Lecture 2
Structured adaptive mesh refinement

Course Block-structured Adaptive Finite Volume Methods in C++

Ralf Deiterding
University of Southampton
Engineering and the Environment
Highfield Campus, Southampton SO17 1BJ, UK
E-mail: r.deiterding@soton.ac.uk
Outline

**Meshes and adaptation**
- Adaptivity on unstructured and structured meshes
- Available SAMR software
Outline

Meshes and adaptation
   Adaptivity on unstructured and structured meshes
   Available SAMR software

The serial Berger-Colella SAMR method
   Data structures and numerical update
   Conservative flux correction
   Level transfer operators
   The basic recursive algorithm
   Block generation and flagging of cells
Outline

Meshes and adaptation
  Adaptivity on unstructured and structured meshes
  Available SAMR software

The serial Berger-Colella SAMR method
  Data structures and numerical update
  Conservative flux correction
  Level transfer operators
  The basic recursive algorithm
  Block generation and flagging of cells

Parallel SAMR method
  Domain decomposition
  A parallel SAMR algorithm
Outline

Meshes and adaptation
  Adaptivity on unstructured and structured meshes
  Available SAMR software

The serial Berger-Colella SAMR method
  Data structures and numerical update
  Conservative flux correction
  Level transfer operators
  The basic recursive algorithm
  Block generation and flagging of cells

Parallel SAMR method
  Domain decomposition
  A parallel SAMR algorithm
Elements of adaptive algorithms

- Base grid
Elements of adaptive algorithms

- Base grid
- Solver
Elements of adaptive algorithms

- Base grid
- Solver
- Error indicators
Elements of adaptive algorithms

- Base grid
- Solver
- Error indicators
- Grid manipulation
Elements of adaptive algorithms

- Base grid
- Solver
- Error indicators
- Grid manipulation
- Interpolation (restriction and prolongation)
Elements of adaptive algorithms

- Base grid
- Solver
- Error indicators
- Grid manipulation
- Interpolation (restriction and prolongation)
- Load-balancing
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
- Neighborhoods have to be stored
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
- Neighborhoods have to store

+ Geometric flexible
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
- Neighborhoods have to stored
  + Geometric flexible
  + No hanging nodes
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
- Neighborhoods have to be stored
  + Geometric flexible
  + No hanging nodes
  + Easy to implement
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
- Neighborhoods have to be stored
  + Geometric flexible
  + No hanging nodes
  + Easy to implement
  - Higher order difficult to achieve
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
- Neighborhoods have to stored
  - Geometric flexible
  - No hanging nodes
  - Easy to implement
  - Higher order difficult to achieve
  - Cell aspect ratio must be considered
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
- Neighborhoods have to stored
  + Geometric flexible
  + No hanging nodes
  + Easy to implement
  - Higher order difficult to achieve
  - Cell aspect ratio must be considered
  - Fragmented data
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
- Neighborhoods have to stored
  + Geometric flexible
  + No hanging nodes
  + Easy to implement
  - Higher order difficult to achieve
  - Cell aspect ratio must be considered
  - Fragmented data
  - Cache-reuse / vectorization nearly impossible
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
- Neighborhoods have to be stored
  + Geometric flexible
  + No hanging nodes
  + Easy to implement
  - Higher order difficult to achieve
  - Cell aspect ratio must be considered
  - Fragmented data
  - Cache-reuse / vectorization nearly impossible
  - Complex load-balancing
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
- Neighborhoods have to be stored
  + Geometric flexible
  + No hanging nodes
  + Easy to implement
    - Higher order difficult to achieve
    - Cell aspect ratio must be considered
    - Fragmented data
    - Cache-reuse / vectorization nearly impossible
    - Complex load-balancing
    - Complex synchronization
Structured mesh refinement techniques

- Block-based data of equal size
Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
- Time-step refinement
Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
- Time-step refinement
- Global index coordinate system

Structured adaptive mesh refinement
Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
- Time-step refinement
- Global index coordinate system
- Neighborhoods need not be stored
Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
- Time-step refinement
- Global index coordinate system
- Neighborhoods need not be stored

Wasted boundary space in a quad-tree
Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
- Time-step refinement
- Global index coordinate system
- Neighborhoods need not be stored
  + Numerical scheme only for single regular block necessary

Wasted boundary space in a quad-tree
Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
- Time-step refinement
- Global index coordinate system
- Neighborhoods need not be stored
- Numerical scheme only for single regular block necessary
- Easy to implement

Wasted boundary space in a quad-tree
Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
- Time-step refinement
- Global index coordinate system
- Neighborhoods need not be stored
  + Numerical scheme only for single regular block necessary
  + Easy to implement
  + Simple load-balancing

Wasted boundary space in a quad-tree
Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
- Time-step refinement
- Global index coordinate system
- Neighborhoods need not be stored
  + Numerical scheme only for single regular block necessary
  + Easy to implement
  + Simple load-balancing
  + Parent/Child relations according to tree

Wasted boundary space in a quad-tree
Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
- Time-step refinement
- Global index coordinate system
- Neighborhoods need not be stored

+ Numerical scheme only for single regular block necessary
+ Easy to implement
+ Simple load-balancing
+ Parent/Child relations according to tree

+/- Cache-reuse / vectorization only in data block

Wasted boundary space in a quad-tree
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
- Block (aka patch) based data structures
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
- Block (aka patch) based data structures
- Global index coordinate system
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
- Block (aka patch) based data structures
- Global index coordinate system
- Numerical scheme only for single patch necessary
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
- Block (aka patch) based data structures
- Global index coordinate system
- Numerical scheme only for single patch necessary
- Efficient cache-reuse / vectorization possible
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
- Block (aka patch) based data structures
- Global index coordinate system
  - Numerical scheme only for single patch necessary
  - Efficient cache-reuse / vectorization possible
  - Simple load-balancing
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
- Block (aka patch) based data structures
- Global index coordinate system
- Numerical scheme only for single patch necessary
- Efficient cache-reuse / vectorization possible
- Simple load-balancing
- Minimal synchronization overhead
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
- Block (aka patch) based data structures
- Global index coordinate system
  + Numerical scheme only for single patch necessary
  + Efficient cache-reuse / vectorization possible
  + Simple load-balancing
  + Minimal synchronization overhead
- Cells without mark are refined
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
- Block (aka patch) based data structures
- Global index coordinate system
  + Numerical scheme only for single patch necessary
  + Efficient cache-reuse / vectorization possible
  + Simple load-balancing
  + Minimal synchronization overhead
- Cells without mark are refined
- Hanging nodes unavoidable
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
- Block (aka patch) based data structures
- Global index coordinate system
- Numerical scheme only for single patch necessary
- Efficient cache-reuse / vectorization possible
- Simple load-balancing
- Minimal synchronization overhead
- Cells without mark are refined
- Hanging nodes unavoidable
- Cluster-algorithm necessary
Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
- Block (aka patch) based data structures
- Global index coordinate system
- Numerical scheme only for single patch necessary
- Efficient cache-reuse / vectorization possible
- Simple load-balancing
- Minimal synchronization overhead
- Cells without mark are refined
- Hanging nodes unavoidable
- Cluster-algorithm necessary
- Difficult to implement
Simplified structured designs

* Distributed memory parallelization fully supported if not otherwise stated. *
Simplified structured designs

*Distributed memory parallelization fully supported if not otherwise states.*

- PARAMESH (Parallel Adaptive Mesh Refinement)
  - Library based on uniform refinement blocks [MacNeice et al., 2000]
  - Both multigrid and explicit algorithms considered
  - [http://sourceforge.net/projects/paramesh](http://sourceforge.net/projects/paramesh)
Simplified structured designs

*Distributed memory parallelization fully supported if not otherwise stated.*

- **PARAMESH** (Parallel Adaptive Mesh Refinement)
  - Library based on uniform refinement blocks [MacNeice et al., 2000]
  - Both multigrid and explicit algorithms considered
  - [http://sourceforge.net/projects/paramesh](http://sourceforge.net/projects/paramesh)

- **Flash code** (AMR code for astrophysical thermonuclear flashes)
  - Built on PARAMESH
  - Solves the magneto-hydrodynamic equations with self-gravitation
  - [http://www.flash.uchicago.edu/site/flashcode](http://www.flash.uchicago.edu/site/flashcode)
Simplified structured designs

*Distributed memory parallelization fully supported if not otherwise stated.*

- **PARAMESH** (Parallel Adaptive Mesh Refinement)
  - Library based on uniform refinement blocks [MacNeice et al., 2000]
  - Both multigrid and explicit algorithms considered
  - http://sourceforge.net/projects/paramesh

- **Flash code** (AMR code for astrophysical thermonuclear flashes)
  - Built on PARAMESH
  - Solves the magneto-hydrodynamic equations with self-gravitation
  - http://www.flash.uchicago.edu/site/flashcode

- **Uintah** (AMR code for simulation of accidental fires and explosions)
  - Only explicit algorithms considered
  - FSI coupling Material Point Method and ICE Method (Implicit, Continuous fluid, Eulerian)
  - http://www.uintah.utah.edu
Simplified structured designs

*Distributed memory parallelization fully supported if not otherwise states.*

- PARAMESH (Parallel Adaptive Mesh Refinement)
  - Library based on uniform refinement blocks [MacNeice et al., 2000]
  - Both multigrid and explicit algorithms considered
  - http://sourceforge.net/projects/paramesh
- Flash code (AMR code for astrophysical thermonuclear flashes)
  - Built on PARAMESH
  - Solves the magneto-hydrodynamic equations with self-gravitation
  - http://www.flash.uchicago.edu/site/flashcode
- Uintah (AMR code for simulation of accidental fires and explosions)
  - Only explicit algorithms considered
  - FSI coupling Material Point Method and ICE Method (Implicit, Continuous fluid, Eulerian)
  - http://www.uintah.utah.edu
- DAGH/Grace [Parashar and Browne, 1997]
  - Just C++ data structures but no methods
  - All grids are aligned to bases mesh coarsened by factor 2
  - http://userweb.cs.utexas.edu/users/dagh
Systems that support general SAMR
Systems that support general SAMR

- **SAMRAI - Structured Adaptive Mesh Refinement Application Infrastructure**
  - Very mature SAMR system [Hornung et al., 2006]
  - Explicit algorithms directly supported, implicit methods through interface to Hypre package
  - Mapped geometry and some embedded boundary support
Systems that support general SAMR

- **SAMRAI** - Structured Adaptive Mesh Refinement Application Infrastructure
  - Very mature SAMR system [Hornung et al., 2006]
  - Explicit algorithms directly supported, implicit methods through interface to Hypre package
  - Mapped geometry and some embedded boundary support

- **BoxLib, AmrLib, MGLib, HGProj**
  - Berkley-Lab-AMR collection of C++ classes by J. Bell et al., 50,000 LOC [Rendleman et al., 2000]
  - Both multigrid and explicit algorithms supported
Systems that support general SAMR

- **SAMRAI - Structured Adaptive Mesh Refinement Application Infrastructure**
  - Very mature SAMR system [Hornung et al., 2006]
  - Explicit algorithms directly supported, implicit methods through interface to Hypre package
  - Mapped geometry and some embedded boundary support

- **BoxLib, AmrLib, MGLib, HGProj**
  - Berkley-Lab-AMR collection of C++ classes by J. Bell et al., 50,000 LOC [Rendleman et al., 2000]
  - Both multigrid and explicit algorithms supported

- **Chombo**
  - Redesign and extension of BoxLib by P. Colella et al.
  - Both multigrid and explicit algorithms demonstrated
  - Some embedded boundary support
  - [https://commons.lbl.gov/display/chombo](https://commons.lbl.gov/display/chombo)
Further SAMR software

- Overture (Object-oriented tools for solving PDEs in complex geometries)
  - Overlapping meshes for complex geometries by W. Henshaw et al. [Brown et al., 1997]
  - Explicit and implicit algorithms supported
  - http://www.overtureframework.org

- AMRClaw within Clawpack [Berger and LeVeque, 1998]
  - Serial 2D Fortran 77 code for the explicit Wave Propagation method with own memory management
  - http://depts.washington.edu/clawpack

- Amrita by J. Quirk
  - Only 2D explicit finite volume methods supported
  - Embedded boundary algorithm
  - http://www.amrita-cfd.org

- Cell-based Cartesian AMR: RAGE
  - Embedded boundary method
  - Explicit and implicit algorithms
  - [Gittings et al., 2008]
Further SAMR software

▶ Overture (Object-oriented tools for solving PDEs in complex geometries)
  ▶ Overlapping meshes for complex geometries by W. Henshaw et al. [Brown et al., 1997]
  ▶ Explicit and implicit algorithms supported
  ▶ http://www.overtureframework.org
Further SAMR software

- Overture (Object-oriented tools for solving PDEs in complex geometries)
  - Overlapping meshes for complex geometries by W. Henshaw et al. [Brown et al., 1997]
  - Explicit and implicit algorithms supported
  - http://www.overtureframework.org

- AMRClaw within Clawpack [Berger and LeVeque, 1998]
  - Serial 2D Fortran 77 code for the explicit Wave Propagation method with own memory management
  - http://depts.washington.edu/clawpack
Further SAMR software

▶ Overture (Object-oriented tools for solving PDEs in complex geometries)
  ▶ Overlapping meshes for complex geometries by W. Henshaw et al. [Brown et al., 1997]
  ▶ Explicit and implicit algorithms supported
  ▶ http://www.overtureframework.org

▶ AMRClaw within Clawpack [Berger and LeVeque, 1998]
  ▶ Serial 2D Fortran 77 code for the explicit Wave Propagation method with own memory management
  ▶ http://depts.washington.edu/clawpack

▶ Amrita by J. Quirk
  ▶ Only 2D explicit finite volume methods supported
  ▶ Embedded boundary algorithm
  ▶ http://www.amrita-cfd.org
Further SAMR software

- Overture (Object-oriented tools for solving PDEs in complex geometries)
  - Overlapping meshes for complex geometries by W. Henshaw et al. [Brown et al., 1997]
  - Explicit and implicit algorithms supported
  - http://www.overtureframework.org

- AMRClaw within Clawpack [Berger and LeVeque, 1998]
  - Serial 2D Fortran 77 code for the explicit Wave Propagation method with own memory management
  - http://depts.washington.edu/clawpack

- Amrita by J. Quirk
  - Only 2D explicit finite volume methods supported
  - Embedded boundary algorithm
  - http://www.amrita-cfd.org

- Cell-based Cartesian AMR: RAGE
  - Embedded boundary method
  - Explicit and implicit algorithms
  - [Gittings et al., 2008]
Outline

Meshes and adaptation
  Adaptivity on unstructured and structured meshes
  Available SAMR software

The serial Berger-Colella SAMR method
  Data structures and numerical update
  Conservative flux correction
  Level transfer operators
  The basic recursive algorithm
  Block generation and flagging of cells

Parallel SAMR method
  Domain decomposition
  A parallel SAMR algorithm
The $m$th refinement grid on level $l$

Notations:

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

Interior grid with buffer cells - $G_{l,m}$
The $m$th refinement grid on level $l$

Notations:

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

Interior grid with buffer cells - $G_{l,m}$
The $m$th refinement grid on level $l$

Notations:

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

Interior grid with buffer cells - $G_{l,m}$
The $m$th refinement grid on level $l$

Notations:
- **Boundary:** $\partial G_{l,m}$
- **Hull:**
  \[ \overline{G}_{l,m} = G_{l,m} \cup \partial G_{l,m} \]
  Complete grid with ghost cells - $G_{l,m}^\sigma$
  Interior grid with buffer cells - $G_{l,m}$
The $m$th refinement grid on level $l$

Notations:

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

Complete grid with ghost cells - $G_{l,m}^\sigma$

Interior grid with buffer cells - $G_{l,m}$
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 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 $r_0 = 1$
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]:
  \[
  \Delta x_{n,l} \approx \prod_{\kappa=l+1}^{l_{\text{max}}} r_{\kappa}
  \]
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]:
  \[
  \Delta x_{n,l} \approx \prod_{\kappa=l+1}^{l_{\text{max}}} r_\kappa
  \]

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

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

- **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: \( \Delta t_l := \frac{\Delta t_{l-1}}{r_l} \) 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]:
  \[
  \Delta x_{n,l} \cong l_{\max} \prod_{\kappa=l+1}^{l_{\max}} r_{\kappa}
  \]

- 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: \( G^1_l \subset G_{l-1} \)
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]:

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

- 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: $G^1_l \subset G_{l-1}$

- Assume a FD scheme with stencil radius $s$. Necessary data:
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]:

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

▶ 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: \( G_{l+1} \subset G_l \)

▶ Assume a FD scheme with stencil radius \( s \). Necessary data:

▶ Vector of state: \( Q^l := \bigcup_m Q(G_{l,m}^s) \)
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]:
  $$\Delta x_{n,l} \equiv \prod_{\kappa=l+1}^{l_{\max}} r_{\kappa}$$
- 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: $G_1 \subset G_{l-1}$
- Assume a FD scheme with stencil radius $s$. Necessary data:
  - Vector of state: $Q^l := \bigcup_m Q(G_{s,l,m})$
  - Numerical fluxes: $F^{n,l} := \bigcup_m F^n(\bar{G}_{l,m})$
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]:
  $$\Delta x_{n,l} \approx l_{\text{max}} \prod_{\kappa=l+1}^{l_{\text{max}}} r_{\kappa}$$

- **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**: $G_{l+1} \subset G_l$

- **Assume a FD scheme with stencil radius** $s$. Necessary data:
  - **Vector of state**: $Q^l := \bigcup_m Q(G_{l,m}^s)$
  - **Numerical fluxes**: $F_{n,l} := \bigcup_m F^n(\bar{G}_{l,m})$
  - **Flux corrections**: $\delta F_{n,l} := \bigcup_m \delta F^n(\partial G_{l,m})$
Setting of ghost cells
Setting of ghost cells
Setting of ghost cells

Synchronization with $G_i - \tilde{S}_{l,m} = \tilde{G}_{l,m} \cap G_i$
Setting of ghost cells

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

Physical boundary conditions - $\tilde{P}_{l,m} = \tilde{G}_{l,m} \setminus G_0$
Setting of ghost cells

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

Time-explicit conservative finite volume scheme

\[ \mathcal{H}^{(\Delta t)} : Q_{jk}(t+\Delta t) = Q_{jk}(t) - \frac{\Delta t}{\Delta x_1} \left( F_{1j, k+\frac{1}{2}} - F_{1j, k-\frac{1}{2}} \right) - \frac{\Delta t}{\Delta x_2} \left( F_{2j, k+\frac{1}{2}} - F_{2j, k-\frac{1}{2}} \right) \]
Numerical update

Time-explicit conservative finite volume scheme

\[ H^{(\Delta t)} : \quad Q_{jk}(t+\Delta t) = Q_{jk}(t) - \frac{\Delta t}{\Delta x_1} \left( F^1_{j+\frac{1}{2},k} - F^1_{j-\frac{1}{2},k} \right) - \frac{\Delta t}{\Delta x_2} \left( F^2_{j,k+\frac{1}{2}} - F^2_{j,k-\frac{1}{2}} \right) \]

UpdateLevel(l)

For all \( m = 1 \) To \( M_l \) Do

\[ Q(G_{l,m}, t) \xrightarrow{H^{(\Delta t_l)}} Q(G_{l,m}, t + \Delta t_l), F^n(\bar{G}_{l,m}, t) \]
Numerical update

Time-explicit conservative finite volume scheme

\[ \mathcal{H}^{(\Delta t)} : Q_{jk}(t+\Delta t) = Q_{jk}(t) - \frac{\Delta t}{\Delta x_1} \left( F_{j+\frac{1}{2},k}^1 - F_{j-\frac{1}{2},k}^1 \right) - \frac{\Delta t}{\Delta x_2} \left( F_{j,k+\frac{1}{2}}^2 - F_{j,k-\frac{1}{2}}^2 \right) \]

UpdateLevel(l)

For all \( m = 1 \) To \( M_l \) Do

\[ Q(G_{l,m}, t) \xrightarrow{\mathcal{H}^{(\Delta t_l)}} Q(G_{l,m}, t + \Delta t_l), F^n(\tilde{G}_{l,m}, t) \]

If level \( l + 1 \) exists

Init \( \delta F^{n,l+1} \) with \( F^n(\tilde{G}_{l,m} \cap \partial G_{l+1}, t) \)
Numerical update

Time-explicit conservative finite volume scheme

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

UpdateLevel(l)

For all \( m = 1 \) To \( M_l \) Do

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

If level \( l > 0 \)

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

If level \( l + 1 \) exists

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

Example: Cell $j, k$

$$\hat{Q}_{jk}^l(t + \Delta t_l) = Q_{jk}^l(t) - \Delta t_l \left( \frac{F_{1,l}^{1/l,j+\frac{1}{2},k} - 1}{r_{l+1}^2} \sum_{\kappa=0}^{r_{l+1}^{-1}} \sum_{\iota=0}^{r_{l+1}^{-1}} F_{1,l+1,v+\frac{1}{2},w+\iota}^{1/l} (t + \kappa \Delta t_{l+1}) \right)$$

$$- \Delta t_l \left( \frac{F_{2,l}^{2/l,j,k+\frac{1}{2}} - F_{2,l}^{2/l,j,k-\frac{1}{2}}}{\Delta x_{2,l}} \right)$$

Correction pass:
Conservative flux correction

Example: Cell $j, k$

$$\dot{Q}_{jk}^{l}(t + \Delta t) = Q_{jk}^{l}(t) - \frac{\Delta t}{\Delta x_{1,l}} \left( F_{1,l}^{1,l+j+\frac{1}{2},k} - \frac{1}{r_{l+1}^{2}} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} F_{1,l+1,v+\frac{1}{2},w+\iota}^{1,l+1} (t + \kappa \Delta t_{l+1}) \right)$$

$$- \frac{\Delta t}{\Delta x_{2,l}} \left( F_{1,l}^{2,l,j,k+\frac{1}{2}} - F_{1,l}^{2,l,j,k-\frac{1}{2}} \right)$$

Correction pass:

1. $\delta F_{1,l+1,j-\frac{1}{2},k}^{1,l+1} := -F_{1,l}^{1,l,j-\frac{1}{2},k}$
Conservative flux correction

Example: Cell \( j, k \)

\[
\dot{Q}^l_{jk}(t + \Delta t_l) = Q^l_{jk}(t) - \frac{\Delta t_l}{\Delta x_1,l} \left( F^{1,l}_{j+\frac{1}{2},k} - \frac{1}{r^{l+1}_l} \sum_{\kappa=0}^{r^{l+1}_l-1} \sum_{\iota=0}^{r^{l+1}_l-1} F^{1,l+1}_{v+\frac{1}{2},w+\iota} (t + \kappa \Delta t^{l+1}_l) \right)
\]

\[
- \frac{\Delta t_l}{\Delta x_2,l} \left( F^{2,l}_{j,k+\frac{1}{2}} - F^{2,l}_{j,k-\frac{1}{2}} \right)
\]

Correction pass:

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

2. \( \delta F^{1,l+1}_{j-\frac{1}{2},k} := \delta F^{1,l+1}_{j-\frac{1}{2},k} + \frac{1}{r^{l+1}_l} \sum_{\iota=0}^{r^{l+1}_l-1} F^{1,l+1}_{v+\frac{1}{2},w+\iota} (t + \kappa \Delta t^{l+1}_l) \)
Conservative flux correction

Example: Cell $j, k$

\[
\bar{Q}_{jk}'(t + \Delta t_l) = Q_{jk}'(t) - \frac{\Delta t_l}{\Delta x_{1,l}} \left( F_{j+\frac{1}{2},k}^{1,l} - \frac{1}{r_{l+1}^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} F_{v+\frac{1}{2},w+\iota}^{1,l+1} (t + \kappa \Delta t_{l+1}) \right) \\
- \frac{\Delta t_l}{\Delta x_{2,l}} \left( F_{j,k+\frac{1}{2}}^{2,l} - F_{j,k-\frac{1}{2}}^{2,l} \right)
\]

Correction pass:

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

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

3. $\bar{Q}_{jk}'(t + \Delta t_l) := Q_{jk}'(t + \Delta t_l) + \frac{\Delta t_l}{\Delta x_{1,l}} \delta F_{j-\frac{1}{2},k}^{1,l+1}$

Structured adaptive mesh refinement
Conservative flux correction II
Conservative flux correction II

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

- Cells to correct
Conservative flux correction II

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

- Cells to correct $\delta F_{n,l+1}$
Conservative flux correction II

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

- Corrections $\delta F_{n,l+1}$ stored on level $l + 1$ along $\partial G_{l+1}$ (lower-dimensional data coarsened by $r_{l+1}$)

- Init $\delta F_{n,l+1}$ with level $l$ fluxes $F_{n,l} (\bar{G}_l \cap \partial G_{l+1})$

- Cells to correct $\bullet F_{n,l} \circ \delta F_{n,l+1}$
Conservative flux correction II

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

- Corrections $\delta F^{n,l+1}$ stored on level $l + 1$ along $\partial G_{l+1}$ (lower-dimensional data coarsened by $r_{l+1}$)

- Init $\delta F^{n,l+1}$ with level $l$ fluxes $F^{n,l}(\tilde{G}_l \cap \partial G_{l+1})$

- Add level $l + 1$ fluxes $F^{n,l+1}(\partial G_{l+1})$ to $\delta F^{n,l}$

- Cells to correct
  - $F^{n,l}$
  - $F^{n,l+1}$
  - $\delta F^{n,l+1}$

Structured adaptive mesh refinement

Conservative flux correction
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

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

Bilinear interpolation (prolongation):

\[
\hat{Q}_{jk}^{l+1} \approx (1 - f_1)(1 - f_2) Q_{j-1,k-1}^l + f_1(1 - f_2) Q_{j,k-1}^l + (1 - f_1)f_2 Q_{j-1,k}^l
\]

with factors

\[
f_1 := \frac{x_{v, l+1} - x_{j-1, l}}{\Delta x_1, l}, \quad f_2 := \frac{x_{w, l+1} - x_{k-1, l}}{\Delta x_2, l}
\]

For boundary conditions on \( s \):

\[
\hat{Q}_{jk}^{l+1}(t + \kappa \Delta t_{l+1}) := (1 - \kappa) \hat{Q}_{jk}^{l+1}(t) + \kappa \hat{Q}_{jk}^{l+1}(t + \Delta t_{l+1})
\]

for \( \kappa = 0, \ldots, r_{l+1} \).
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

\[
\hat{Q}_{jk}^l := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\nu=0}^{r_{l+1}-1} Q_{v+\kappa,w+\nu}^{l+1}
\]

Bilinear interpolation (prolongation):

\[
\bar{Q}_{vw}^{l+1} := (1 - f_1)(1 - f_2) Q_{j-1,k-1}^l + f_1(1 - f_2) Q_{j,k-1}^l + (1 - f_1)f_2 Q_{j-1,k}^l + f_1 f_2 Q_{jk}^l
\]

with factors \( f_1 := \frac{x_{1,l+1}^v - x_{1,l}^{j-1}}{\Delta x_{1,l}} \), \( f_2 := \frac{x_{2,l+1}^w - x_{2,l}^{k-1}}{\Delta x_{2,l}} \) derived from the spatial coordinates of the cell centers \((x_{1,l}^{j-1}, x_{2,l}^{k-1})\) and \((x_{1,l+1}^v, x_{2,l+1}^w)\).
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

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

Bilinear interpolation (prolongation):

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

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

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

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

- **Root Level**
  - $r_0 = 1$

- **Level 1**
  - $r_1 = 4$

- **Level 2**
  - $r_2 = 2$

**Time**

- Regridding of finer levels.
- Base level (●) stays fixed.
Recursive integration order

- Space-time interpolation of coarse data to set $I^s_l, l > 0$

---

**Root Level**

- $r_0 = 1$

**Level 1**

- $r_1 = 4$

**Level 2**

- $r_2 = 2$

---

Regridding of finer levels.

Base level (●) stays fixed.
Recursive integration order

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

Root Level

- $r_0 = 1$

Level 1

- $r_1 = 4$

Level 2

- $r_2 = 2$

Base level (●) stays fixed.
Recursive integration order

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

```
<table>
<thead>
<tr>
<th>Level</th>
<th>Root Level</th>
<th>Level 1</th>
<th>Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$r_0 = 1$</td>
<td>$r_1 = 4$</td>
<td>$r_2 = 2$</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2, 5, 8, 11</td>
<td>3, 4, 6, 7, 9, 10, 12, 13</td>
</tr>
</tbody>
</table>
```

Regridding of finer levels.
Base level (○) stays fixed.
The basic recursive algorithm

AdvanceLevel(l)

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

UpdateLevel(l)

\[ t := t + \Delta t_l \]
The basic recursive algorithm

AdvanceLevel(/)

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

UpdateLevel(/)
If level \( l + 1 \) exists?
Set ghost cells of \( Q^l(t + \Delta t_i) \)
AdvanceLevel(/ + 1)

\( t := t + \Delta t_i \)
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 $Q^l(t + \Delta t_l)$
- AdvanceLevel($l + 1$)
- Average $Q^{l+1}(t + \Delta t_l)$ onto $Q^l(t + \Delta t_l)$
- Correct $Q^l(t + \Delta t_l)$ with $\delta F^{l+1}$

$t := t + \Delta t_l$
The basic recursive algorithm

AdvanceLevel(/)

Repeat \( r \) times

Set ghost cells of \( Q^l(t) \)

If time to regrid?

Regrid(/)

UpdateLevel(/)

If level \( l + 1 \) exists?

Set ghost cells of \( Q^l(t + \Delta t_l) \)

AdvanceLevel(/ + 1)

Average \( Q^{l+1}(t + \Delta t_l) \) onto \( Q^l(t + \Delta t_l) \)

Correct \( Q^l(t + \Delta t_l) \) with \( \delta F^{l+1} \)

\( t := t + \Delta t_l \)

- Recursion
- Restriction and flux correction
- Re-organization of hierarchical data
The basic recursive algorithm

\textbf{AdvanceLevel}(l)

\textbf{Repeat } \textbf{r}_l \textbf{ times}

\begin{itemize}
  \item Set ghost cells of \( Q^l(t) \)
  \item If time to regrid?
    \begin{itemize}
      \item \textbf{Regrid}(l)
    \end{itemize}
  \item \textbf{UpdateLevel}(l)
  \item If level \( l+1 \) exists?
    \begin{itemize}
      \item Set ghost cells of \( Q^{l+1}(t + \Delta t_l) \)
      \item \textbf{AdvanceLevel}(l + 1)
      \item Average \( Q^{l+1}(t + \Delta t_l) \) onto \( Q^l(t + \Delta t_l) \)
      \item Correct \( Q^l(t + \Delta t_l) \) with \( \delta F^{l+1} \)
    \end{itemize}
\end{itemize}

\texttt{t := t + \Delta t_l}

Start - Start integration on level 0

\( l = 0, \ r_0 = 1 \)

\textbf{AdvanceLevel}(1)
The basic recursive algorithm

\texttt{AdvanceLevel}(l)

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

Start - Start integration on level 0

\( l = 0, \ r_0 = 1 \)
\texttt{AdvanceLevel}(l)

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

Regrid(\(l\)) - Regrid all levels \(i > l\)

For \(i = i_f\) DownTo \(l\) Do

Flag \(N^i\) according to \(Q^i(t)\)
Regridding algorithm

Regrid(/) - Regrid all levels \( i > l \)

For \( i = l_f \) Downto \( l \) Do

Flag \( N^i \) according to \( Q^i(t) \)

Refinement flags:
\[
N^i := \bigcup_m N(\partial G_{i,m})
\]
Regridding algorithm

Regrid(/) - Regrid all levels $i > l$

For $i = l_f$ Do downto $l$ Do
   Flag $N^i$ according to $Q^i(t)$
   If level $i + 1$ exists?
      Flag $N^i$ below $\mathcal{G}^{i+2}$

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

> Activate flags below higher levels
Regridding algorithm

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

For \( l = l_f \) Down to \( l \) Do

- Flag \( N^l \) according to \( Q^l(t) \)
- If level \( l + 1 \) exists?
  - Flag \( N^l \) below \( \tilde{G}^{l+2} \)
  - Flag buffer zone on \( N^l \)

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

- Activate flags below higher levels

- Flag buffer cells of \( b > \kappa_r \) cells,
  \( \kappa_r \) steps between calls of
  Regrid(l)
The basic recursive algorithm

Regridding algorithm

Regrid(/) - Regrid all levels $i > l$

For $i = l_f$ Down to $l$ Do

- Flag $N^i$ according to $Q^i(t)$
- If level $i + 1$ exists?
  - Flag $N^i$ below $\tilde{G}^{i+2}$
- Flag buffer zone on $N^i$
- Generate $\tilde{G}^{i+1}$ from $N^i$

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

Regrid(/) - Regrid all levels \( i > l \)

For \( i = l_f \) Down to \( i \) Do
   Flag \( N^i \) according to \( Q^i(t) \)
   If level \( i + 1 \) exists?
      Flag \( N^i \) below \( \tilde{G}^{i+2} \)
   Flag buffer zone on \( N^i \)
   Generate \( \tilde{G}^{i+1} \) from \( N^i \)

\[ \tilde{G}_i := G_i \]

For \( i = l \) To \( l_f \) Do
   \[ C \tilde{G}_i := G_0 \setminus \tilde{G}_l \]
   \[ \tilde{G}_{l+1} := \tilde{G}_{l+1} \setminus C \tilde{G}_l \]

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

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

For \( l = l_f \) Down to \( l \) Do
  Flag \( N^l \) according to \( Q^l(t) \)
  If level \( l + 1 \) exists?
    Flag \( N^l \) below \( \tilde{G}^{l+2} \)
    Flag buffer zone on \( N^l \)
    Generate \( \tilde{G}^{l+1} \) from \( N^l \)
  Flag buffer zone on \( N^l \)
  Generate \( \tilde{G}^{l+1} \) from \( N^l \)
  Generate \( \tilde{G}_l := G_l \)
  For \( l = l \) To \( l_f \) Do
    \( C \tilde{G}_l := G_0 \setminus \tilde{G}_l \)
    \( \tilde{G}_{l+1} := \tilde{G}_{l+1} \setminus C \tilde{G}_l \)
  Recompose(l)
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 \( \tilde{G}_\iota \) empty (no coarsening!)
Recomposition of data

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

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

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

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

For $l = l + 1$ To $l_f + 1$ Do

- Interpolate $Q_{l-1}(t)$ onto $\tilde{Q}_l(t)$
- Copy $Q_l(t)$ onto $\tilde{Q}_l(t)$

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

Recompose(/) - Reorganize all levels \( \nu > / \)

For \( \nu = / + 1 \) To \( /_f + 1 \) Do

- Interpolate \( Q^{\nu-1}(t) \) onto \( \tilde{Q}^{\nu}(t) \)
- Copy \( Q^{\nu}(t) \) onto \( \tilde{Q}^{\nu}(t) \)
- Set ghost cells of \( \tilde{Q}^{\nu}(t) \)

- Creates max. 1 level above \( /_f \), but can remove multiple level if \( \tilde{G}_\nu \)
  empty (no coarsening!)
- Use spatial interpolation on entire data \( \tilde{Q}^{\nu}(t) \)
- Overwrite where old data exists
- Synchronization and physical boundary conditions
Recomposition of data

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

For \( l = l + 1 \) To \( l_f + 1 \) Do
- Interpolate \( Q^{l-1}(t) \) onto \( \tilde{Q}^l(t) \)
- Copy \( Q^l(t) \) onto \( \tilde{Q}^l(t) \)
- Set ghost cells of \( \tilde{Q}^l(t) \)
- \( Q^l(t) := \tilde{Q}^l(t), \ G_l := \tilde{G}_l \)

- Creates max. 1 level above \( l_f \), but can remove multiple level if \( \tilde{G}_l \) empty (no coarsening!)
- Use spatial interpolation on entire data \( \tilde{Q}^l(t) \)
- Overwrite where old data exists
- Synchronization and physical boundary conditions
Clustering by signatures

Flagged cells per row/column

\[ \Delta = \gamma_{\nu+1} - 2\gamma_\nu + \gamma_{\nu-1} \]

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

Flagged cells per row/column

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

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

Flagged cells per row/column

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

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

\[ \Upsilon \] Flagged cells per row/column
\[ \Delta \] Second derivative of \( \Upsilon \), \[ \Delta = \Upsilon_{\nu+1} - 2\Upsilon_{\nu} + \Upsilon_{\nu-1} \]

Technique from image detection: [Bell et al., 1994], see also [Berger and Rigoutsos, 1991], [Berger, 1986]
### Block generation and flagging of cells

<table>
<thead>
<tr>
<th>Block Generation</th>
<th>Serial SAMR Method</th>
<th>Parallel SAMR Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursive generation of $\tilde{G}_{l,m}$</td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

#### Recursive Generation of $\tilde{G}_{l,m}$

1. 0 in $\Gamma$
2. Largest difference in $\Delta$
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$
Block generation and flagging of cells

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

1. 0 in $\Upsilon$
2. Largest difference in $\Delta$
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$
Recursive generation of $\mathcal{G}_{l,m}$

1. 0 in $\Gamma$
2. Largest difference in $\Delta$
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$
Recursive generation of $\tilde{G}_{l,m}$

1. 0 in $\Upsilon$
2. Largest difference in $\Delta$
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$
Refinement criteria

Scaled gradient of scalar quantity $w$

$$|w(Q_{j+1,k}) - w(Q_{jk})| > \epsilon_w, \quad |w(Q_{j,k+1}) - w(Q_{jk})| > \epsilon_w, \quad |w(Q_{j+1,k+1}) - w(Q_{jk})| > \epsilon_w$$
Refinement criteria

Scaled gradient of scalar quantity $w$

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

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

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

Scaled gradient of scalar quantity $w$

$$|w(Q_{j+1,k}) - w(Q_{jk})| > \epsilon_w, \quad |w(Q_{j,k+1}) - w(Q_{jk})| > \epsilon_w, \quad |w(Q_{j+1,k+1}) - w(Q_{jk})| > \epsilon_w$$

Heuristic error estimation [Berger, 1982]:

Local truncation error of scheme of order $o$

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

For $q$ smooth after 2 steps $\Delta t$

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

Scaled gradient of scalar quantity \( w \)
\[
|w(Q_{j+1,k}) - w(Q_{jk})| > \epsilon_w, \quad |w(Q_{j,k+1}) - w(Q_{jk})| > \epsilon_w, \quad |w(Q_{j+1,k+1}) - w(Q_{jk})| > \epsilon_w
\]

Heuristic error estimation [Berger, 1982]:
Local truncation error of scheme of order \( o \)
\[
q(x, t + \Delta t) - \mathcal{H}^{(\Delta t)}(q(\cdot,t)) = C\Delta t^{o+1} + O(\Delta t^{o+2})
\]
For \( q \) smooth after 2 steps \( \Delta t \)
\[
q(x, t + \Delta t) - \mathcal{H}_{2}^{(\Delta t)}(q(\cdot,t - \Delta t)) = 2C\Delta t^{o+1} + O(\Delta t^{o+2})
\]
and after 1 step with \( 2\Delta t \)
\[
q(x, t + \Delta t) - \mathcal{H}^{(2\Delta t)}(q(\cdot,t - \Delta t)) = 2^{o+1}C\Delta t^{o+1} + O(\Delta t^{o+2})
\]
Refinement criteria

Scaled gradient of scalar quantity $w$

$$|w(Q_{j+1,k}) - w(Q_{jk})| > \epsilon_w, \quad |w(Q_{j,k+1}) - w(Q_{jk})| > \epsilon_w, \quad |w(Q_{j+1,k+1}) - w(Q_{jk})| > \epsilon_w$$

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

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

For $q$ smooth after 2 steps $\Delta t$

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

and after 1 step with $2\Delta t$

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

Gives

$$\mathcal{H}_2^{(\Delta t)}(q(\cdot, t - \Delta t)) - \mathcal{H}^{(2\Delta t)}(q(\cdot, t - \Delta t)) = (2^{o+1} - 2)C\Delta t^{o+1} + O(\Delta t^{o+2})$$
Heuristic error estimation for FV methods

1. Error estimation on interior cells
Heuristic error estimation for FV methods

1. Error estimation on interior cells

\[ \mathcal{H}^{\Delta t_l} Q'(t_l - \Delta t_l) \]
Heuristic error estimation for FV methods

1. Error estimation on interior cells

$\mathcal{H}^{\Delta t_l} Q'(t_l - \Delta t_l)$
Heuristic error estimation for FV methods

1. Error estimation on interior cells

\[ \mathcal{H}^{\Delta t_i} Q'(t_i - \Delta t_i) \]

1. Error estimation on interior cells

\[ \mathcal{H}^{\Delta t_i} (\mathcal{H}^{\Delta t_i} Q'(t_i - \Delta t_i)) = \mathcal{H}^{\Delta t_i} Q'(t_i - \Delta t_i) \]

Structured adaptive mesh refinement
Heuristic error estimation for FV methods

1. Error estimation on interior cells
2. Create temporary Grid coarsened by factor 2
   Initialize with fine-grid-values of preceding time step

\[ \mathcal{H}^{t_l} Q^l(t_l - \Delta t_l) = \mathcal{H}^{t_l} (\mathcal{H}^{t_l} Q^l(t_l - \Delta t_l)) = \mathcal{H}_2^{t_l} Q^l(t_l - \Delta t_l) \]
Heuristic error estimation for FV methods

1. Error estimation on interior cells
2. Create temporary Grid coarsened by factor 2
   Initialize with fine-grid-values of preceding time step

\[ \mathcal{H}^{\Delta t_l} Q^l(t_l - \Delta t_l) = \mathcal{H}^{\Delta t_l} (\mathcal{H}^{\Delta t_l} Q^l(t_l - \Delta t_l)) = \mathcal{H}_2^{\Delta t_l} Q^l(t_l - \Delta t_l) \]

\[ \mathcal{H}^{2\Delta t_l} \bar{Q}^l(t_l - \Delta t_l) \]
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

\[ \mathcal{H}^{\Delta t_l} Q^l(t_l - \Delta t_l) \]

3. Compare temporary solutions

\[ \mathcal{H}^{\Delta t_l} (\mathcal{H}^{\Delta t_l} Q^l(t_l - \Delta t_l)) = \mathcal{H}_2^{\Delta t_l} Q^l(t_l - \Delta t_l) \]

\[ \mathcal{H}^{2\Delta t_l} \bar{Q}^l(t_l - \Delta t_l) \]
Usage of heuristic error estimation

Current solution integrated tentatively 1 step with $\Delta t_l$ and coarsened

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

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

$$Q(t_l + \Delta t_l) := \mathcal{H}_2^{2\Delta t_l} \text{Restrict} \left( Q^l(t_l - \Delta t_l) \right)$$
Usage of heuristic error estimation

Current solution integrated tentatively 1 step with $\Delta t_l$ and coarsened

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

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

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

Local error estimation of scalar quantity $w$

$$\tau_{jk}^w := \frac{|w(\bar{Q}_{jk}(t + \Delta t)) - w(Q_{jk}(t + \Delta t))|}{2^{o+1} - 2}$$
Usage of heuristic error estimation

Current solution integrated tentatively 1 step with $\Delta t_l$ and coarsened

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

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

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

Local error estimation of scalar quantity $w$

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

In practice [Deiterding, 2003] use

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

Meshes and adaptation
   Adaptivity on unstructured and structured meshes
   Available SAMR software

The serial Berger-Colella SAMR method
   Data structures and numerical update
   Conservative flux correction
   Level transfer operators
   The basic recursive algorithm
   Block generation and flagging of cells

Parallel SAMR method
   Domain decomposition
   A parallel SAMR algorithm
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

Processor 1

Processor 2
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

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

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

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

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

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

Higher level domains $G_l$ follow decomposition of root level

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

With $N_l(\cdot)$ denoting number of cells, we estimate the workload as

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\text{max}}} \left[ N_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right]$$
Rigorous domain decomposition formalized

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

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

Higher level domains $G_l$ follow decomposition of root level

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

With $N_l(\cdot)$ denoting number of cells, we estimate the workload as

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

Equal work distribution necessitates

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

[Deiterding, 2005]
Ghost cell setting

Ghost cell values:

- Interpolation
- Local synchronization
- Parallel synchronization
- Physical boundary
Ghost cell setting

Ghost cell values:
- Interpolation
- Local synchronization
- Parallel synchronization
- Physical boundary

Diagram showing overlapping grids for Processor 1 and Processor 2, with ghost cell values indicated in different colors.
Ghost cell setting

Local synchronization
\[ \tilde{S}_{l,m}^{s,p} = \tilde{G}_{l,m}^{s,p} \cap G_l^p \]
Ghost cell setting

Local synchronization
\[ \tilde{S}_{l,m}^s,p = \tilde{G}_{l,m}^s \cap G_p \]

Parallel synchronization
\[ \tilde{S}_{l,m}^s,q = \tilde{G}_{l,m}^s \cap G_q^p, q \neq p \]
Ghost cell setting

Local synchronization
\[ \tilde{S}_{l,m}^s \cap G_p \]

Parallel synchronization
\[ \tilde{S}_{l,m}^{s,q} \cap G^q_p \]

Interpolation and physical boundary conditions remain strictly local

- Scheme $H(\Delta t_l)$ evaluated locally
- Restriction and prolongation local

Ghost cell values:
- Interpolation
- Local synchronization
- Parallel synchronization
- Physical boundary
Parallel flux correction

Node p

Node q

Structured adaptive mesh refinement
Parallel flux correction

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

---

Node p

$w$

Node q

$\mathbf{F}^{n,l}$
Parallel flux correction

1. Strictly local: Init $\delta F_{n,l+1}$ with $F^n(\tilde{G}_{l,m} \cap \partial G_{l+1}, t)$
Parallel flux correction

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

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

\textbf{AdvanceLevel}(\ell)

Repeat \(r_\ell\) times
\begin{itemize}
  \item Set ghost cells of \(Q^\ell(t)\)
  \item If time to regrid?
    \begin{itemize}
      \item \textbf{Regrid}(\ell)
    \end{itemize}
  \item \textbf{UpdateLevel}(\ell)
  \item If level \(\ell + 1\) exists?
    \begin{itemize}
      \item Set ghost cells of \(Q^\ell(t + \Delta t_\ell)\)
      \item \textbf{AdvanceLevel}(\ell + 1)
      \item Average \(Q^{\ell + 1}(t + \Delta t_\ell)\) onto \(Q^\ell(t + \Delta t_\ell)\)
      \item Correct \(Q^\ell(t + \Delta t_\ell)\) with \(\delta F^{\ell + 1}\)
      \end{itemize}
  \end{itemize}
\end{itemize}

\(t := t + \Delta t_\ell\)

\textbf{UpdateLevel}(\ell)

For all \(m = 1\) To \(M_\ell\) Do
\begin{itemize}
  \item \(Q(G^\ell_{l,m}, t) \xrightarrow{H(\Delta t_\ell)} Q(G_{l,m}, t + \Delta t_\ell), F^n(\bar{G}_{l,m}, t)\)
  \item If level \(\ell > 0\)
    \begin{itemize}
      \item Add \(F^n(\partial G_{l,m}, t)\) to \(\delta F^{n,l}\)
    \end{itemize}
  \item If level \(\ell + 1\) exists
    \begin{itemize}
      \item Init \(\delta F^{n,l+1}\) with \(F^n(\bar{G}_{l,m} \cap \partial G_{l+1}, t)\)
    \end{itemize}
\end{itemize}
The recursive algorithm in parallel

AdvanceLevel(/)

Repeat \( r_l \) times

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

\( t := t + \Delta t_l \)

UpdateLevel(/)

For all \( m = 1 \) To \( M_l \) Do

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

If level \( l > 0 \)
    Add \( F^n(\partial G_{l,m}, t) \) to \( \delta F^{n,l} \)
If level \( l + 1 \) exists
    Init \( \delta F^{n,l+1} \) with \( F^n(\bar{G}_{l,m} \cap \partial G_{l+1}, t) \)
The recursive algorithm in parallel

AdvanceLevel(/)

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

UpdateLevel(/)

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

Numerical update
strictly local

Inter-level transfer local
The recursive algorithm in parallel

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

\textbf{UpdateLevel}(/) \\
\textbf{For all } m = 1 \textbf{ To } M_l \textbf{ Do} \\
\hspace{2em} Q(G_{l,m}, t) \xrightarrow{\mathcal{H}_{(\Delta t_l)}} Q(G_{l,m}, t + \Delta t_l), F^n(\tilde{G}_{l,m}, t) \\
\hspace{1em} \textbf{If } \text{level } l > 0 \\
\hspace{2em} \add{F^n(\partial G_{l,m}, t)} \textbf{ to } \delta F^n,l \\
\hspace{1em} \textbf{If level } l + 1 \textbf{ exists} \\
\hspace{2em} \textbf{Init } \delta F^{n,l+1} \textbf{ with } F^n(\tilde{G}_{l,m} \cap \partial G_{l+1}, t) \\

\begin{itemize}
  \item \textbf{Numerical update}  \\
  \item \textbf{strictly local}  \\
  \item \textbf{Inter-level transfer} \textbf{ local}  \\
  \item \textbf{Parallel synchronization}
\end{itemize}
The recursive algorithm in parallel

\textbf{AdvanceLevel}(l)

\textbf{Repeat} \(r_l\) \textbf{times}

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

\textbf{UpdateLevel}(l)

\textbf{For all} \(m = 1\) \textbf{To} \(M_l\) \textbf{Do}

- \(Q(G^s_l,m,t) \xrightarrow{H(\Delta t_l)} Q(G_l,m,t + \Delta t_l), F^n(\tilde{G}_l,m,t)\)
- If level \(l > 0\)
  - Add \(F^n(\partial G_l,m,t)\) to \(\delta F^{n,l}\)
- If level \(l + 1\) exists
  - \textbf{Init} \(\delta F^{n,l+1}\) with \(F^n(\tilde{G}_l,m \cap \partial G_{l+1},t)\)

- Numerical update strictly local
- Inter-level transfer local
- Parallel synchronization
- Application of \(\delta F^{l+1}\) on \(\partial G^q_l\)
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 $Q^{l+1}(t + \Delta t_l)$
    AdvanceLevel($l+1$)
    Average $Q^{l+1}(t + \Delta t_l)$ onto $Q^l(t + \Delta t_l)$
    Correct $Q^l(t + \Delta t_l)$ with $\delta F^{l+1}$
    $t := t + \Delta t_l$

UpdateLevel($l$)

For all $m = 1$ To $M_l$ Do
  $Q(G^s_l, m, t) \xrightarrow{H(\Delta t_l)} Q(G_l, m, t + \Delta t_l), F^n(\tilde{G}_l, m, t)$
  If level $l > 0$
    Add $F^n(\partial G_l, m, t)$ to $\delta F^n,l$
  If level $l + 1$ exists
    Init $\delta F^n,l+1$ with $F^n(\tilde{G}_l, m \cap \partial G_{l+1}, t)$

Numerical update
  strictly local

Inter-level transfer local

Parallel synchronization

Application of $\delta F^{l+1}$ on $\partial G^q_l$
Regridding algorithm in parallel

Regrid(\ell) - Regrid all levels \( \ell > l \)

For \( \ell = l_f \) Down to \( l \) Do

Flag \( N^\ell \) according to \( Q^\ell(t) \)
If level \( \ell + 1 \) exists? 
Flag \( N^\ell \) below \( \tilde{G}^{\ell+2} \)
Flag buffer zone on \( N^\ell \)
Generate \( \tilde{G}^{\ell+1} \) from \( N^\ell \)

\( \tilde{G}_l := G_l \)

For \( \ell = l \) To \( l_f \) Do

\( C\tilde{G}_l := G_0 \setminus \tilde{G}_l \)
\( \tilde{G}_{l+1} := \tilde{G}_{l+1} \setminus C\tilde{G}_l \)

Recompose(\( l \))
A parallel SAMR algorithm

Regridding algorithm in parallel

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

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

$\mathcal{G}_l := G_l$

For $l = l$ to $l_f$ Do
   $C\mathcal{G}_{l_t} := G_0 \setminus \mathcal{G}_l$
   $\mathcal{G}_{l+1} := \mathcal{G}_{l+1} \setminus C\mathcal{G}_l$

Recompose($l$)
Regridding algorithm in parallel

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

For \(i = i_f \) Down to \(i = l\) Do

    Flag \(N^i\) according to \(Q^i(t)\)

    If level \(i + 1\) exists?
        Flag \(N^i\) below \(\tilde{G}^{i+2}\)
    Flag buffer zone on \(N^i\)
    Generate \(\tilde{G}^{i+1}\) from \(N^i\)

\(\tilde{G}_l := \tilde{G}_l\)

For \(i = l\) To \(i_f\) Do

\(C\tilde{G}_l := \tilde{G}_0 \setminus \tilde{G}_l\)

\(\tilde{G}_{l+1} := \tilde{G}_{l+1} \setminus C\tilde{G}_l\)

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 $i > l$

For $i = l_f$ Down to $l$ Do
  Flag $N_i$ according to $Q_i(t)$
  If level $i+1$ exists?
    Flag $N_i$ below $\tilde{G}_{i+2}$
  Flag buffer zone on $N_i$
  Generate $\tilde{G}_{i+1}$ from $N_i$

$\tilde{G}_i := G_i$
For $i = l$ To $l_f$ Do
  $C\tilde{G}_i := G_0 \setminus \tilde{G}_i$
  $\tilde{G}_{i+1} := \tilde{G}_{i+1} \setminus C\tilde{G}_i$
  $\tilde{G}_{i+1} := \tilde{G}_{i+1} \setminus C\tilde{G}_i$

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 $\tilde{G}_{i+1}$
Regridding algorithm in parallel

Regrid(l) – Regrid all levels $l > l$

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

$\tilde{G}_l := G_l$

For $l = l$ To $l_f$ Do
  $C\tilde{G}_l := G_0 \setminus \tilde{G}_l$
  $\tilde{G}_{l+1} := \tilde{G}_{l+1} \setminus C\tilde{G}_l$
  $\tilde{G}^{l+1}_l := \tilde{G}^{l+1}_{l+1} \setminus C\tilde{G}_{l}^1$

Recompose(/)

- 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 $\tilde{G}^{l+1}$
Regridding algorithm in parallel

Regrid(\(l\)) - Regrid all levels \(i > l\)

For \(i = l_f\) DownTo \(l\) Do
   Flag \(N^i\) according to \(Q^i(t)\)
   If level \(i + 1\) exists?
      Flag \(N^i\) below \(\mathcal{G}^{i+2}\)
      Flag buffer zone on \(N^i\)
      Generate \(\mathcal{G}^{i+1}\) from \(N^i\)

\(\mathcal{G}_l := G_l\)

For \(i = l\) To \(l_f\) Do
   \(C\mathcal{G}_l := G_0 \setminus \mathcal{G}_l\)
   \(\mathcal{G}_{l+1} := \mathcal{G}_{l+1} \setminus C\mathcal{G}_l^{1}\)
   \(\mathcal{G}_{l+1} := \mathcal{G}_{l+1} \setminus C\mathcal{G}_l^{1}\)

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 \(\mathcal{G}^{i+1}\)
Recomposition algorithm in parallel

Recompose(/) - Reorganize all levels

For \( i = l + 1 \) To \( l_f + 1 \) Do

Interpolate \( Q^{i-1}(t) \) onto \( \tilde{Q}^i(t) \)

Copy \( Q^i(t) \) onto \( \tilde{Q}^i(t) \)
Set ghost cells of \( \tilde{Q}^i(t) \)
\( Q^i(t) := \tilde{Q}^i(t) \)
\( G_i := \tilde{G}_i \)
Recomposition algorithm in parallel

Recompose($l$) - Reorganize all levels

Generate $G^p_0$ from $\{G_0, ..., G_l, \tilde{G}_{l+1}, ..., \tilde{G}_{l_f+1}\}$
For $\iota = 0$ To $l_f + 1$ Do

Interpolate $Q^{\iota-1}(t)$ onto $\tilde{Q}^\iota(t)$

Copy $Q^\iota(t)$ onto $\tilde{Q}^\iota(t)$
Set ghost cells of $\tilde{Q}^\iota(t)$
$Q^\iota(t) := \tilde{Q}^\iota(t)$
$G^p_\iota := \tilde{G}^p_\iota$, $G_\iota := \bigcup_p G^p_\iota$

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 \{\( G_0, ..., G_l, \tilde{G}_{l+1}, ..., \tilde{G}_{l_f+1} \)\}

For \( \iota = 0 \) To \( l_f + 1 \) Do
  If \( \iota > l \)
    \( \tilde{G}^p_\iota := \tilde{G}_\iota \cap G_0^p \)
    Interpolate \( Q^{\iota-1}(t) \) onto \( \tilde{Q}^\iota(t) \)
  Else
    \( \tilde{G}^p_\iota := G_\iota \cap G_0^p \)
    Copy \( Q^\iota(t) \) onto \( \tilde{Q}^\iota(t) \)
    Set ghost cells of \( \tilde{Q}^\iota(t) \)
    \( Q^\iota(t) := \tilde{Q}^\iota(t) \)
  End If
  \( G^p_\iota := \tilde{G}^p_\iota, G_\iota := \bigcup_p G^p_\iota \)
End For

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

▷ When \( \iota > l \) do nothing special

▷ For \( \iota \leq l \), redistribute additionally
Recomposition algorithm in parallel

Recompose($l$) - Reorganize all levels

Generate $G_0^p$ from \{$G_0, ..., G_l, \tilde{G}_{l+1}, ..., \tilde{G}_{l_f+1}$\}

For $\iota = 0$ To $l_f + 1$ Do

If $\iota > l$

\[ \tilde{G}_\iota^p := \tilde{G}_\iota \cap G_0^p \]

Interpolate $Q_{\iota-1}(t)$ onto $\tilde{Q}_{\iota}(t)$

else

\[ \tilde{G}_\iota^p := G_\iota \cap G_0^p \]

If $\iota > 0$

Copy $\delta F_{n,\iota}$ onto $\tilde{\delta F}_{n,\iota}$

\[ \delta F_{n,\iota} := \tilde{\delta F}_{n,\iota} \]

Copy $Q_{\iota}(t)$ onto $\tilde{Q}_{\iota}(t)$

Set ghost cells of $\tilde{Q}_{\iota}(t)$

\[ Q_{\iota}(t) := \tilde{Q}_{\iota}(t) \]

\[ G_i^p := \tilde{G}_i^p, G_i := \bigcup_p G_i^p \]

- Global redistribution can also be required when regridding higher levels and $G_0, ..., G_l$ do not change (drawback of domain decomposition)
- When $\iota > l$ do nothing special
- For $\iota \leq l$, redistribute additionally
  - Flux corrections $\delta F_{n,\iota}$
Recomposition algorithm in parallel

Recompose(l) - Reorganize all levels

Generate $G^p_0$ from $\{G_0, ..., G_l, \tilde{G}_{l+1}, ..., \tilde{G}_{l_f+1}\}$

For $\iota = 0$ To $l_f + 1$ Do
  If $\iota > l$
    $\tilde{G}^p_\iota := \tilde{G}_\iota \cap G^p_0$
    Interpolate $Q^{\iota - 1}(t)$ onto $\tilde{Q}^\iota(t)$
  else
    $\tilde{G}^p_\iota := G_\iota \cap G^p_0$
    If $\iota > 0$
      Copy $\delta F^{n,\iota}$ onto $\tilde{\delta F}^{n,\iota}$
      $\delta F^{n,\iota} := \tilde{\delta F}^{n,\iota}$
  If $\iota \geq l$ then $\kappa_\iota = 0$ else $\kappa_\iota = 1$
  For $\kappa = 0$ To $\kappa_\iota$ Do
    Copy $Q^\iota(t + \kappa \Delta t_\iota)$ onto $\tilde{Q}^\iota(t + \kappa \Delta t_\iota)$
    Set ghost cells of $\tilde{Q}^\iota(t + \kappa \Delta t_\iota)$
    $Q^\iota(t + \kappa \Delta t_\iota) := \tilde{Q}^\iota(t + \kappa \Delta t_\iota)$
    $G^p_\iota := \tilde{G}^p_\iota$, $G_\iota := \bigcup_p G^p_\iota$

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

Recompose(l) - Reorganize all levels

Generate $G_0^p$ from $\{G_0, ..., G_l, \tilde{G}_{l+1}, ..., \tilde{G}_{l_f+1}\}$

For $\iota = 0$ To $l_f + 1$ Do
  If $\iota > l$
    $\tilde{G}_l^p := \tilde{G}_l \cap G_0^p$
    Interpolate $Q^{l-1}(t)$ onto $\tilde{Q}^l(t)$
  else
    $\tilde{G}_l^p := G_l \cap G_0^p$
    If $\iota > 0$
      Copy $\delta F_{n,\iota}$ onto $\tilde{\delta F}_{n,\iota}$
      $\delta F_{n,\iota} := \tilde{\delta F}_{n,\iota}$
    If $\iota \ge l$ then $\kappa_l = 0$ else $\kappa_l = 1$
    For $\kappa = 0$ To $\kappa_l$ Do
      Copy $Q^l(t + \kappa \Delta t_l)$ onto $\tilde{Q}^l(t + \kappa \Delta t_l)$
      Set ghost cells of $\tilde{Q}^l(t + \kappa \Delta t_l)$
      $Q^l(t + \kappa \Delta t_l) := \tilde{Q}^l(t + \kappa \Delta t_l)$
      $G_l^p := \tilde{G}_l^p$, $G_l := \bigcup_p G_l^p$

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

<table>
<thead>
<tr>
<th>Calculation domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Workload</td>
</tr>
<tr>
<td>Medium Workload</td>
</tr>
<tr>
<td>Low Workload</td>
</tr>
</tbody>
</table>

A parallel SAMR algorithm
Space-filling curve algorithm

Calculation domain

- High Workload
- Medium Workload
- Low Workload

Structured adaptive mesh refinement
Space-filling curve algorithm

Calculation domain

Necessary domain of Space-Filling Curve

- High Workload
- Medium Workload
- Low Workload
Space-filling curve algorithm

Calculation domain

High Workload
Medium Workload
Low Workload

Necessary domain of Space-Filling Curve
Space-filling curve algorithm

Calculation domain

Necessary domain of Space-Filling Curve

- Proc. 1
- High Workload
- Medium Workload
- Low Workload

Structured adaptive mesh refinement
Meshes and adaptation

Serial SAMR method

Parallel SAMR method

References

A parallel SAMR algorithm

**Space-filling curve algorithm**

- High Workload
- Medium Workload
- Low Workload

Calculation domain

Necessary domain of Space-Filling Curve

Proc. 1

Proc. 2

Structured adaptive mesh refinement
Meshes and adaptation

Serial SAMR method

Parallel SAMR method

References

A parallel SAMR algorithm

Space-filling curve algorithm

Calculation domain

Necessary domain of Space-Filling Curve

Proc. 1

Proc. 2

Proc. 3

High Workload

Medium Workload

Low Workload

Structured adaptive mesh refinement
References I


References II


References III


