Adaptive meshes

Contents

- Why adaptive meshes
- Different strategies
- AMRCART: block structured meshes
- TR3D: cell - by - cell refinemet

Why adaptive meshes?

Because of: spatial size of the object

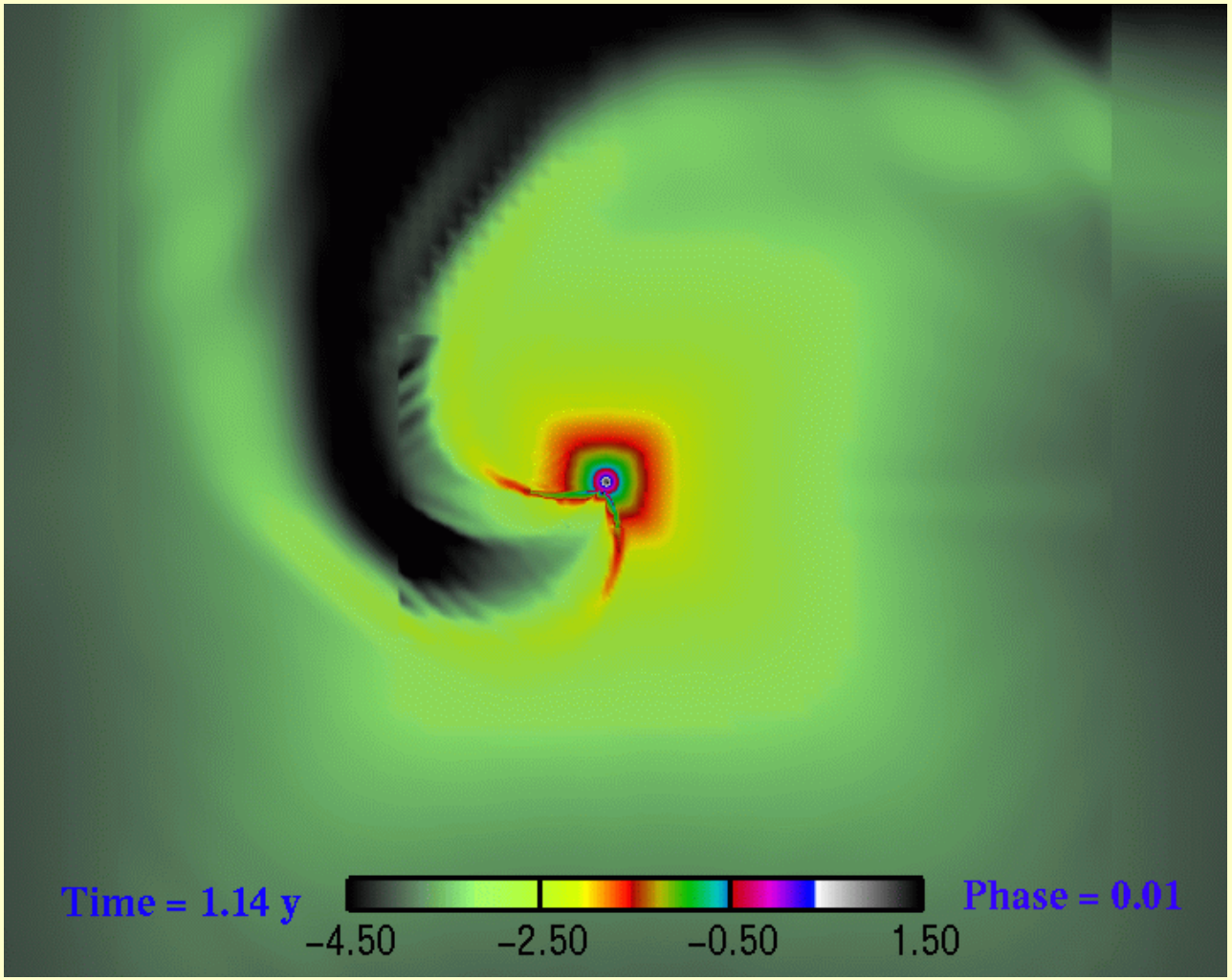
3D hydro simulation of WR 140, a WR+O binary.

To catch large scale density distribution

and

wind- wind collision zone between the two stars you need adaptive grids

Grid cells around stars are 512 times finer than on coarsest grid (catching the global scale).

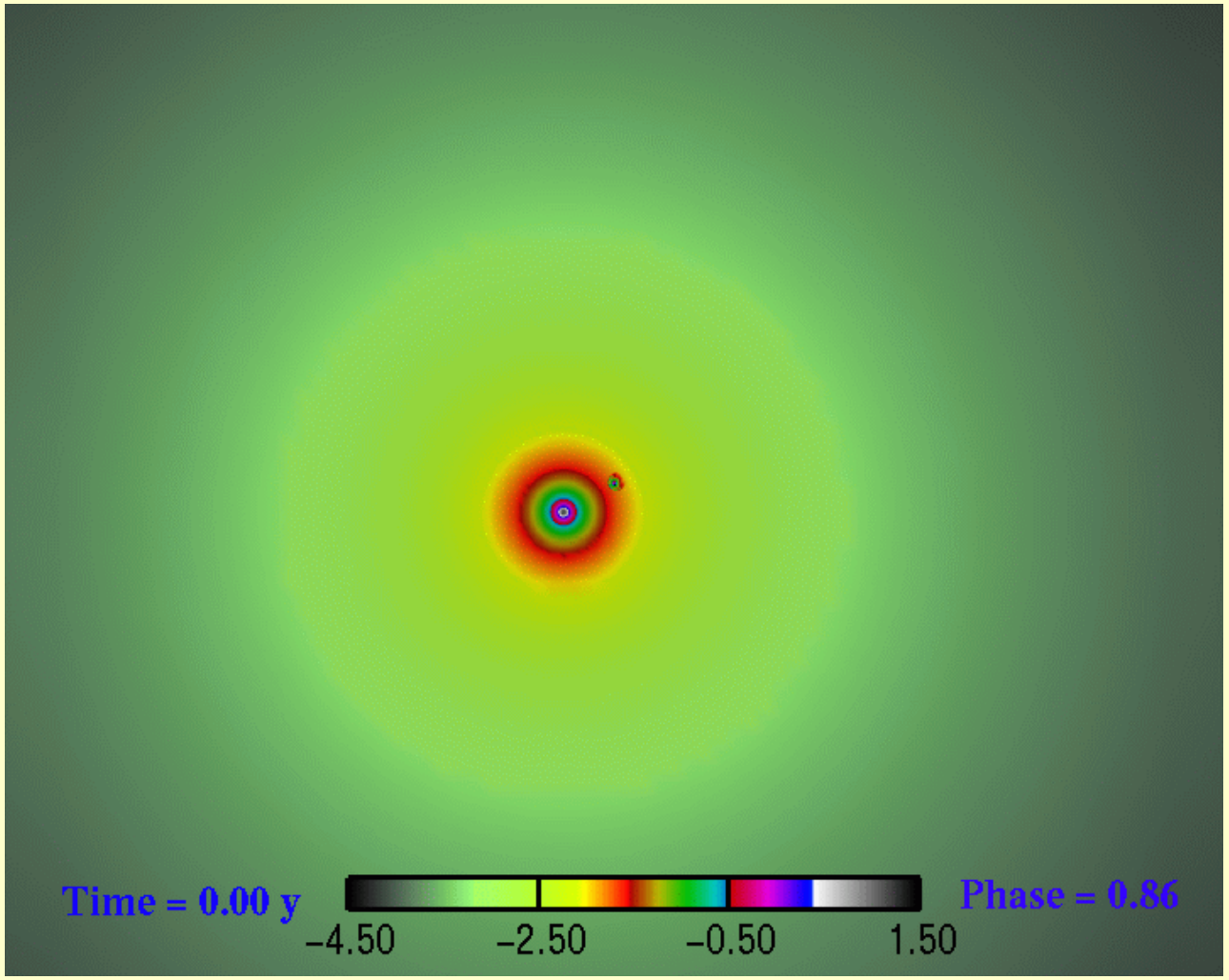


Because of: geometry / boundary

3D hydro simulation of WR 140, a WR+O binary.

To catch (spherical) outflow of each star boundary of star has to be sufficiently well resolved → need adaptive grids

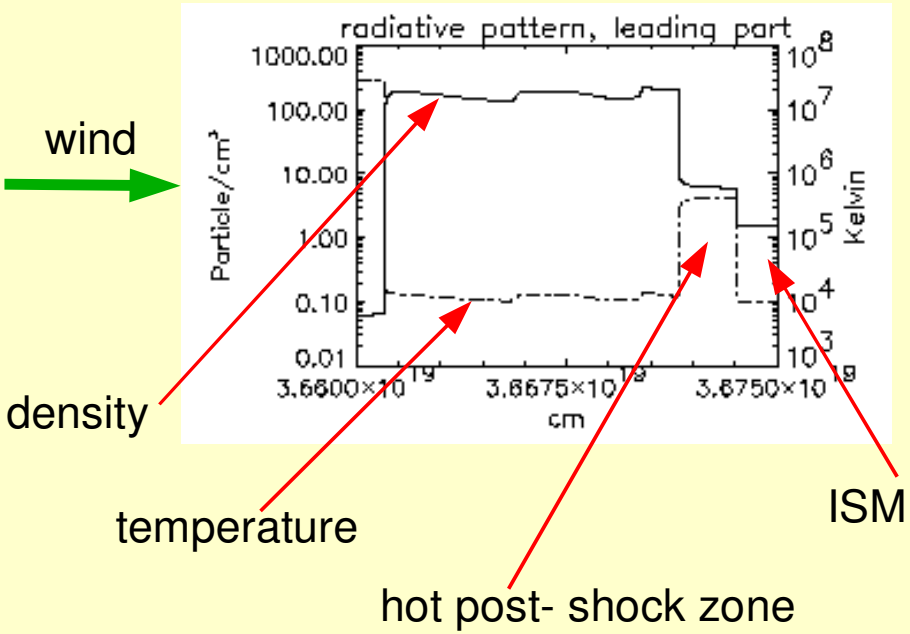
Grid cells around stars are 512 times finer than on coarsest grid (catching the global scale).



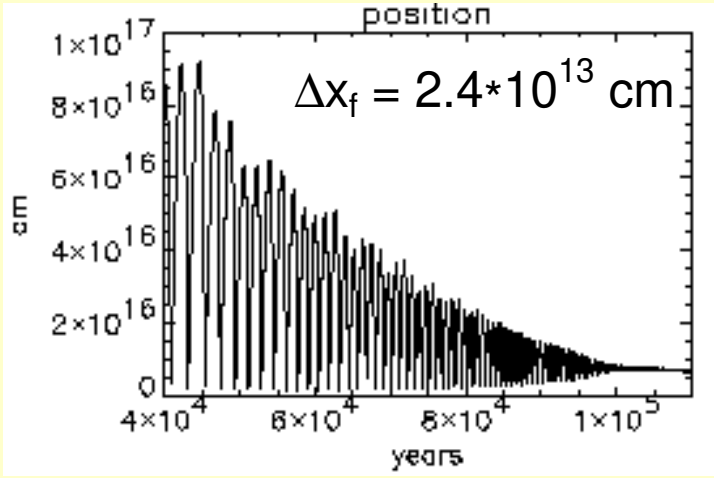
Because of: accuracy

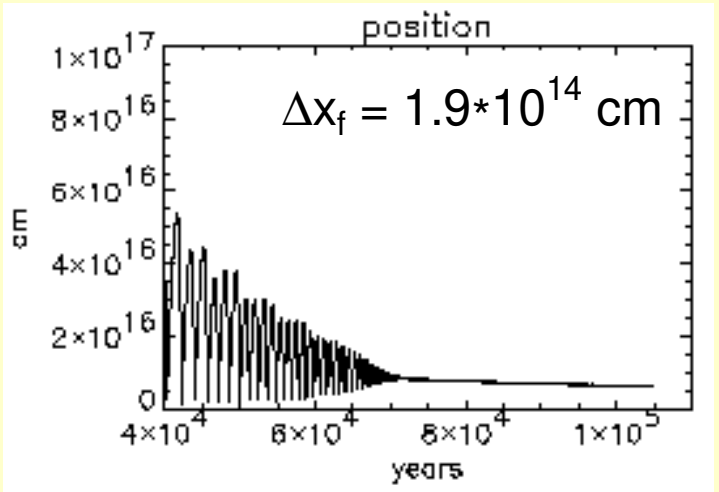
1D hydro simulation of radiative colliding flows.

To capture thermal instability of hot post- shock zone:
need adaptive meshes.



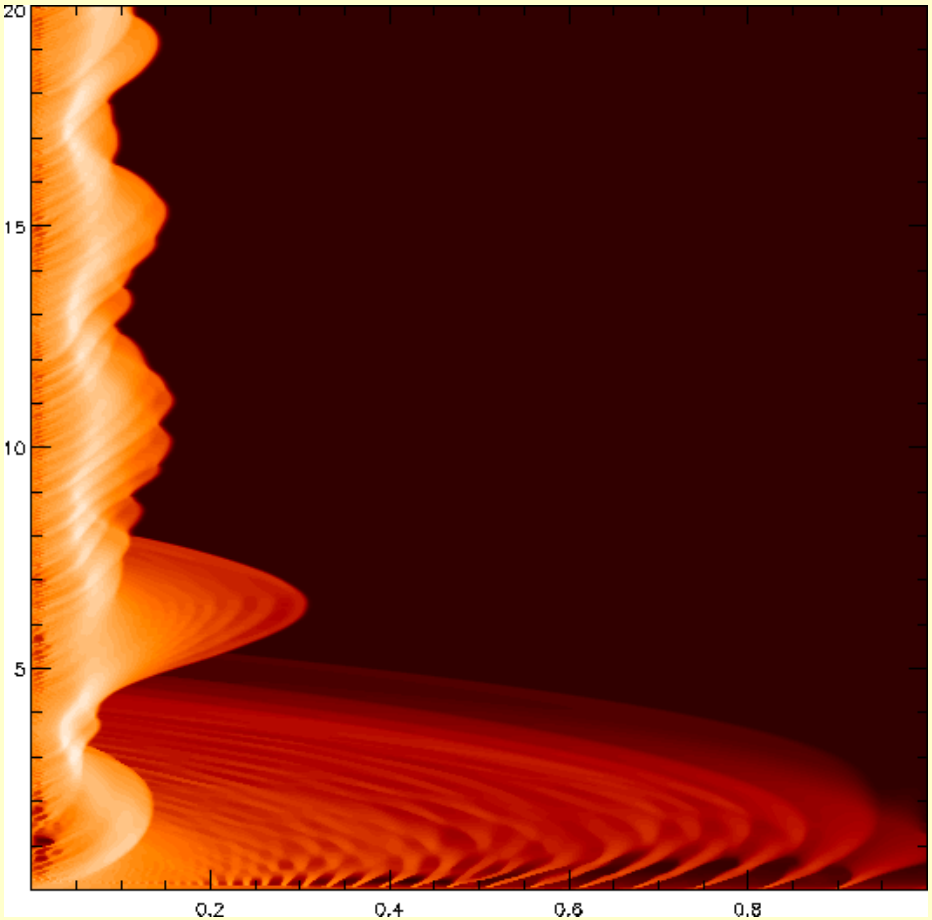
size of hot post-shock zone as function of time for two fine-grid spatial resolutions



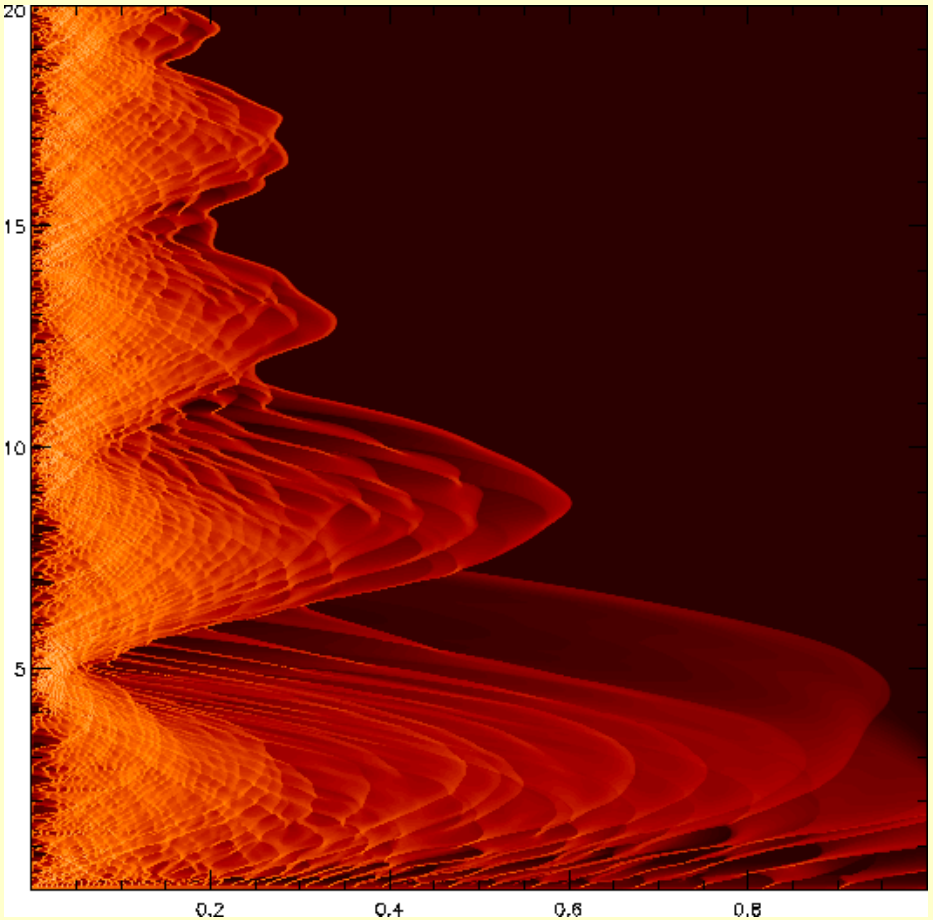


Note: adaptive mesh useful for local refinement only

If you have structures to resolve all over
your domain, adaptive meshes wont help...



1000 cells



5000 cells

1D MHD simulations of Alfvéen- wave in plane parallel, self- gravitating slab

Because of: physics that must be resolved

If your physics happens on a scale Δx

and

if this physics is important for your model

and

if you cannot account for this physics by subgrid- scale modeling (as e.g. LES)

$\Rightarrow$ you must resolve the scale Δx!

Different strategies

Time- dependent or not?

Time- dependent problems:

You must adapt your grid to your solution as this solution evolves.

Stationary problems:

You may desigen your grid in advance, before starting the actual simulation. Note: such grid generation may take as much CPU- time as the computation itself.

Different ways of adaptive meshes

block- structured meshes (Cartesian mesh):

variable number of cells, refine rectangular regions

Pros: efficient storage of grid

Cons: refine larger area than is needed

cell- by- cell (Cartesion mesh)

variable number of cells, refine only the 'bad' cells

Pros: you calculate only what is necessary

Cons: need to store the grid

'moving grids' (time- dependent simulations only)

fixed number of cells, move cells to where needed

Pros: know your memory

Cons: you may run out of points

cell- by- cell (triangular meshes)

variable number of cells, refine only the 'bad' cells

Pros: fit best to your bodies / solution

Cons: expensive storage and computation

AMRCART: block structured meshes

Adaptive meshes at the example of a jet calculation

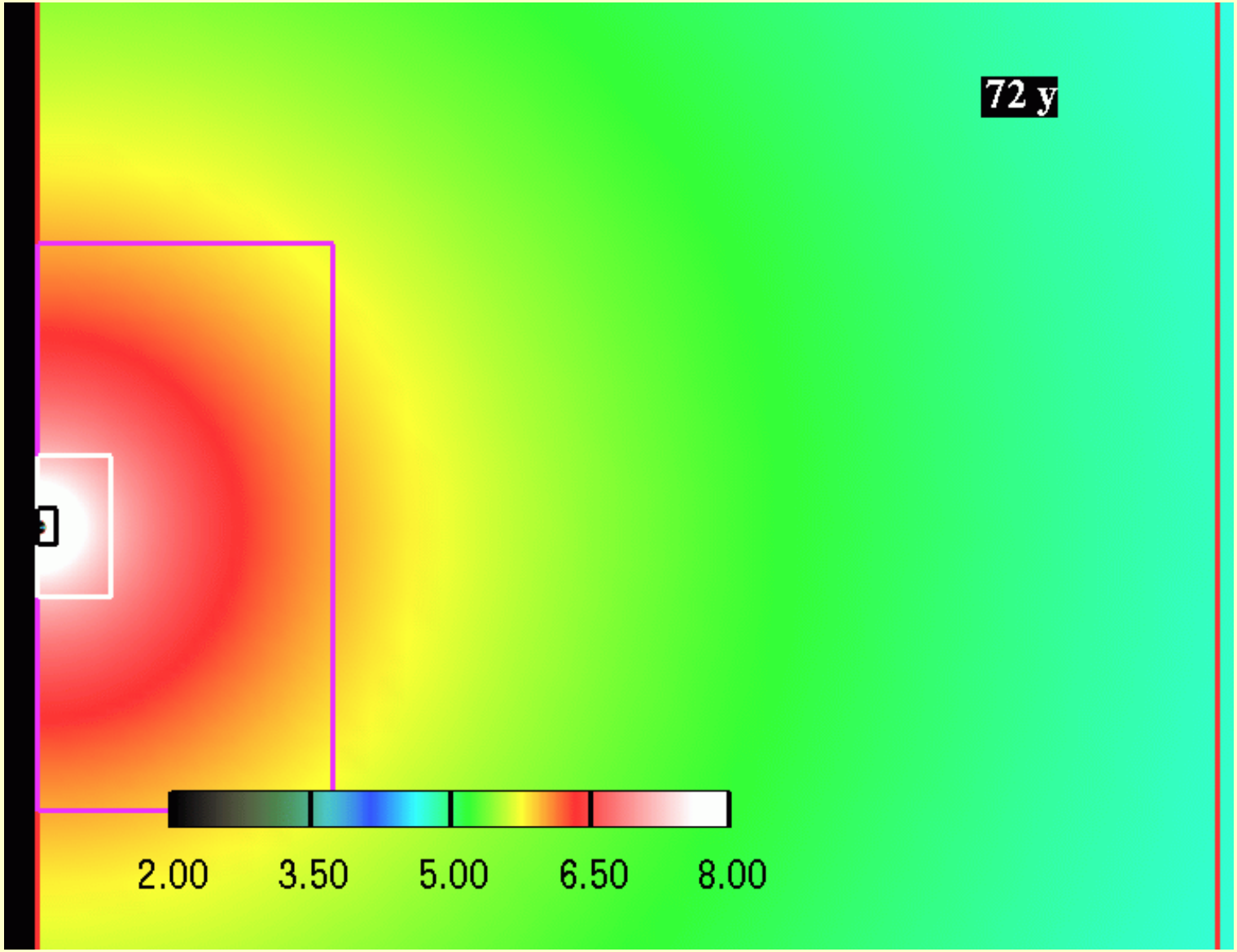
level 2

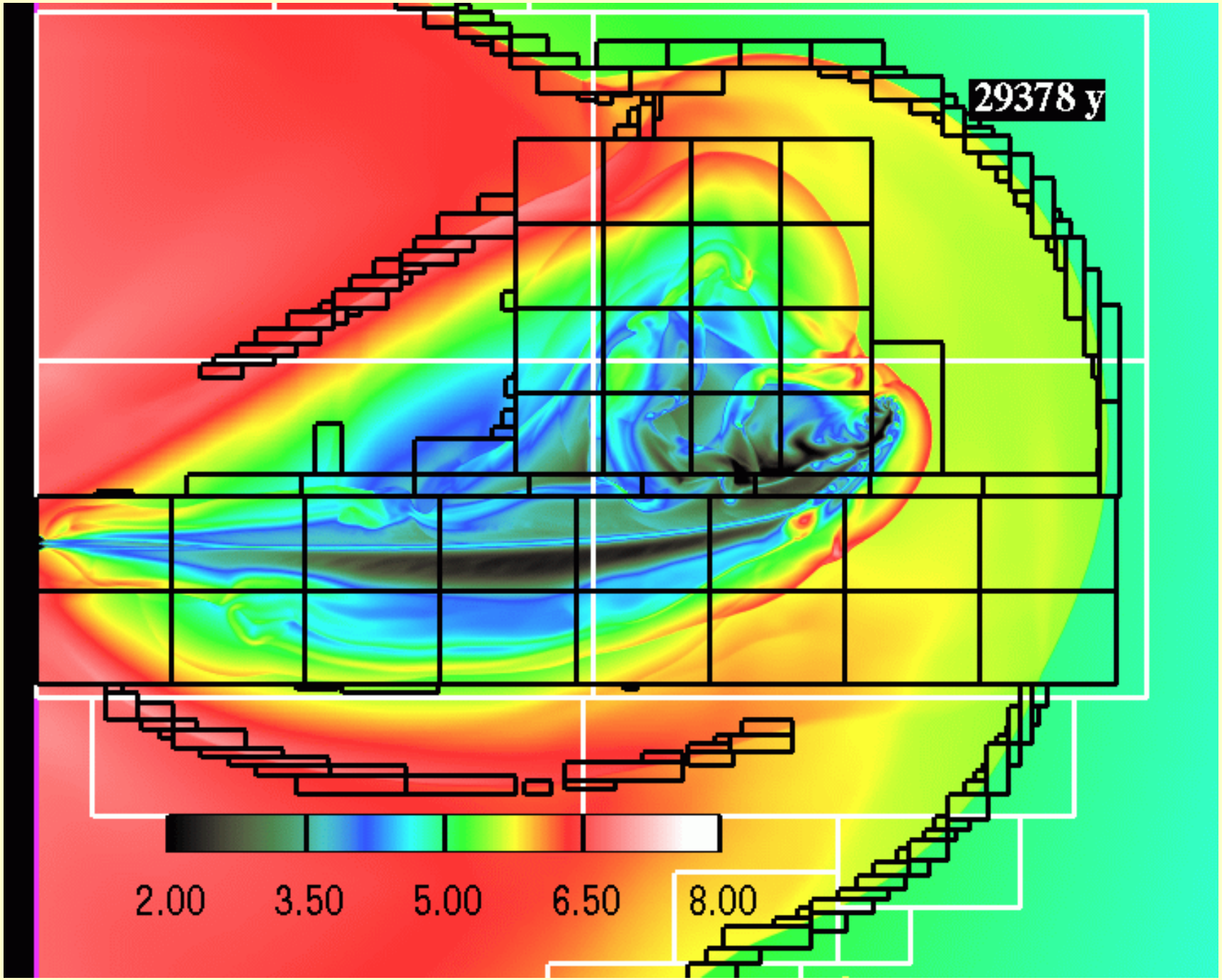
level 3

level 4
(white)

level 5

level 1:
not shown




29378 y
2.00
3.50
5.00
6.50
8.00

Basic idea

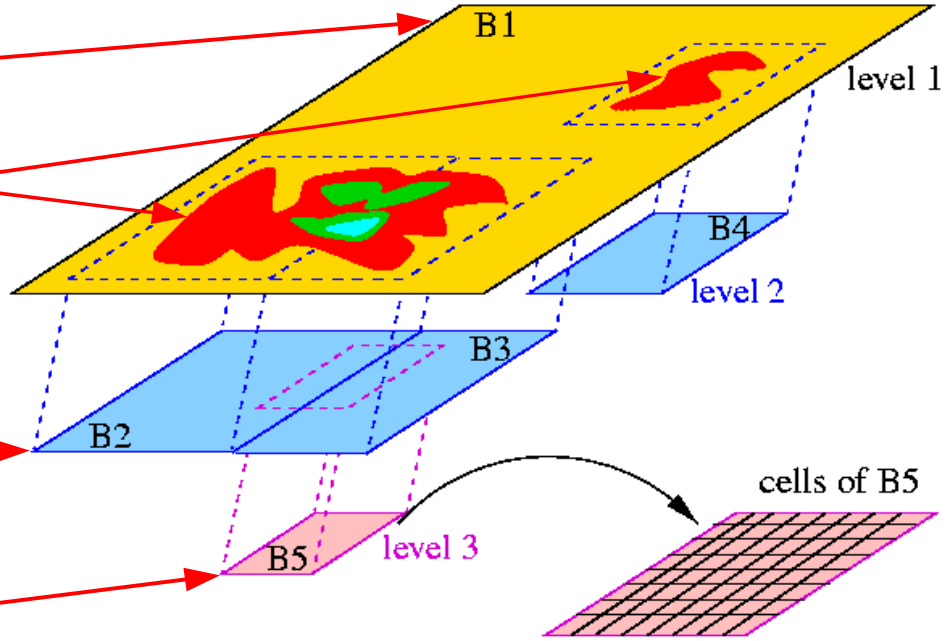
total computational domain: level 1

regions to be refined

refined rectangular blocks around these regions: level 2

level 3

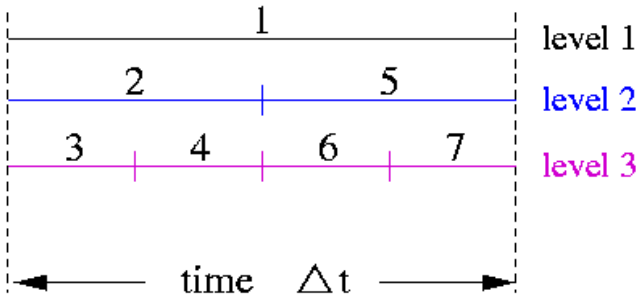
block structured A M R



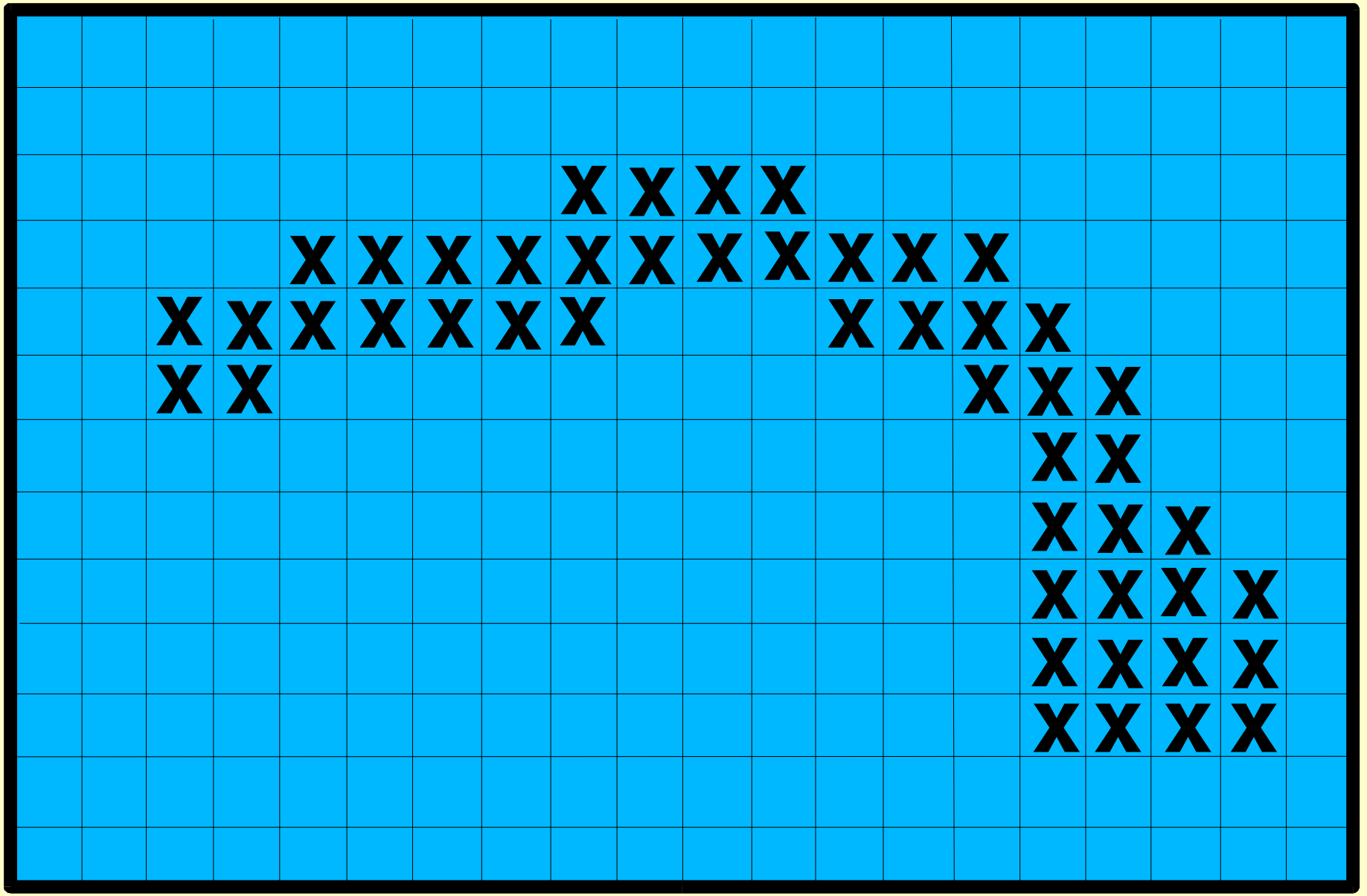
storing the grid: list of independent blocks
refinement ratios between levels

storing a block: l_x, l_y, l_z, one corner of each block
pointer to ...

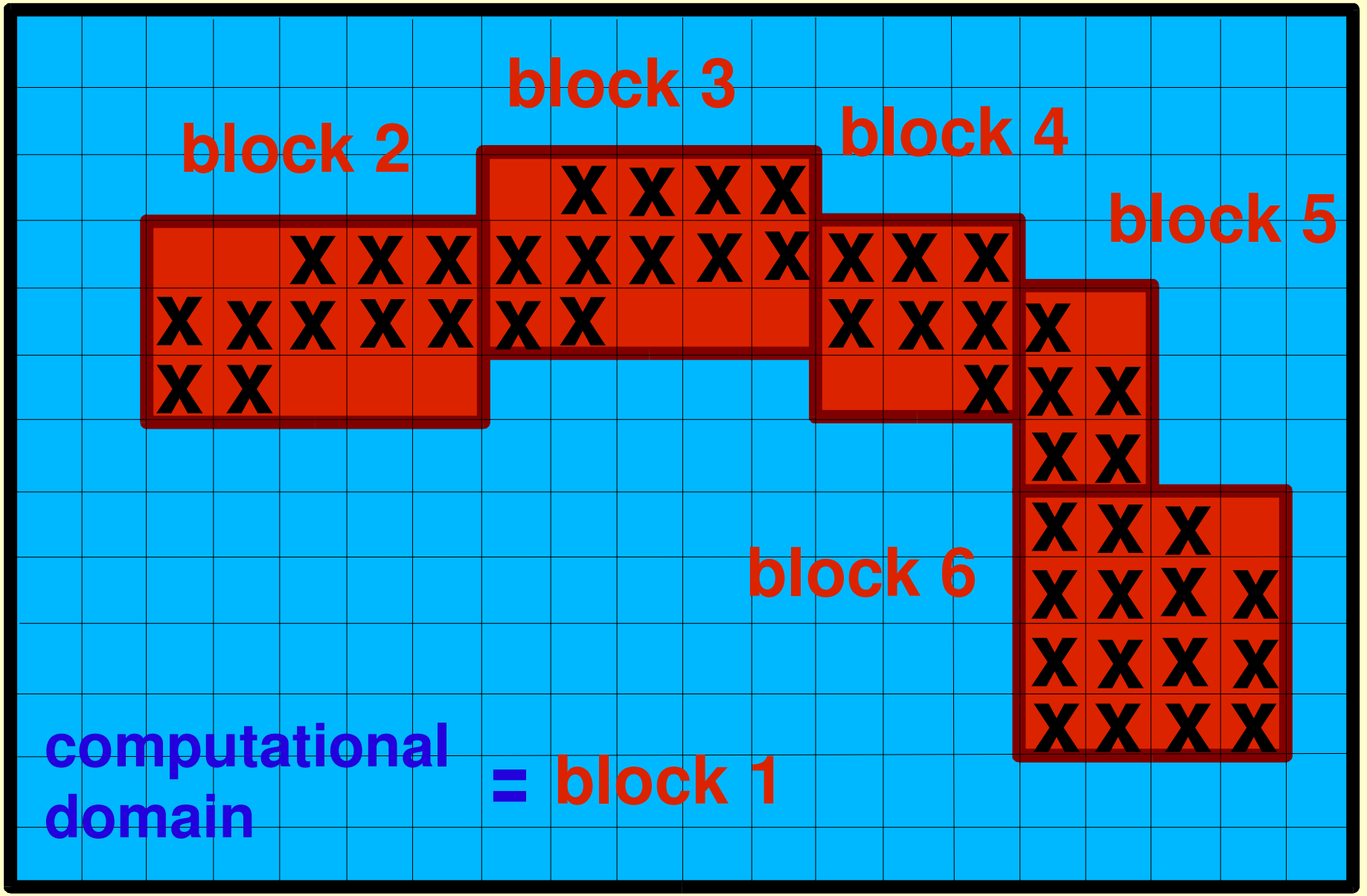
integration scheme:



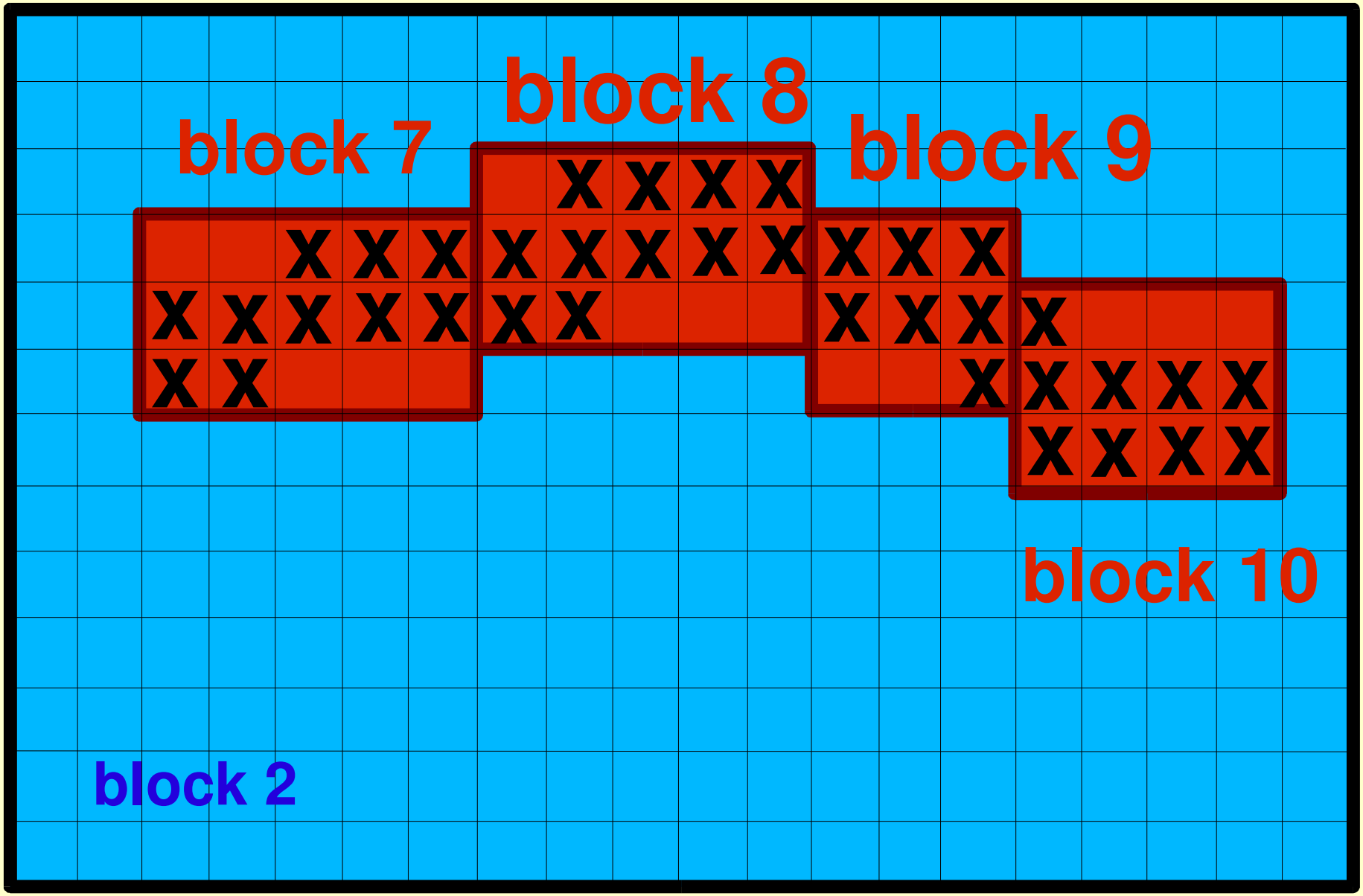
cells to be refined: x



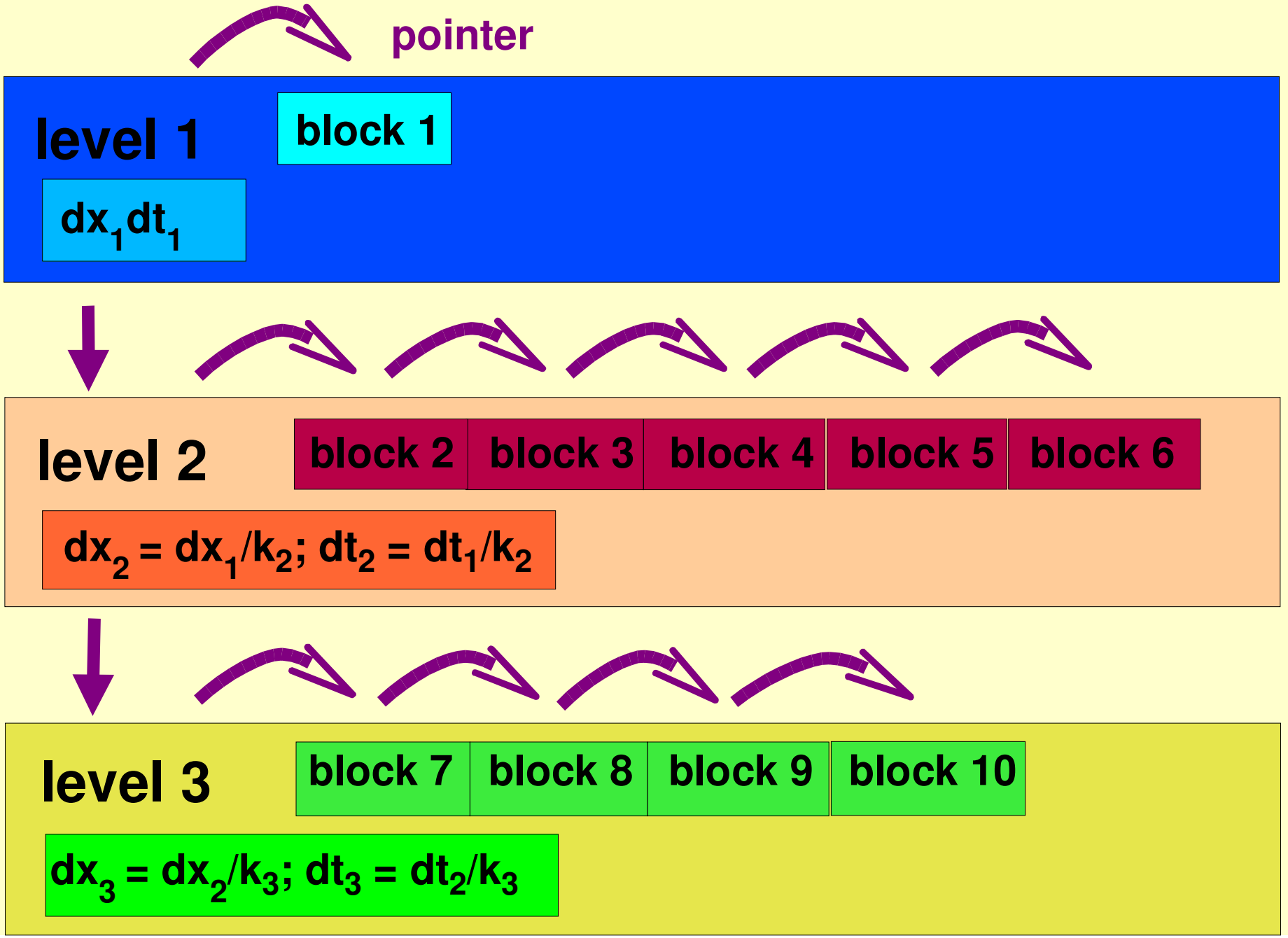
block-structered refinement I



block-structered refinement II

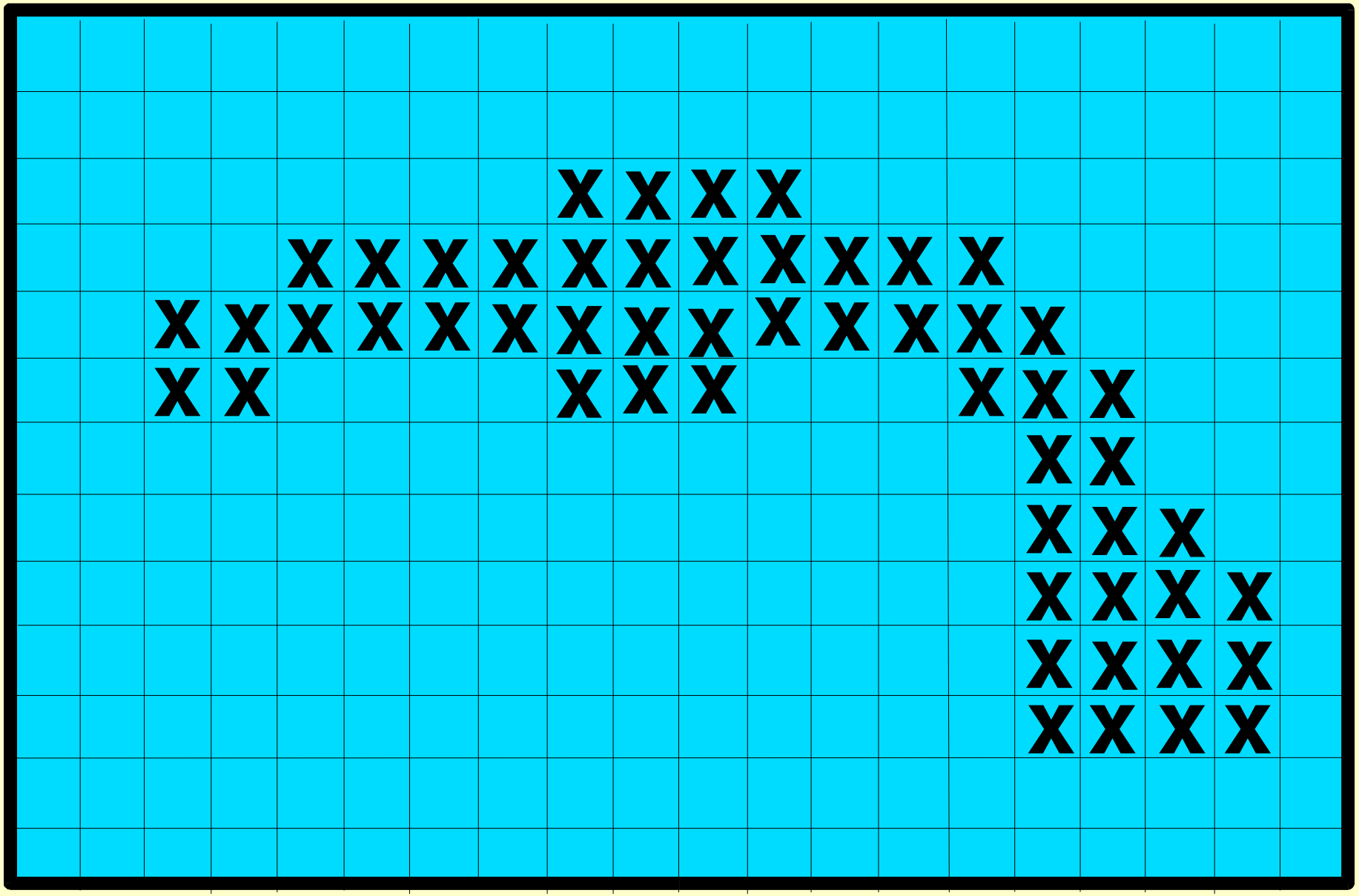


block-structered refinement III

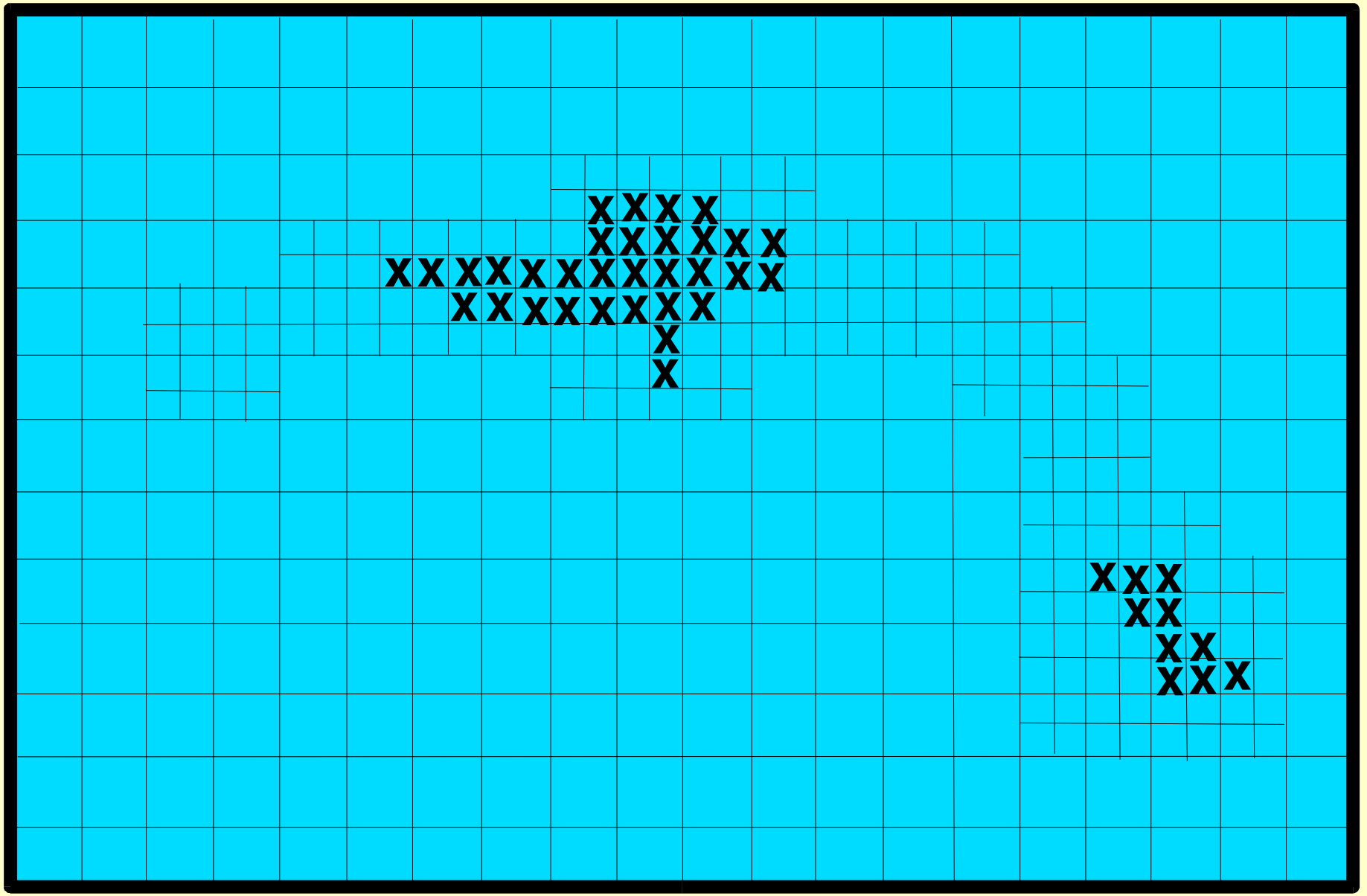


TR3D: cell- by- cell refinement

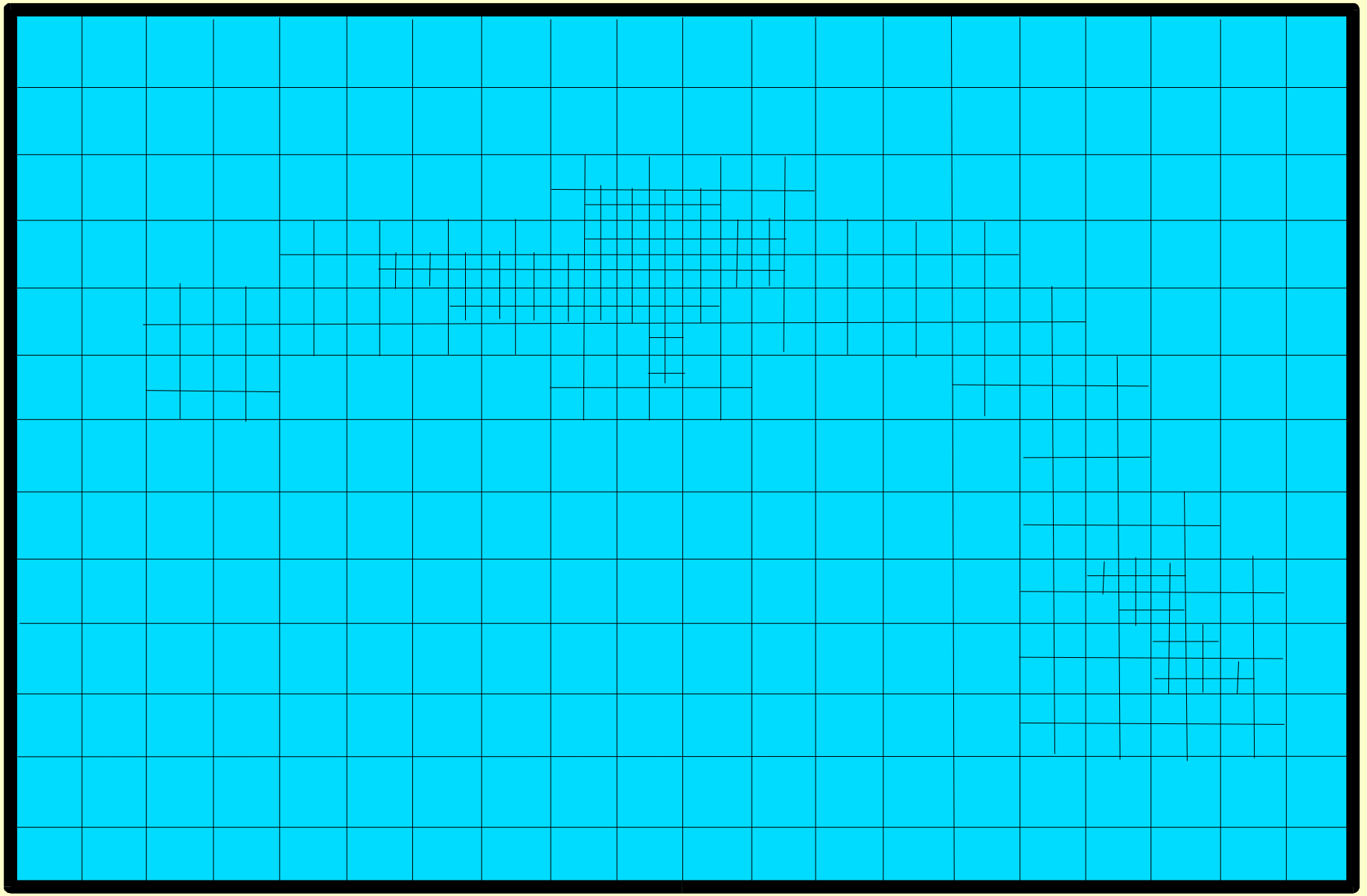
cells to be refined: x



cells to be refined: x



cells to be refined: x x

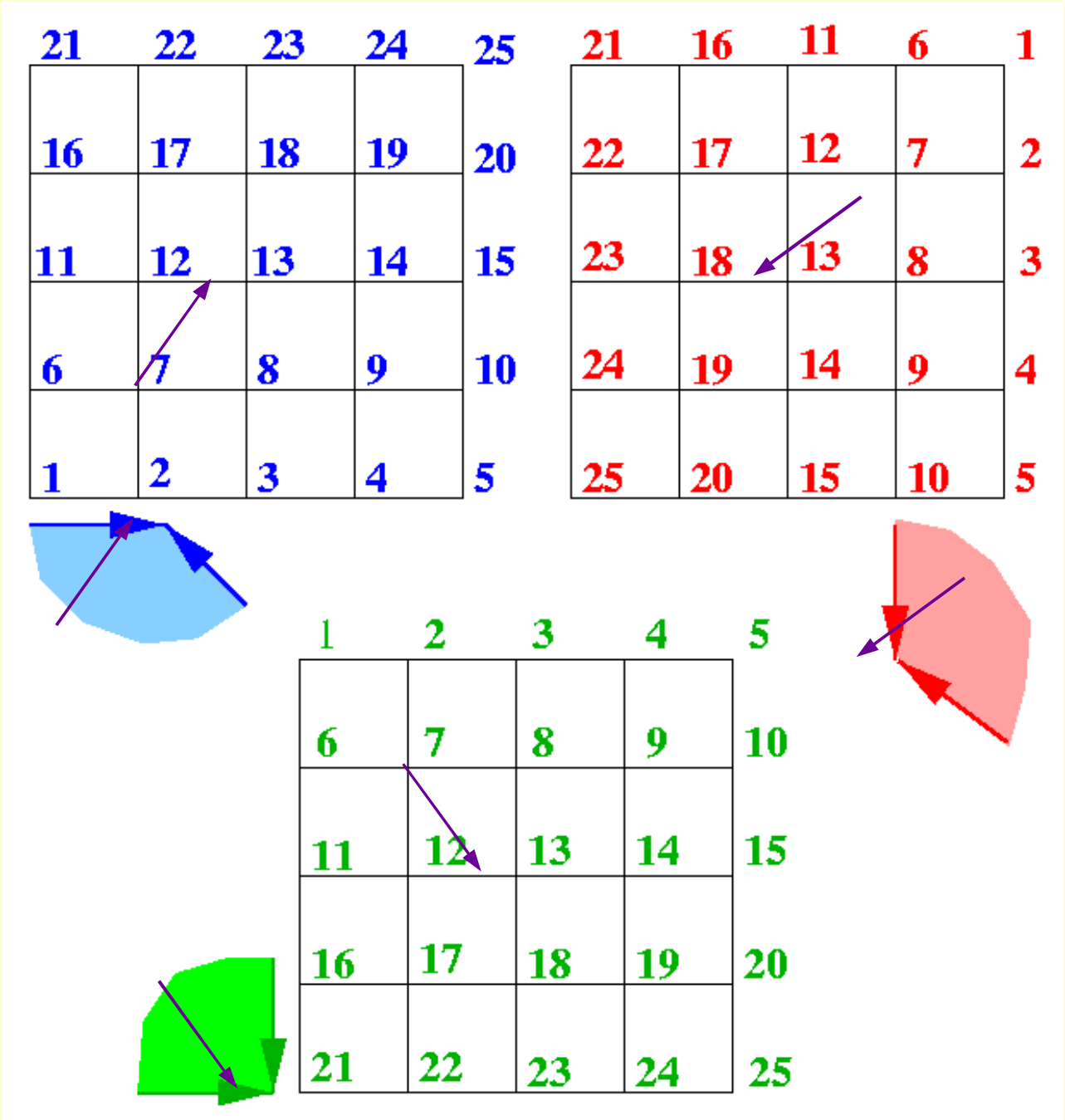


Remember: have to renumber grid such that T_h^m is lower triagonal

Earlier 2D regular grid example:

Any given discrete ordinate direction m will fall into one of the three ranges of angel shown (blue, red, or green)

The numbering for each of these three ranges is such that the point the ordinate m is going to has always a higher number than any points limiting cell boundaries which have been crossed already by the ordinate.



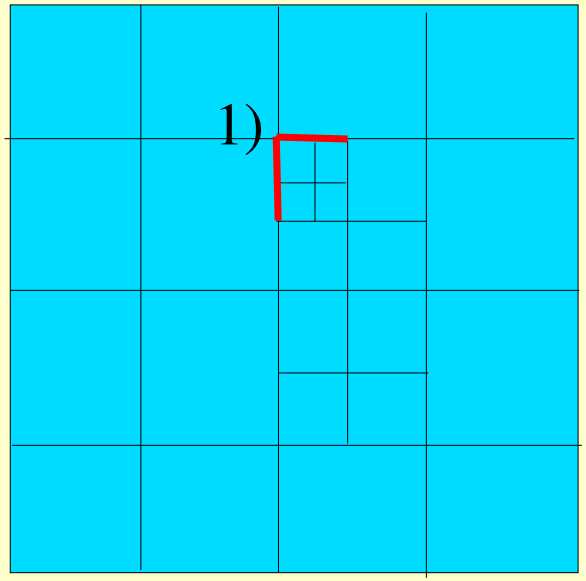
Cell- by- cell refinement in TR3D: restrictions

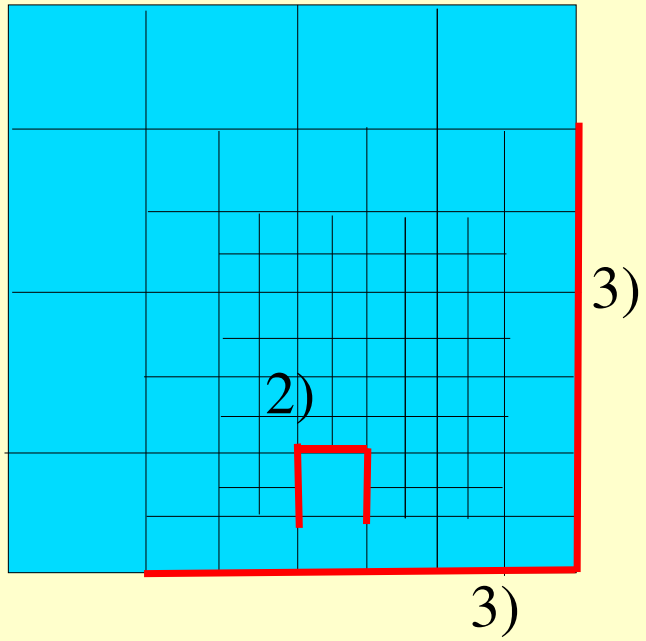
In order to be able to number grid appropriately and efficiently (not too many different numberings):

1) refinement ratio from between cells must not exceed 2

2) must not have concave regions of only 2 cells or less

3) boundary of computational domain must be coarse grid





Summary

Adaptive meshes are neede because of

- spatial size of an object
- geometry / boundary of an object
- accuracy of calculation (numerical diffusion)
- the physics has to be resolved

Hydro- and MHD- codes in A- MAZE use block- structured meshes

Pros: need not store grid $\rightarrow$ good for memory

Cons: refined region too large

(but only 8 variables per cell $\rightarrow$ not too bad)

TR3D uses cell- by- cell refinement (# 100 variables per cell)

Pros: refine exactly what you need

$\rightarrow$ with a few 100 variables per cell: good for memory

Cons: data structure (need to store grid)

D3NEBEL solves an ODE. In that sense: no grid at all.