Adaptive Mesh Refinement for 1D Hyperbolic PDEs

Saumya Sinha, Kenneth J. Roche
University of Washington
Department of Applied Mathematics
AM574, Prof. R. J. LeVeque

wave propagation method for Riemann Problem

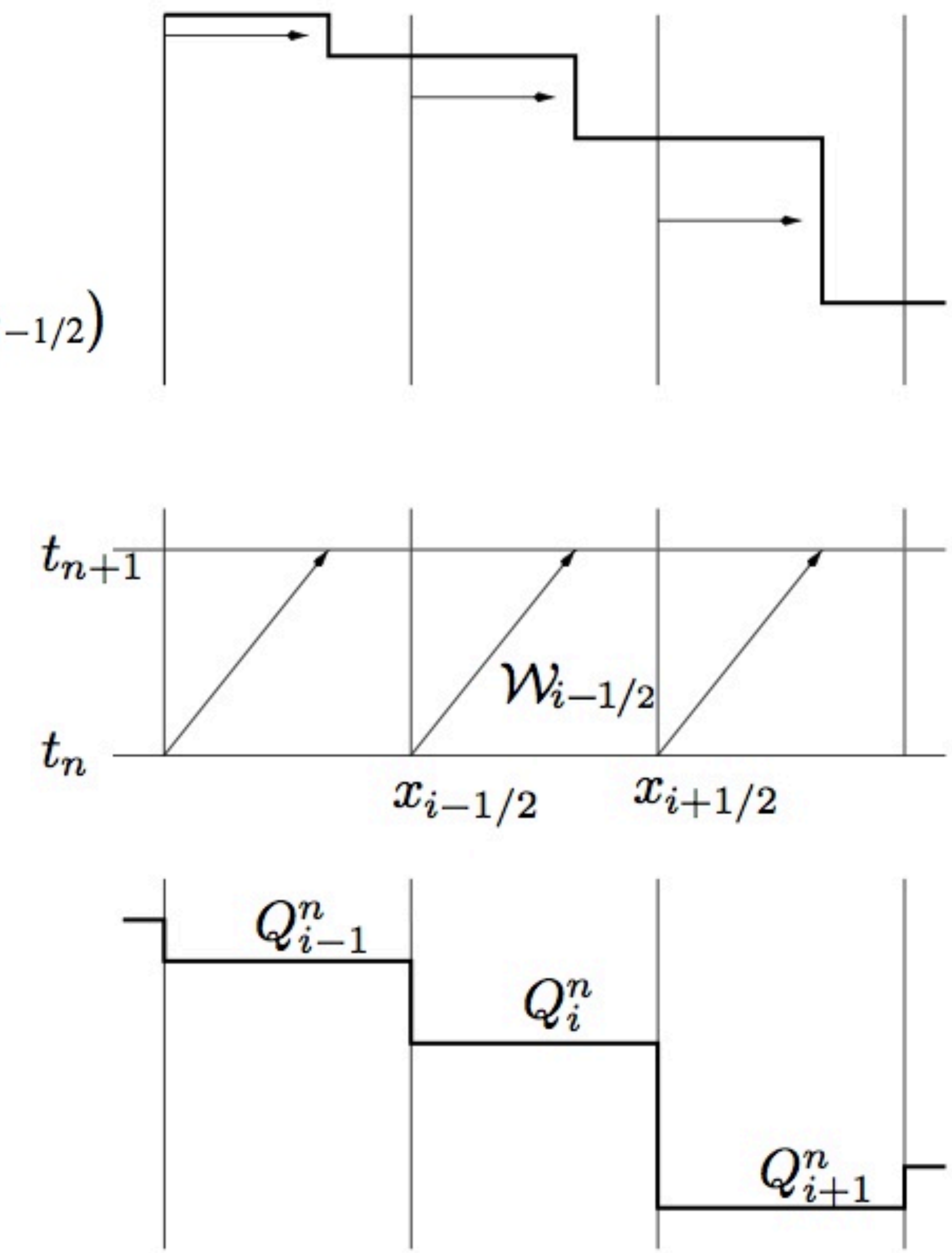
Godunov, “2nd order” correction, limiters

$$Q_i^{n+1} = Q_i - \frac{\Delta t}{\Delta x}\left(A^+\Delta Q_{i-1/2} + A^-\Delta Q_{i+1/2}\right) - \frac{\Delta t}{\Delta x}\left(\tilde{F}_{i+1/2} - \tilde{F}_{i-1/2}\right)$$

$$A^+\Delta Q_{i-1/2} = \sum_{p=1}^{m} (\lambda^p)^+ \mathcal{W}_{i-1/2}^p,$$

$$A^-\Delta Q_{i-1/2} = \sum_{p=1}^{m} (\lambda^p)^- \mathcal{W}_{i-1/2}^p,$$

$$\tilde{F}_{i-1/2} = \frac{1}{2}\sum_{p=1}^{m} |\lambda^p| \left(1 - \frac{\Delta t}{\Delta x}|\lambda^p|\right) \widetilde{\mathcal{W}}_{i-1/2}^p$$



color equation, variable coefficient Riemann problem

$$q(x,t)_t + u(x)q(x,t)_x = 0$$

$$u(x) = \begin{cases} u_{i-1}, & x < x_{i-\frac{1}{2}} \\ u_i, & x > x_{i-\frac{1}{2}} \end{cases} \qquad q(x,0) := \phi(x) = \begin{cases} q_{i-1}, & x < x_{i-\frac{1}{2}} \\ q_i, & x > x_{i-\frac{1}{2}} \end{cases}$$

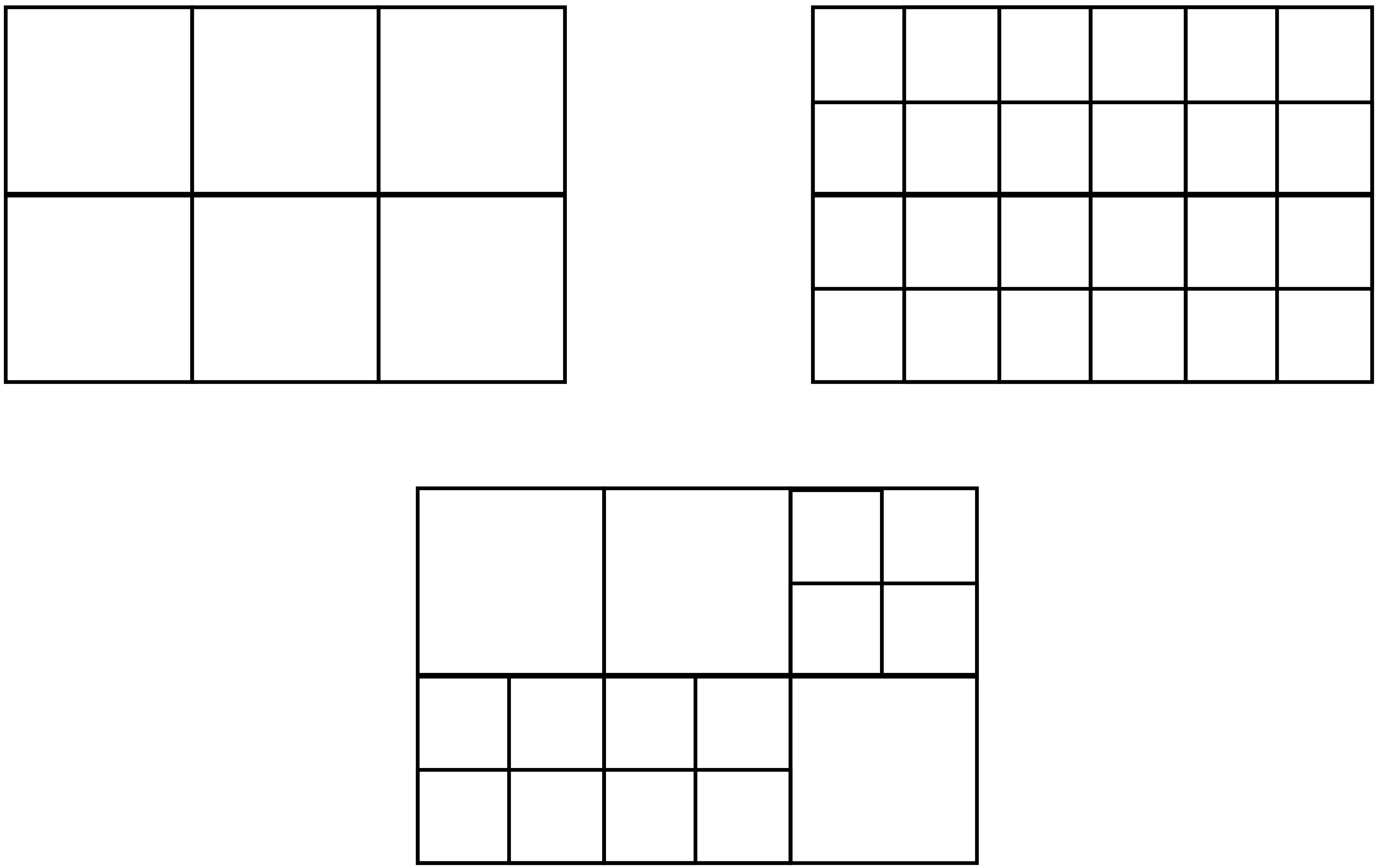
$$q_i^{n+1} = q_i^n - \frac{k}{h}(u_i^+(q_i^n - q_{i-1}^n) + u_i^-(q_{i+1}^n - q_i^n)) - \frac{k}{h}(\tilde{F}_{i+1} - \tilde{F}_i)$$

$$\tilde{F}_i = \frac{1}{2}|u_i|(1 - \frac{k}{h}|u_i|)\tilde{W}_i$$

$$\tilde{W}_i = \begin{cases} limiter(W_i, W_{i-1}), & u_i > 0 \\ limiter(W_i, W_{i+1}), & u_i < 0 \end{cases}$$

Why AMR

- resolve sharp discontinuities without all the overhead
- keep track of physically interesting regions



The remainder of the talk ...

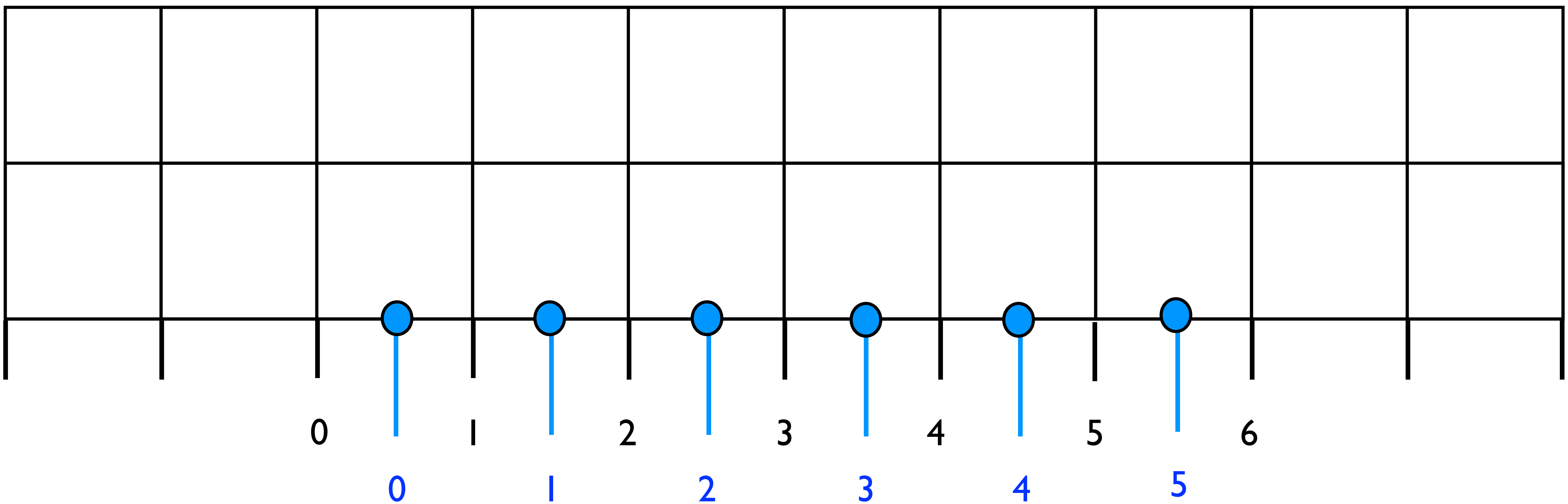
- detailed breakdown of 1D AMR algorithm

- talk about our implementation efforts (ehem)

- some results

AMR steps : initialize values in each cell

(a) initialize coarsest grid (called level $i = 0$)



initial values of q

ncells=6

dx=1/mx

xc=dx/2,3dx/2,...

AMR step : flag, cluster, initialize fine grids

(b) *recurse*: if level ($i > levels_{max}$) *break*

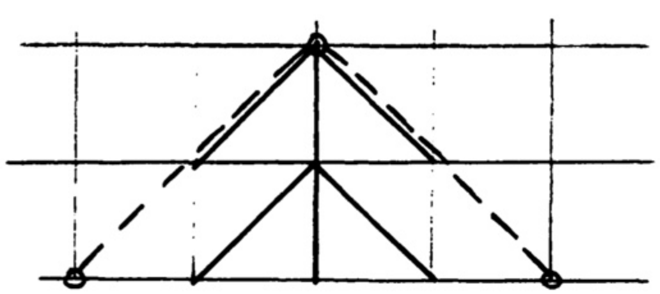
- flag cells for current level i
  - estimate error in each level i cell (compare $D^2(i)$,$D_{2h(i)}$)
    - ∗ if error condition violated, bump $flagged$
- if ($flagged > 0$) construct level $i+1$ grid
  - cluster (based on buffer region and ghost cells) level i grid cells
  - initialize $i+1$ grid for current time
    - ∗ linear interpolation using level i values at current time
- $i \leftarrow i + 1$

AMR step : flag cells at current level for refinement

(b) *recurse*: if level ($i > levels_{max}$) *break*

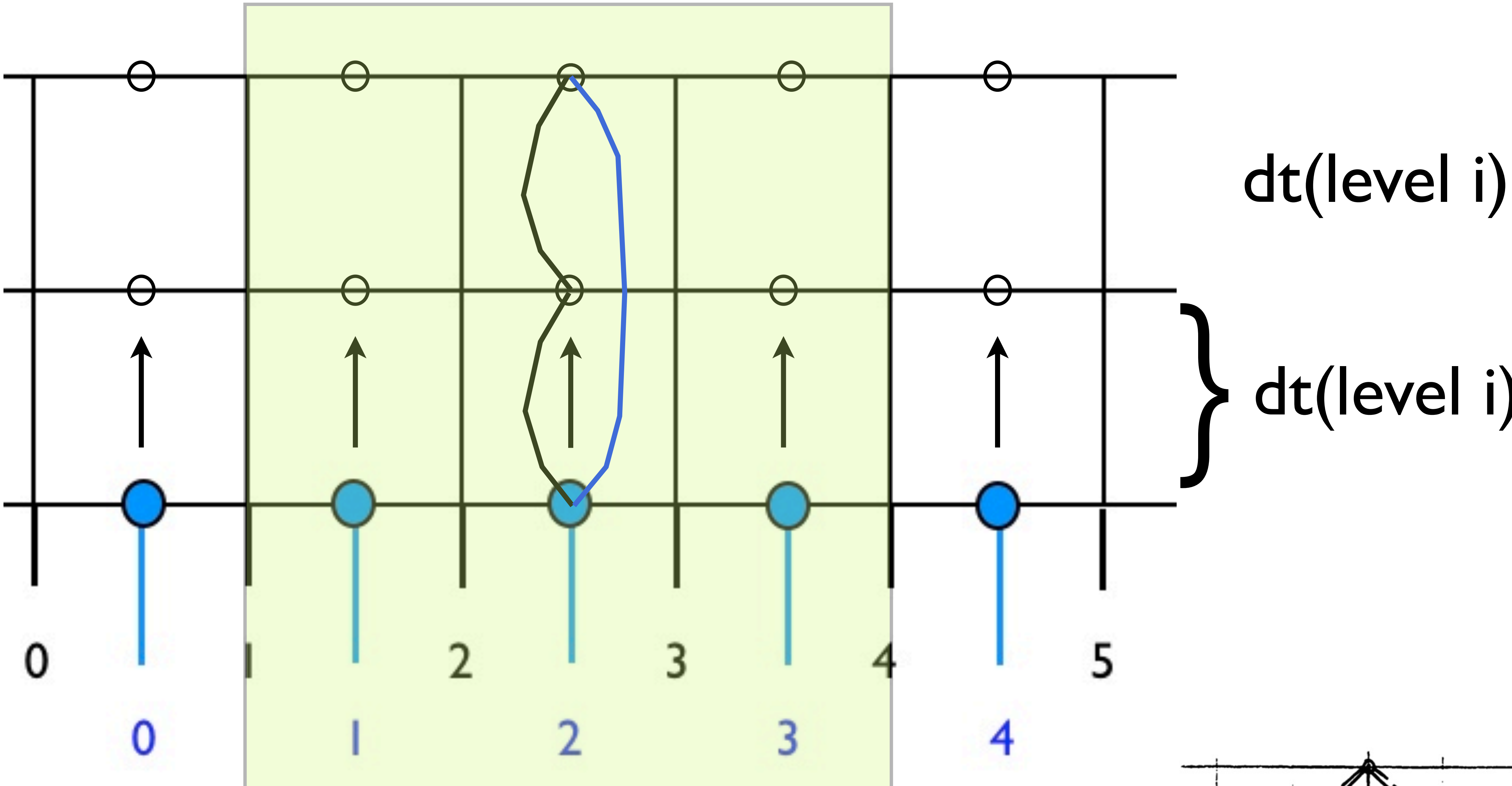
- flag cells for current level i
  - estimate error in each level i cell (compare $D^2(i)$,$D_{2h(i)}$)
    - if error condition violated, bump *flagged*
- if ($flagged > 0$) construct level $i+1$ grid
  - cluster (based on buffer region and ghost cells) level i grid cells
  - initialize $i+1$ grid for current time
    - linear interpolation using level i values at current time
- $i \leftarrow i+1$

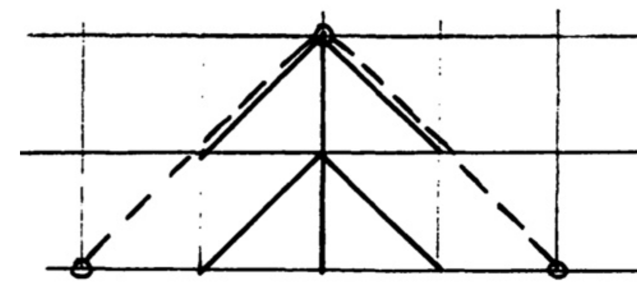
$$\frac{D^2 q(x,t) - D_{2h} q(x,t)}{2^{s+1} - 2} = \tau(x,t) + O(h^{s+2})$$



D := $\quad Q_i^{n+1} = Q_i - \frac{\Delta t}{\Delta x}(A^+ \Delta Q_{i-1/2} + A^- \Delta Q_{i+1/2}) - \frac{\Delta t}{\Delta x}(\tilde{F}_{i+1/2} - \tilde{F}_{i-1/2})$

estimate error in each cell





- stored
- temporary

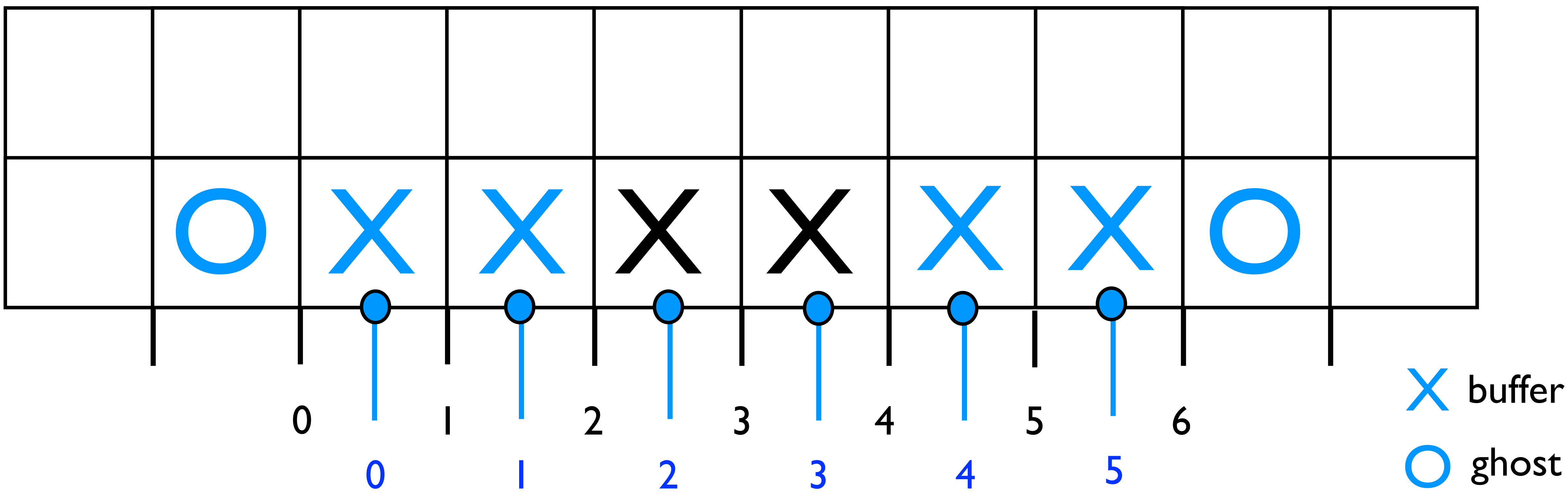
$$\frac{\boxed{D^2 q}(x,t) - \boxed{D_{2h} q}(x,t)}{2^{s+1}-2} = \tau(x,t) + O(h^{s+2})$$

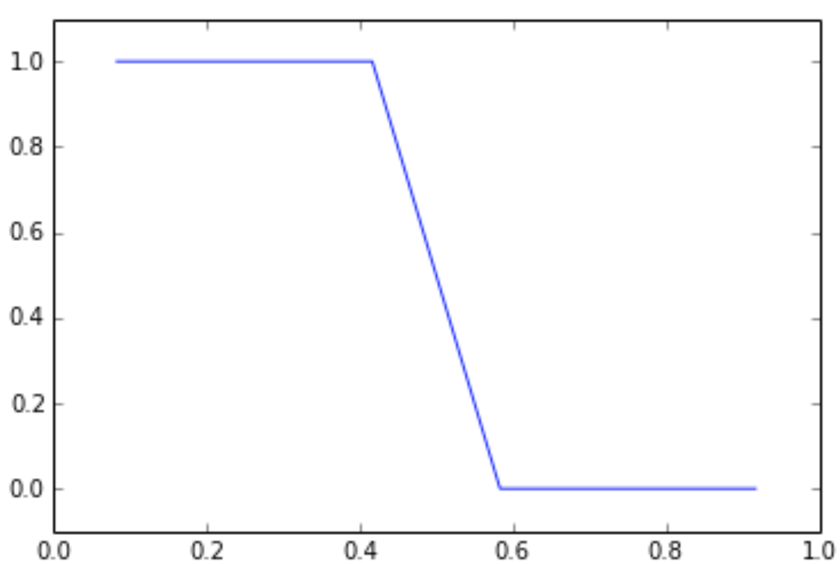
AMR step : cluster flagged cells

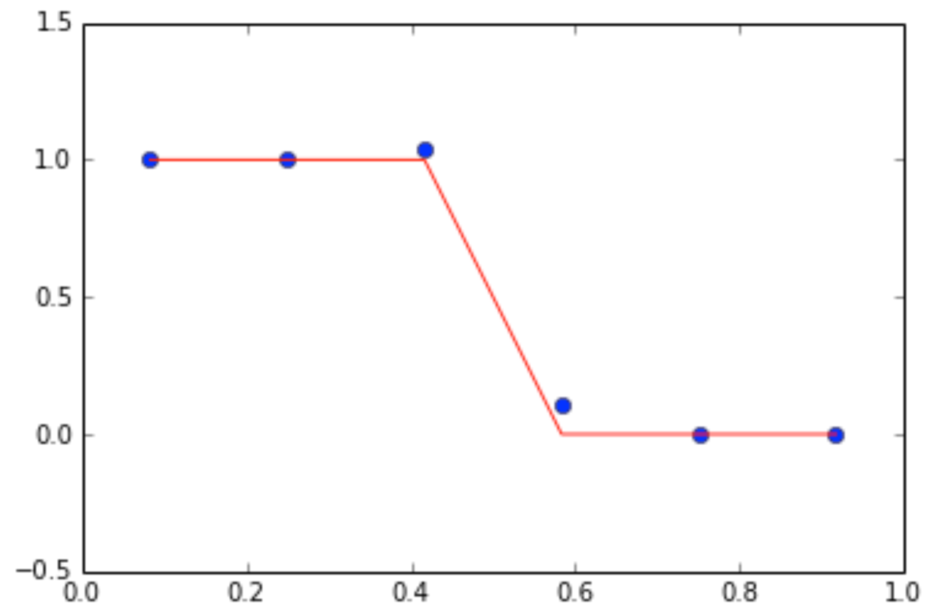
(b) *recurse*: if level ($i > levels_{max}$) *break*

- flag cells for current level i
  - estimate error in each level i cell (compare $D^2(i)$,$D_{2h(i)}$)
    - if error condition violated, bump $flagged$
- if ($flagged > 0$) construct level $i+1$ grid
  - cluster (based on buffer region and ghost cells) level i grid cells
  - initialize $i+1$ grid for current time
    - linear interpolation using level i values at current time
- $i \leftarrow i+1$

AMR step : cluster flagged cells, determine L(i+1) grid

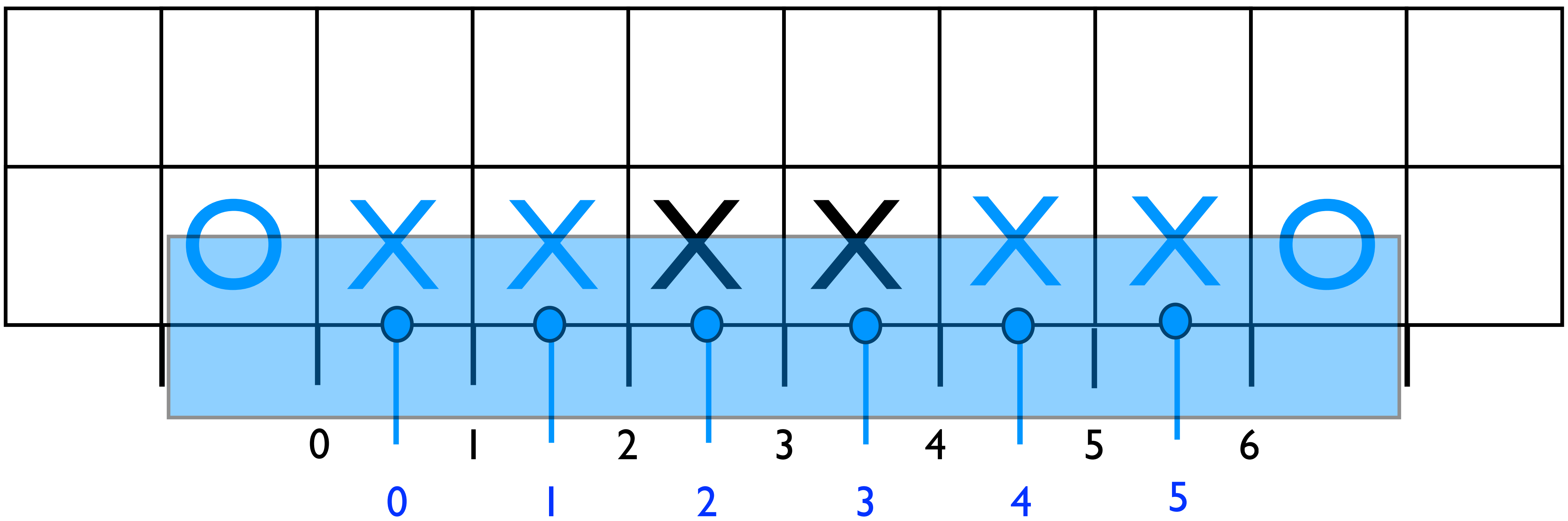


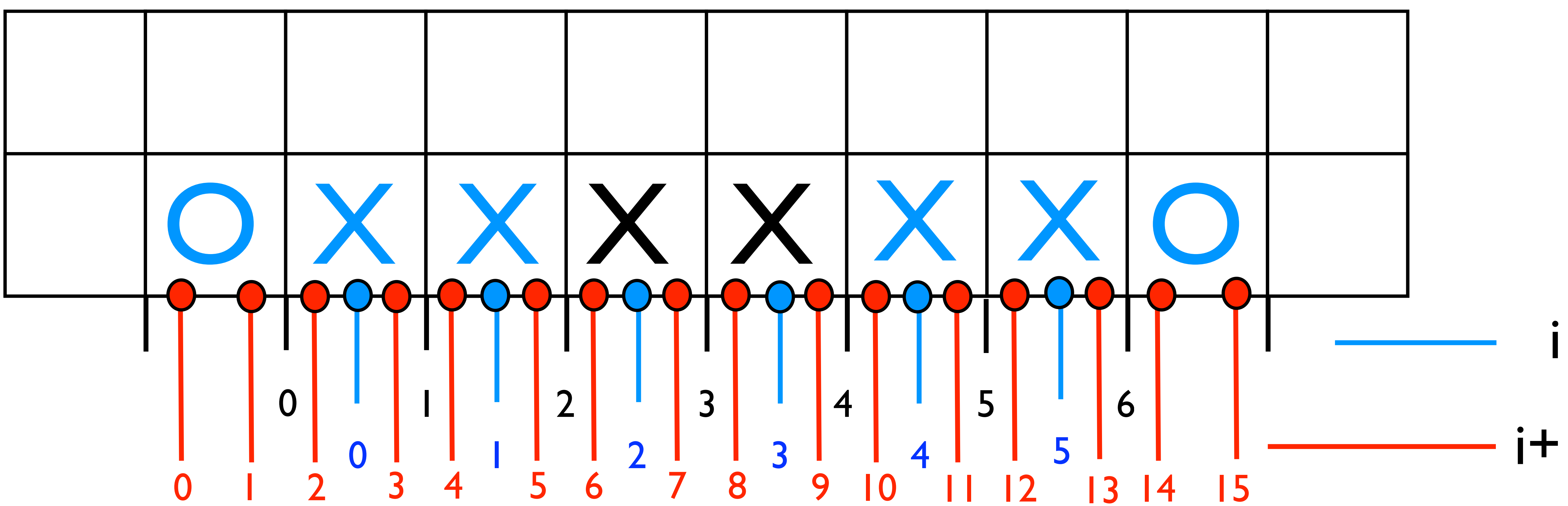




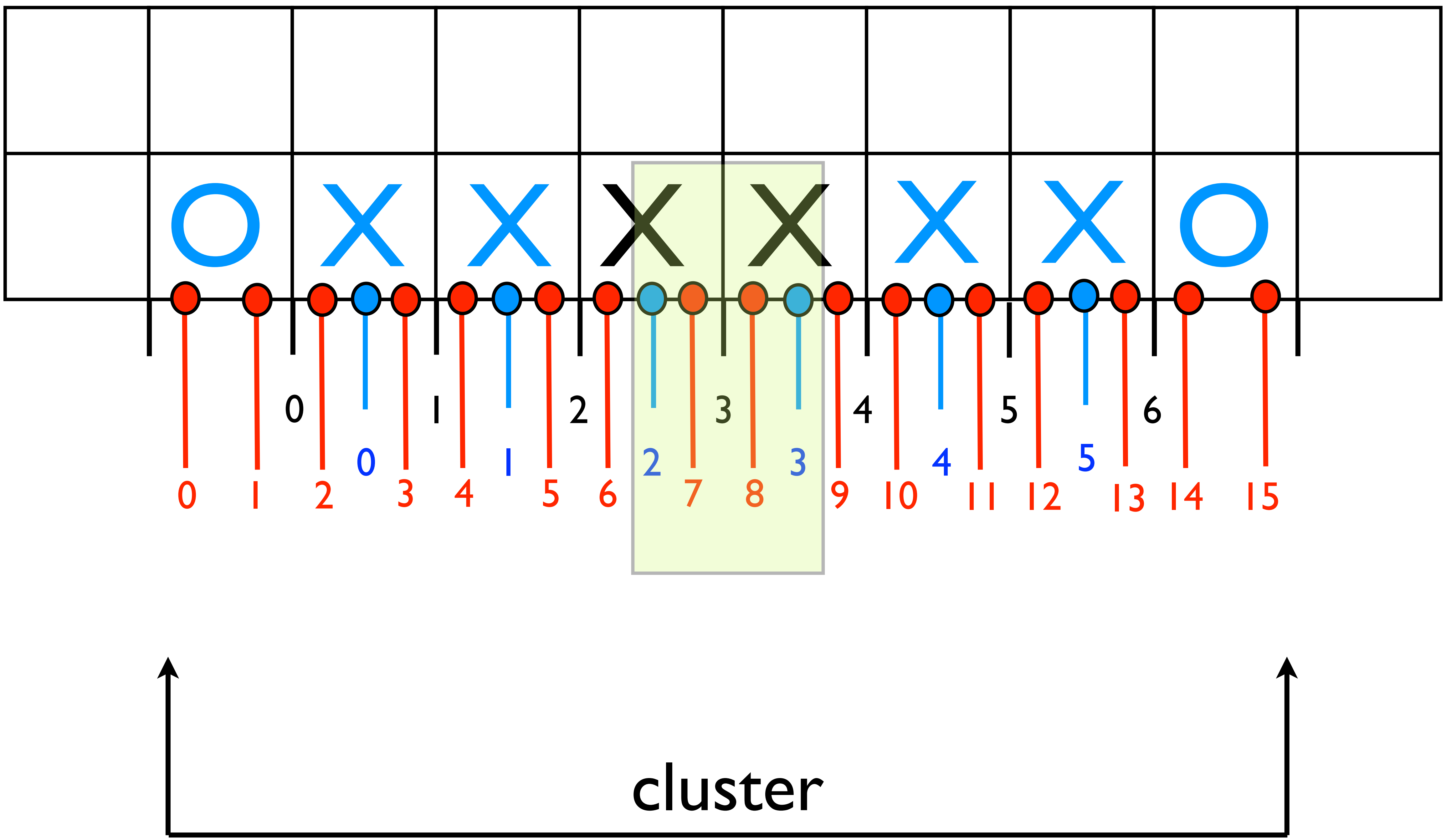
[1. 1. 1. 1. 1. 0. 0. 0. 0. 0.]
2 number of flagged cells
[2. 3.]
start of cluster 0
end of cluster 5
-1 6
cluster, fine grid size 0 16

AMR step : cluster flagged cells, determine L(i+1) grid





AMR step : initialize L(i+1) grid - interpolate L(i)



AMR step : do it all again until happy

(b) *recurse*: if level ($i > levels_{max}$) *break*

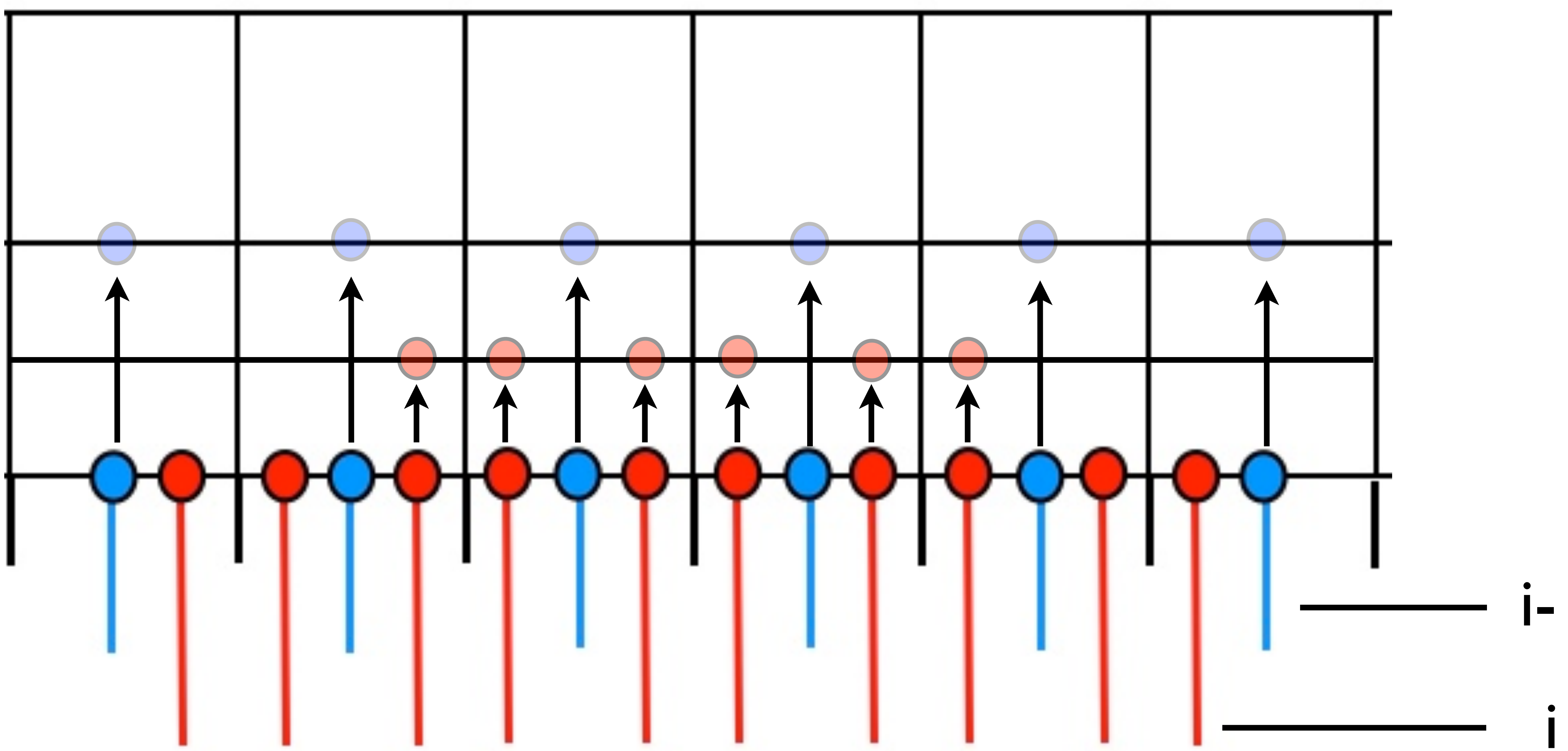
- flag cells for current level i
  - estimate error in each level i cell (compare $D^2(i)$,$D_{2h(i)}$)
    - if error condition violated, bump *flagged*
- if ($flagged > 0$) construct level $i+1$ grid
  - cluster (based on buffer region and ghost cells) level i grid cells
  - initialize $i+1$ grid for current time
    - linear interpolation using level i values at current time

- $i \leftarrow i + 1$

AMR step : intermediate time stepping

(c) partial time evolution

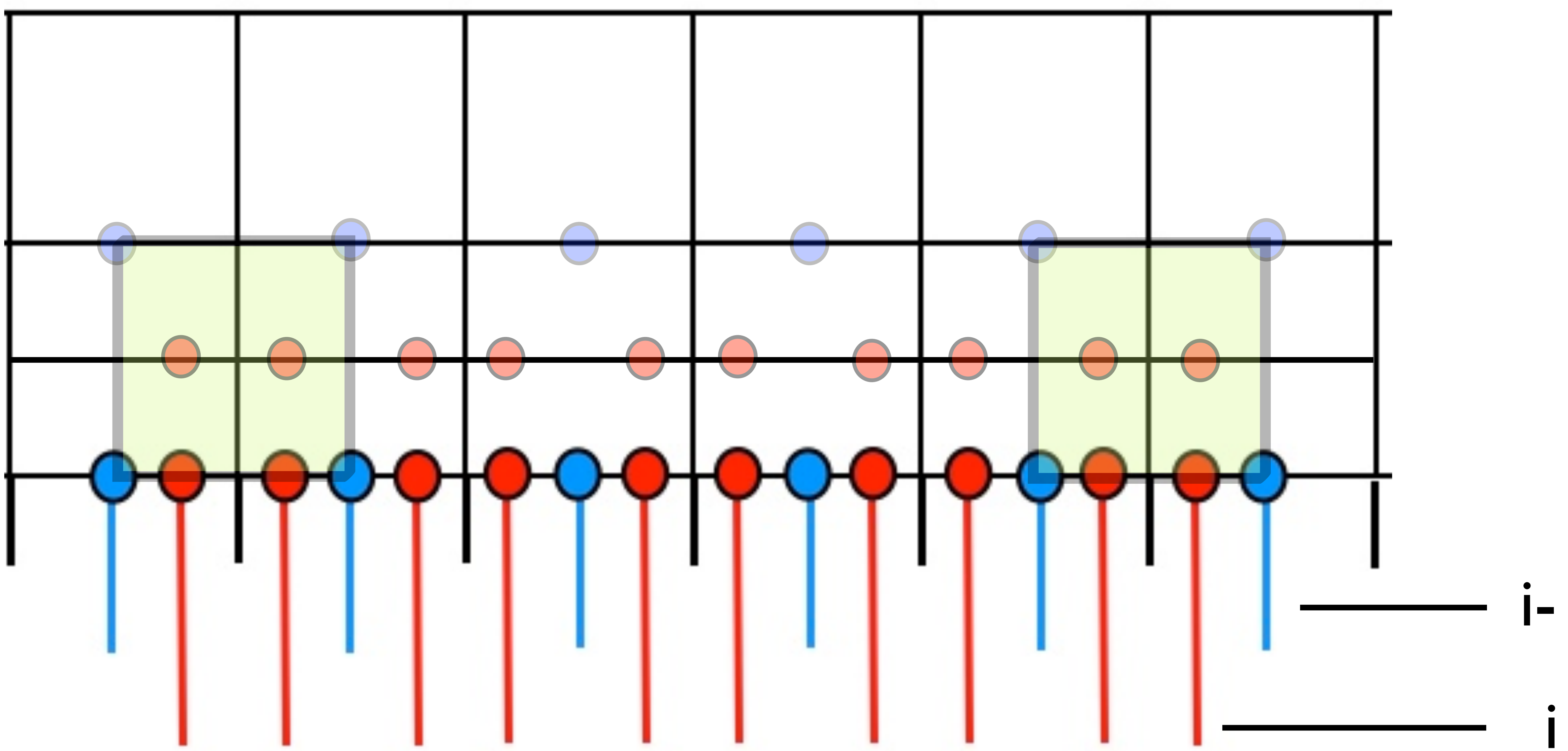
- $\forall levels(i)$ take single $dt(level(i))$ step by evaluating method
  - at fine-coarse interfaces evaluate points at $dt(level(i))$ time
    - requires interpolating function formed with bilinear interpolation with $level(i-1)$ values



AMR step : intermediate time stepping

(c) partial time evolution

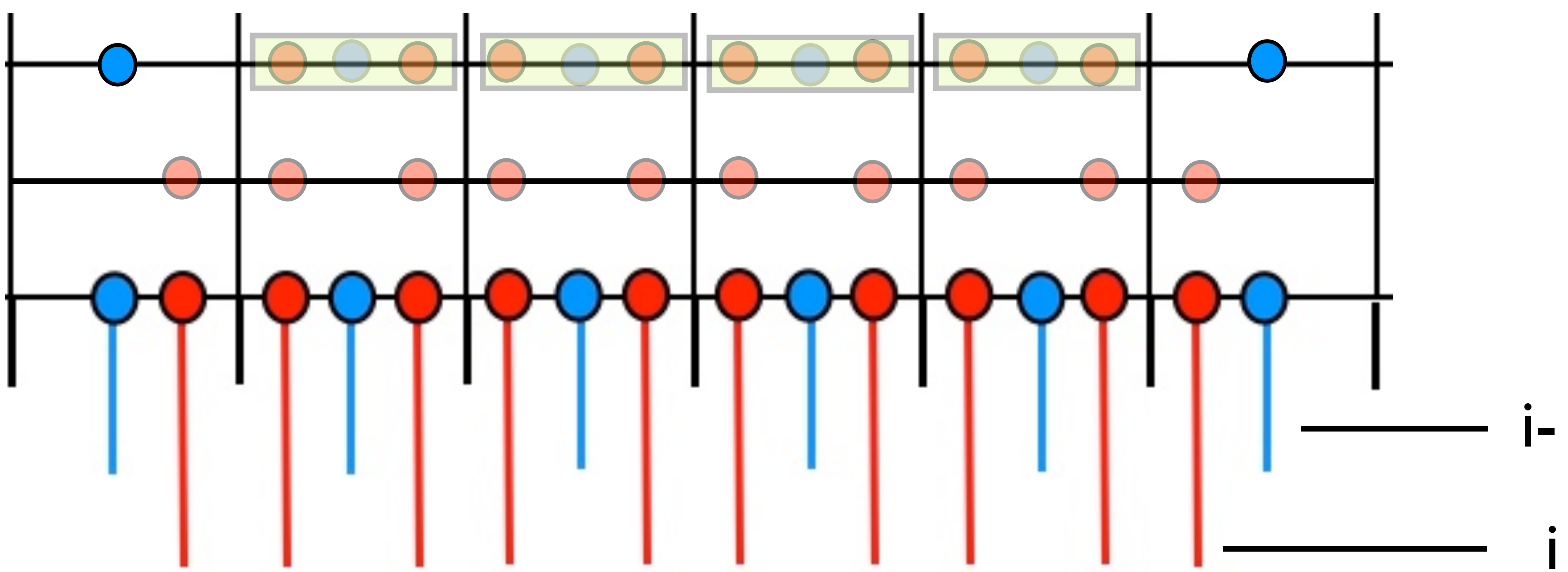
- $\forall levels(i)$ take single $dt(level(i))$ step by evaluating method
  - at fine-coarse interfaces evaluate points at $dt(level(i))$ time
    - requires interpolating function formed with bilinear interpolation with $level(i-1)$ values



AMR step : promote solutions to full time step

(d) promote grid solutions per grid level to full time step

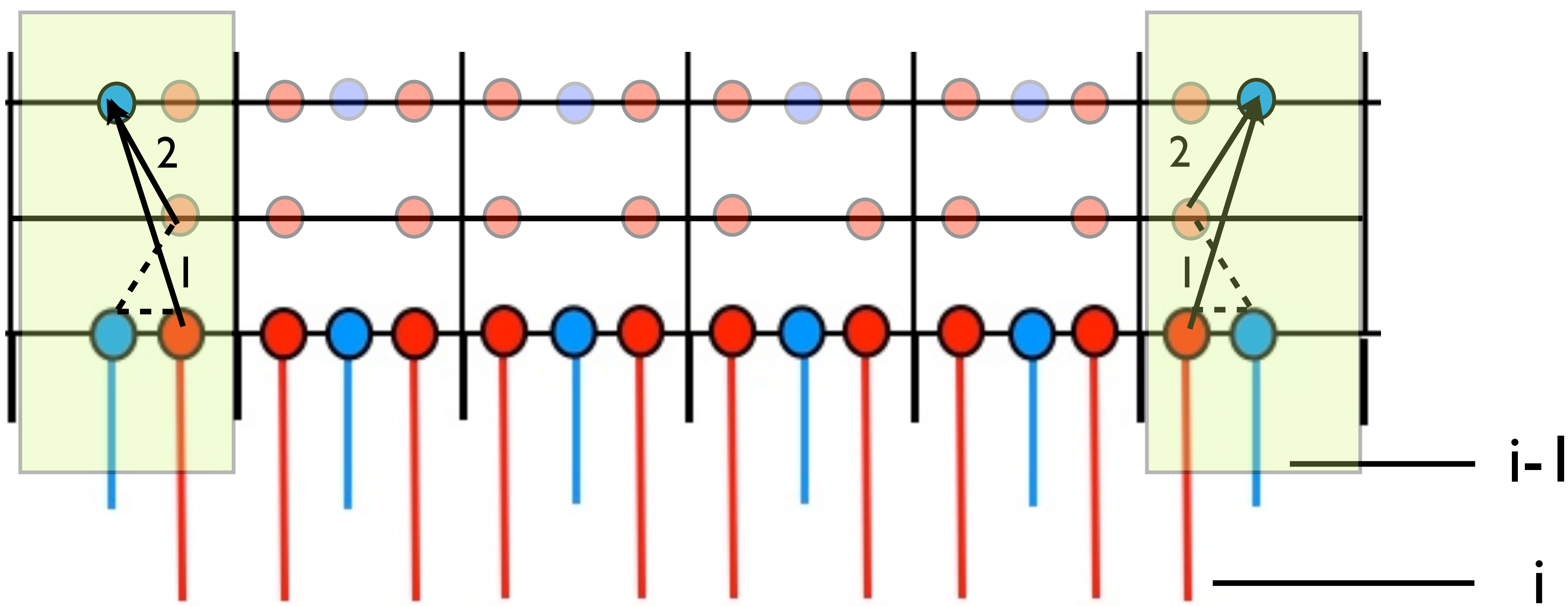
- $\forall i = L, 1$ $level(i)$ updates interior cells at $level(i-1)$
  - evaluate method using $level(i)$ points and average to form value
- correction step to ensure conservation at fine-coarse interfaces
  - sequentially solve Riemann problems between $level(i)$ values closest to interface and value at $level(i-1)$
  - after each solve, contribution is added to value at $level(i-1)$



AMR step : interface corrections to full time step

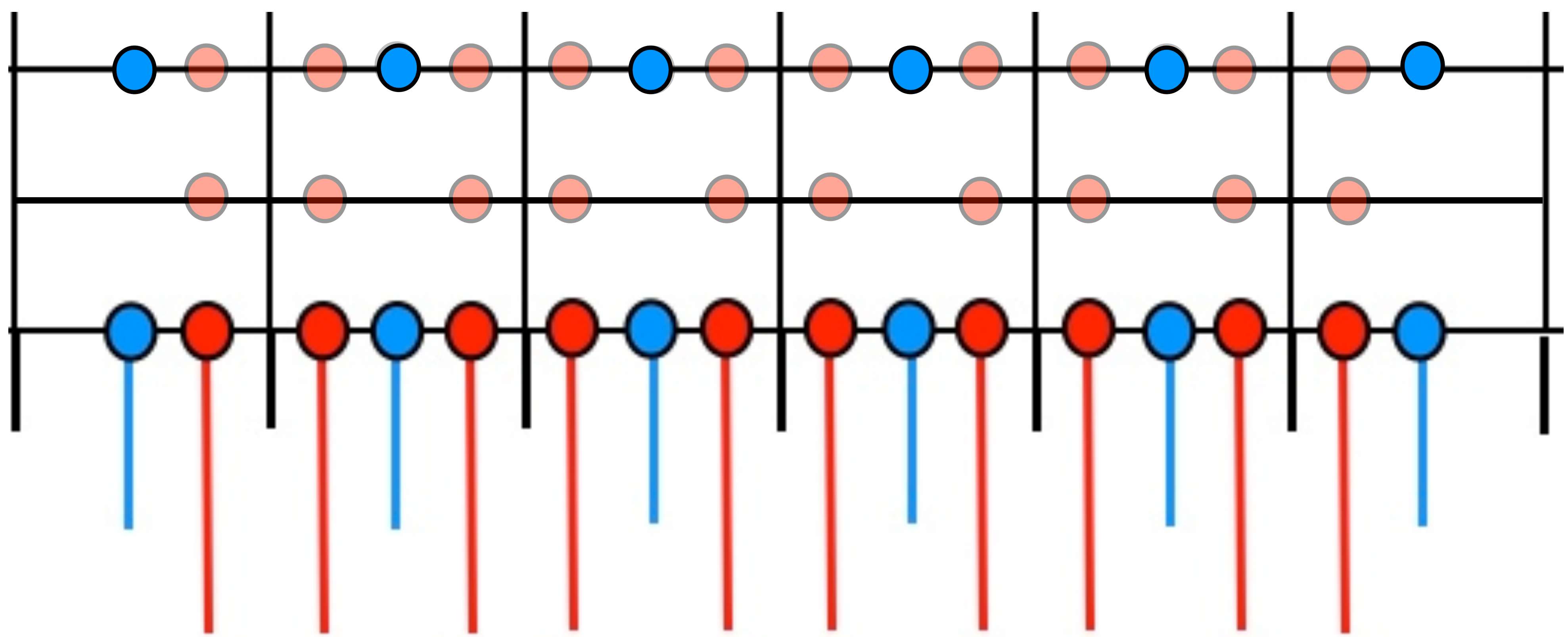
(d) promote grid solutions per grid level to full time step

- $\forall i = L, 1$ $level(i)$ updates interior cells at $level(i-1)$
  - evaluate method using $level(i)$ points and average to form value
- correction step to ensure conservation at fine-coarse interfaces
  - sequentially solve Riemann problems between $level(i)$ values closest to interface and value at $level(i-1)$
  - after each solve, contribution is added to value at $level(i-1)$



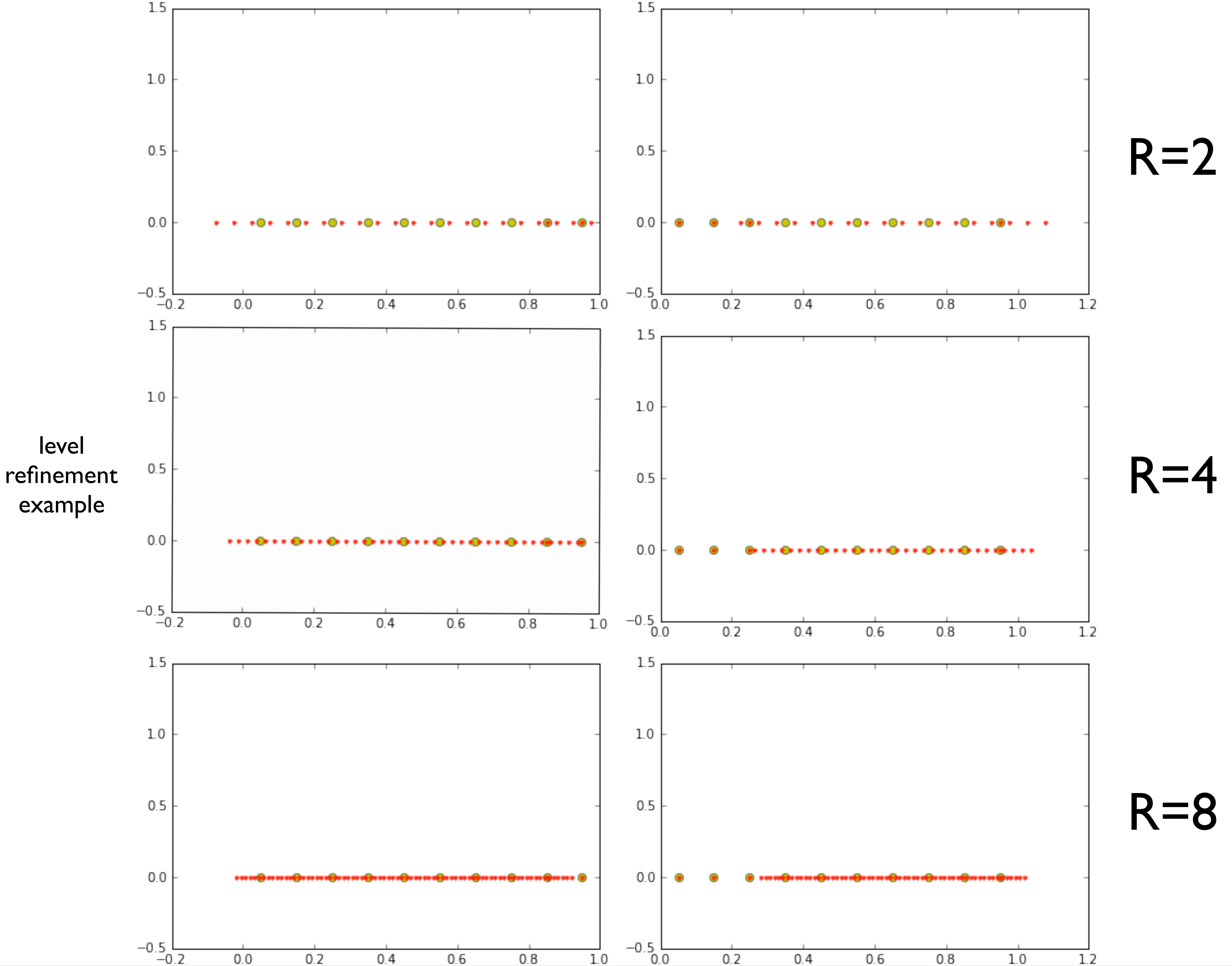
AMR step : take K steps if you like (K ~ buffer size)

(e) completes single time step on coarse grid $level(0)$, go to step b



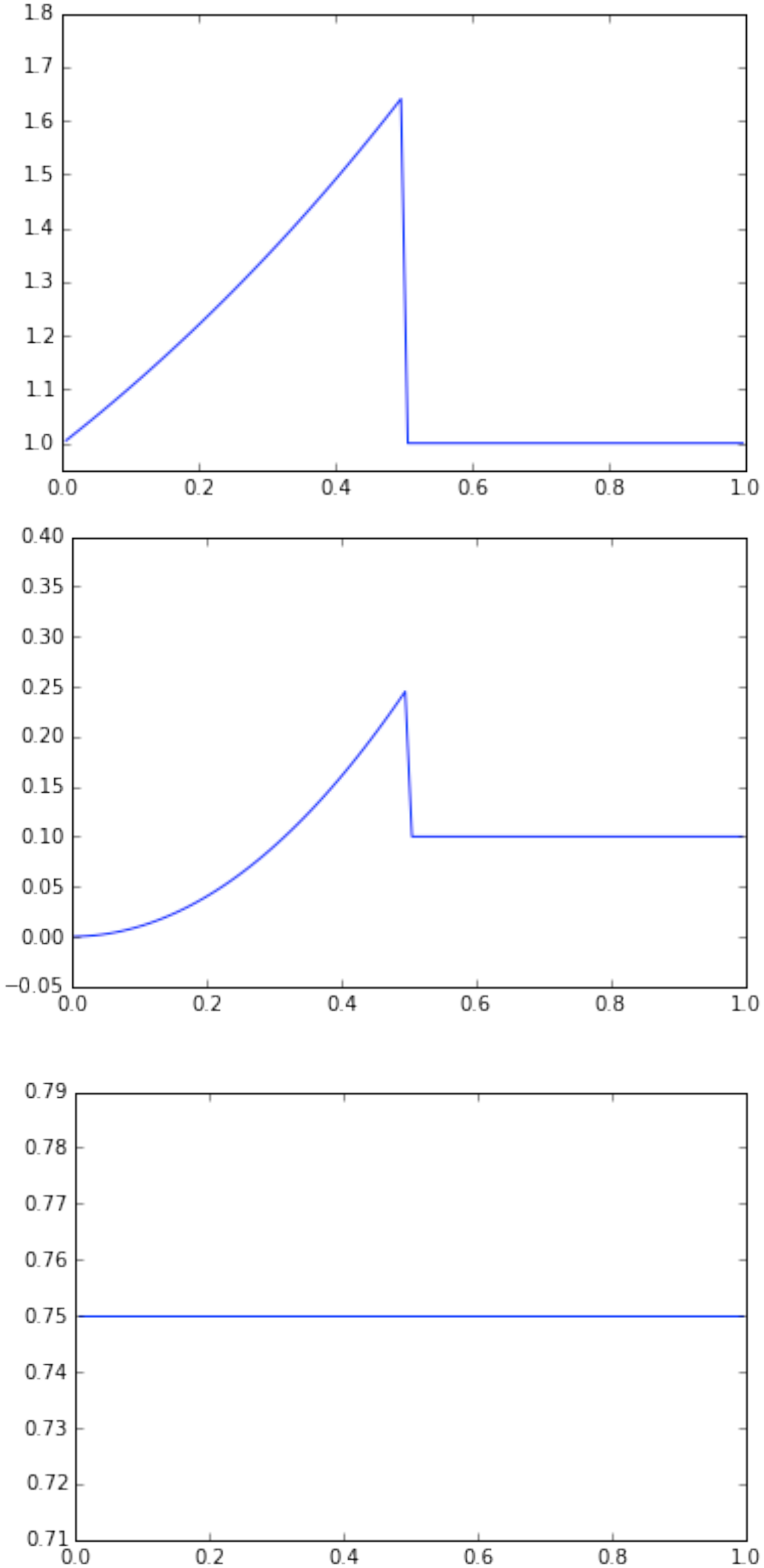
Implementation efforts

- implemented wave propagation method as base solver
    - upwind
    - Lax - Wendroff
    - limiters (we used mostly minmod)
- 1 level amr - 2^p refinement choice
- recursive amr - sadly broken this moment

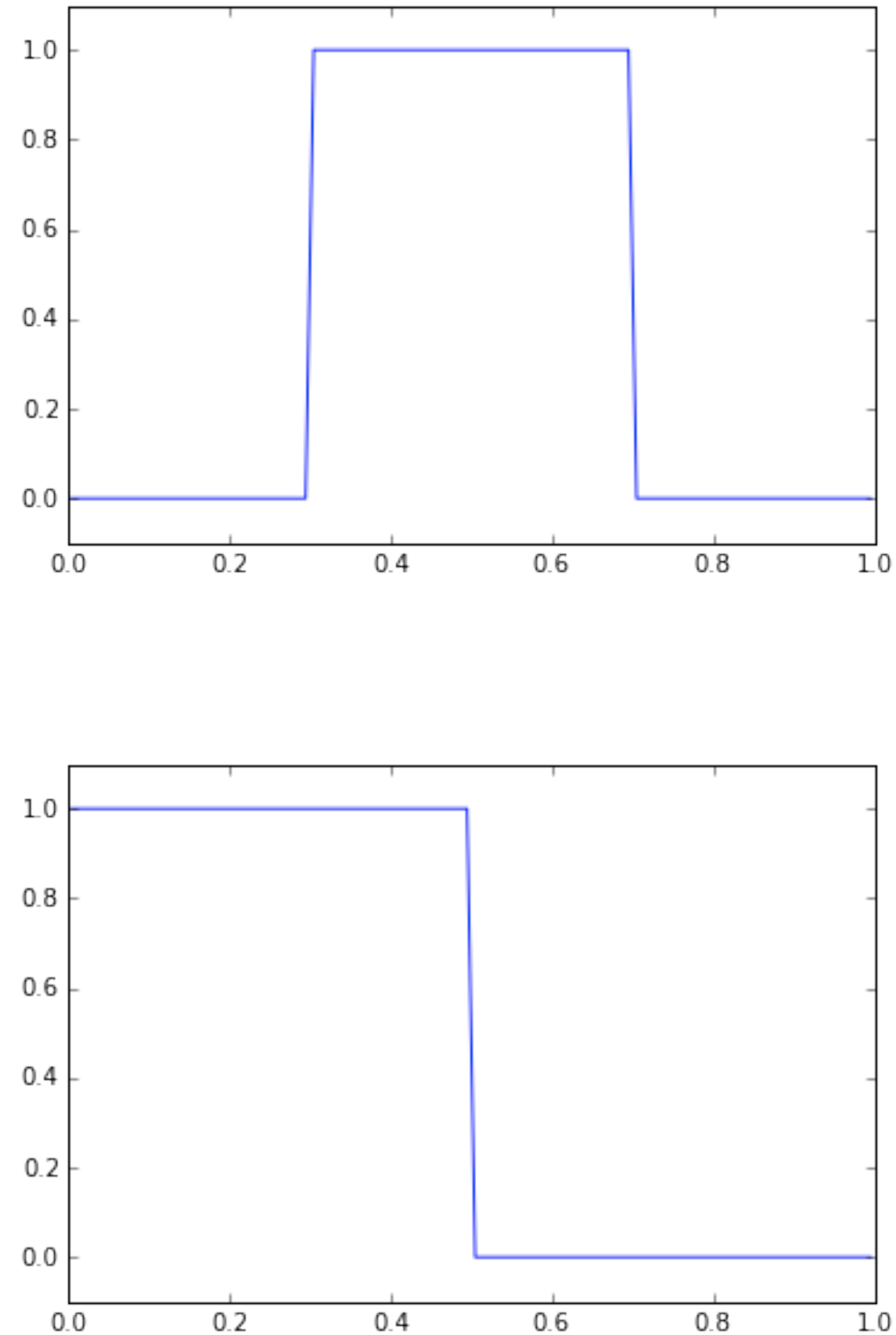


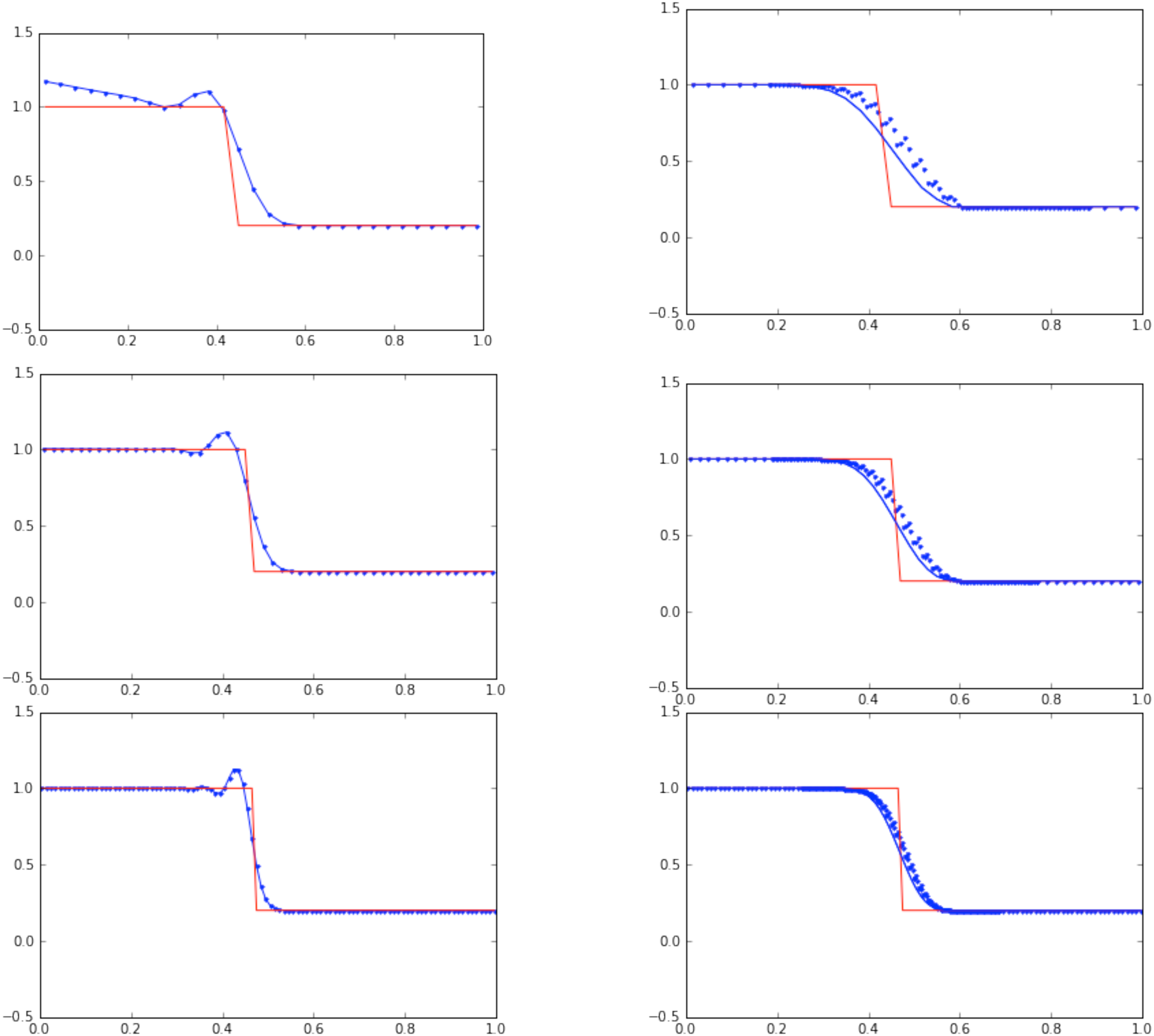
R=2
R=4
R=8
level refinement example

u(x)


1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0.0 0.2 0.4 0.6 0.8 1.0
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
0.0 0.2 0.4 0.6 0.8 1.0
0.79
0.78
0.77
0.76
0.75
0.74
0.73
0.72
0.71
0.0 0.2 0.4 0.6 0.8 1.0

q(x)


1.0
0.8
0.6
0.4
0.2
0.0
0.0 0.2 0.4 0.6 0.8 1.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0 0.2 0.4 0.6 0.8 1.0



1.5
1.0
0.5
0.0
−0.5
0.0
0.2
0.4
0.6
0.8
1.0

AMR

studies that we did not present but that matter

- static global refinement versus locally adaptive
    - performance and accuracy tradeoffs

- error improvement as function of level refinement

- role of error tolerance in flagging in grid construction and numerical quality

AMR

difficulties we encountered

- something hosed in recursive implementation
  - close but no cigar

- spreading in refined regions
  - culprit perhaps over-refinement

- overly refined regions - what are the controlling knobs here and how to balance them
  - seems flagging algorithm and clustering but ...

(and questions)

References

[1] http://www.clawpack.org

[2] R. Leveque Finite Volume Methods for Hyperbolic Problems Cambridge University Press (2002)

[3] M. Berger and P. Collela, Local Adaptive Mesh Refinement for Shock Hydrodynamics Journal of Computational Physics 82, 64-84 (1989)

[4] M. Berger and R. Leveque, Adaptive Mesh Refinement Using Wave-Propagation Algorithms for Hyperbolic Systems SIAM Journal of Numerical Analysis, 35(6):2298–2316, (1998)

[5] M. Berger, Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations, *PhD Thesis*, Department of Computer Science, Stanford University, Stanford, CA 94305 (1982)