Grid adaptivity for systems of conservation laws

M. Semplice[1] G. Puppo[2]

[1]Dipartimento di Matematica
Università degli Studi di Torino

[2]Dipartimento di Scienze Matematiche
Politecnico di Torino

HYP2012
14th International Conference on Hyperbolic Problems:
Theory, Numerics and Applications
Padova, 25-29 June 2012

A recipe for an adaptive scheme

Ingredients

- (finite volume) grid
- numerical fluxes
- time marching method
- refine/coarsen strategy (and indicator)

Recipe

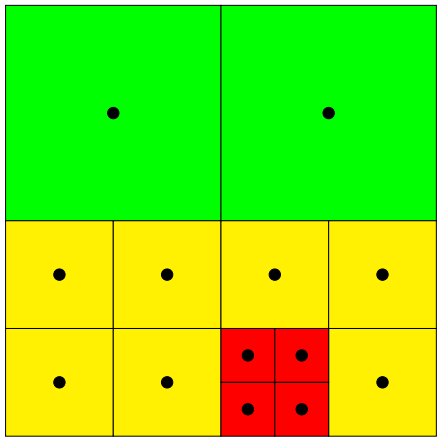
Mix all ingredients, stir until it compiles and runs smoothly, enjoy the movie show!

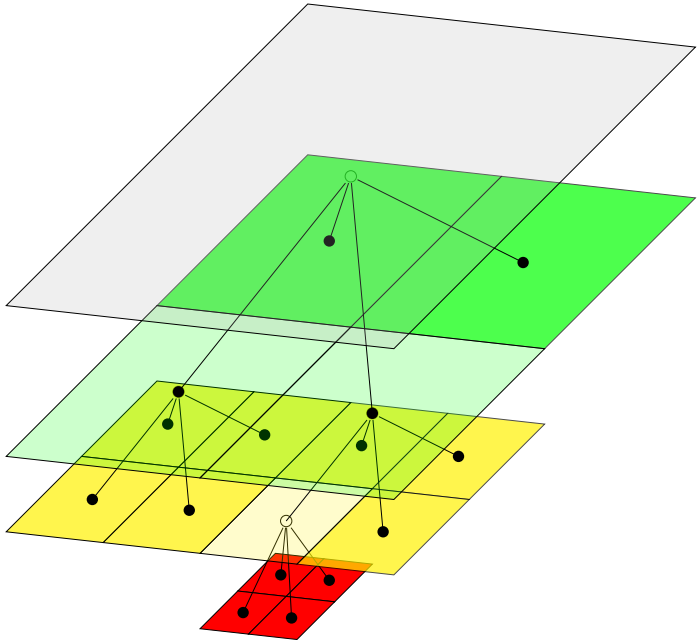
Tools

Object Oriented Programming and a library like DUNE[a] allows to code each part independently (e.g. the Runge-Kutta does not know the number of spatial dimensions)

[a] Dune (Distributed and Unified Numerics Environment),

2D grids and quad-trees





Data are attached only to coloured cells (leaves)

The grid manager (`dune-grid` for us) allows to iterate easily

- on all the leaves
- on a given level

Note: similar structures holds cubes, triangles, tetrahedra, ...

Semidiscrete scheme

$$\partial_t u + \partial_x f(u) = 0$$

R-K: $\begin{array}{c|c} A \\ \hline b \end{array}$

$$\text{stages } \overline{u}_j^{(i)} = \overline{u}_j^n - \lambda \sum_{k=1}^{i-1} a_{ik} \left(F_{j+1/2}^{(k)} - F_{j-1/2}^{(k)} \right) \qquad \lambda = \frac{\Delta t}{h}$$

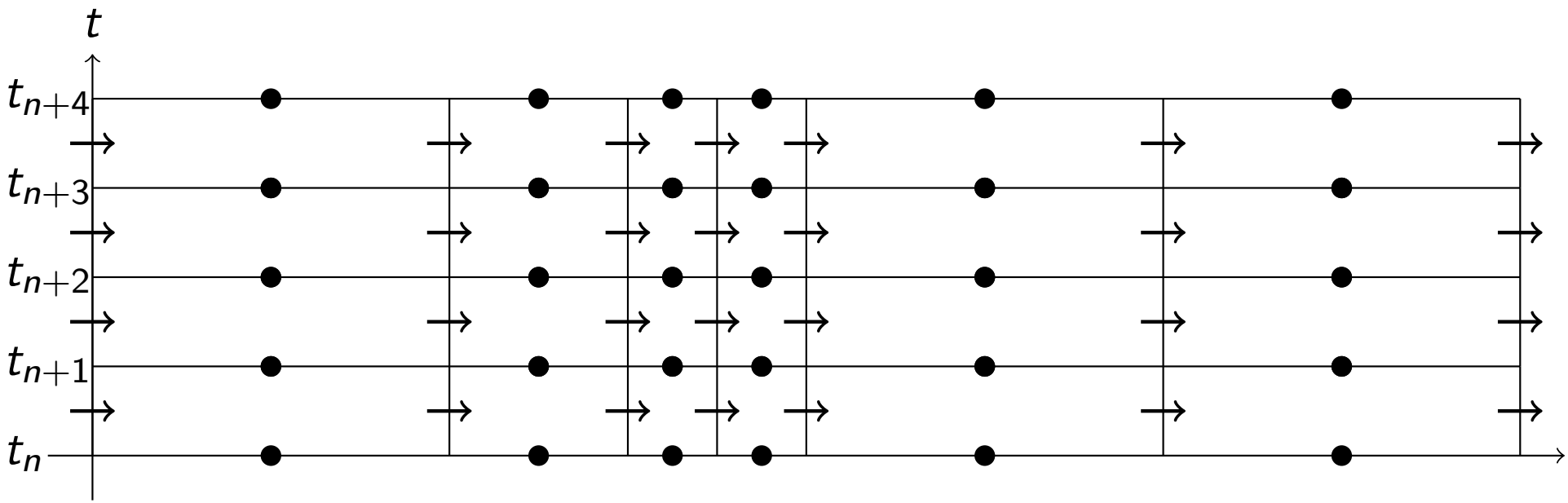
$$\text{fluxes } F_{j+1/2}^{(i)} = \mathcal{F}\left(u_{j+1/2}^{(i),-}, u_{j+1/2}^{(i),+} \right)$$

$$\text{step } \overline{u}_j^{n+1} = \overline{u}_j^n - \lambda \sum_{i=0}^{\nu} b_i \left(F_{j+1/2}^{(i)} - F_{j-1/2}^{(i)} \right)$$

where $u_{j+1/2}^{(i),\pm}$ are some (non-oscillatory) reconstructions.

Imposing the CFL globally

$$\text{CFL} \Longrightarrow \forall k:\ \Delta t \leq \lambda h_k = \lambda h_0 2^{-k} \Longrightarrow \Delta t = \lambda h_0 2^{-k_{\max}}$$



•: cell averages $\qquad \rightarrow : \sum_i b_i F^{(i)}$

Local timestepping

Choosing Δt globally:

- easier to program
- wastes CPU resources for big cells, where the solution is well approximated
- (in theory) unneeded numerical diffusion if many cells have sub-optimal (local) mesh ratio

Local timestepping

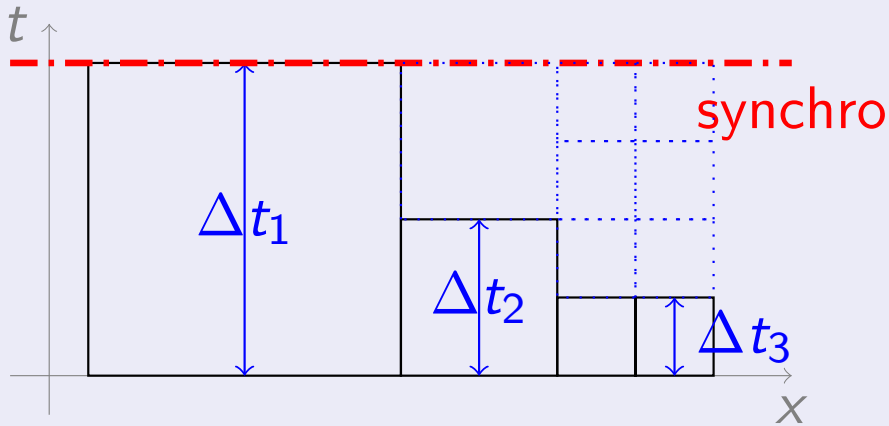
Choosing Δt globally:

- easier to program
- wastes CPU resources for big cells, where the solution is well approximated
- (in theory) unneeded numerical diffusion if many cells have sub-optimal (local) mesh ratio

Local timestepping

Each cell advances with different stepsizes

$$\Delta t_k = \lambda h_k = \lambda h_0 2^{-k}$$



Review of local timestepping

Motivated by approximating (on uniform grids) conservation laws with strong spatial variations of the velocity:

- First order Osher-Sanders (1983)
- Second order, TVD property Dawson-Kirby (2001), Kirby (2003)
- multiresolution (wavelets) Muller-Stiriba(2007)

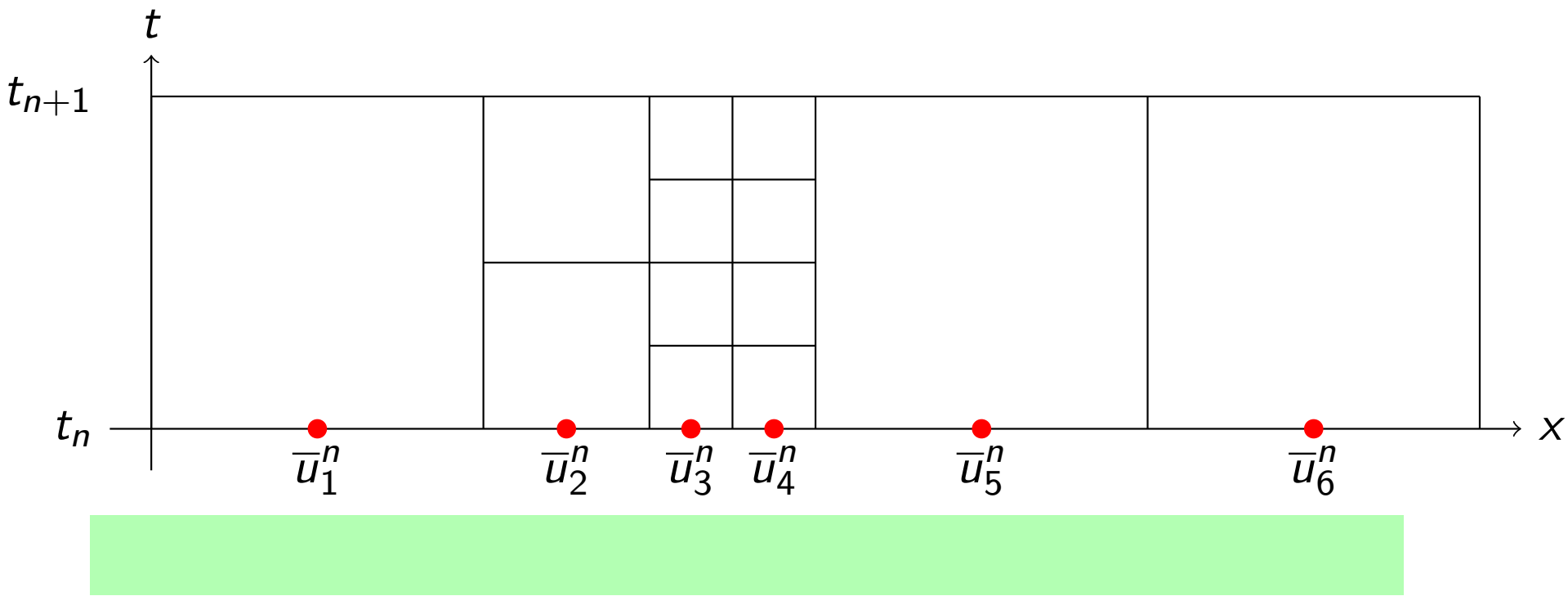
More recent work has focused on AMR:

- Second order Runge-Kutta Constantinescu-Sandu (2007), Krivodonova (2010)

First order scheme on nonuniform grid – local timestepping

$$\Delta t_k = \lambda h_k \qquad \Delta t = \Delta t_{\text{macro}}$$

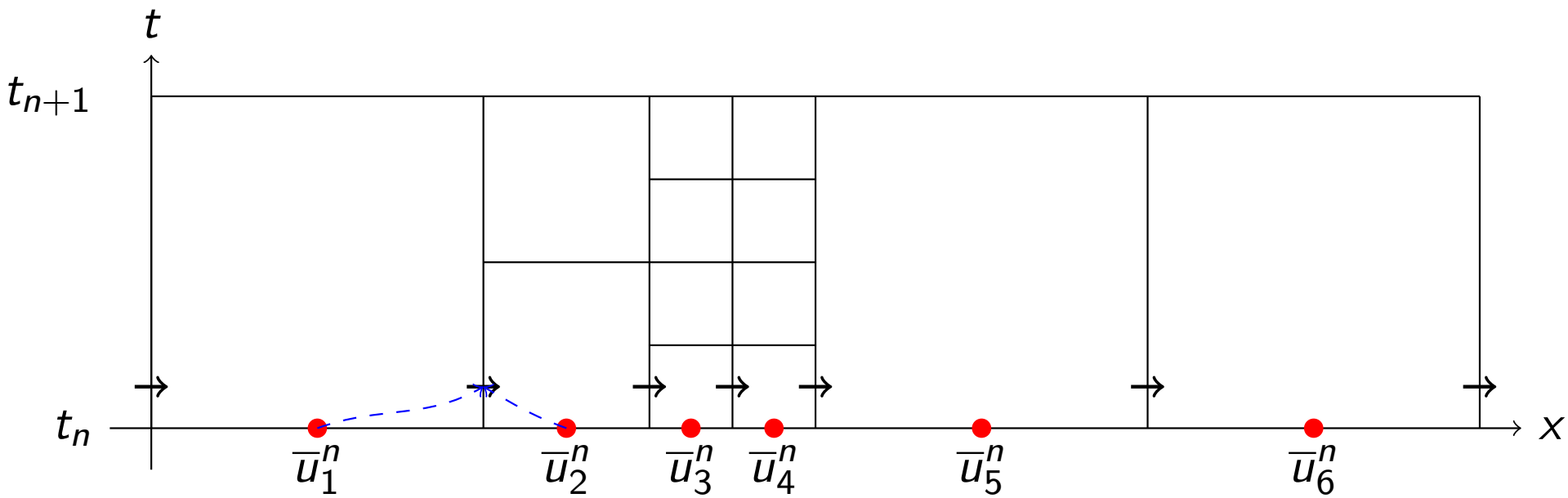
● ● cell averages → fluxes ○ temporaries



First order scheme on nonuniform grid – local timestepping

$$\Delta t_k = \lambda h_k \qquad \Delta t = \Delta t_{\text{macro}}$$

● ● cell averages → fluxes ○ temporaries

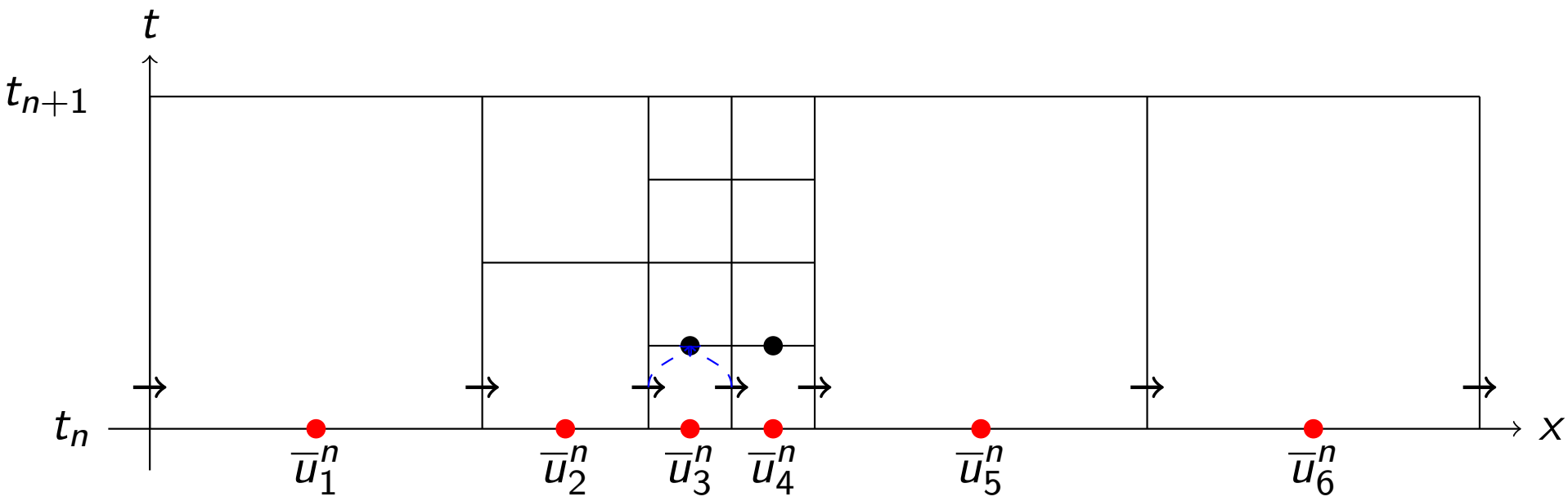


everywhere

First order scheme on nonuniform grid – local timestepping

$$\Delta t_k = \lambda h_k \qquad \Delta t = \Delta t_{\text{macro}}$$

• • cell averages → fluxes ○ temporaries

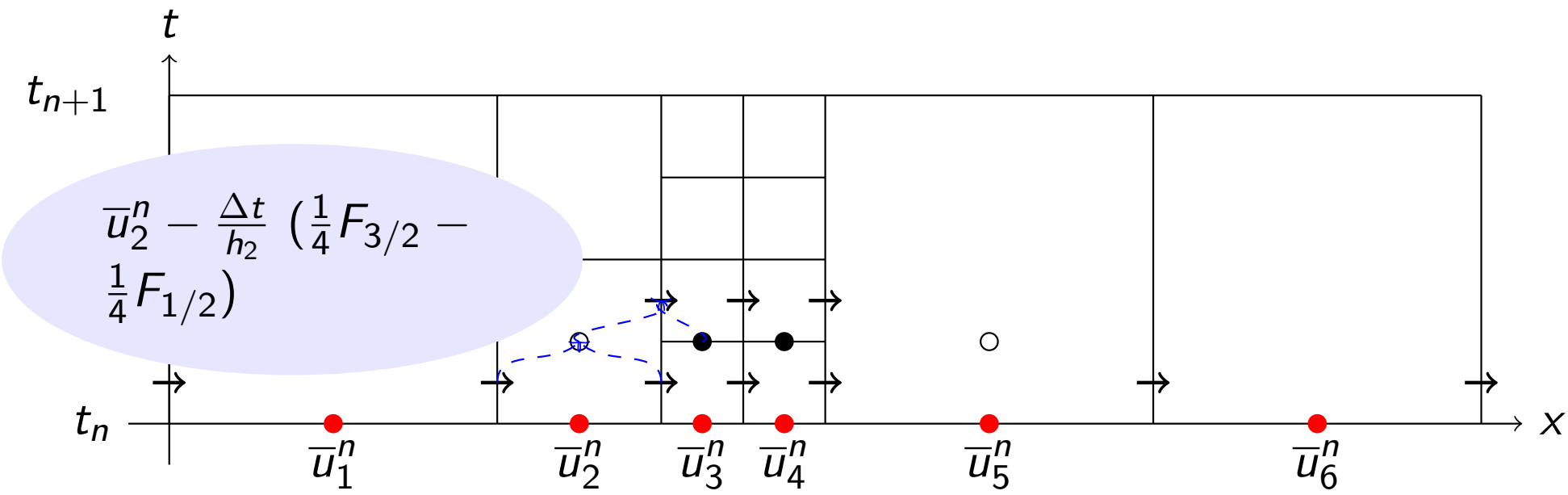


small

First order scheme on nonuniform grid – local timestepping

$$\Delta t_k = \lambda h_k \qquad \Delta t = \Delta t_{\text{macro}}$$

● ● cell averages → fluxes ○ temporaries

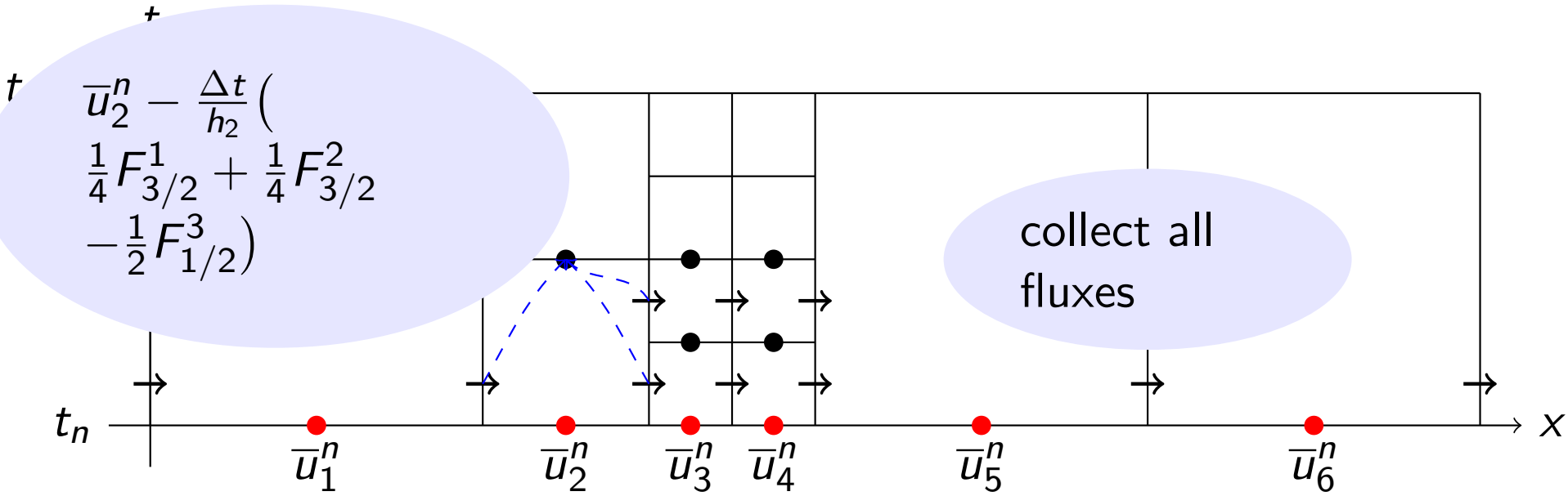


small

First order scheme on nonuniform grid – local timestepping

$$\Delta t_k = \lambda h_k \qquad \Delta t = \Delta t_{\text{macro}}$$

• • cell averages → fluxes ○ temporaries

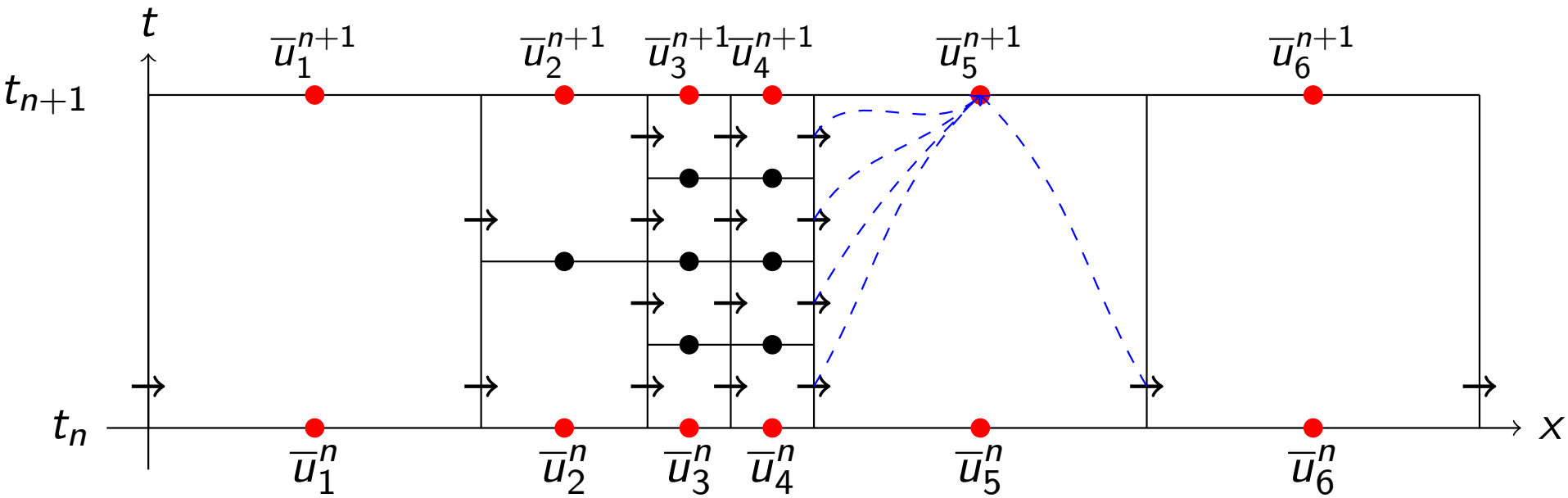


small, medium

First order scheme on nonuniform grid – local timestepping

$$\Delta t_k = \lambda h_k \qquad \Delta t = \Delta t_{\text{macro}}$$

● ● cell averages → fluxes ○ temporaries

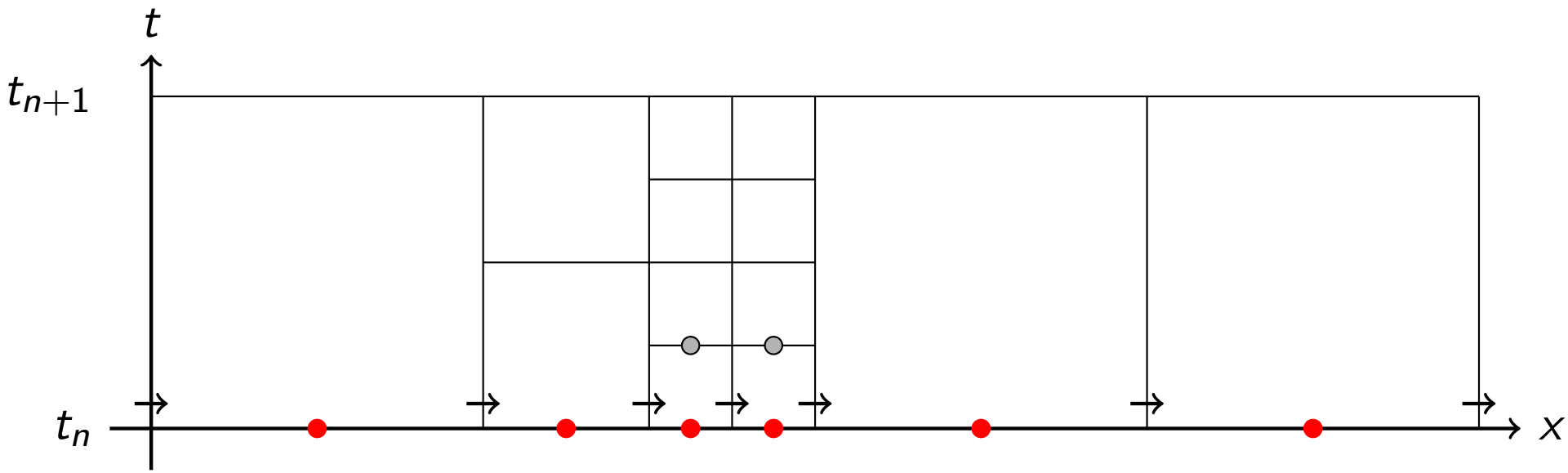


Second order scheme on nonuniform grid – global CFL

cell averages ● and ●

stage values ○

fluxes → $F^{(1)}_{j+1/2}$ and → $F^{(2)}_{j+1/2}$ ○ temporaries

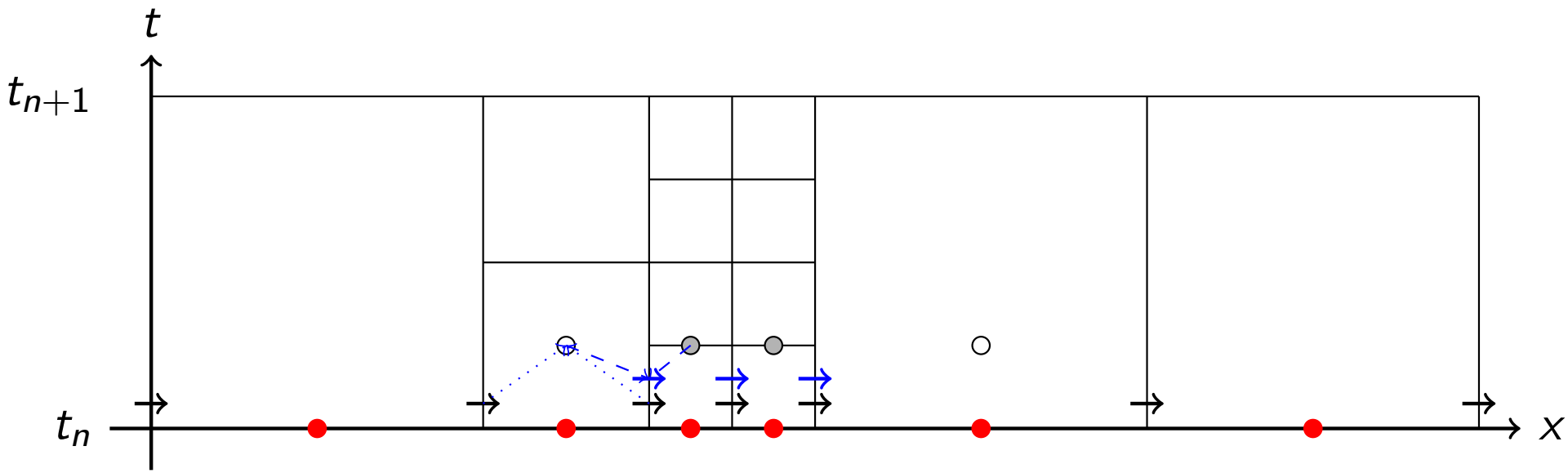


Second order scheme on nonuniform grid – global CFL

cell averages ● and ●

stage values ○

fluxes → $F^{(1)}_{j+1/2}$ and → $F^{(2)}_{j+1/2}$ ○ temporaries

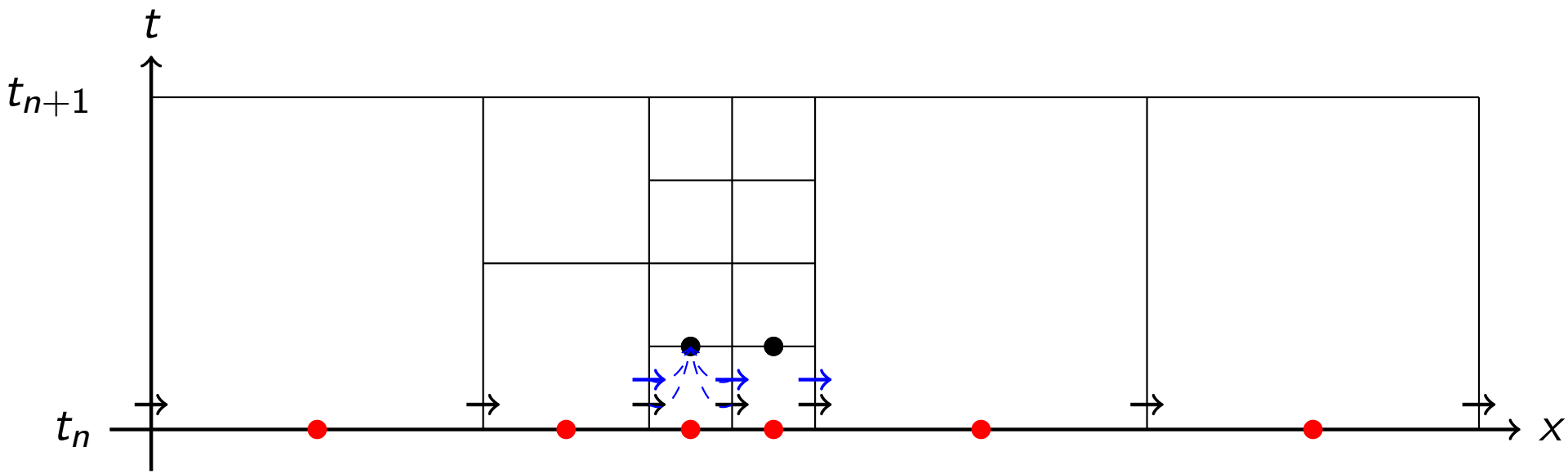


Second order scheme on nonuniform grid – global CFL

cell averages ● and ●

stage values ○

fluxes → $F^{(1)}_{j+1/2}$ and → $F^{(2)}_{j+1/2}$ ○ temporaries



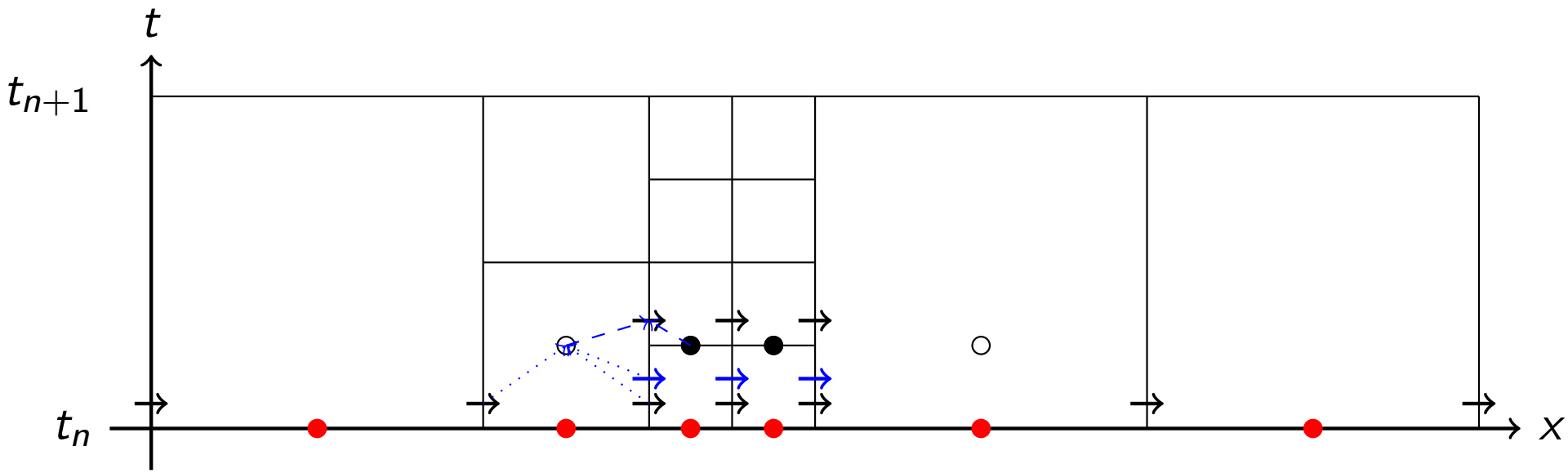
Second order scheme on nonuniform grid – global CFL

cell averages ● and ●

stage values ○

fluxes → $F^{(1)}_{j+1/2}$ and → $F^{(2)}_{j+1/2}$

○ temporaries

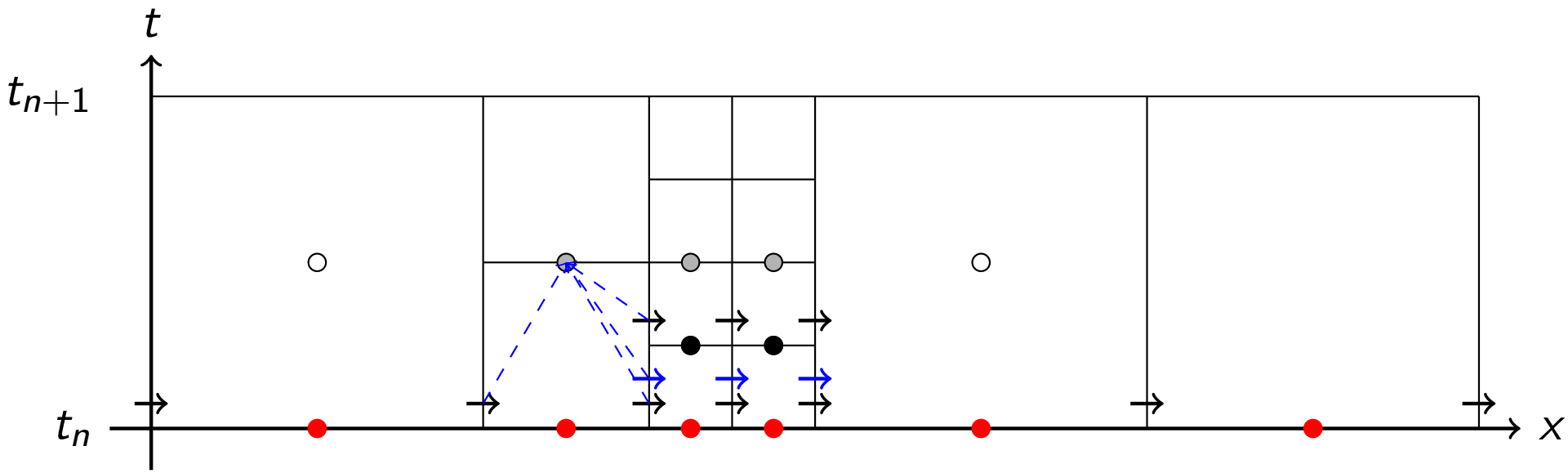


Second order scheme on nonuniform grid – global CFL

cell averages ● and ●

stage values ○

fluxes → $F^{(1)}_{j+1/2}$ and → $F^{(2)}_{j+1/2}$ ○ temporaries

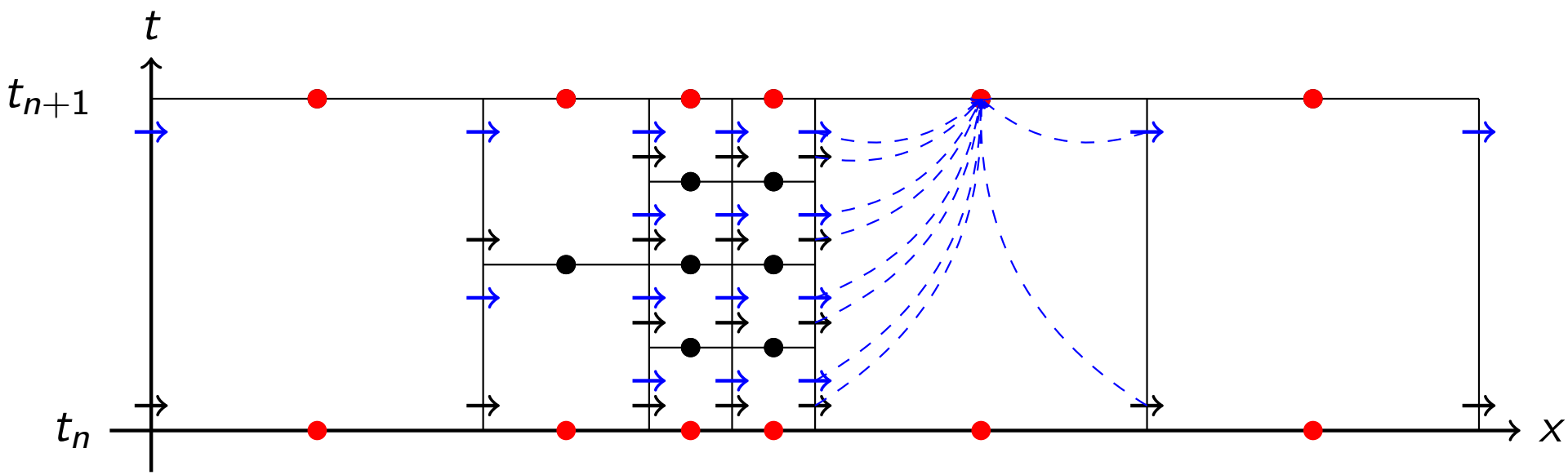


Second order scheme on nonuniform grid – global CFL

cell averages • and •

stage values ○

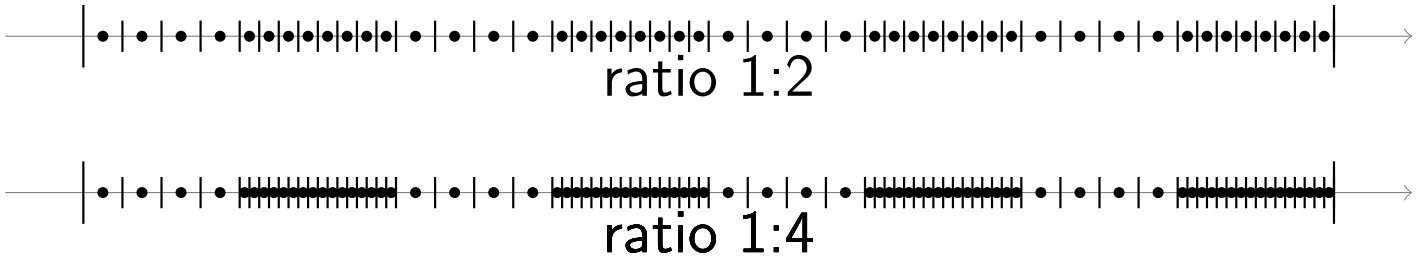
fluxes → $F^{(1)}_{j+1/2}$ and → $F^{(2)}_{j+1/2}$ ○ temporaries



- automatically conservative!
- large reconstruction stencil ⇒ many temporaries

Let's test the performance of nonuniform grids

1. build a nonuniform grid



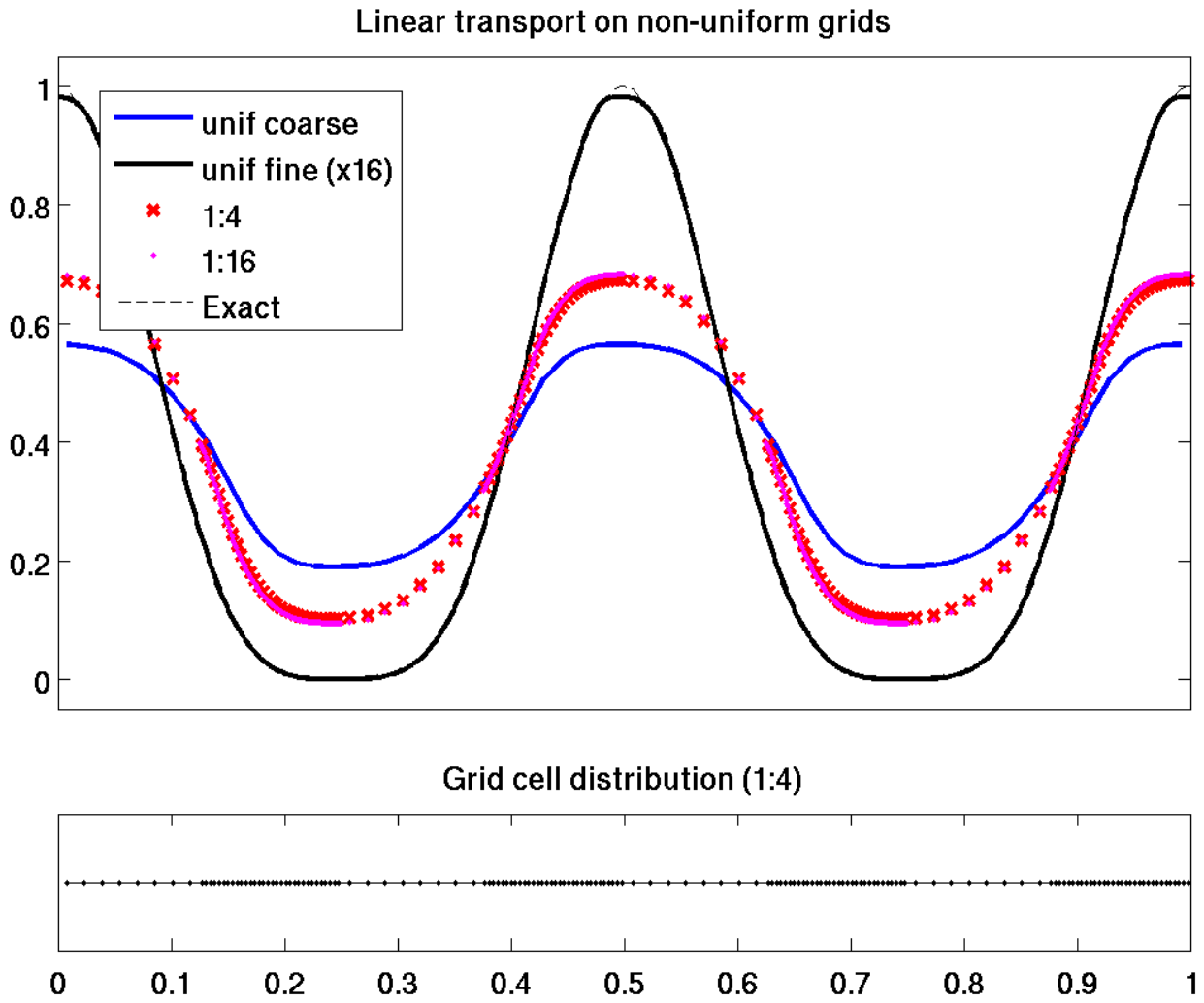
2. leave it fixed in time
3. make different kinds of waves travel along them, crossing grid discontinuities.

We expect:

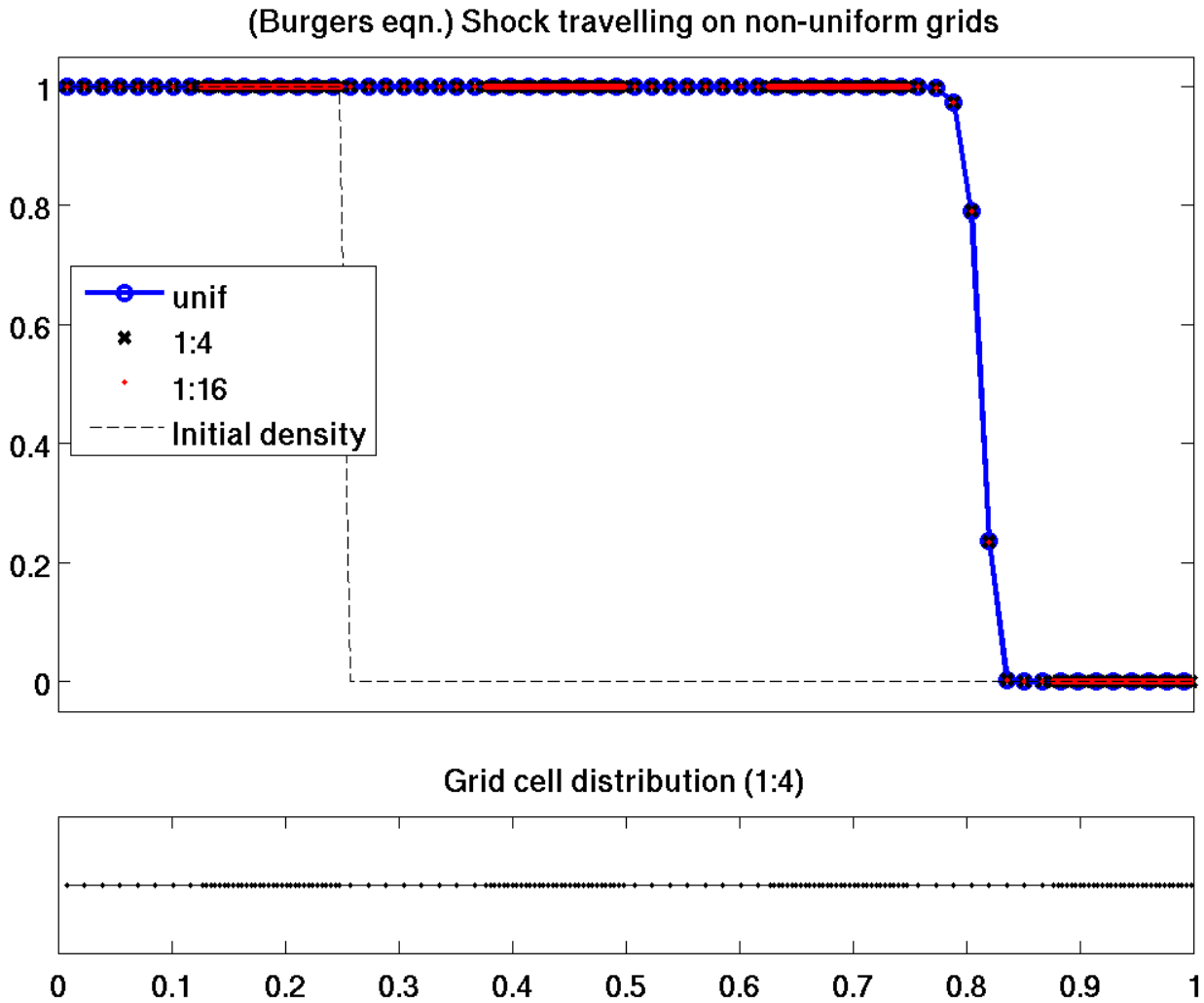
- poor behaviour on error (the grid is fixed!)
- check if grid discontinuities can deform waves

Smooth solution (linear transport)

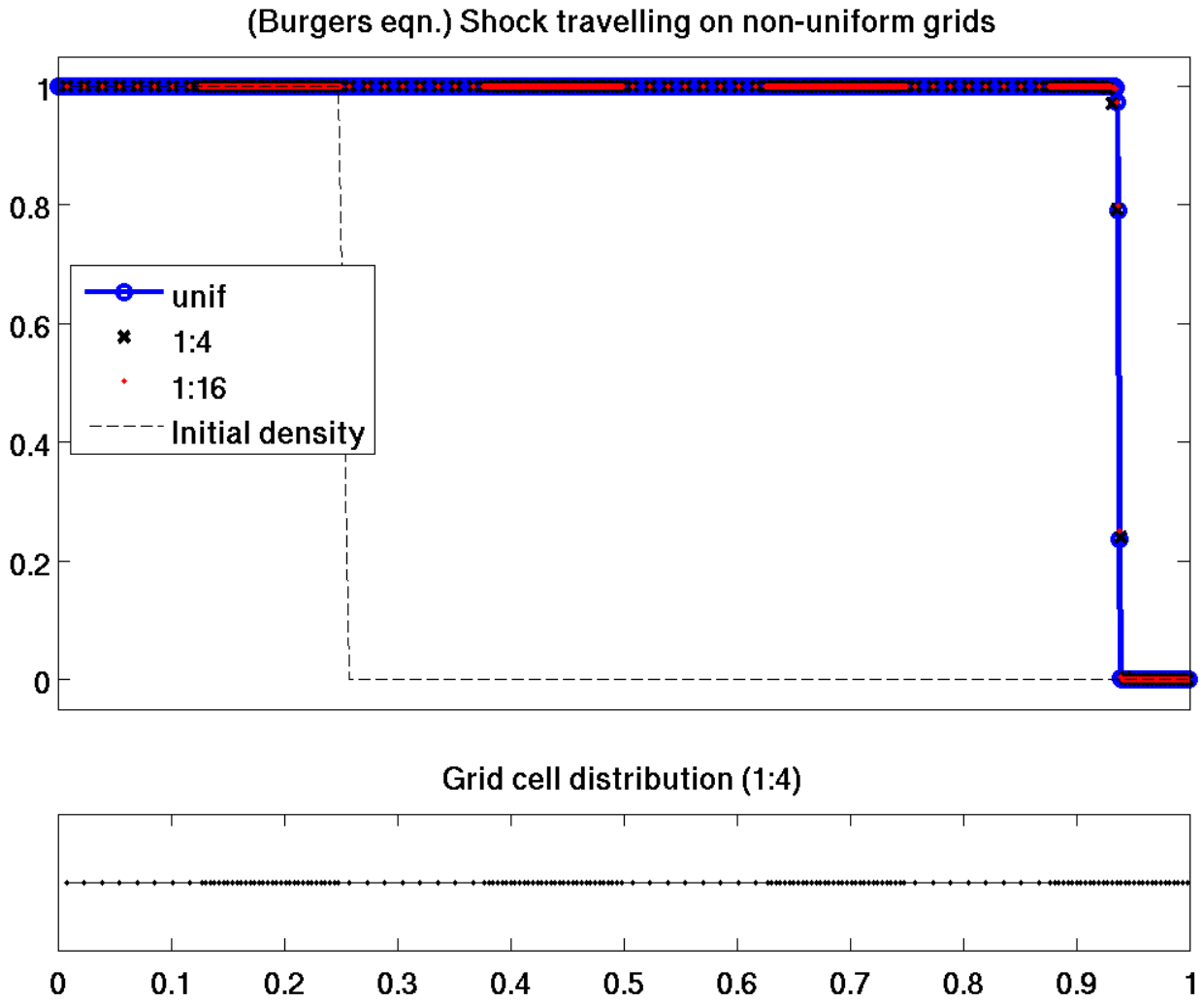
$u_0(x) = [\cos(2\pi x)]^4$



Shock (Burgers equation)

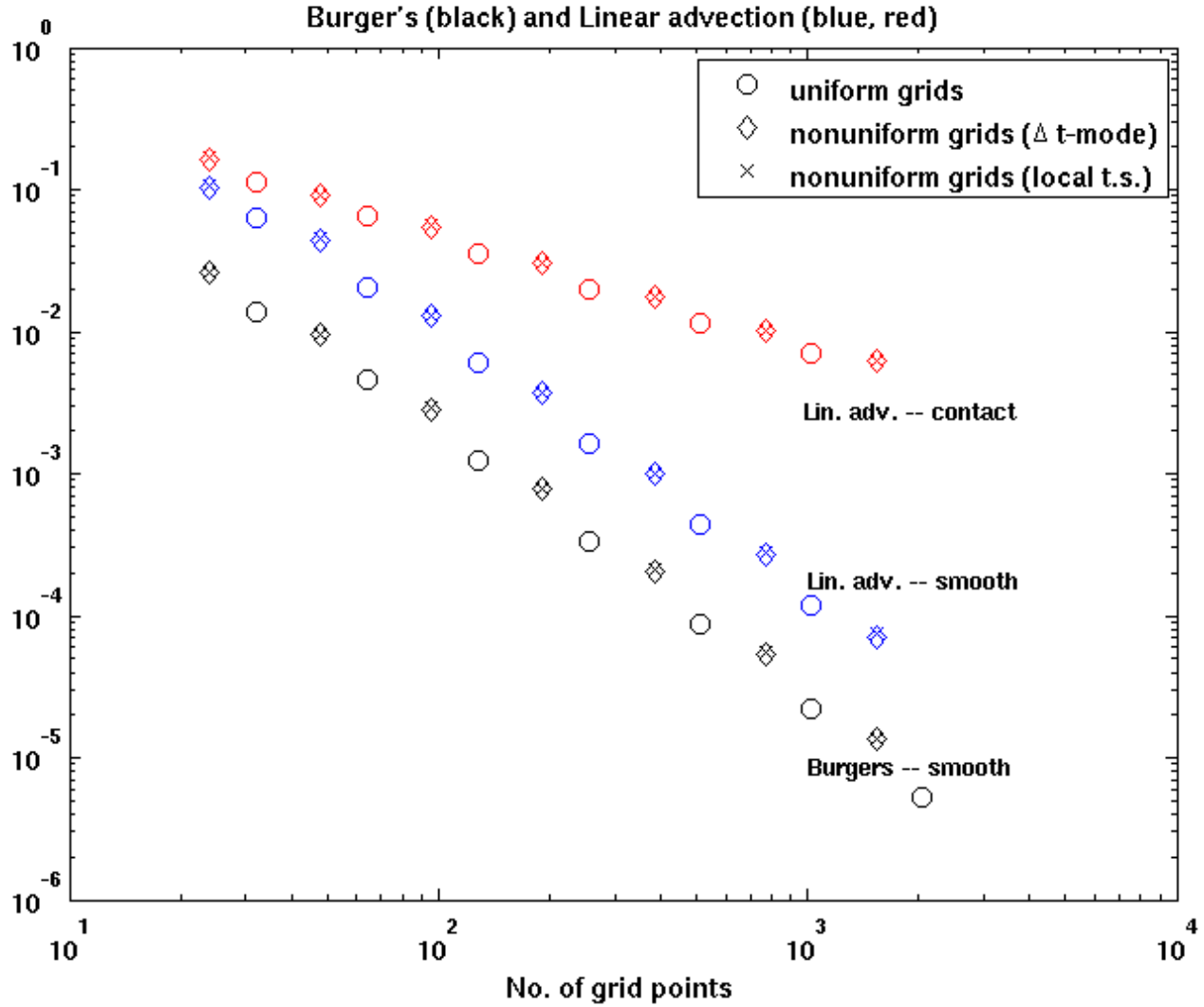


Shock (Burgers equation)

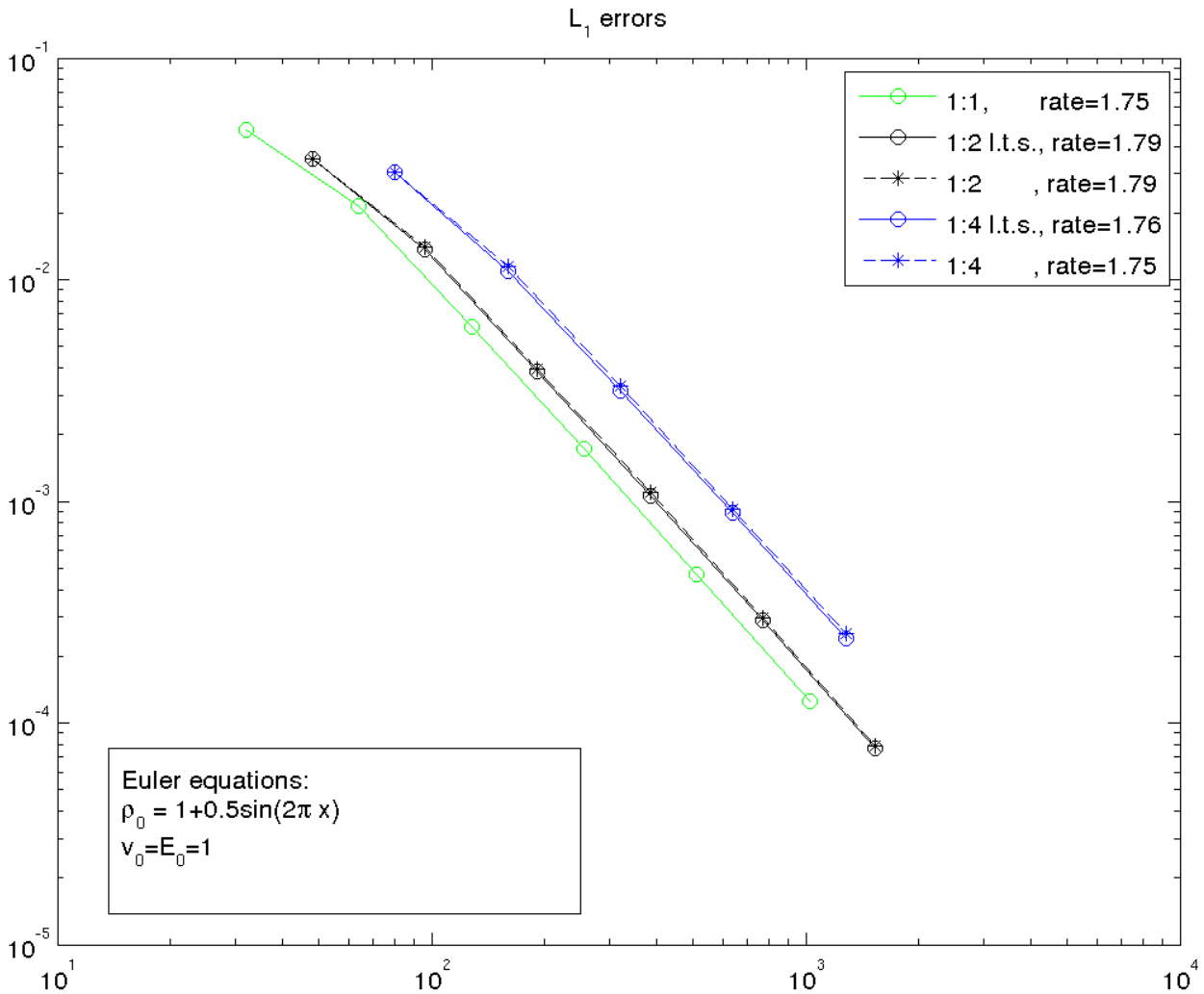


Errors on uniform/nonuniform grids

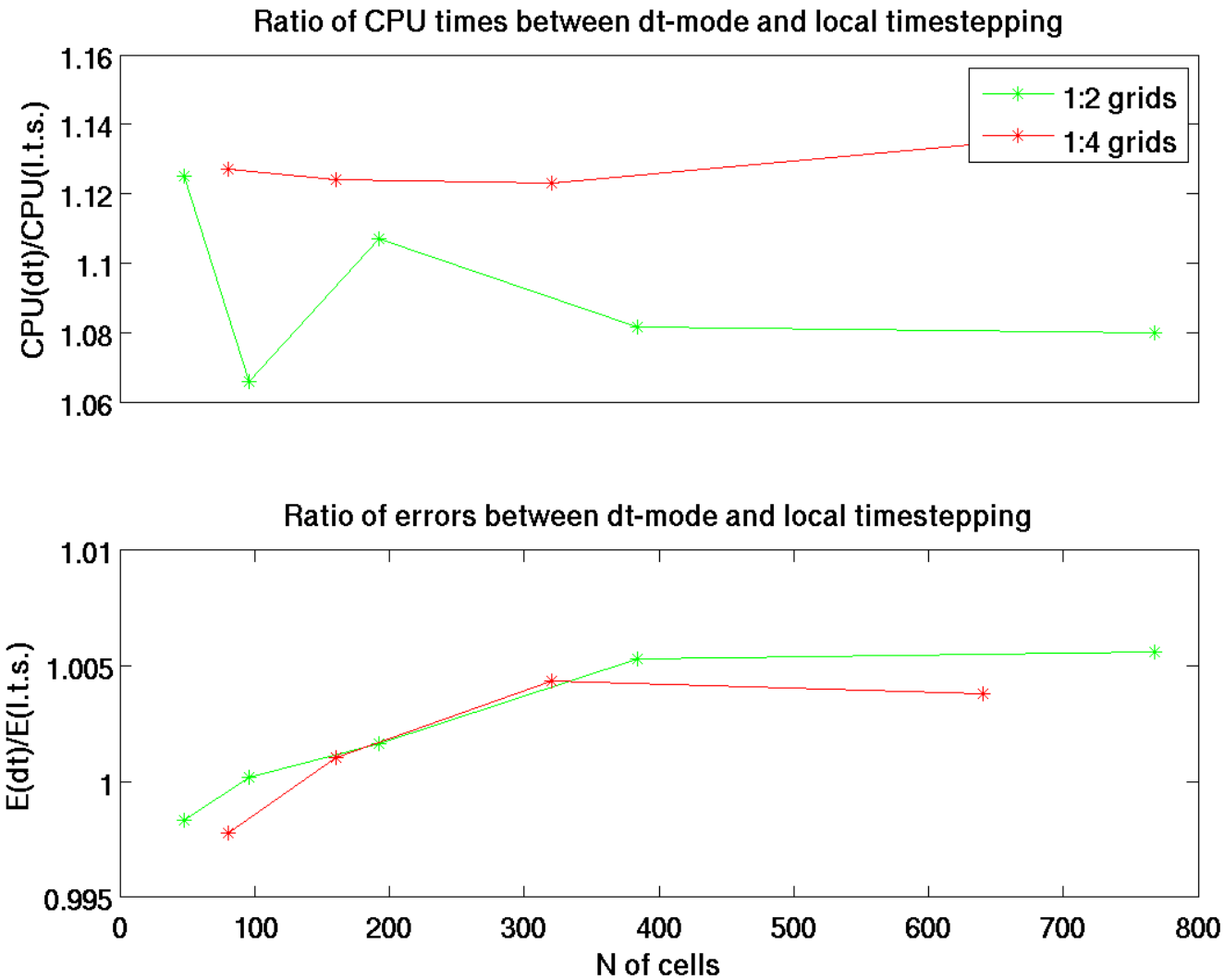
grid ratio 1:2



Smooth solution (Euler equations)



Smooth solution (Euler equations)



Semidiscrete scheme with numerical entropy production

$$\partial_t u + \partial_x f(u) = 0 \qquad \partial_t \eta(u) + \partial_x \psi(u) \le 0 \qquad \text{R-K: } \begin{array}{c|c} A \\ \hline b \end{array}$$

$$\text{stages } \overline{u}_j^{(i)} = \overline{u}_j^n - \lambda \sum_{k=1}^{i-1} a_{ik} \left(F_{j+1/2}^{(k)} - F_{j-1/2}^{(k)} \right) \qquad \lambda = \frac{\Delta t}{h}$$

$$\text{fluxes } F_{j+1/2}^{(i)} = \mathcal{F}\left(u_{j+1/2}^{(i),-}, u_{j+1/2}^{(i),+} \right) \qquad \Psi_{j+1/2}^{(i)} = \Psi\left(u_{j+1/2}^{(i),-}, u_{j+1/2}^{(i),+} \right)$$

$$\text{step } \overline{u}_j^{n+1} = \overline{u}_j^n - \lambda \sum_{i=0}^{\nu} b_i \left(F_{j+1/2}^{(i)} - F_{j-1/2}^{(i)} \right)$$

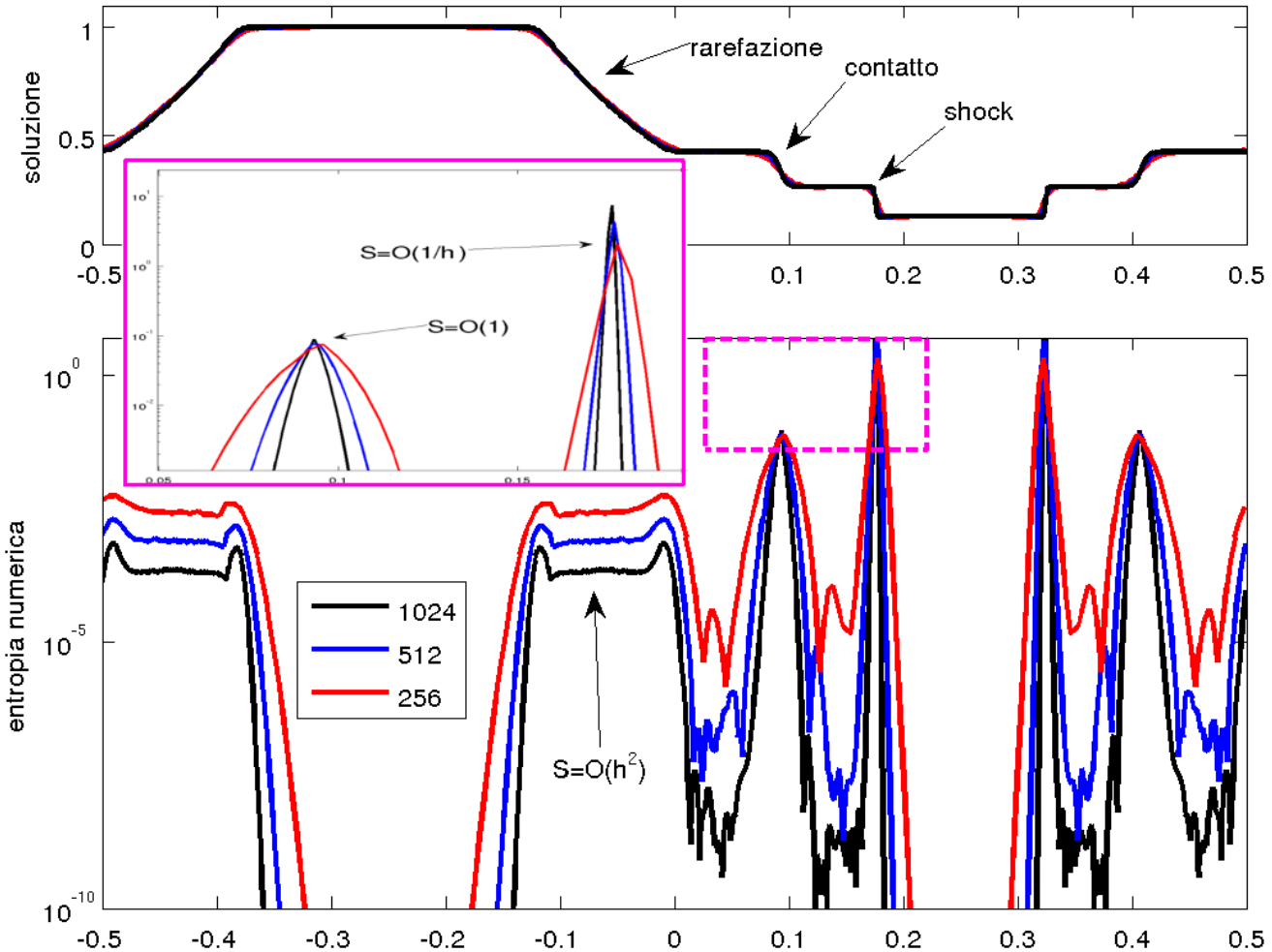
$$\text{entropy } S_j^n = \frac{1}{\Delta t} \left\{ \overline{\eta\left(u_j^{n+1}\right)} - \overline{\eta\left(u_j^n\right)} + \lambda \sum_{i=1}^{\nu} b_i \left(\Psi_{j+1/2}^{(i)} - \Psi_{j-1/2}^{(i)} \right) \right\}$$

Note: most of the seminar is independent from this choice of error indicator, although the numerical entropy production works well in general

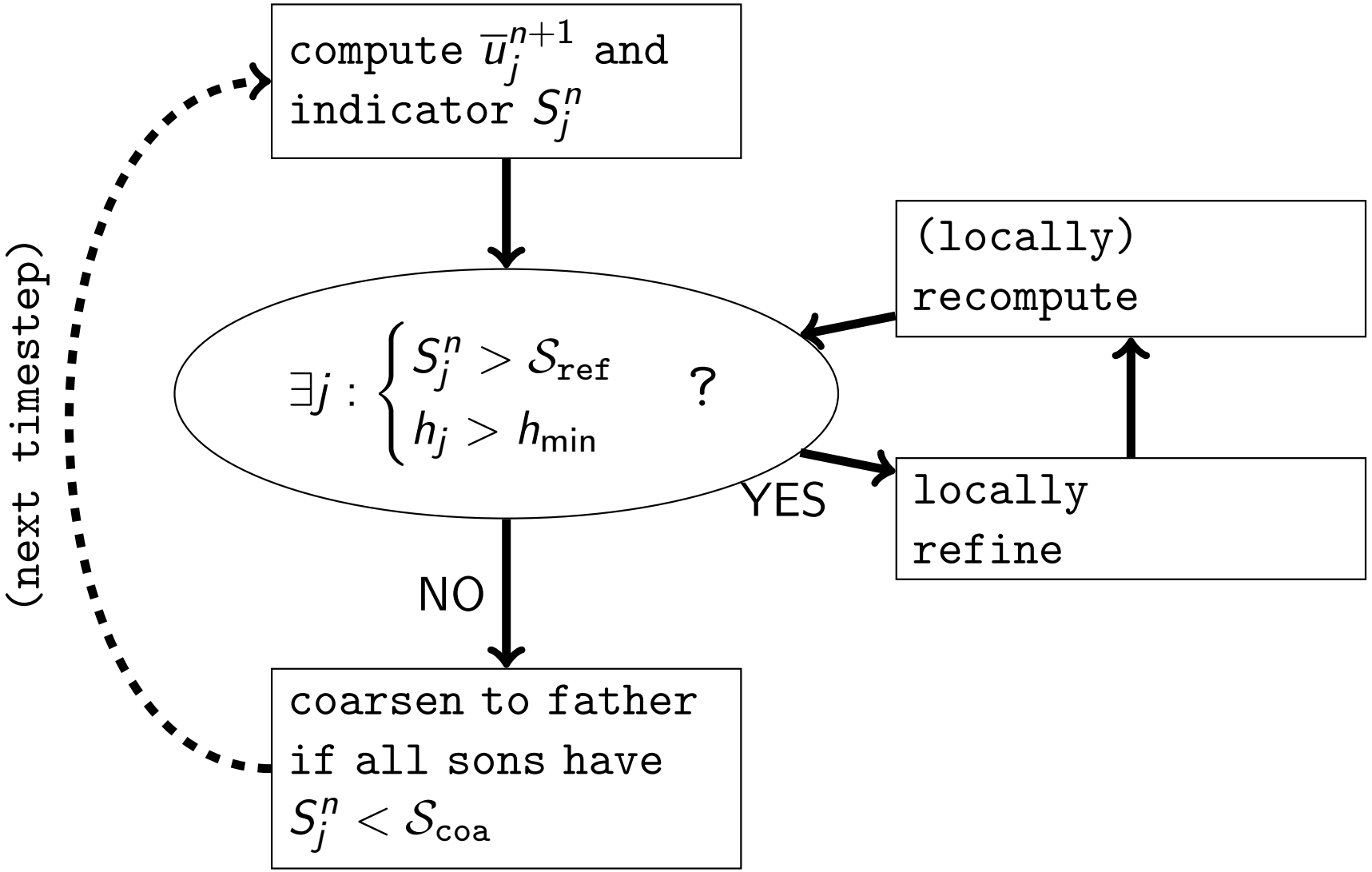
Numerical entropy production in the Sod test

With the second order scheme (Heun + KT flux + minmod), using periodic b.c.

$O(1)$ on contacts $\Uparrow$ threshold



Choose your favourite indicator and loop until final time:



How to set coarsen thresholds?

Coarsening will succede (in $\mathbb{R}^d$) if 2^d neighbouring cells have $S_j^n < S_{\text{coa}}$. We expect to coarsen on smooth regions of the flow, so assume that at the next timestep the *father cell* might produce numerical entropy of size

$$S_{\text{father}}^{n+1} \sim 2^d S_{\text{coa}}$$

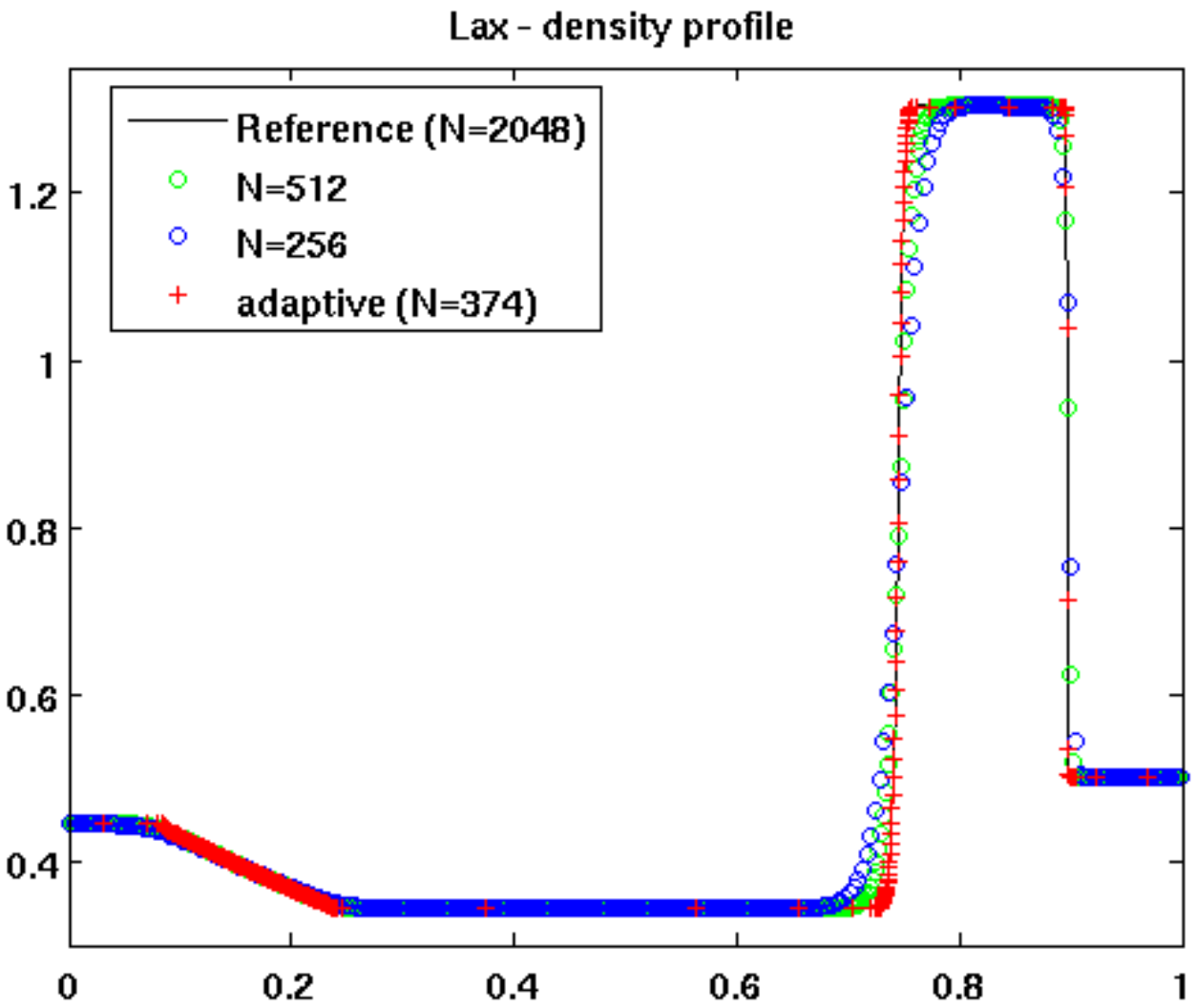
We want to avoid to immediately refine the cell again, so we need

$$2^d S_{\text{coa}} < \eta S_{\text{ref}} \text{ for some } \eta < 1$$

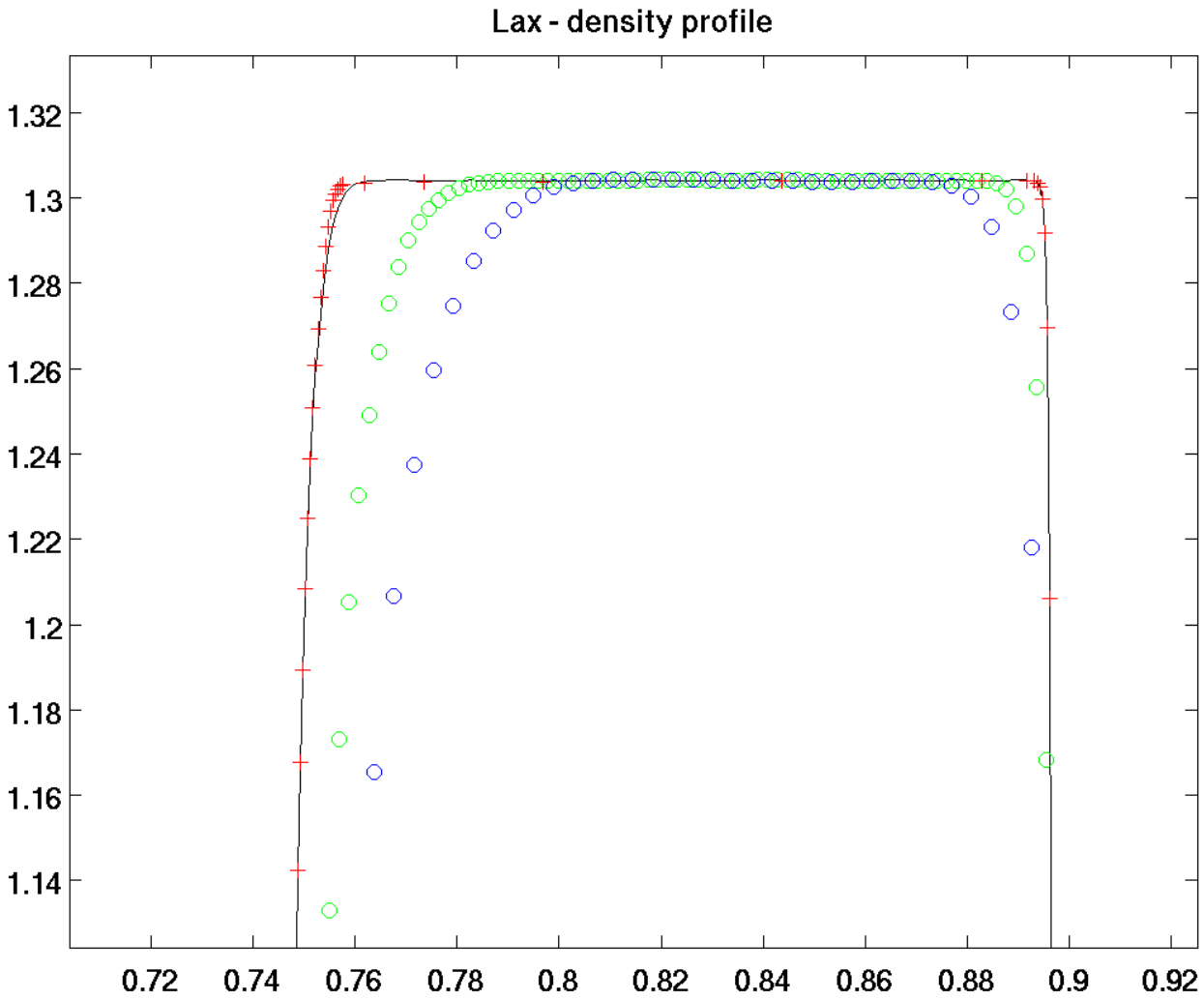
Practical guideline:

$$\eta = \tfrac{1}{4} \quad \Rightarrow \quad S_{\text{coa}} = \frac{S_{\text{ref}}}{2^{d+2}}$$

Lax test (1)

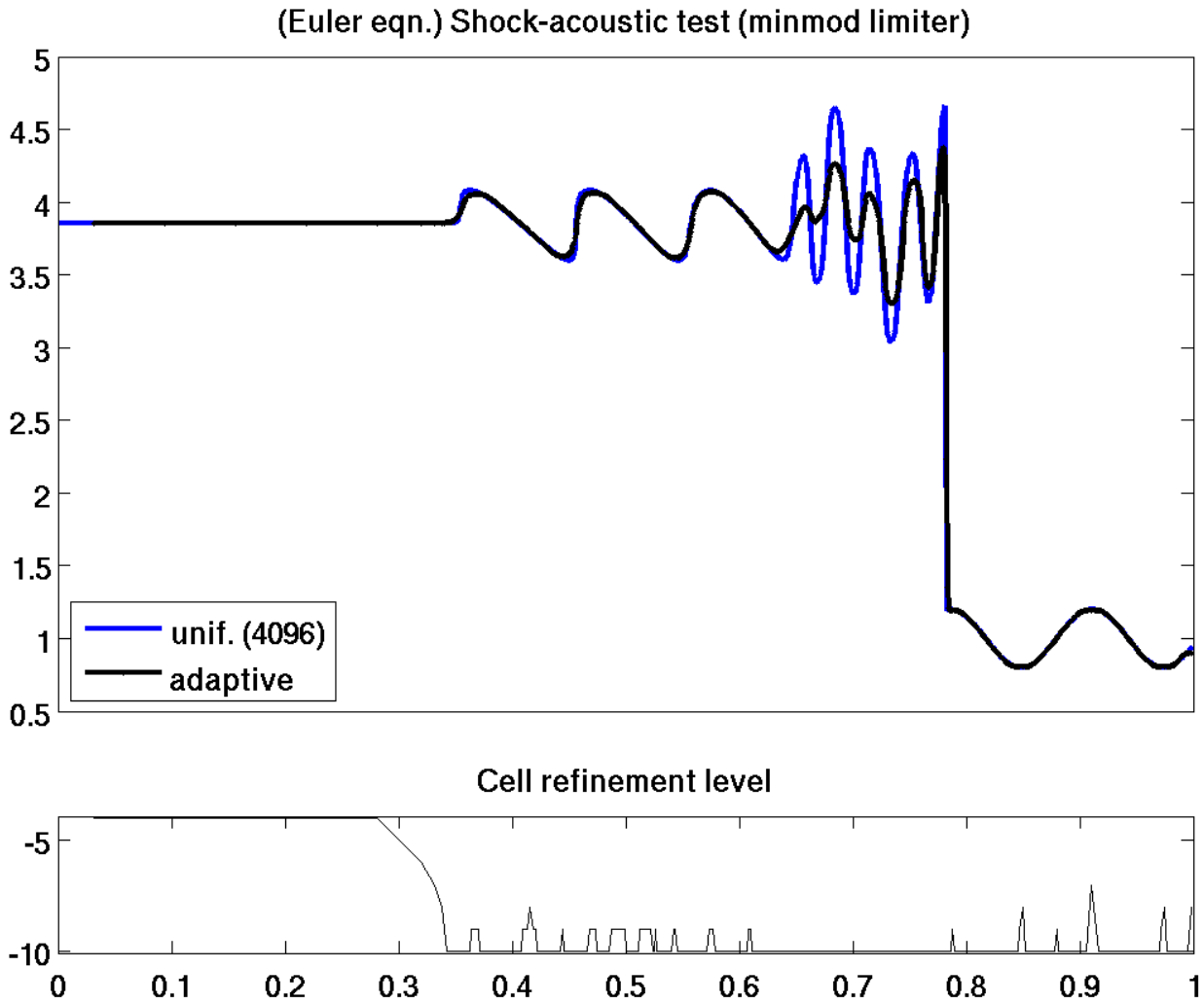

Lax - density profile
Reference (N=2048)
N=512
N=256
adaptive (N=374)
1.2
1
0.8
0.6
0.4
0
0.2
0.4
0.6
0.8
1

Lax test (2)

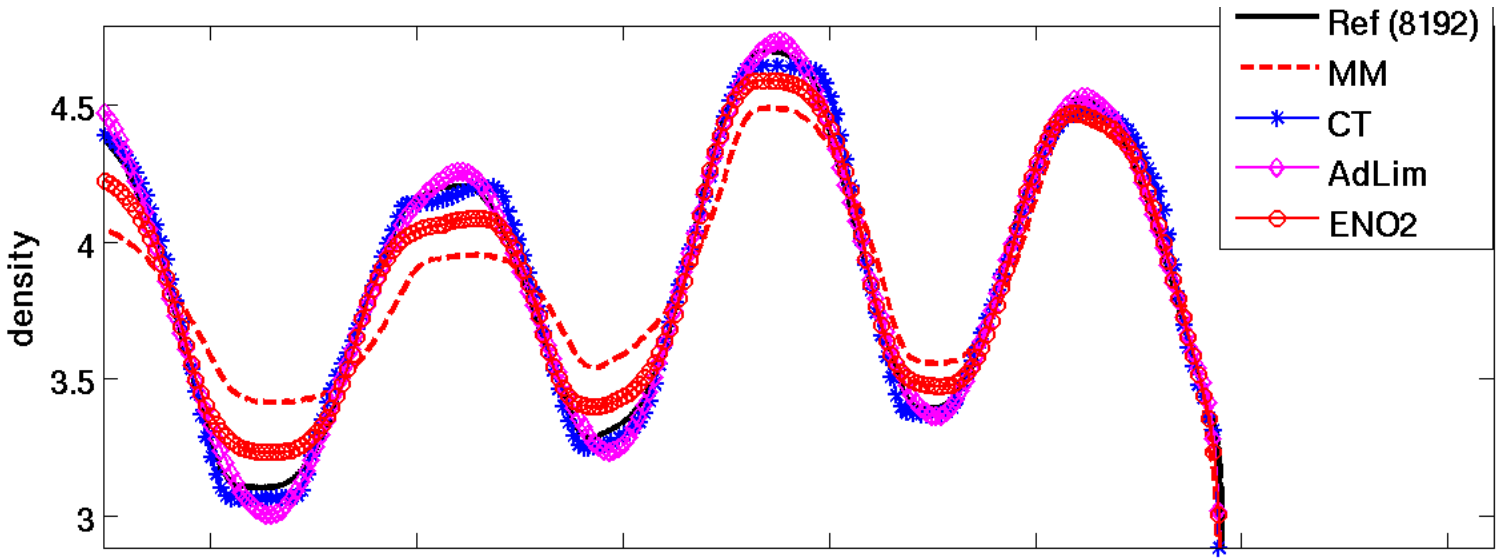

Lax - density profile
1.32
1.3
1.28
1.26
1.24
1.22
1.2
1.18
1.16
1.14
0.72
0.74
0.76
0.78
0.8
0.82
0.84
0.86
0.88
0.9
0.92

Shock-acoustic test problem (1)

Using `minmod` limiter



Shock-acoustic test problem (2)



MM minmod limiter

CT limiter from *Cada, Torrilhon* – J. Comput. Phys. (2009)

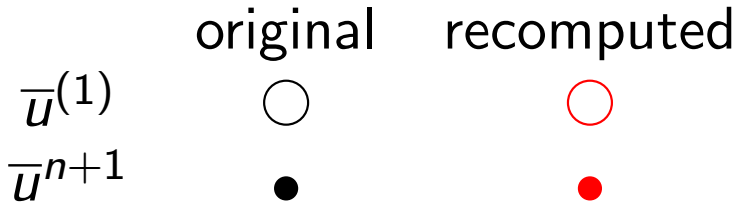
ENO2 second order ENO

AdLim minmod limiter applied only where error indicator is high

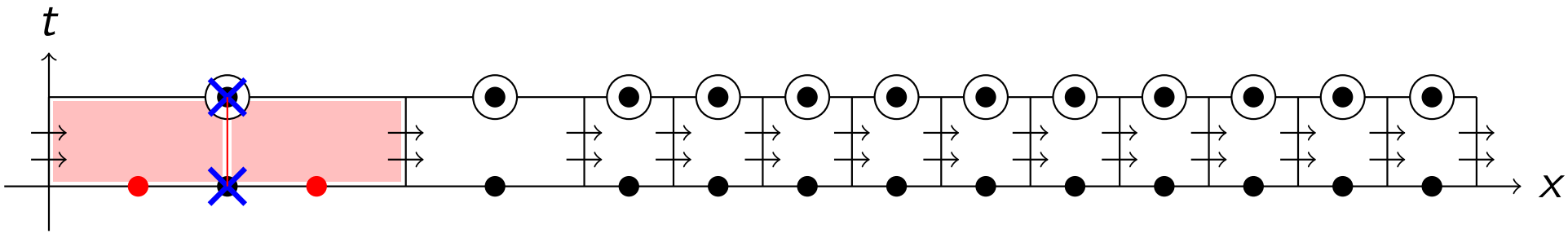
Need for local recomputation

The refine/recompute loop is highly inefficient if we recompute the solution over the whole grid!
Using the numerical domain of dependence, it is possible to predict which cells need recomputing after a given cell is refined.

Recomputation after refinement – uniform Δt

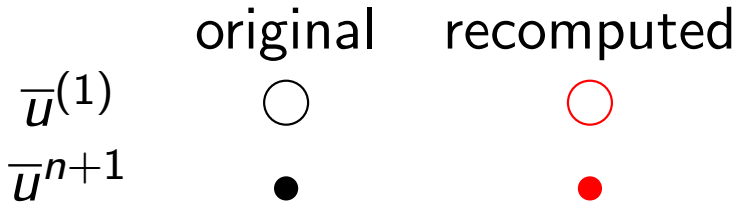


Example: 2-stage RK + linear reconstructions

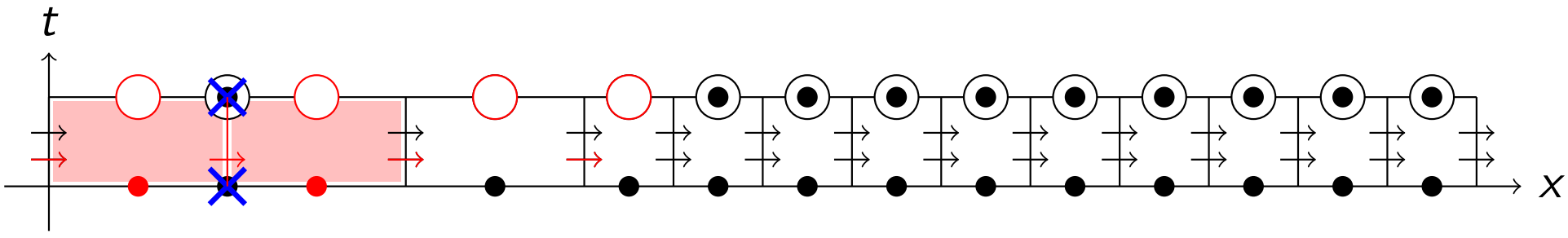


Note: the domain of dependence is enlarged by number of RK stages and reconstruction stencil

Recomputation after refinement – uniform Δt

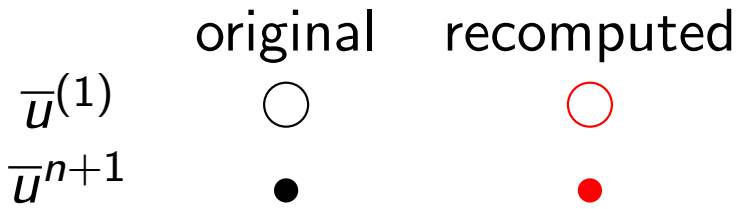


Example: 2-stage RK + linear reconstructions

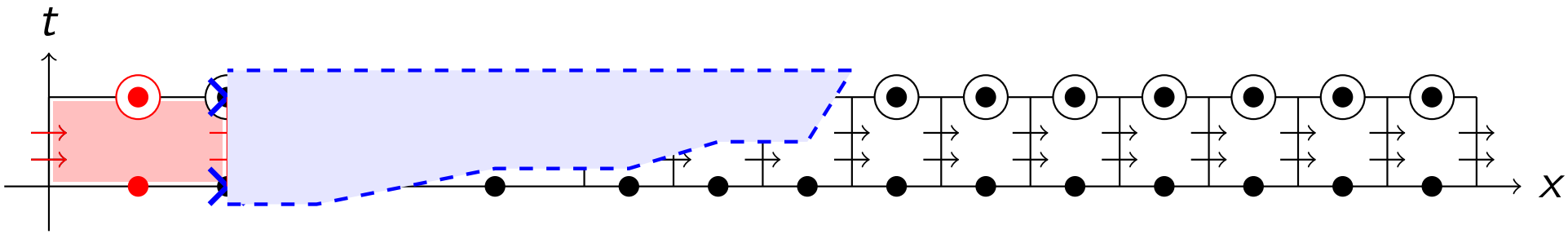


Note: the domain of dependence is enlarged by number of RK stages and reconstruction stencil

Recomputation after refinement – uniform Δt



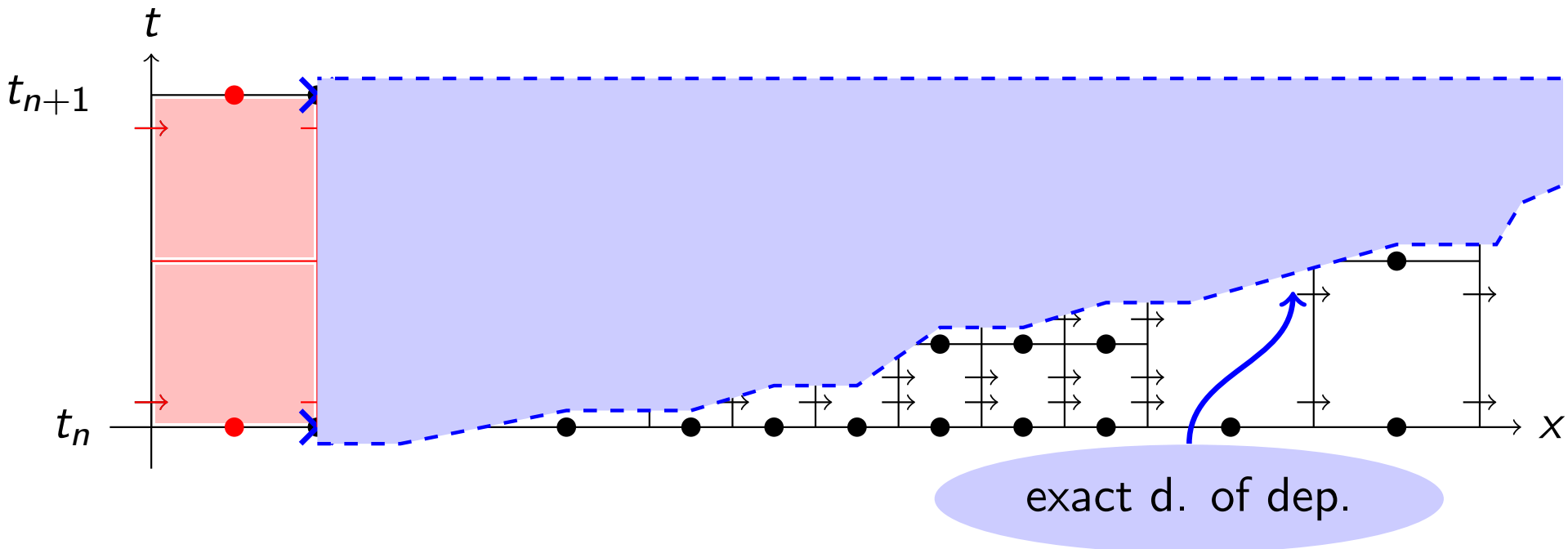
Example: 2-stage RK + linear reconstructions



Note: the domain of dependence is enlarged by number of RK stages and reconstruction stencil

Recomputation after refinement – local timestepping

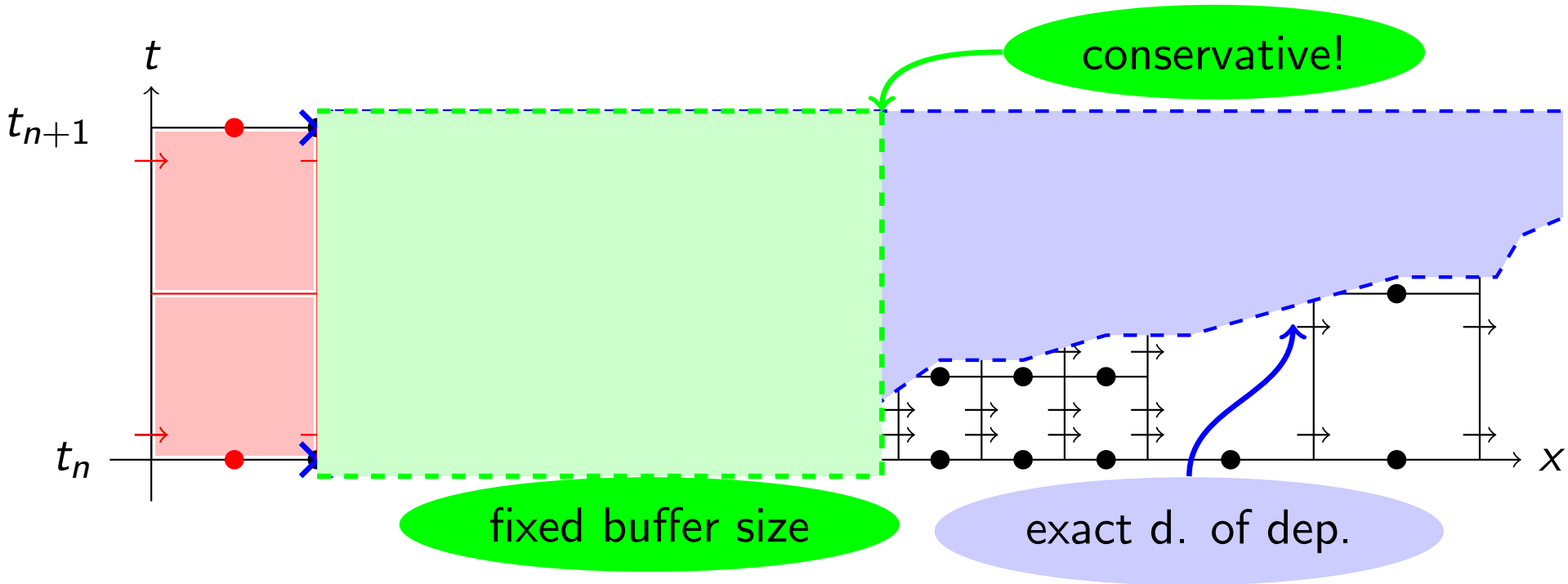
Following the domain of dependence is hard, computationally expensive (and unneeded since we work with an adaptive framework)
The error indicator was happy about the solution away from the first cell, so only recompute red quantitities in the diagram:



We can accumulate fluxes and don't have to store them for each substep!

Recomputation after refinement – local timestepping

Following the domain of dependence is hard, computationally expensive (and unneeded since we work with an adaptive framework)
The error indicator was happy about the solution away from the first cell, so only recompute red quantitities in the diagram:



We can accumulate fluxes and don't have to store them for each substep!

Doswell "frontogenesis" problem

Scalar conservation law, mimicking the mixing of a cold and hot front:

$$u_t + v(x,y)\nabla \cdot u = 0 \quad \text{on } \Omega = [-4,4]^2$$

with velocity

$$v(x,y) = \frac{f(r)}{.385}\left[-\frac{y}{r}, \frac{x}{r}\right]$$

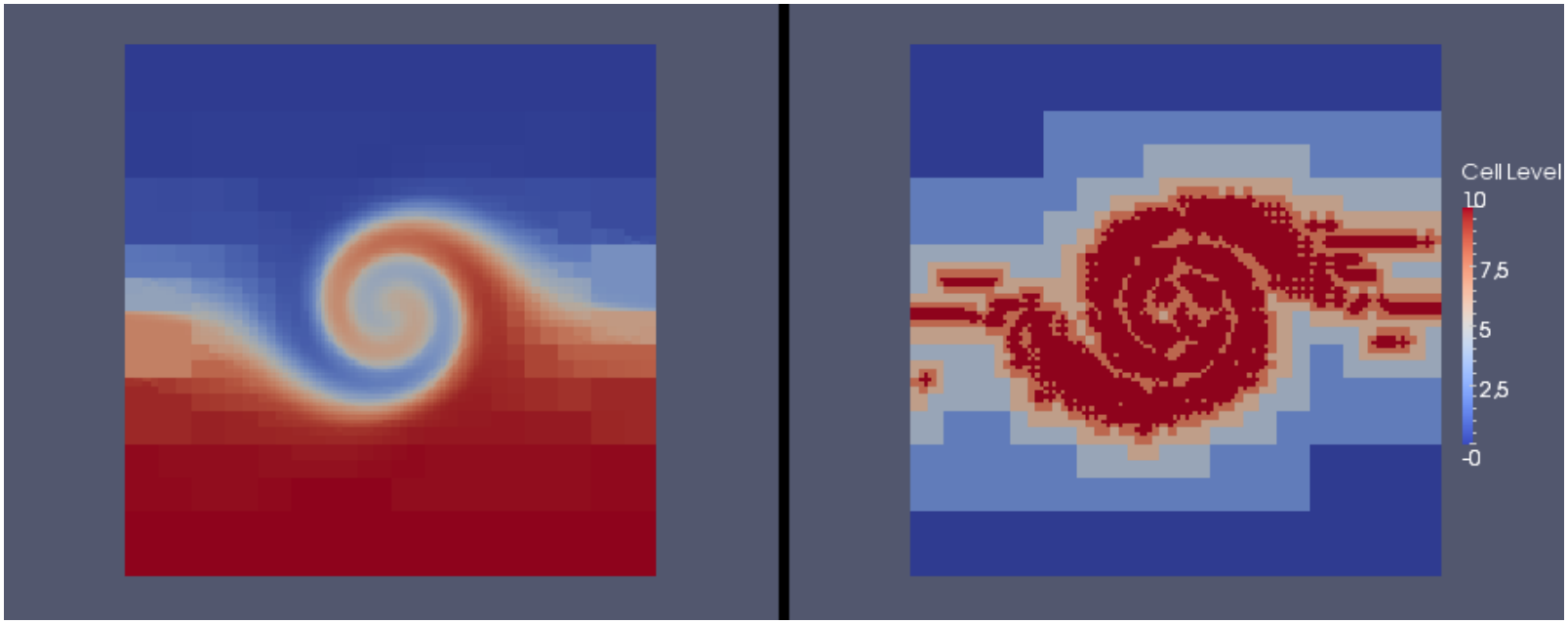
with $r = \sqrt{x^2+y^2}$ and $f = \tanh(r)/(\cosh(r))^2$
and initial data

$$u_0(x,y) = -\tanh(y/2).$$

Atmospheric instability: the solution

solution

cell level

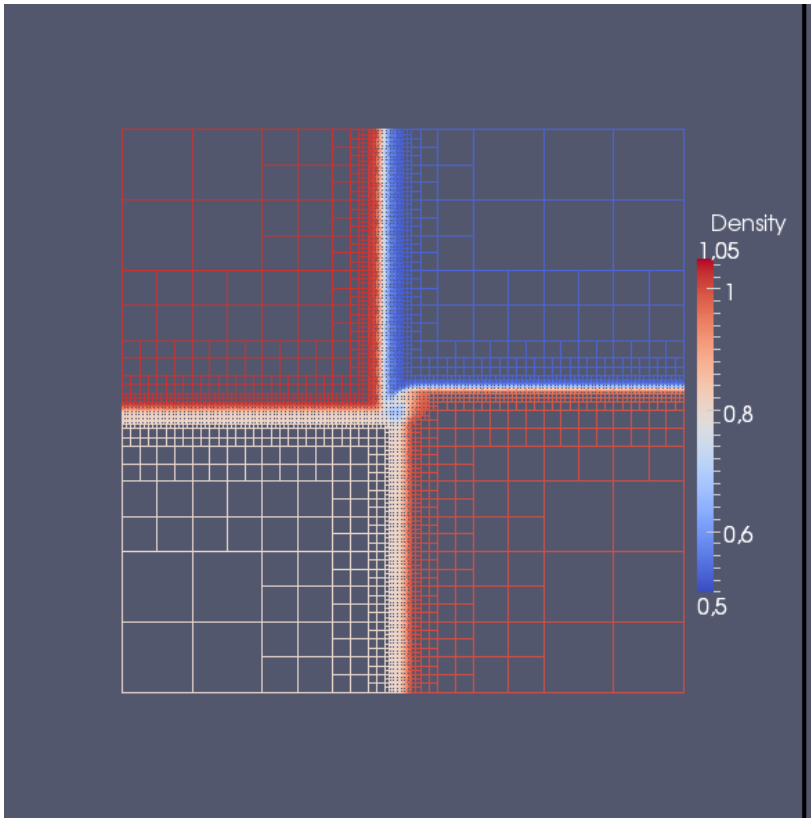


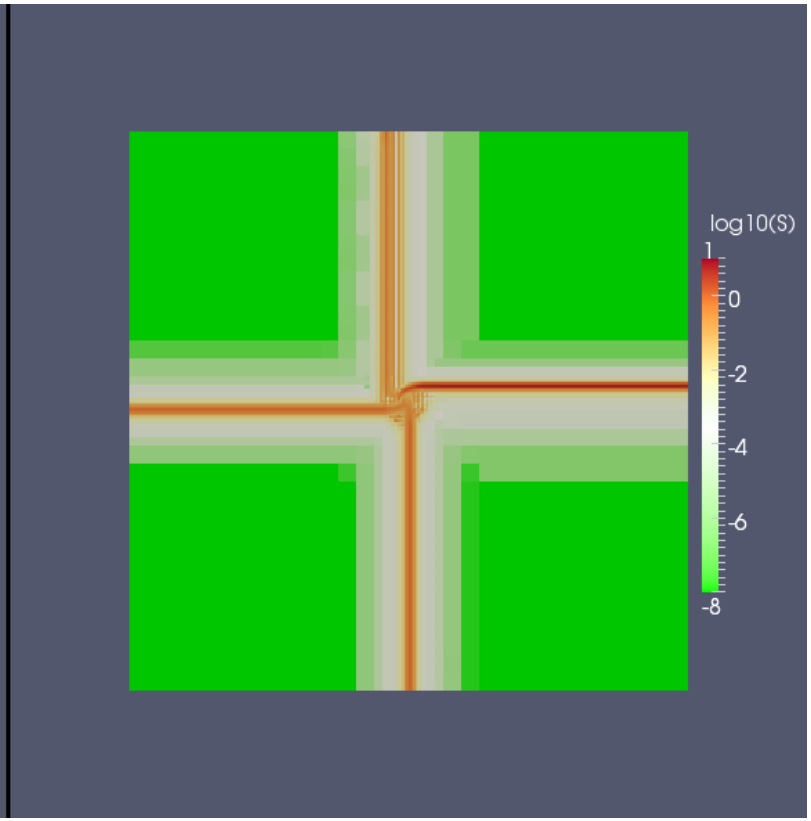
Euler equations: 2D Riemann problem

solution and cell size

entropy indictor (log scale)

initial

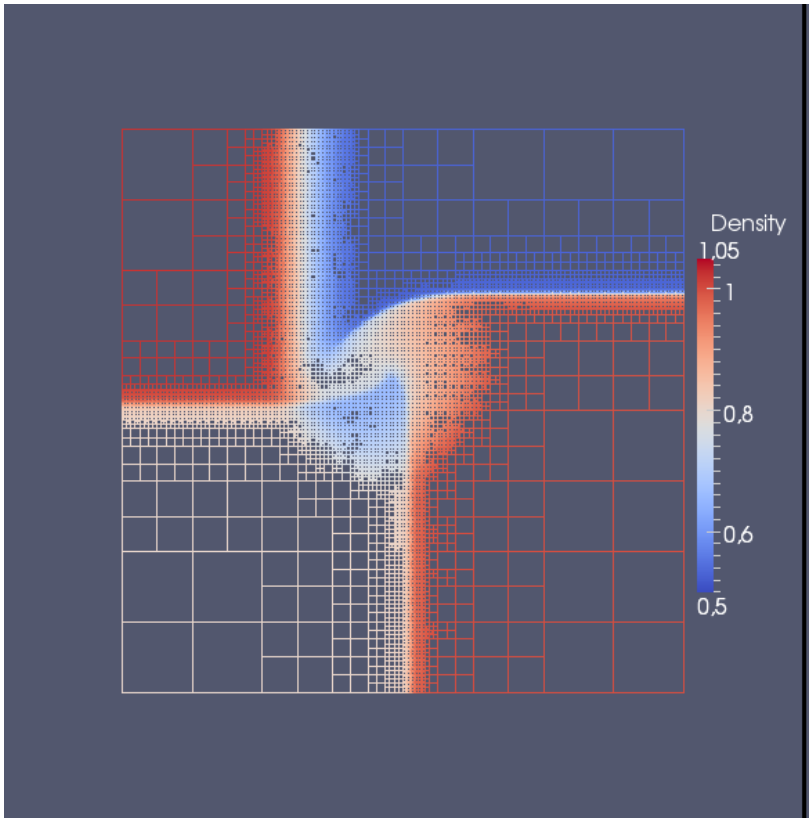




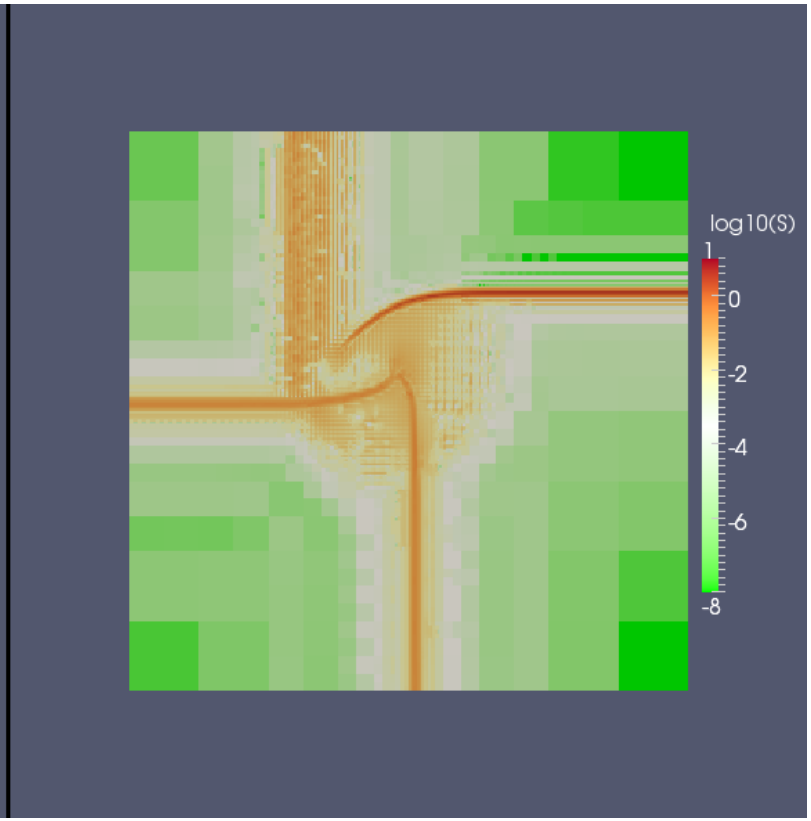
Euler equations: 2D Riemann problem

middle

solution and cell size



entropy indictor (log scale)

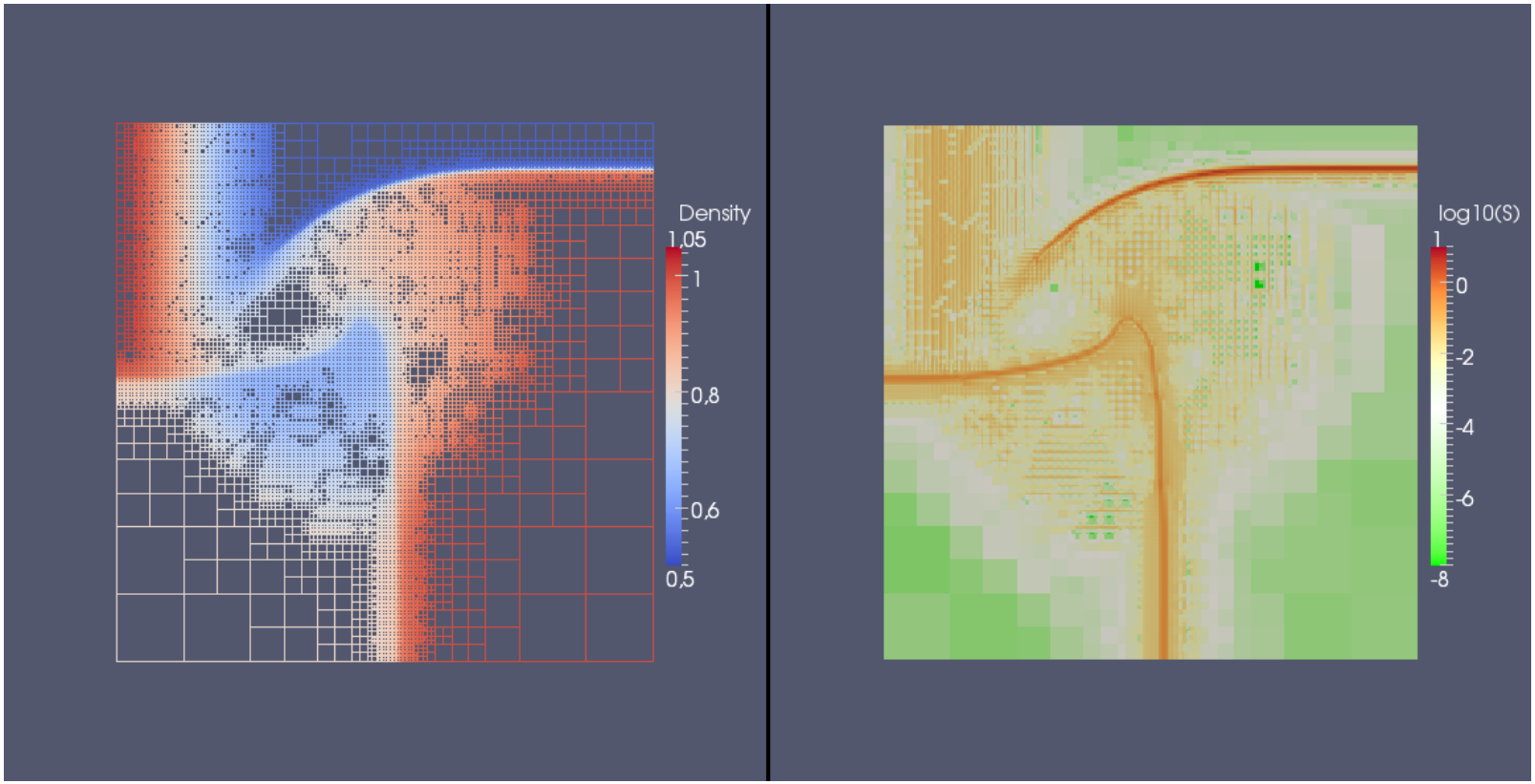


Euler equations: 2D Riemann problem

solution and cell size

entropy indictor (log scale)

final



Perspectives

- third order CWENO reconstructions on nonuniform grids (with G. Russo, A. Coco)
- third order in time (done for global timestepping), coming for local timestepping ...
- source terms and well balancing (with S. Noelle, ...)

All tests were run using a finite volume library that I developed on top of the DUNE environment.
The source code will be released under the GPL2 licence (by the end of 2012). Watch out on *http://www.unito.it/dipmatematica*