Introduction to Block-Structured Adaptive Mesh Refinement (AMR)

Ann S. Almgren

Center for Computational Sciences and Engineering
(Building 50A)
Lawrence Berkeley National Laboratory

asalmgren@lbl.gov

Types of Refinement

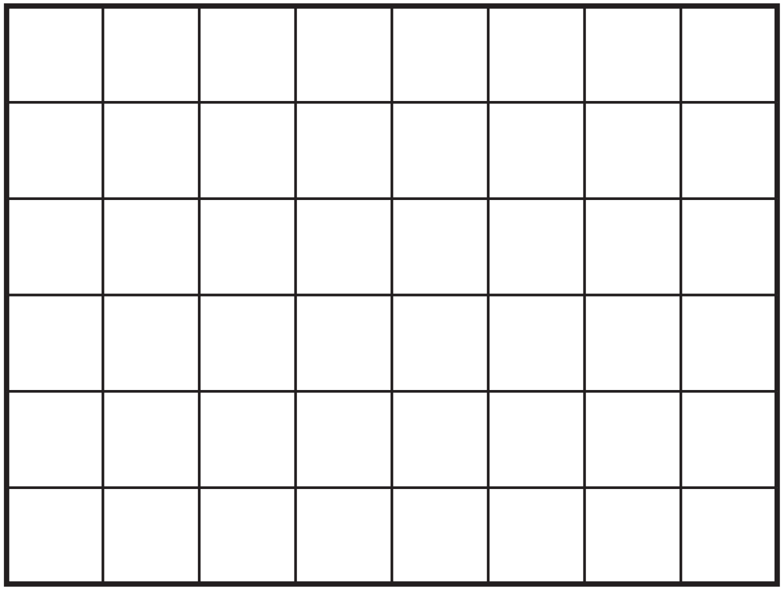
AMR approaches

- Unstructured
-
-
-

Types of Refinement

AMR approaches

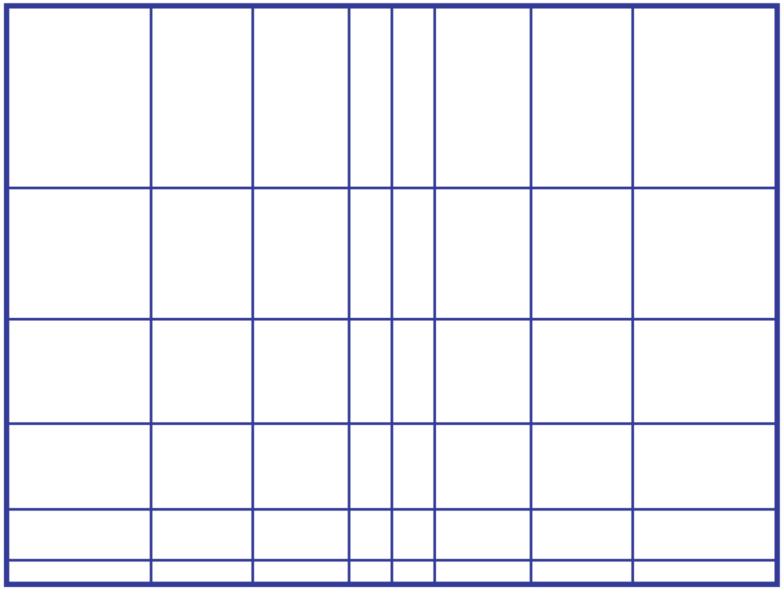
- Unstructured
- Mesh distortion
-
-



Types of Refinement

AMR approaches

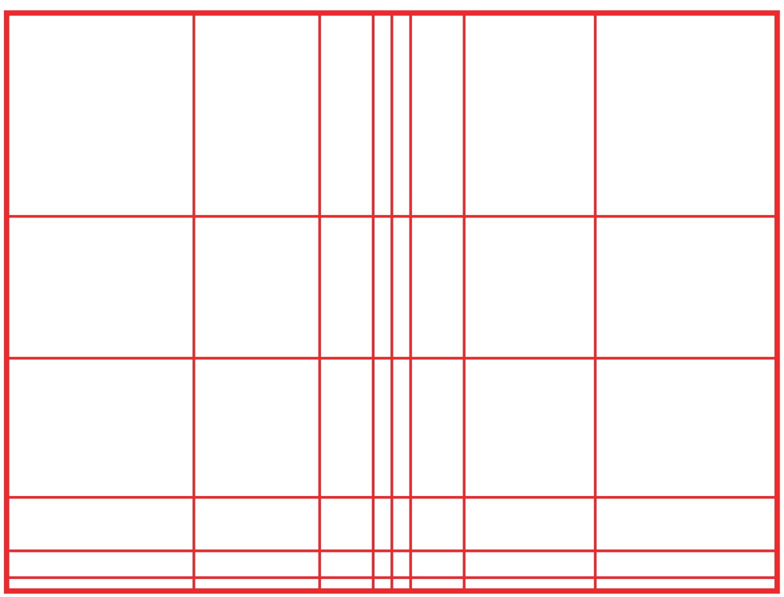
- Unstructured
- Mesh distortion
-
-



Types of Refinement

AMR approaches

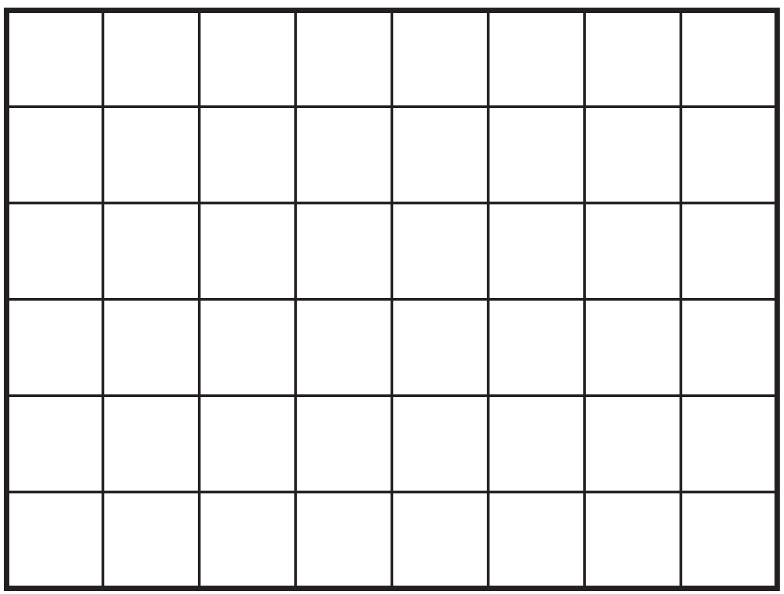
- Unstructured
- Mesh distortion
-
-



Types of Refinement

AMR approaches

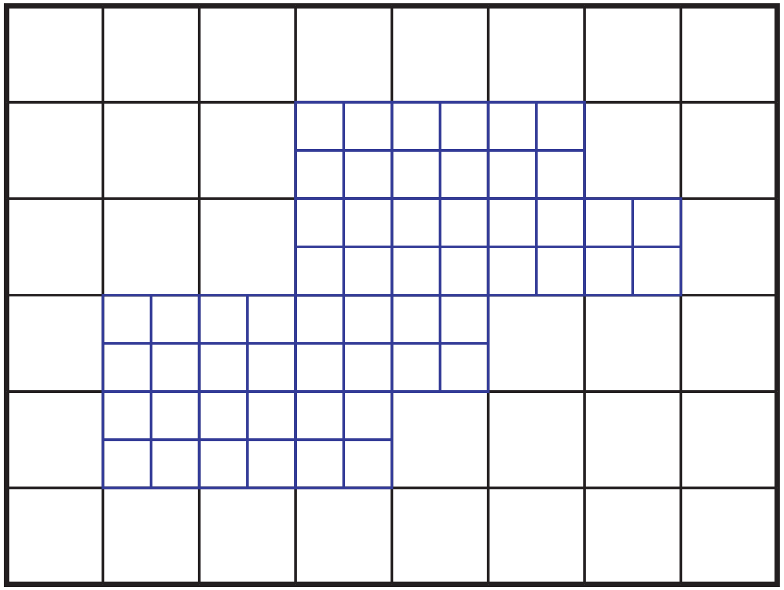
- Unstructured
- Mesh distortion
- Point-wise structured refinement
-



Types of Refinement

AMR approaches

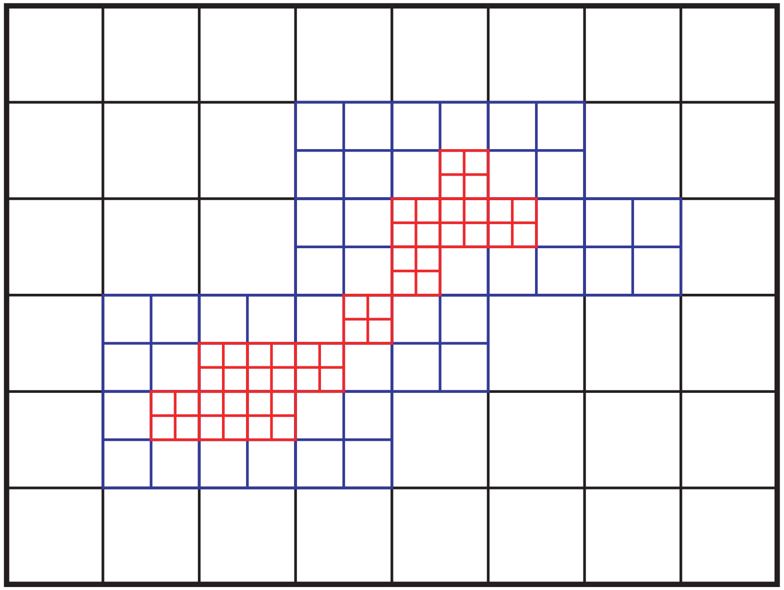
- Unstructured
- Mesh distortion
- Point-wise structured refinement
-



Types of Refinement

AMR approaches

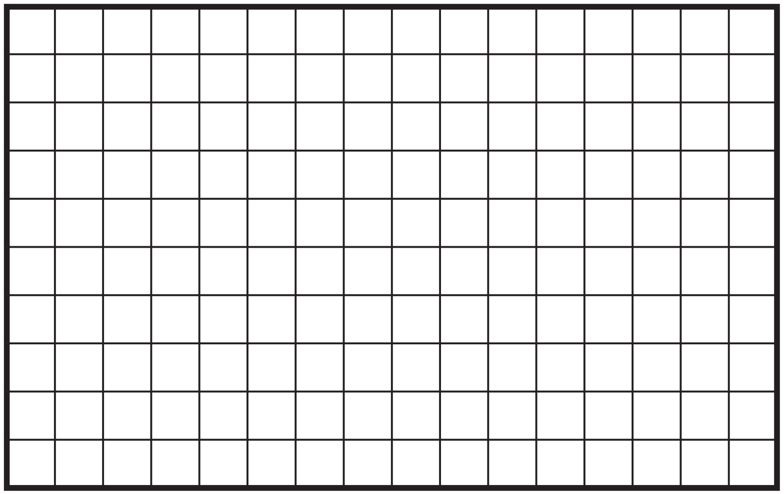
- Unstructured
- Mesh distortion
- Point-wise structured refinement
-



Types of Refinement

AMR approaches

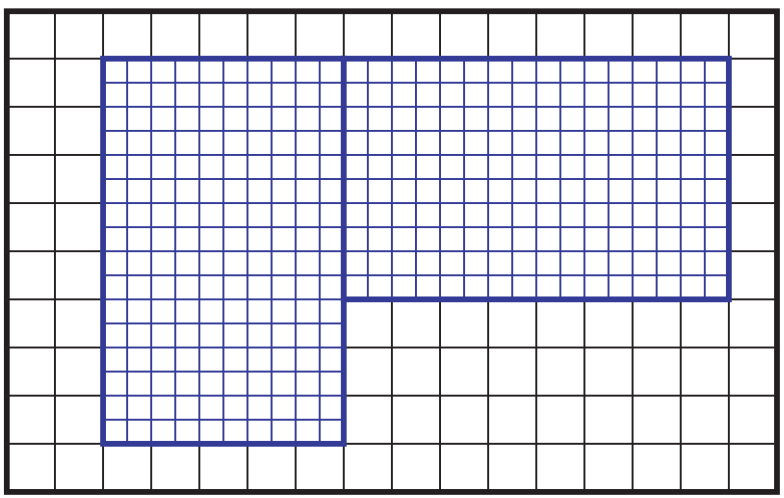
- Unstructured
- Mesh distortion
- Point-wise structured refinement
- Block structured



Types of Refinement

AMR approaches

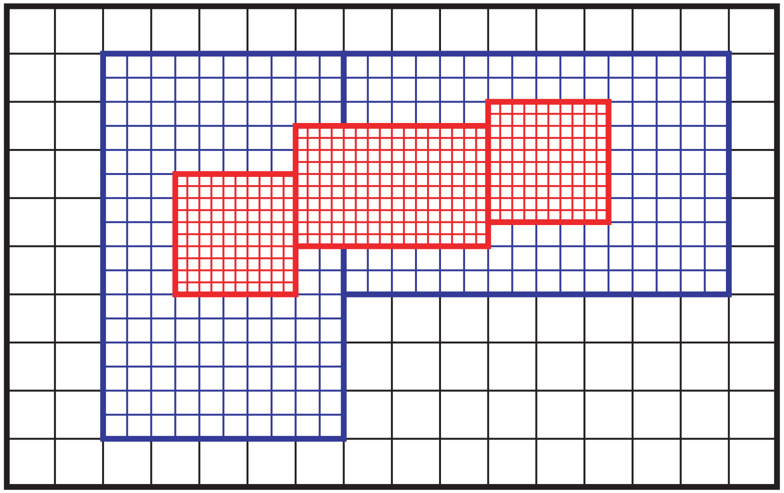
- Unstructured
- Mesh distortion
- Point-wise structured refinement
- Block structured



Types of Refinement

AMR approaches

- Unstructured
- Mesh distortion
- Point-wise structured refinement
- Block structured



AMR for Conservation Laws

Consider the 2-D hyperbolic conservation law

$$U_t + \mathbf{F}_x + \mathbf{G}_y = 0$$

where

$$\mathbf{F} = \mathbf{F}(U), \mathbf{G} = \mathbf{G}(U)$$

Basic discretization:

- Finite volume approach with cell-centered data
- Flux-based representation of conservation law
- Explicit in time update

$$\frac{U^{n+1} - U^n}{\Delta t} = \frac{\mathbf{F}^{n+\frac{1}{2}}_{i-1/2,j} - \mathbf{F}^{n+\frac{1}{2}}_{i+1/2,j}}{\Delta x} + \frac{\mathbf{G}^{n+\frac{1}{2}}_{i,j-1/2} - \mathbf{G}^{n+\frac{1}{2}}_{i,j+1/2}}{\Delta y}$$

Numerical fluxes computed from data at t^n in neighborhood of edge

Basic Structured Grid AMR Ideas

- Cover regions requiring high resolution with finer grids
- Use (higher-order upwind) methodology for regular grids to integrate solution

Need to know

- how to generate the grid hierarchy
  - initially
  - every time you regrid
- how to integrate the solution forward in time
  - Integration of data on a patch
  - Synchronization of levels

Original references:

- 2-D: Berger and Colella, JCP 1989
- 3-D: Bell,Berger,Saltzman and Welcome, JCP 1994

AMR: To Subcycle or Not To Subcycle?

Should we refine in time as well as space?

- Yes – subcyle in time (CASTRO, Enzo)
  - $\Delta t_c = r \Delta t_f$
  - Maintain CFL (accuracy of advection scheme) across levels
  - Reduce total number of cells advanced
- No – don't subcycle (FLASH, RAGE)
  - $\Delta t_c = \Delta t_f$
  - Compute time step every fine grid time
  - Simpler synchronization algorithm
  - Simpler software framework

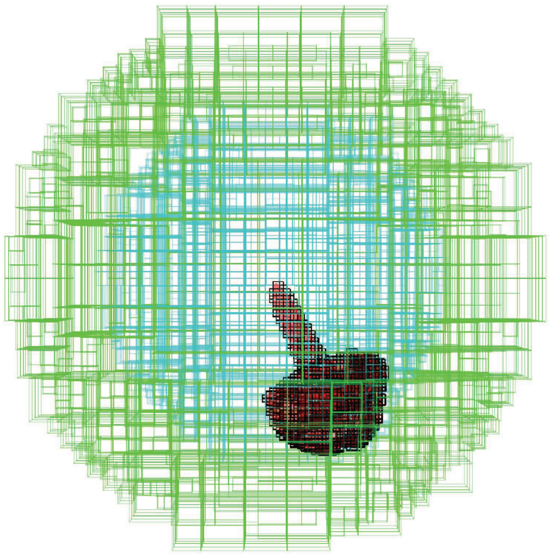
AMR: Subcycling in Time?

Suppose we have N cells at each level and L levels with factor 2 refinement.To reach time T:

With $L = 4$:

- Subcycling: must advance
  $N + 2N + 4N + 8N = 15N$ cell-steps
- No subcycling: must advance
  $4 \cdot 8N = 32N$ cell-steps

Type Ia Supernova



With 2 total levels there is a factor of $4/3$ more work with no subcycling.

With 3 total levels there is a factor of $12/7$ more work with no subcycling.

With 4 levels ... $32/15$ ...

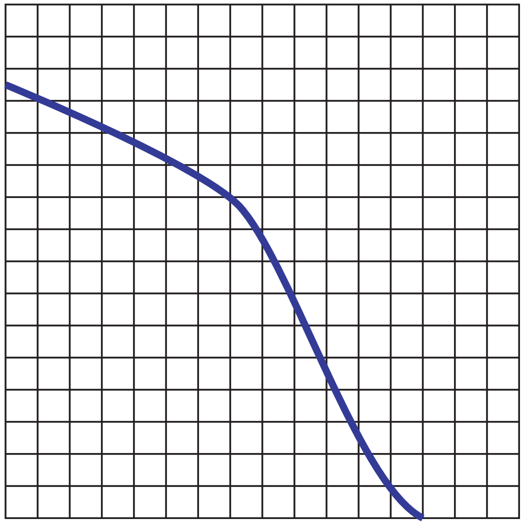
With L levels ... ??
(Note that answer changes if N not constant over levels.)

Building the initial grid hierarchy

- Fill data at level 0
- Estimate where refinement is needed and buffer
- Group cells into patches according to a prescribed "grid efficiency" and refine $\Rightarrow B_1, ..., B_n$ (Berger and Rigoustos, 1991)
- Repeat for next level and adjust for proper nesting

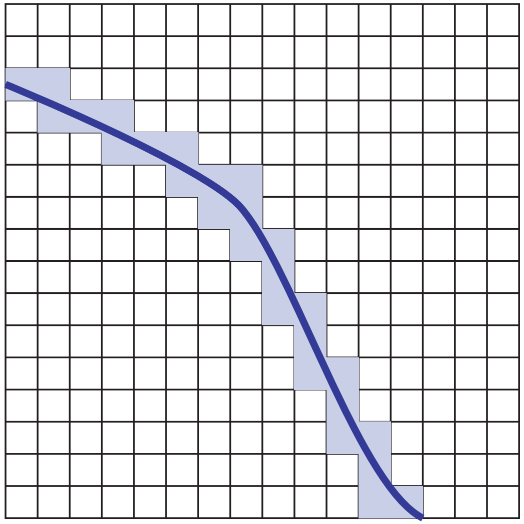
Building the initial grid hierarchy

- Fill data at level 0
- Estimate where refinement is needed and buffer
- Group cells into patches according to a prescribed “grid efficiency” and refine $\Rightarrow B_1, \ldots, B_n$ (Berger and Rigoustos, 1991)
- Repeat for next level and adjust for proper nesting



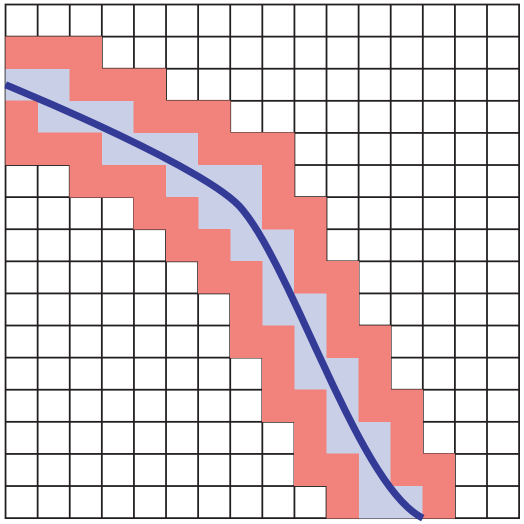
Building the initial grid hierarchy

- Fill data at level 0
- Estimate where refinement is needed and buffer
- Group cells into patches according to a prescribed “grid efficiency” and refine $\Rightarrow B_1, ..., B_n$ (Berger and Rigoustos, 1991)
- Repeat for next level and adjust for proper nesting



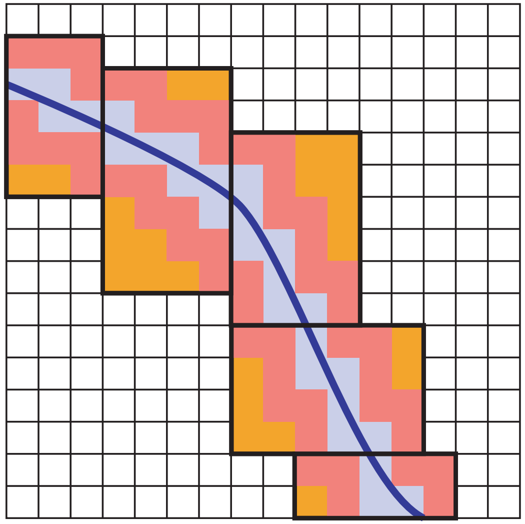
Building the initial grid hierarchy

- Fill data at level 0
- Estimate where refinement is needed and buffer
- Group cells into patches according to a prescribed “grid efficiency” and refine $\Rightarrow B_1, ..., B_n$ (Berger and Rigoustos, 1991)
- Repeat for next level and adjust for proper nesting



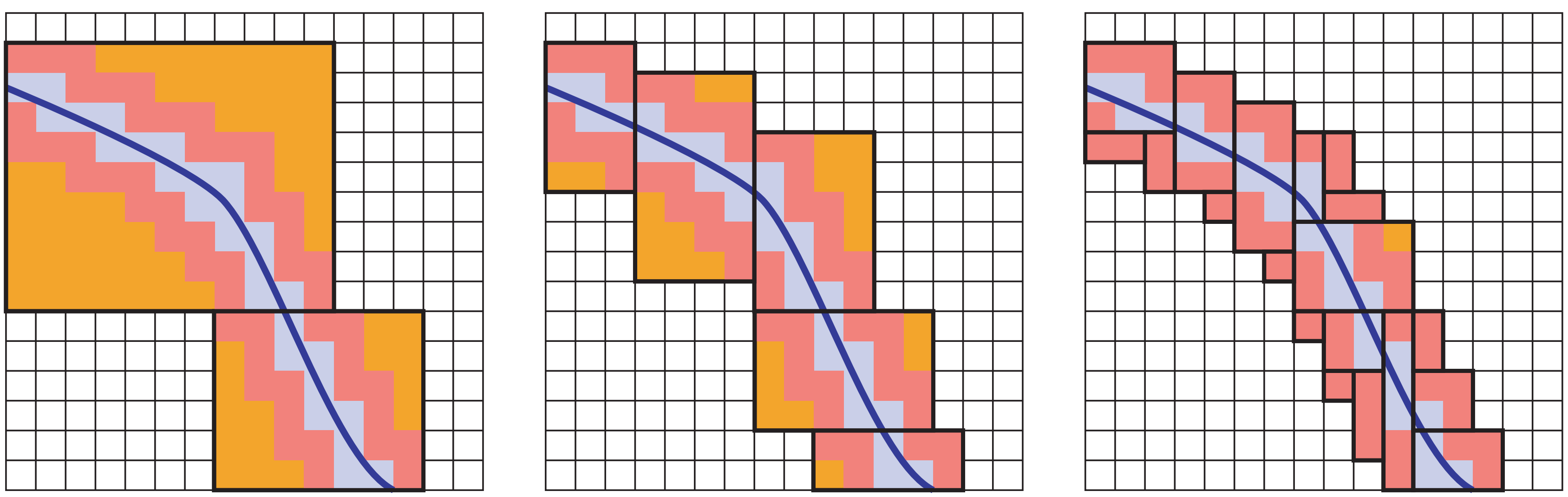
Building the initial grid hierarchy

- Fill data at level 0
- Estimate where refinement is needed and buffer
- Group cells into patches according to a prescribed “grid efficiency” and refine $\Rightarrow B_1, ..., B_n$ (Berger and Rigoustos, 1991)
- Repeat for next level and adjust for proper nesting



Building the initial grid hierarchy

- Fill data at level 0
- Estimate where refinement is needed and buffer
- Group cells into patches according to a prescribed “grid efficiency” and refine $\Rightarrow B_1, ..., B_n$ (Berger and Rigoustos, 1991)
- Repeat for next level and adjust for proper nesting



Efficiency = 0.5 Efficiency = 0.7 Efficiency = 0.9

Adaptive integration algorithm

Consider two levels, coarse and fine, with refinement ratio r

$$\Delta x_f = \Delta x_c / r \quad , \quad \Delta t_f = \Delta t_c / r,$$

To integrate

- Advance coarse grids in time $t_c \rightarrow t_c + \Delta t_c$
- Advance fine grids in time r times
- Synchronize coarse and fine data

Extend recursively to arbitrary number of refinement levels.

Adaptive integration algorithm

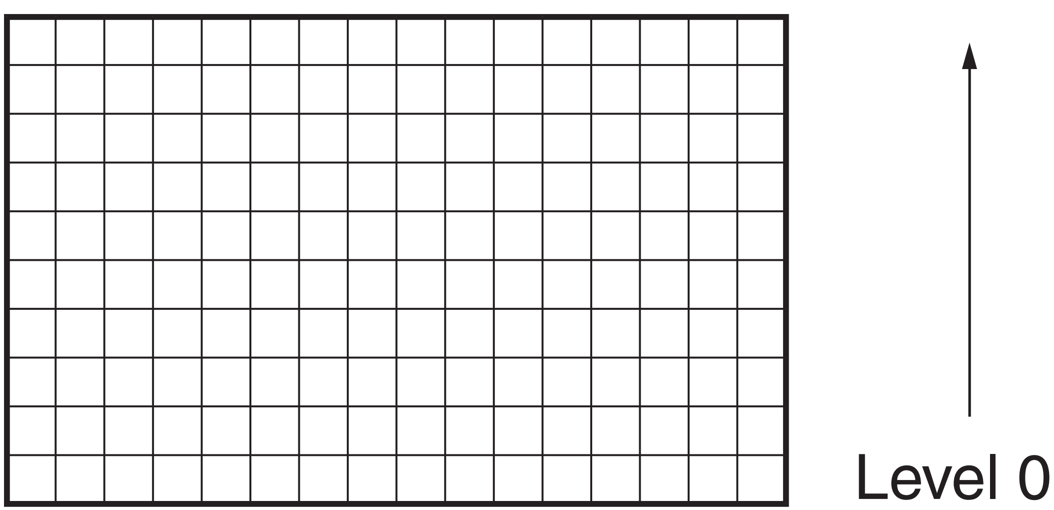
Consider two levels, coarse and fine, with refinement ratio r

$$\Delta x_f = \Delta x_c / r \quad , \quad \Delta t_f = \Delta t_c / r,$$

To integrate

- Advance coarse grids in time $t_c \to t_c + \Delta t_c$
- Advance fine grids in time r times
- Synchronize coarse and fine data

Extend recursively to arbitrary number of refinement levels.



Adaptive integration algorithm

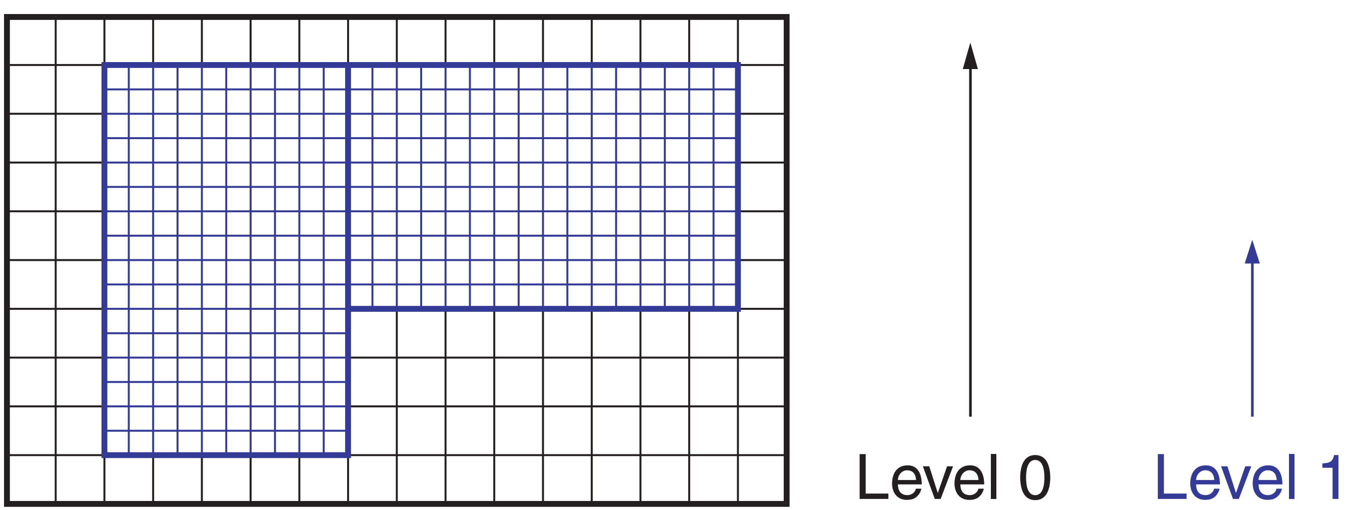
Consider two levels, coarse and fine, with refinement ratio r

$$\Delta x_f = \Delta x_c / r \quad , \quad \Delta t_f = \Delta t_c / r,$$

To integrate

- Advance coarse grids in time $t_c \to t_c + \Delta t_c$
- Advance fine grids in time r times
- Synchronize coarse and fine data

Extend recursively to arbitrary number of refinement levels.



Adaptive integration algorithm

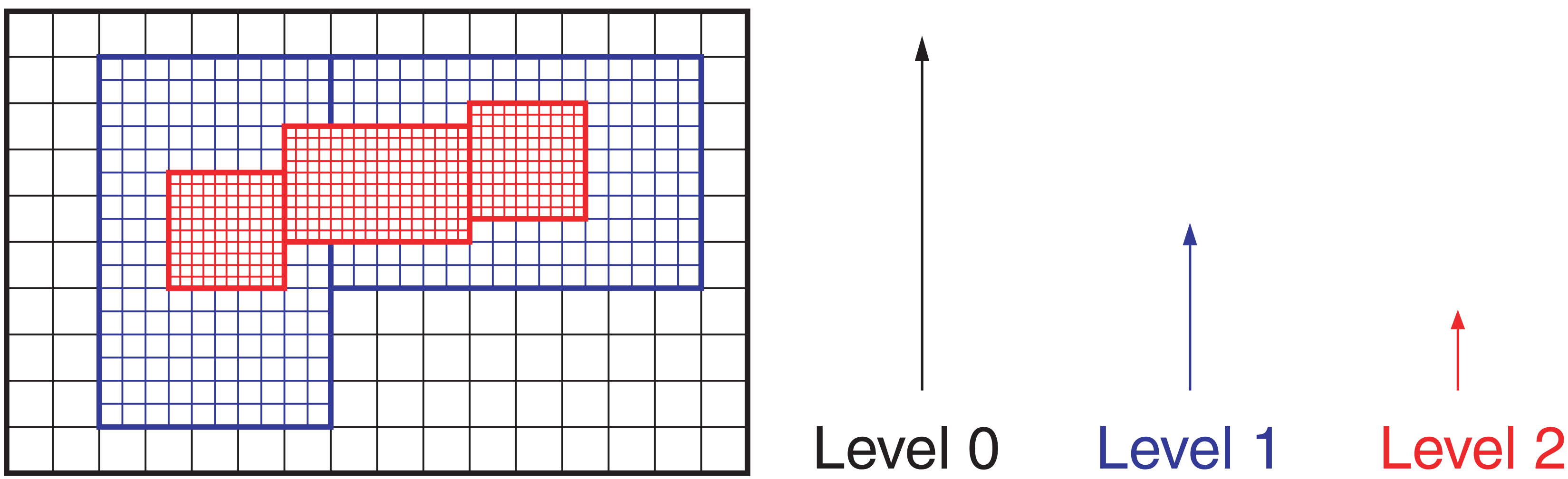
Consider two levels, coarse and fine, with refinement ratio r

$$\Delta x_f = \Delta x_c / r \quad , \quad \Delta t_f = \Delta t_c / r,$$

To integrate

- Advance coarse grids in time $t_c \rightarrow t_c + \Delta t_c$
- Advance fine grids in time r times
- Synchronize coarse and fine data

Extend recursively to arbitrary number of refinement levels.



Adaptive integration algorithm

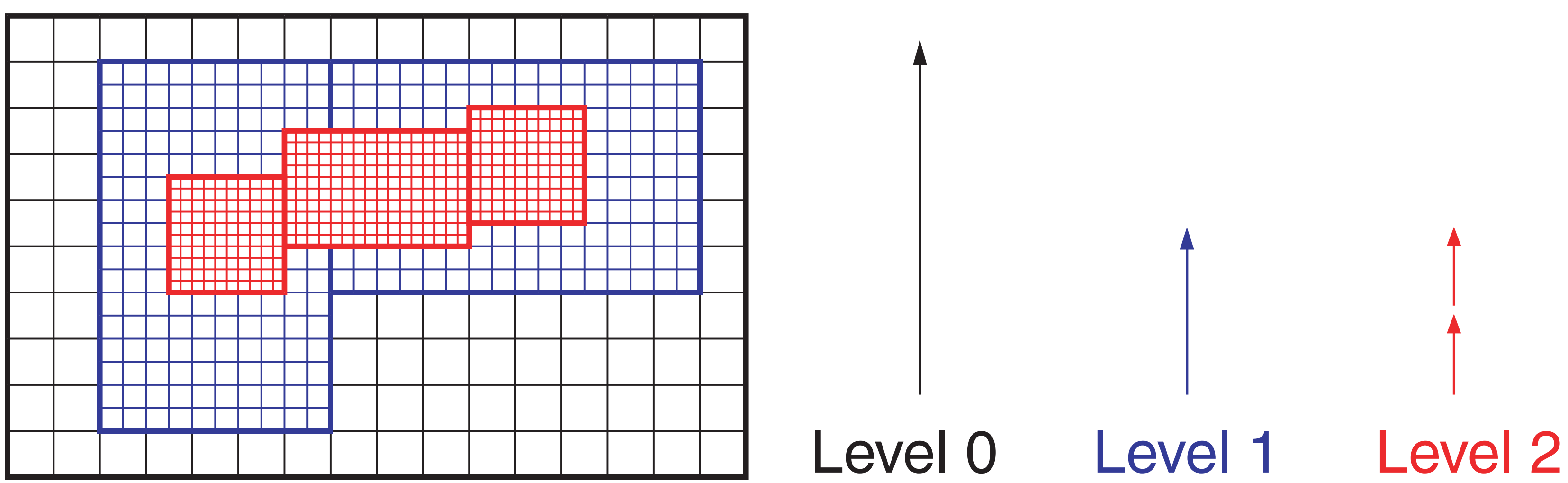
Consider two levels, coarse and fine, with refinement ratio r

$$\Delta x_f = \Delta x_c / r \quad , \quad \Delta t_f = \Delta t_c / r,$$

To integrate

- Advance coarse grids in time $t_c \rightarrow t_c + \Delta t_c$
- Advance fine grids in time r times
- Synchronize coarse and fine data

Extend recursively to arbitrary number of refinement levels.



Adaptive integration algorithm

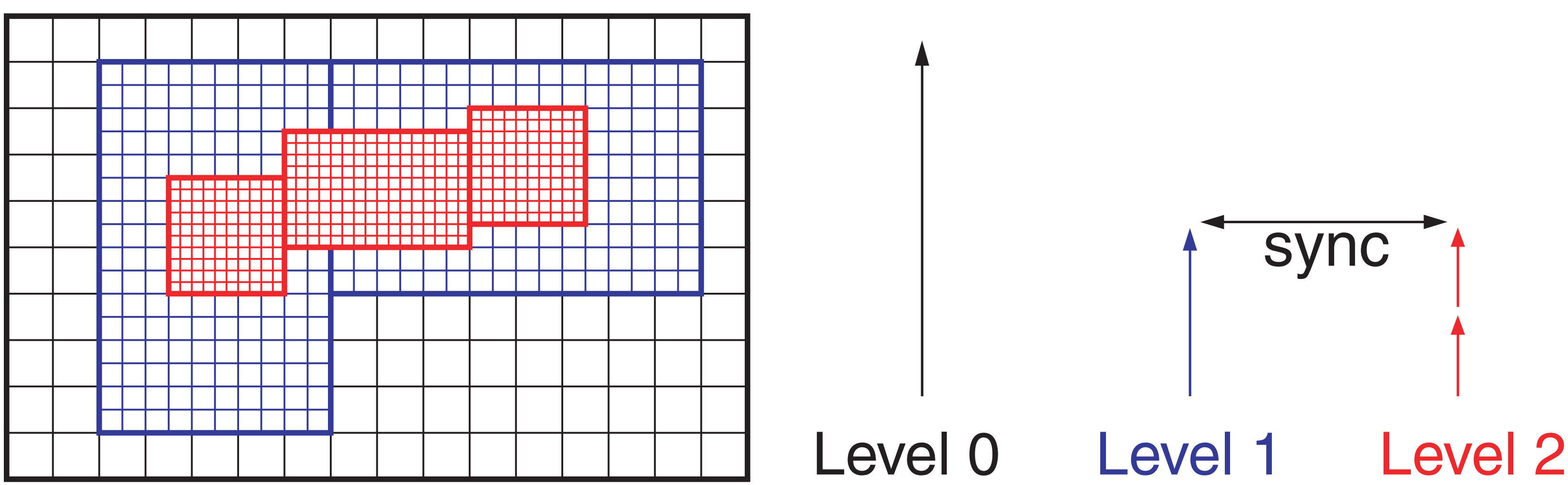
Consider two levels, coarse and fine, with refinement ratio r

$$\Delta x_f = \Delta x_c / r \quad , \quad \Delta t_f = \Delta t_c / r,$$

To integrate

- Advance coarse grids in time $t_c \to t_c + \Delta t_c$
- Advance fine grids in time r times
- Synchronize coarse and fine data

Extend recursively to arbitrary number of refinement levels.



Adaptive integration algorithm

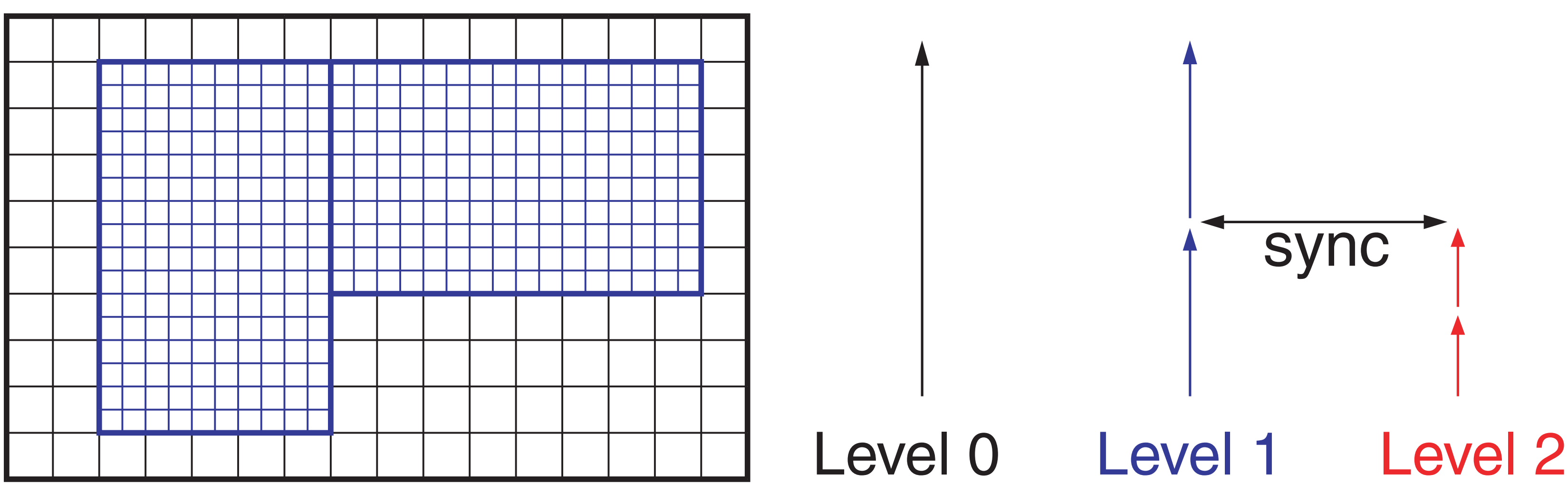
Consider two levels, coarse and fine, with refinement ratio r

$$\Delta x_f = \Delta x_c / r \quad , \quad \Delta t_f = \Delta t_c / r,$$

To integrate

- Advance coarse grids in time $t_c \to t_c + \Delta t_c$
- Advance fine grids in time r times
- Synchronize coarse and fine data

Extend recursively to arbitrary number of refinement levels.



Adaptive integration algorithm

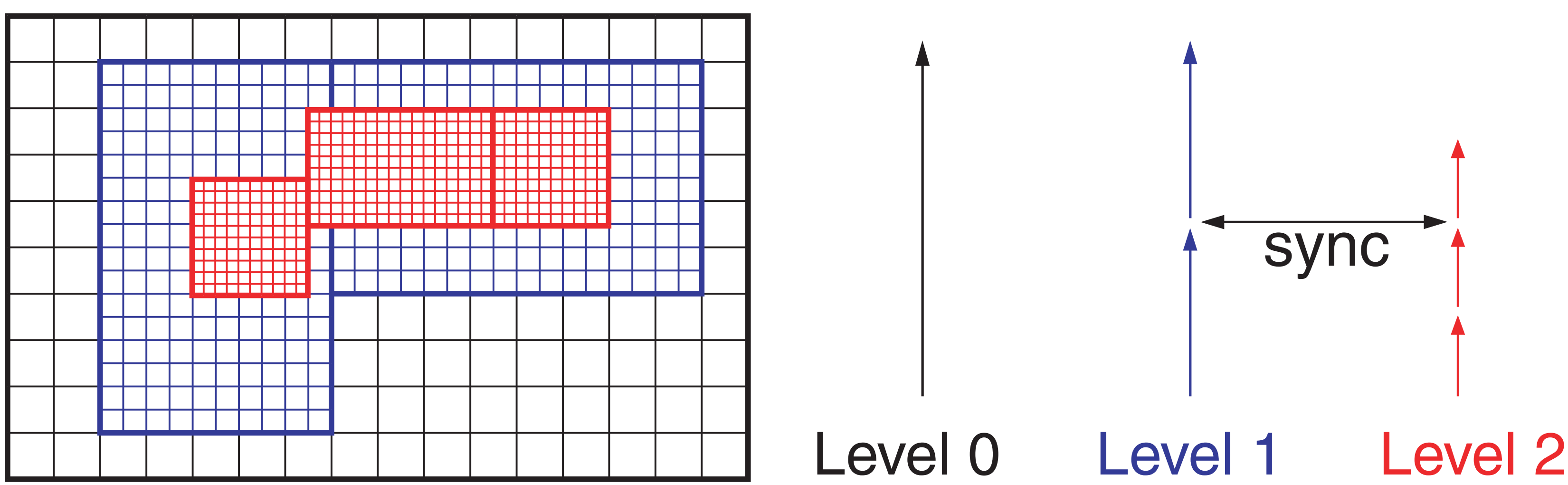
Consider two levels, coarse and fine, with refinement ratio r

$$\Delta x_f = \Delta x_c / r \quad , \quad \Delta t_f = \Delta t_c / r,$$

To integrate

- Advance coarse grids in time $t_c \rightarrow t_c + \Delta t_c$
- Advance fine grids in time r times
- Synchronize coarse and fine data

Extend recursively to arbitrary number of refinement levels.



Adaptive integration algorithm

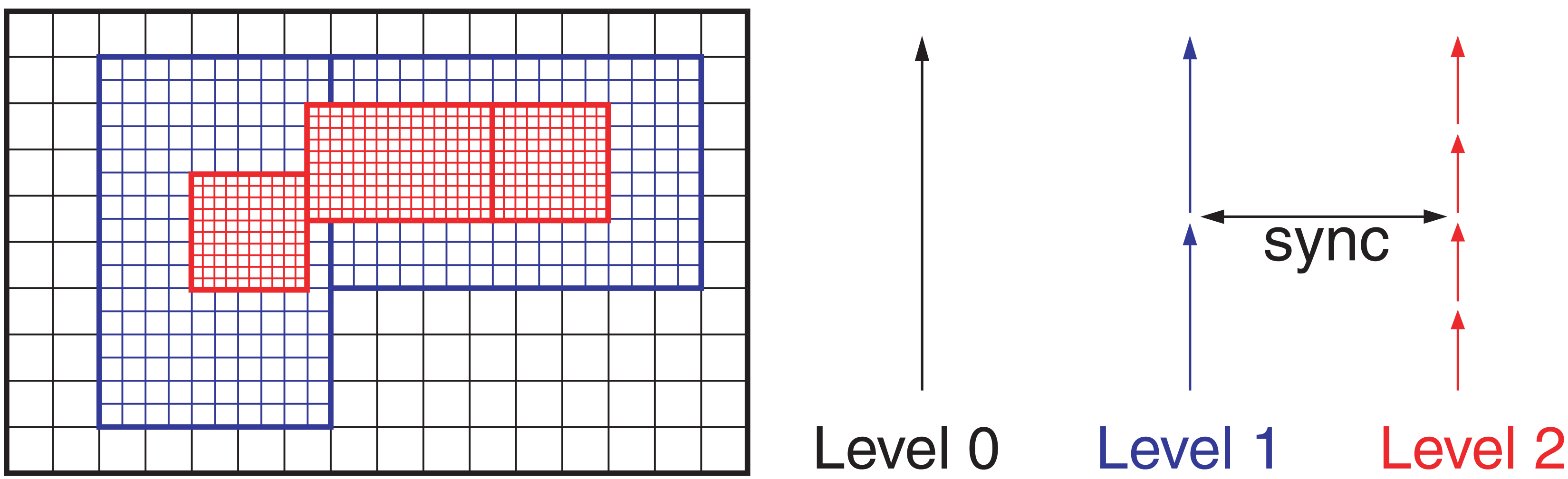
Consider two levels, coarse and fine, with refinement ratio r

$$\Delta x_f = \Delta x_c / r \quad , \quad \Delta t_f = \Delta t_c / r,$$

To integrate

- Advance coarse grids in time $t_c \rightarrow t_c + \Delta t_c$
- Advance fine grids in time r times
- Synchronize coarse and fine data

Extend recursively to arbitrary number of refinement levels.



Adaptive integration algorithm

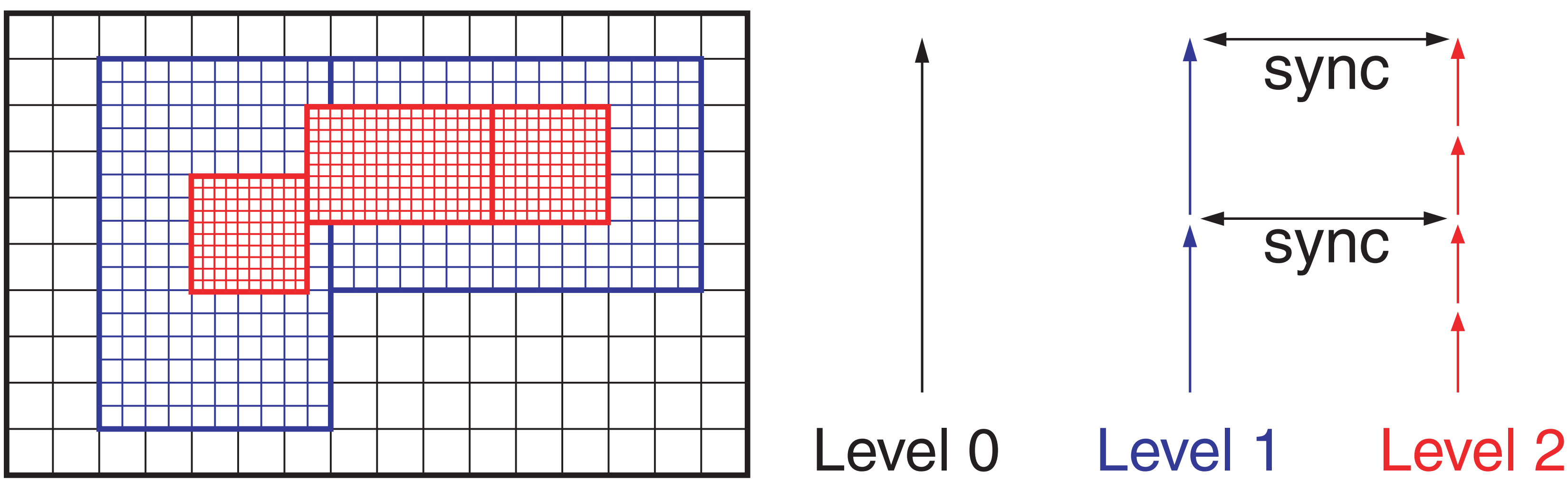
Consider two levels, coarse and fine, with refinement ratio r

$$\Delta x_f = \Delta x_c / r \quad , \quad \Delta t_f = \Delta t_c / r,$$

To integrate

- Advance coarse grids in time $t_c \rightarrow t_c + \Delta t_c$
- Advance fine grids in time r times
- Synchronize coarse and fine data

Extend recursively to arbitrary number of refinement levels.



Adaptive integration algorithm

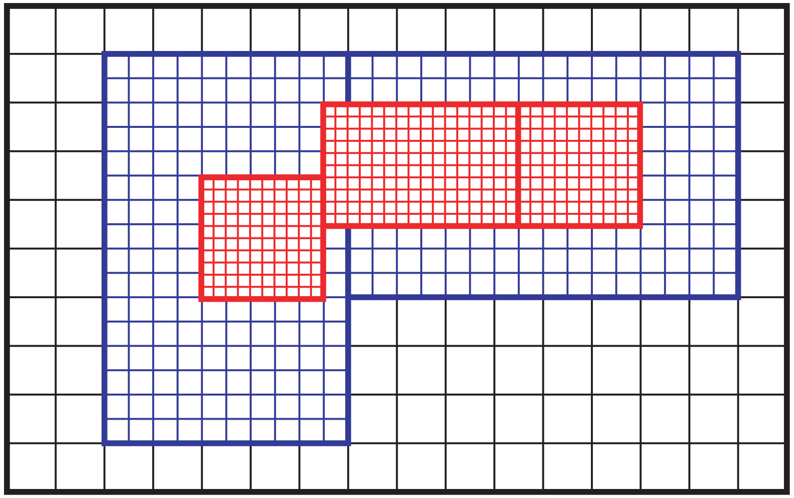
Consider two levels, coarse and fine, with refinement ratio r

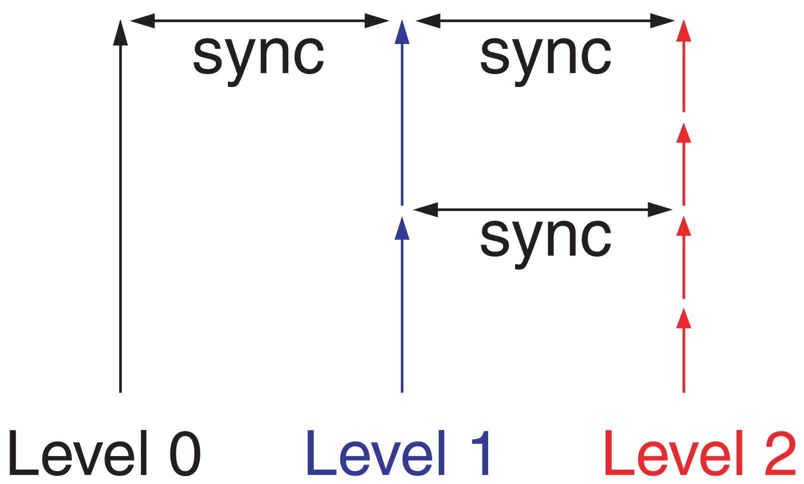
$$\Delta x_f = \Delta x_c / r \quad , \quad \Delta t_f = \Delta t_c / r,$$

To integrate

- Advance coarse grids in time $t_c \to t_c + \Delta t_c$
- Advance fine grids in time r times
- Synchronize coarse and fine data

Extend recursively to arbitrary number of refinement levels.





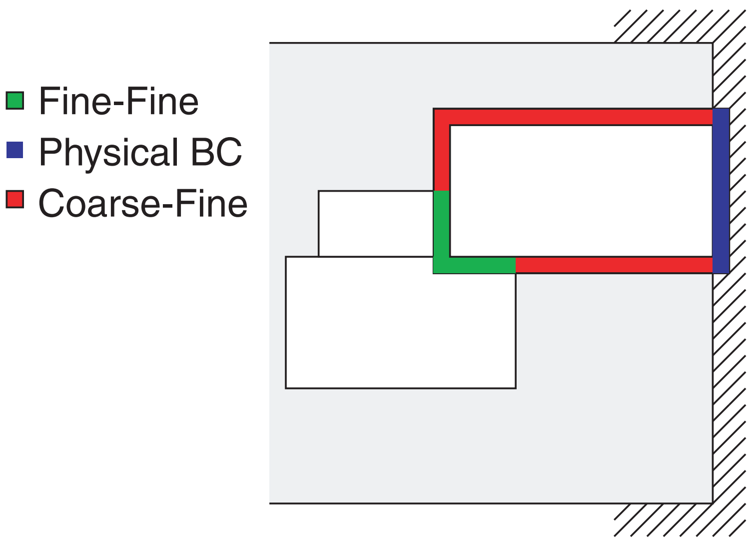
Integrating on grid patch

How do you integrate a patch of data at level ℓ?

Obtain boundary data needed to call integrator on uniform grid of data.

- Assume explicit scheme with stencil width s_d

- Enlarge patch by s_d cells in each direction and fill with data using
  - Physical boundary conditions
  - Other patches at the same level
  - Coarse grid data (fillpatch)

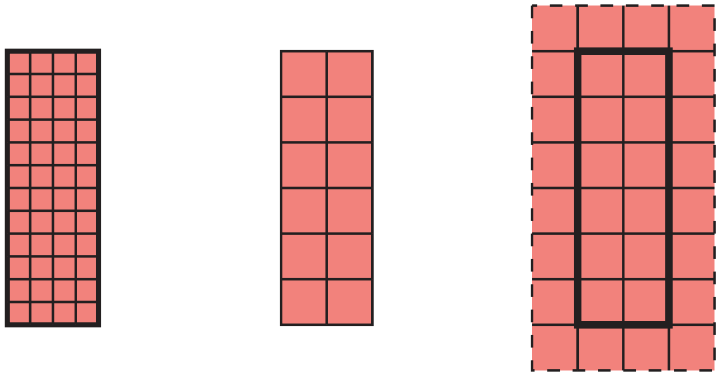


- Advance grid in time $t \rightarrow t + \Delta t$

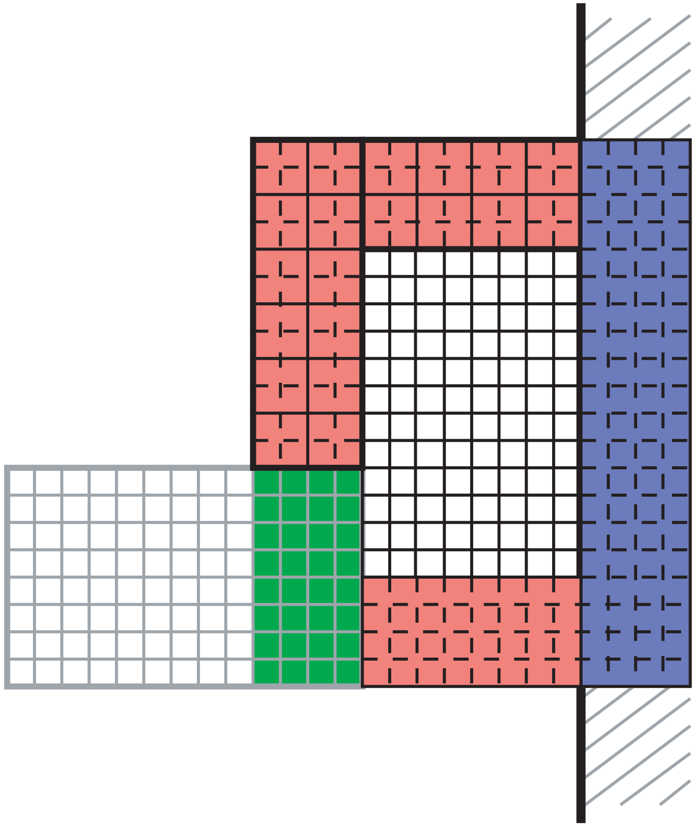
FillPatch Operation

To fill fine grid "ghost cells" at $t+k\Delta t_f$, $k = 0, ..., r-1$, using coarse grid data

- Define coarse patch needed for interpolation



- Fill coarse patch at time t and $t + \Delta t_c$
- Time-interpolate data on coarse patch to time $t+k\Delta t_f$
- Interpolate coarse data to fine patch



Synchronization

After coarse grid time step and the subcycled advance of fine data, we have

- U^c at t_c^{n+1}
- U^f at t_c^{n+1}

However, U^c and U^f are not consistent

- Coarse data is not necessarily equal to the average of the fine grid data “over” it.
- Scheme violates conservation because of inconsistent fluxes at coarse-fine interface

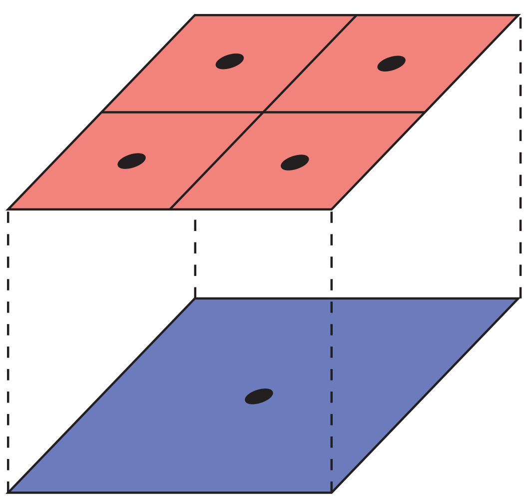
Synchronization

After coarse grid time step and the subcycled advance of fine data, we have

- U^c at t_c^{n+1}
- U^f at t_c^{n+1}

However, U^c and U^f are not consistent

- Coarse data is not necessarily equal to the average of the fine grid data "over" it.
- Scheme violates conservation because of inconsistent fluxes at coarse-fine interface



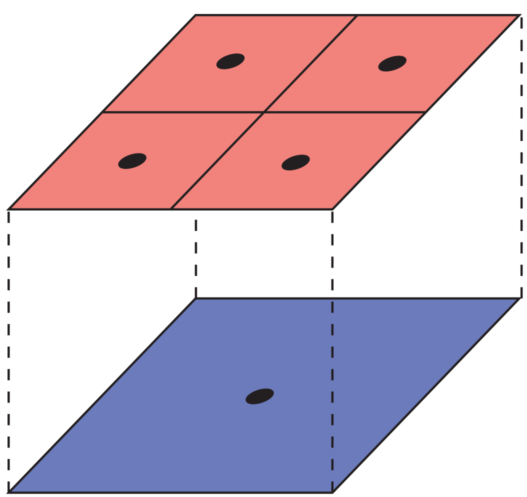
Synchronization

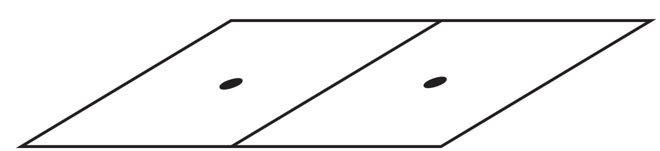
After coarse grid time step and the subcycled advance of fine data, we have

- U^c at t_c^{n+1}
- U^f at t_c^{n+1}

However, U^c and U^f are not consistent

- Coarse data is not necessarily equal to the average of the fine grid data “over” it.
- Scheme violates conservation because of inconsistent fluxes at coarse-fine interface





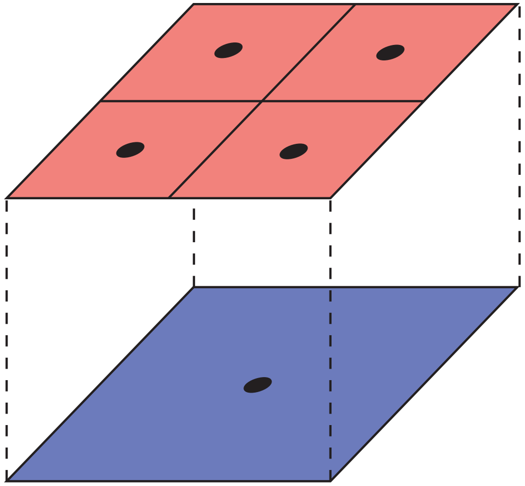
Synchronization

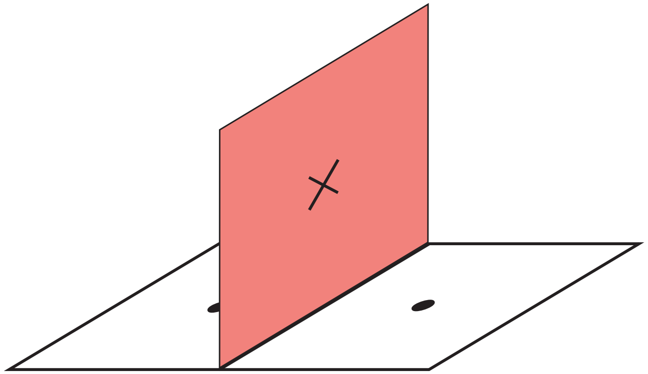
After coarse grid time step and the subcycled advance of fine data, we have

- U^c at t_c^{n+1}
- U^f at t_c^{n+1}

However, U^c and U^f are not consistent

- Coarse data is not necessarily equal to the average of the fine grid data "over" it.
- Scheme violates conservation because of inconsistent fluxes at coarse-fine interface





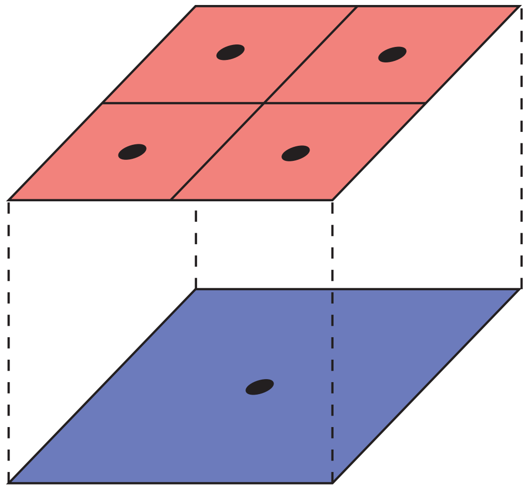
Synchronization

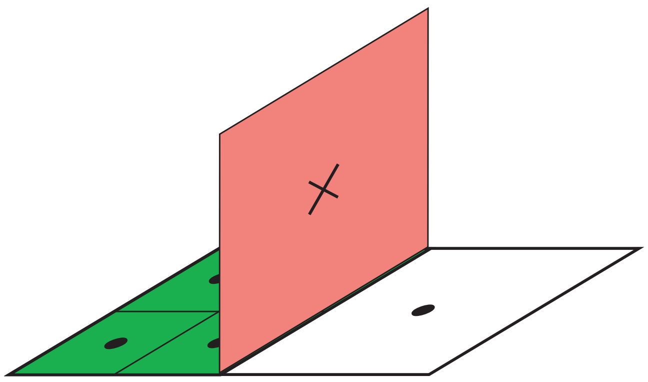
After coarse grid time step and the subcycled advance of fine data, we have

- U^c at t_c^{n+1}
- U^f at t_c^{n+1}

However, U^c and U^f are not consistent

- Coarse data is not necessarily equal to the average of the fine grid data "over" it.
- Scheme violates conservation because of inconsistent fluxes at coarse-fine interface





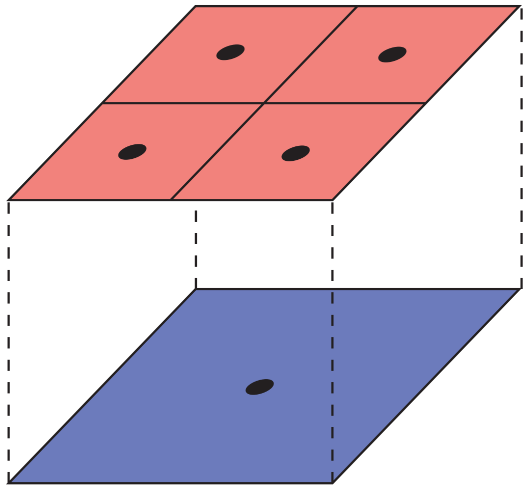
Synchronization

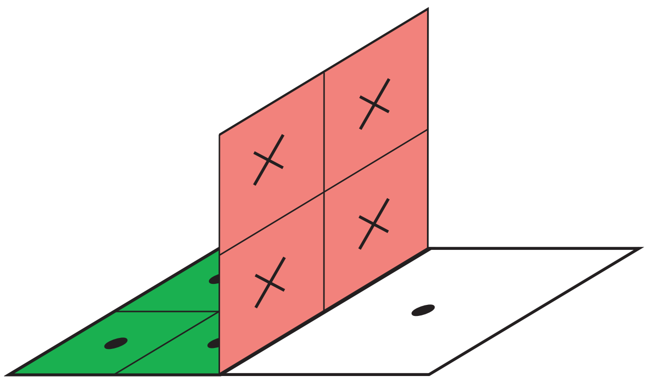
After coarse grid time step and the subcycled advance of fine data, we have

- U^c at t_c^{n+1}
- U^f at t_c^{n+1}

However, U^c and U^f are not consistent

- Coarse data is not necessarily equal to the average of the fine grid data “over” it.
- Scheme violates conservation because of inconsistent fluxes at coarse-fine interface





Synchronization (p2)

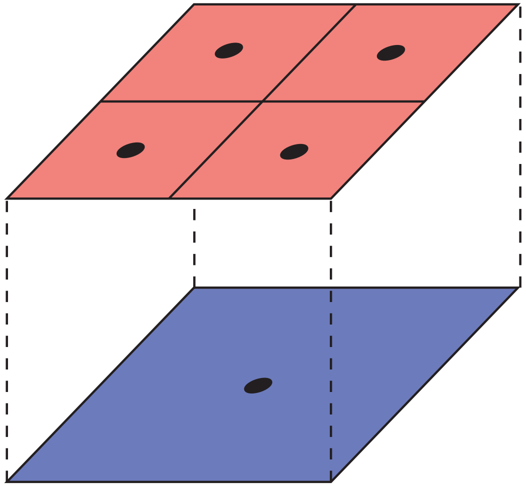
How do we address these problems with the solution?

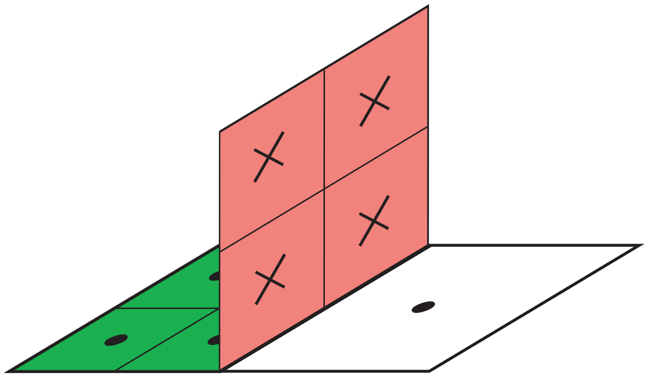
- Average down the fine grid data onto all underlying coarse cells

$$U^c = \frac{1}{r^d} \sum U^f$$

- Reflux at coarse-fine interfaces

$$\Delta x_c \Delta y_c U^c = \Delta x_c \Delta y_c U^c - \Delta t^c A^c \mathbf{F}^c + \sum \Delta t^f A^f \mathbf{F}^f$$





Regridding

Compute "error" at each cell : if "too big" then flag cell for refinement

- Richardson extrapolation
  - Coarsen data on a patch at t^{n-1} and advance by $2\Delta t$
  - Advance data at t^n by Δt and coarsen
  - Difference of these two solutions is proportional to error
- Functions of solution (e.g., vorticity)
- Geometric considerations

Compute refined patches as initially

Fill newly refined regions using conservative interpolation from coarse grid

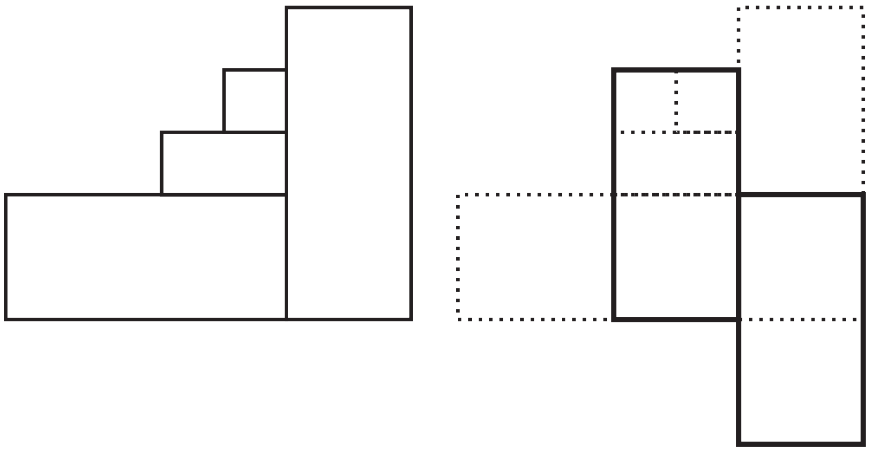
Regridding

Compute "error" at each cell : if "too big" then flag cell for refinement

- Richardson extrapolation
  - Coarsen data on a patch at t^{n-1} and advance by $2\Delta t$
  - Advance data at t^n by Δt and coarsen
  - Difference of these two solutions is proportional to error
- Functions of solution (e.g., vorticity)
- Geometric considerations

Compute refined patches as initially

Fill newly refined regions using conservative interpolation from coarse grid



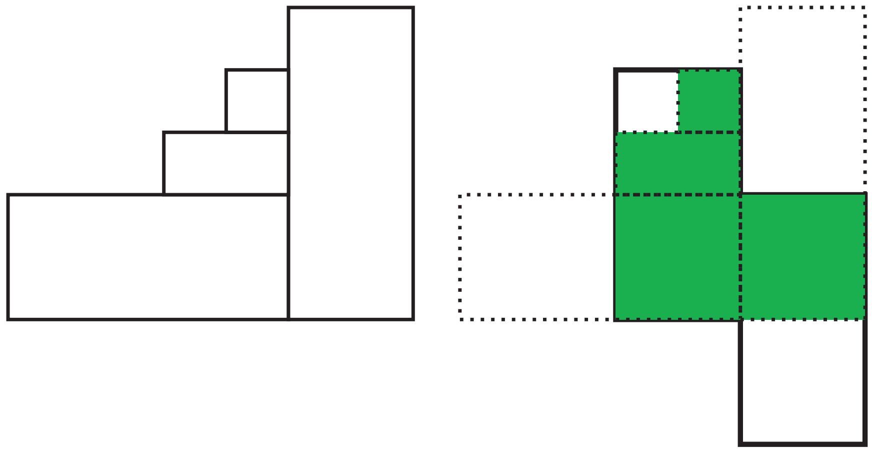
Regridding

Compute “error” at each cell : if “too big” then flag cell for refinement

- Richardson extrapolation
  - Coarsen data on a patch at t^{n-1} and advance by $2\Delta t$
  - Advance data at t^n by Δt and coarsen
  - Difference of these two solutions is proportional to error
- Functions of solution (e.g., vorticity)
- Geometric considerations

Compute refined patches as initially

Fill newly refined regions using conservative interpolation from coarse grid



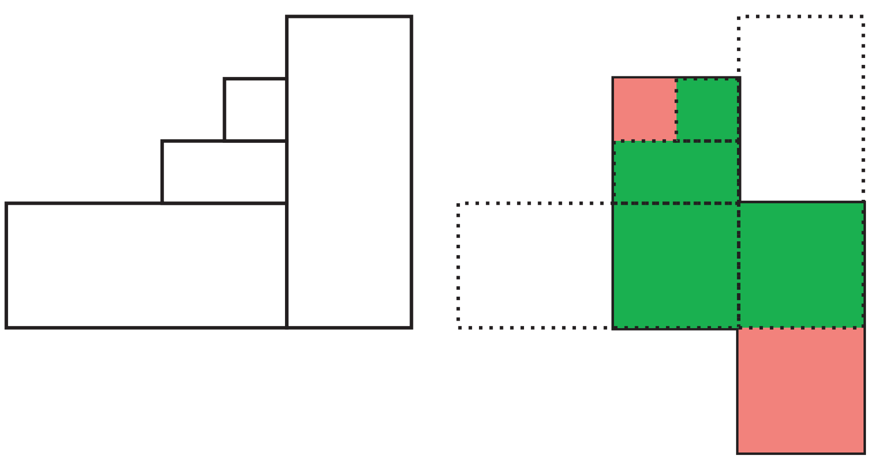
Regridding

Compute "error" at each cell : if "too big" then flag cell for refinement

- Richardson extrapolation
  - Coarsen data on a patch at t^{n-1} and advance by $2\Delta t$
  - Advance data at t^n by Δt and coarsen
  - Difference of these two solutions is proportional to error
- Functions of solution (e.g., vorticity)
- Geometric considerations

Compute refined patches as initially

Fill newly refined regions using conservative interpolation from coarse grid



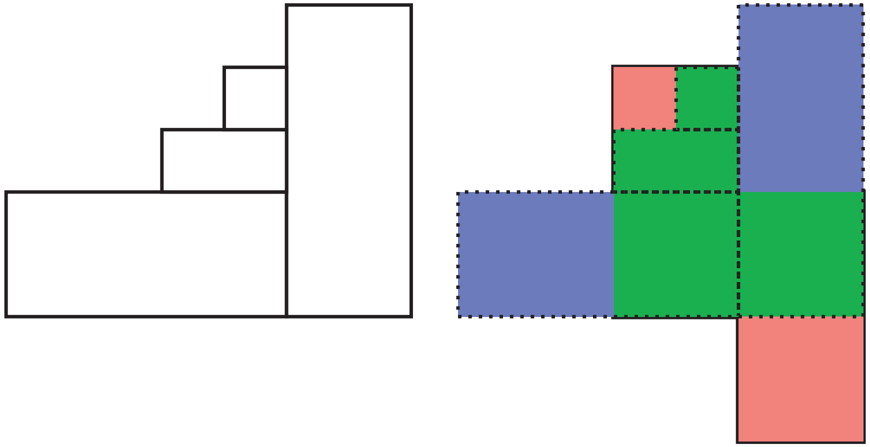
Regridding

Compute "error" at each cell : if "too big" then flag cell for refinement

- Richardson extrapolation
  - Coarsen data on a patch at t^{n-1} and advance by $2\Delta t$
  - Advance data at t^n by Δt and coarsen
  - Difference of these two solutions is proportional to error
- Functions of solution (e.g., vorticity)
- Geometric considerations

Compute refined patches as initially

Fill newly refined regions using conservative interpolation from coarse grid



Summary of Algorithm

Hyperbolic AMR

For $n = 1, \ldots, N_{final}$
  Advance$(0, t_0^n)$

Advance (ℓ, t)
  If (time to regrid) **then**
    Regrid(ℓ)
  FillPatch$(\ell$,t)
  Integrate$(\ell, t, \Delta t_\ell)$
  If $(\ell < \ell_{finest})$ **then**
    For $i_{sub} = 1, \ldots, r_\ell$
      Advance$(\ell{+}1,\ t + (i_{sub} - 1)\Delta t_{\ell+1})$
    Average down$(\ell, t + \Delta t_\ell)$
    Reflux$(\ell, t + \Delta t_\ell)$
  End If

Regrid(ℓ): generate new grids at levels $\ell+1$ and higher

FillPatch$(\ell$,t): fill patch of data at level ℓ and time t

Integrate$(\ell, t, \Delta t)$: Advance data at level ℓ from t to $t + \Delta t$, averaging and storing fluxes at boundaries of level ℓ grids if $\ell > 0$ and level ℓ cells at boundary of $\ell + 1$

Average down(ℓ, t): average (in space) level $\ell+1$ data at time t to level ℓ

Reflux(ℓ, t): Add (time- and space-averaged) refluxing corrections to level ℓ cells at time t adjacent to level $\ell+1$ grids

Review of Data Operations

Single-level operations

- Fill a patch with data from same-level grids
- Fill data using physical boundary conditions
- Interpolate data in time
- Add corrections from stored fluxes at same resolution
- Integrate patch of data in time
- Find union of rectangles that contain a set of tagged points

Multi-level operations

- Map regions between different levels of refinement
- Interpolate : coarse $\rightarrow$ fine
- Average : fine $\rightarrow$ coarse
- Store fluxes from fine grid boundaries at coarse resolution

Parallel Issues for AMR

Issues are primarily

- Data distribution – obvious idea is distributing grids to processors
  - Gridding strategy depends on approach to parallelization
    - Pure MPI
    - Hybrid: MPI + OpenMP
  - Size of grids depends on memory usage, parallelization strategy and additional physics (e.g. Poisson solve for self-gravity)
- Dynamic load balancing
  - Need good work estimate
  - Data locality
- Parallel in space, serial in time ...

We will talk more about these next week