomposition algorithm in parallel

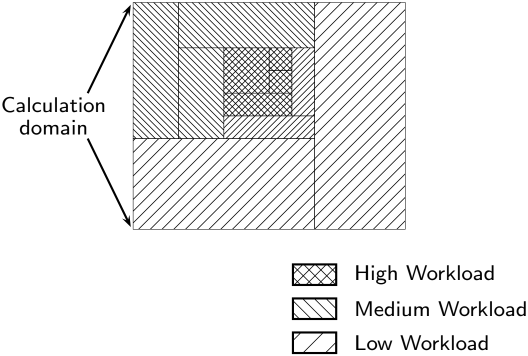
```
Recompose(l) - Reorganize all levels

  Generate G_0^p from {G_0, ..., G_l, Ğ_{l+1}, ..., Ğ_{l_f+1}}
  For ι = 0 To l_f + 1 Do
      If ι > l
          Ğ_ι^p := Ğ_ι ∩ G_0^p
          Interpolate Q^{ι-1}(t) onto Q̌^ι(t)
      else
          Ğ_ι^p := G_ι ∩ G_0^p
          If ι > 0
              Copy δF^{n,ι} onto δF̌^{n,ι}
              δF^{n,ι} := δF̌^{n,ι}
      If ι ≥ l then κ_ι = 0 else κ_ι = 1
      For κ = 0 To κ_ι Do
          Copy Q^ι(t + κΔt_ι) onto Q̌^ι(t + κΔt_ι)
          Set ghost cells of Q̌^ι(t + κΔt_ι)
          Q^ι(t + κΔt_ι) := Q̌^ι(t + κΔt_ι)
      G_ι^p := Ğ_ι^p,  G_ι := ⋃_p G_ι^p
```

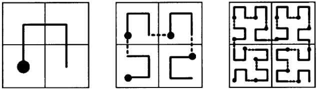
- Global redistribution can also be required when regridding higher levels and $G_0, \dots, G_l$ do not change (drawback of domain decomposition)
- When $\iota > l$ do nothing special
- For $\iota \le l$, redistribute additionally
  - Flux corrections $\delta\mathbf{F}^{n,\iota}$
  - Already updated time level $\mathbf{Q}^{\iota}(t + \kappa\Delta t_{\iota})$

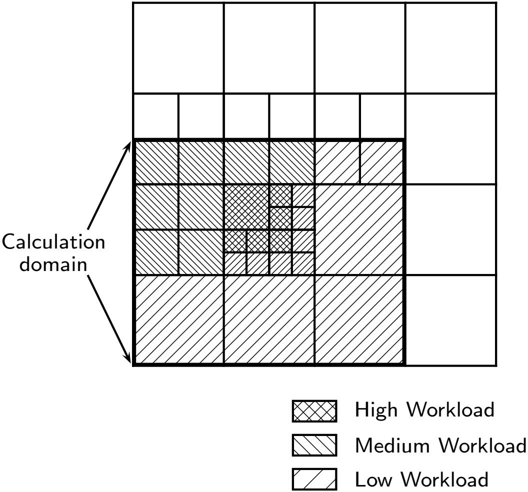
Space-filling curve algorithm



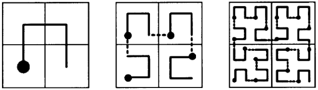


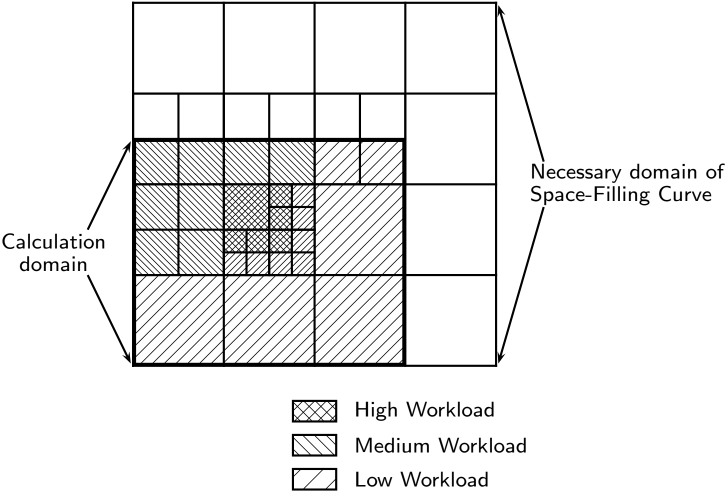
Space-filling curve algorithm



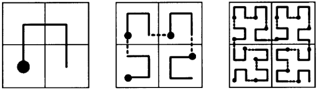


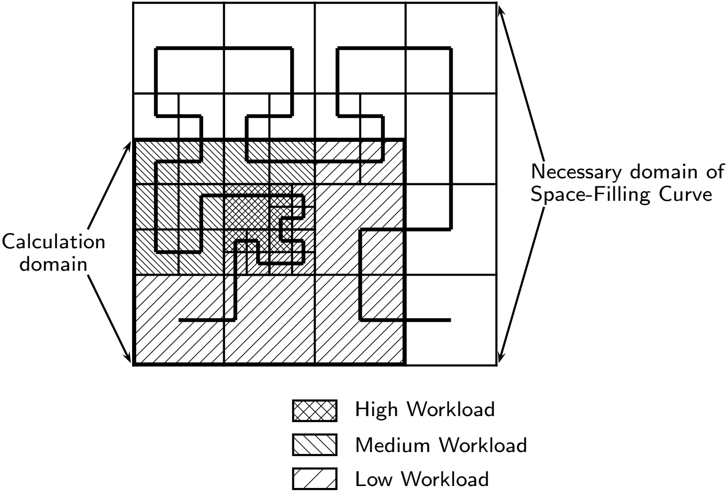
Space-filling curve algorithm



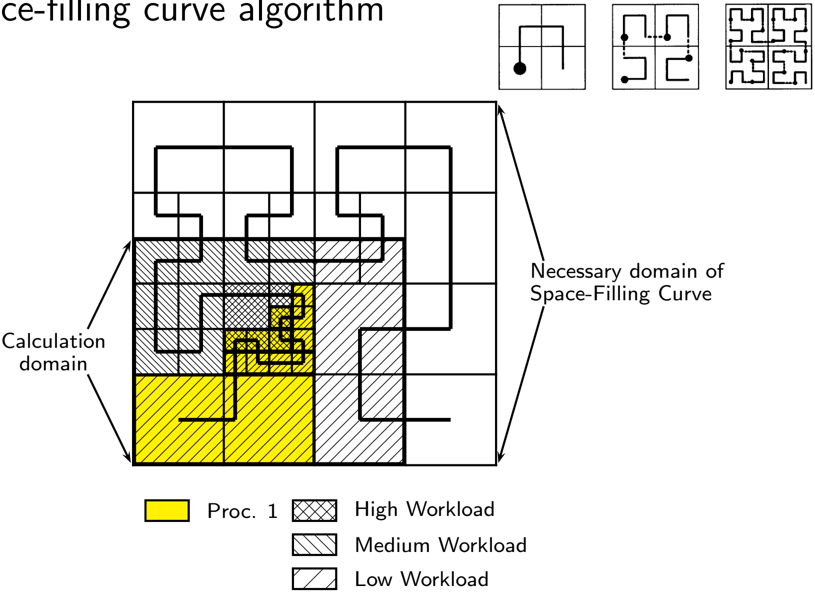


Space-filling curve algorithm

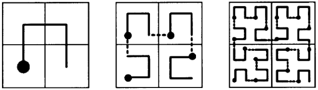


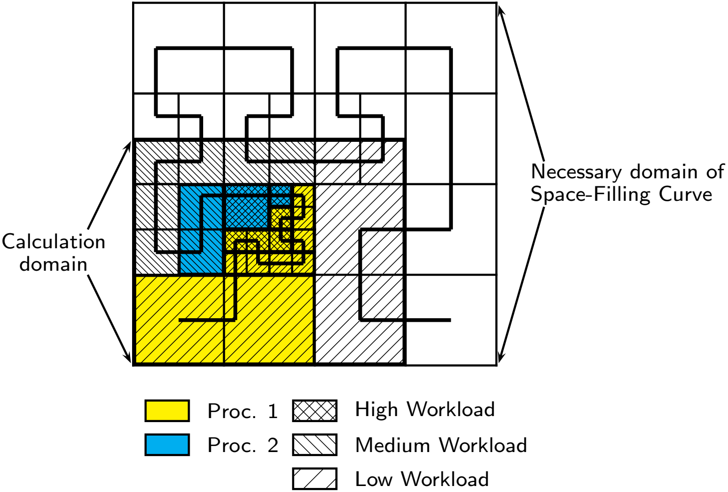


Space-filling curve algorithm

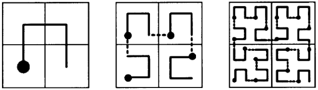


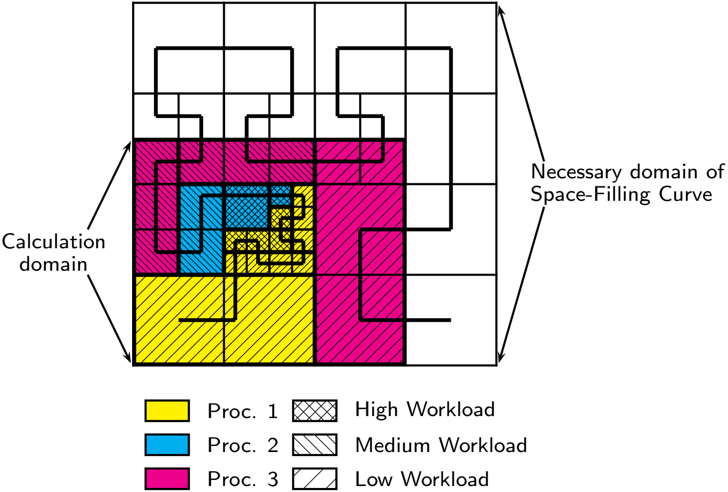
Space-filling curve algorithm





Space-filling curve algorithm





Outline

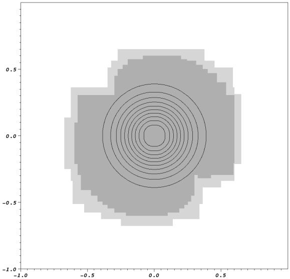
SAMR accuracy verification

Gaussian density shape

$$\rho(x_1,x_2) = 1 + e^{-\left(\frac{\sqrt{x_1^2+x_2^2}}{R}\right)^2}$$

is advected with constant velocities $u_1 = u_2 \equiv 1$, $p_0 \equiv 1$, $R = 1/4$

- Domain $[-1,1] \times [-1,1]$, periodic boundary conditions, $t_{end} = 2$
- Two levels of adaptation with $r_{1,2} = 2$, finest level corresponds to $N \times N$ uniform grid



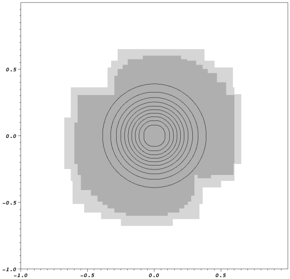
SAMR accuracy verification

Gaussian density shape

$$\rho(x_1, x_2) = 1 + e^{-\left(\frac{\sqrt{x_1^2+x_2^2}}{R}\right)^2}$$

is advected with constant velocities $u_1 = u_2 \equiv 1$, $p_0 \equiv 1$, $R = 1/4$

- Domain $[-1, 1] \times [-1, 1]$, periodic boundary conditions, $t_{end} = 2$
- Two levels of adaptation with $r_{1,2} = 2$, finest level corresponds to $N \times N$ uniform grid



Use *locally* conservative interpolation

$$\check{\mathbf{Q}}^l_{v,w} := \mathbf{Q}^l_{ij} + f_1(\mathbf{Q}^l_{i+1,j} - \mathbf{Q}^l_{i-1,j}) + f_2(\mathbf{Q}^l_{i,j+1} - \mathbf{Q}^l_{i,j-1})$$

with factor $f_1 = \dfrac{x^v_{1,l+1} - x^i_{1,l}}{2\Delta x_{1,l}}$, $f_2 = \dfrac{x^w_{2,l+1} - x^j_{2,l}}{2\Delta x_{2,l}}$ to also test flux correction

This prolongation operator is not monotonicity preserving! Only applicable to smooth problems.

SAMR accuracy verification: results

VanLeer flux vector splitting with dimensional splitting, Minmod limiter

N	Unigrid		SAMR - fixup			SAMR - no fixup		
	Error	Order	Error	Order	$\Delta\rho$	Error	Order	$\Delta\rho$
20	0.10946400							
40	0.04239430	1.369						
80	0.01408160	1.590	0.01594820		0	0.01595980		2e-5
160	0.00492945	1.514	0.00526693	1.598	0	0.00530538	1.589	2e-5
320	0.00146132	1.754	0.00156516	1.751	0	0.00163837	1.695	-1e-5
640	0.00041809	1.805	0.00051513	1.603	0	0.00060021	1.449	-6e-5

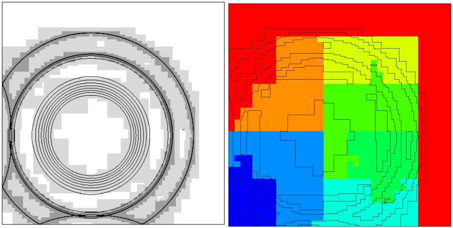
Fully two-dimensional Wave Propagation Method, Minmod limiter

N	Unigrid		SAMR - fixup			SAMR - no fixup		
	Error	Order	Error	Order	$\Delta\rho$	Error	Order	$\Delta\rho$
20	0.10620000							
40	0.04079600	1.380						
80	0.01348250	1.598	0.01536580		0	0.01538820		2e-5
160	0.00472301	1.513	0.00505406	1.604	0	0.00510499	1.592	5e-5
320	0.00139611	1.758	0.00147218	1.779	0	0.00152387	1.744	7e-5
640	0.00039904	1.807	0.00044500	1.726	0	0.00046587	1.710	6e-5

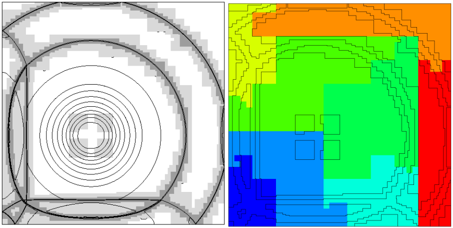
Benchmark run: blast wave in 2D

- 2D-Wave-Propagation Method with Roe's approximate solver
- Base grid 150 × 150
- 2 levels: factor 2, 4

Task [%]	P=1	P=2	P=4	P=8	P=16
Update by $\mathcal{H}^{(\cdot)}$	86.6	83.4	76.7	64.1	51.9
Flux correction	1.2	1.6	3.0	7.9	10.7
Boundary setting	3.5	5.7	10.1	15.6	18.3
Recomposition	5.5	6.1	7.4	9.9	14.0
Misc.	4.9	3.2	2.8	2.5	5.1
Time [min]	**151.9**	**79.2**	**43.4**	**23.3**	**13.9**
Efficiency [%]	**100.0**	**95.9**	**87.5**	**81.5**	**68.3**



After 38 time steps



After 79 time steps

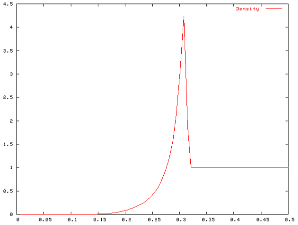
Benchmark run 2: point-explosion in 3D

- Benchmark from the Chicago workshop on AMR methods, September 2003
- Sedov explosion - energy deposition in sphere of radius 4 finest cells
- 3D-Wave-Prop. Method with hybrid Roe-HLL scheme
- Base grid 32^3
- Refinement factor $r_l = 2$
- Effective resolutions: 128^3, 256^3, 512^3, 1024^3
- Grid generation efficiency $\eta_{tol} = 85\%$
- Proper nesting enforced
- Buffer of 1 cell

Benchmark run 2: point-explosion in 3D

- Benchmark from the Chicago workshop on AMR methods, September 2003
- Sedov explosion - energy deposition in sphere of radius 4 finest cells
- 3D-Wave-Prop. Method with hybrid Roe-HLL scheme
- Base grid 32^3
- Refinement factor $r_l = 2$
- Effective resolutions: 128^3, 256^3, 512^3, 1024^3
- Grid generation efficiency $\eta_{tol} = 85\%$
- Proper nesting enforced
- Buffer of 1 cell

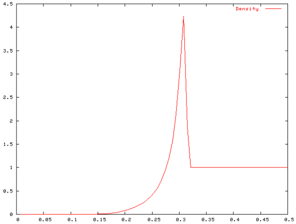
$l_{\text{max}} = 4$ solution



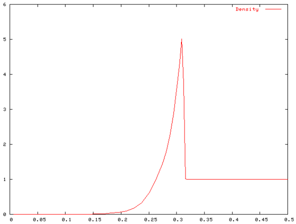
Benchmark run 2: point-explosion in 3D

- Benchmark from the Chicago workshop on AMR methods, September 2003
- Sedov explosion - energy deposition in sphere of radius 4 finest cells
- 3D-Wave-Prop. Method with hybrid Roe-HLL scheme
- Base grid 32^3
- Refinement factor $r_l = 2$
- Effective resolutions: 128^3, 256^3, 512^3, 1024^3
- Grid generation efficiency $\eta_{tol} = 85\%$
- Proper nesting enforced
- Buffer of 1 cell

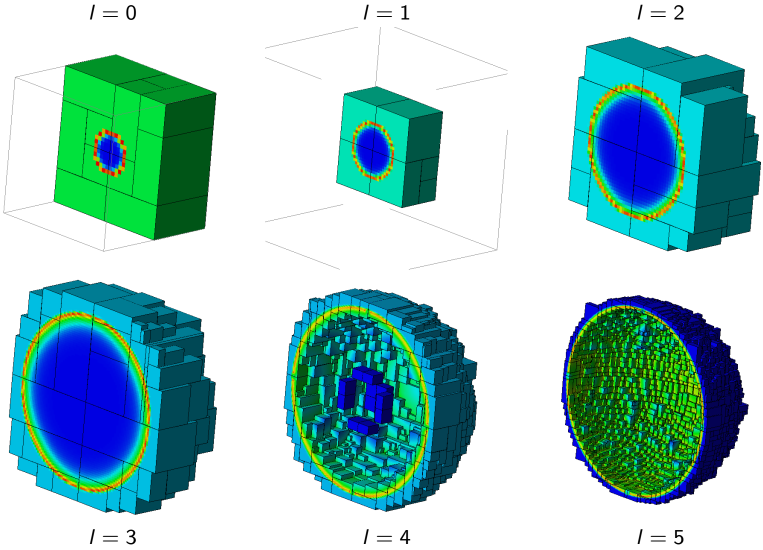
$l_{\max} = 4$ solution



$l_{\max} = 5$ solution



Benchmark run 2: visualization of refinement



Benchmark run 2: performance results

Number of grids and cells

l	$l_{\max} = 2$		$l_{\max} = 3$		$l_{\max} = 4$		$l_{\max} = 5$	
	Grids	Cells	Grids	Cells	Grids	Cells	Grids	Cells
0	28	32,768	28	32,768	33	32,768	34	32,768
1	8	32,768	14	32,768	20	32,768	20	32,768
2	63	115,408	49	116,920	43	125,680	50	125,144
3			324	398,112	420	555,744	193	572,768
4					1405	1,487,312	1,498	1,795,048
5							5,266	5,871,128
$\sum$		180,944		580,568		2,234,272		8,429,624

Benchmark run 2: performance results

Number of grids and cells

l	$l_{max}=2$		$l_{max}=3$		$l_{max}=4$		$l_{max}=5$	
	Grids	Cells	Grids	Cells	Grids	Cells	Grids	Cells
0	28	32,768	28	32,768	33	32,768	34	32,768
1	8	32,768	14	32,768	20	32,768	20	32,768
2	63	115,408	49	116,920	43	125,680	50	125,144
3			324	398,112	420	555,744	193	572,768
4					1405	1,487,312	1,498	1,795,048
5							5,266	5,871,128
$\sum$		180,944		580,568		2,234,272		8,429,624

Breakdown of CPU time on 8 nodes SGI Altix 3000 (Linux-based shared memory system)

Task [%]	$l_{max}=2$		$l_{max}=3$		$l_{max}=4$		$l_{max}=5$	
Integration	73.7		77.2		72.9		37.8	
Fixup	2.6	*46*	3.1	*58*	2.6	*42*	2.2	*45*
Boundary	10.1	*79*	6.3	*78*	5.1	*56*	6.9	*78*
Recomposition	7.4		8.0		15.1		50.4	
Clustering	0.5		0.6		0.7		1.0	
Output/Misc	5.7		4.0		3.6		1.7	
Time [min]	0.5		5.1		73.0		2100.0	
Uniform [min]	5.4		160		∼5,000		∼180,000	
Factor of AMR savings	11		31		69		86	
Time steps	15		27		52		115	

References I

[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, 1982] Berger, M. (1982). *Adaptive Mesh Refinement for Hyperbolic Differential Equations*. PhD thesis, Stanford University. Report No. STAN-CS-82-924.

[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.

[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.

References II

[Deiterding, 2003] Deiterding, R. (2003). *Parallel adaptive simulation of multi-dimensional detonation structures*. PhD thesis, Brandenburgische Technische Universität Cottbus.

[Deiterding, 2005] Deiterding, R. (2005). Construction and application of an AMR algorithm for distributed memory computers. In Plewa, T., Linde, T., and Weirs, V. G., editors, *Adaptive Mesh Refinement - Theory and Applications*, volume 41 of *Lecture Notes in Computational Science and Engineering*, pages 361–372. Springer.

[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.