PiTP Summer School 2009

Plan for my lectures

Volker Springel

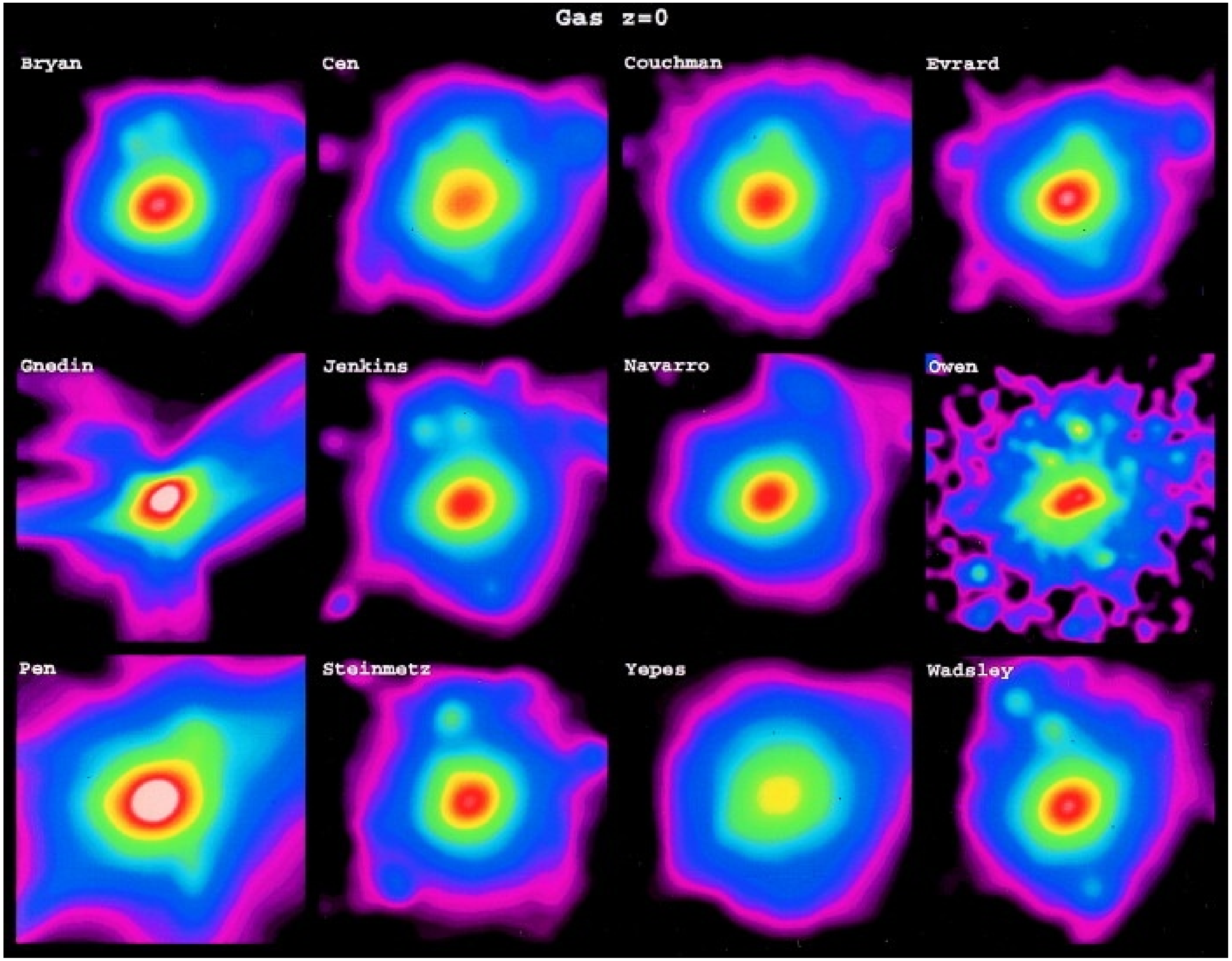
Lecture 1	▶ Basics of collisionless dynamics and the N-body approach
Lecture 2	▶ Gravitational solvers suitable for collisionless dynamics, parallelization
Lecture 3	▶ More parallelization, Introduction to smoothed particle hydrodynamics
Lecture 4	▶ Algorithmic aspects of SPH, caveats, applications
Lecture 5	▶ Comparison of SPH to finite volume methods, Moving-mesh hydrodynamics

Accuracy issues in cosmological simulations

Different hydrodynamical simulation codes are broadly in agreement, but show substantial scatter and differences in detail

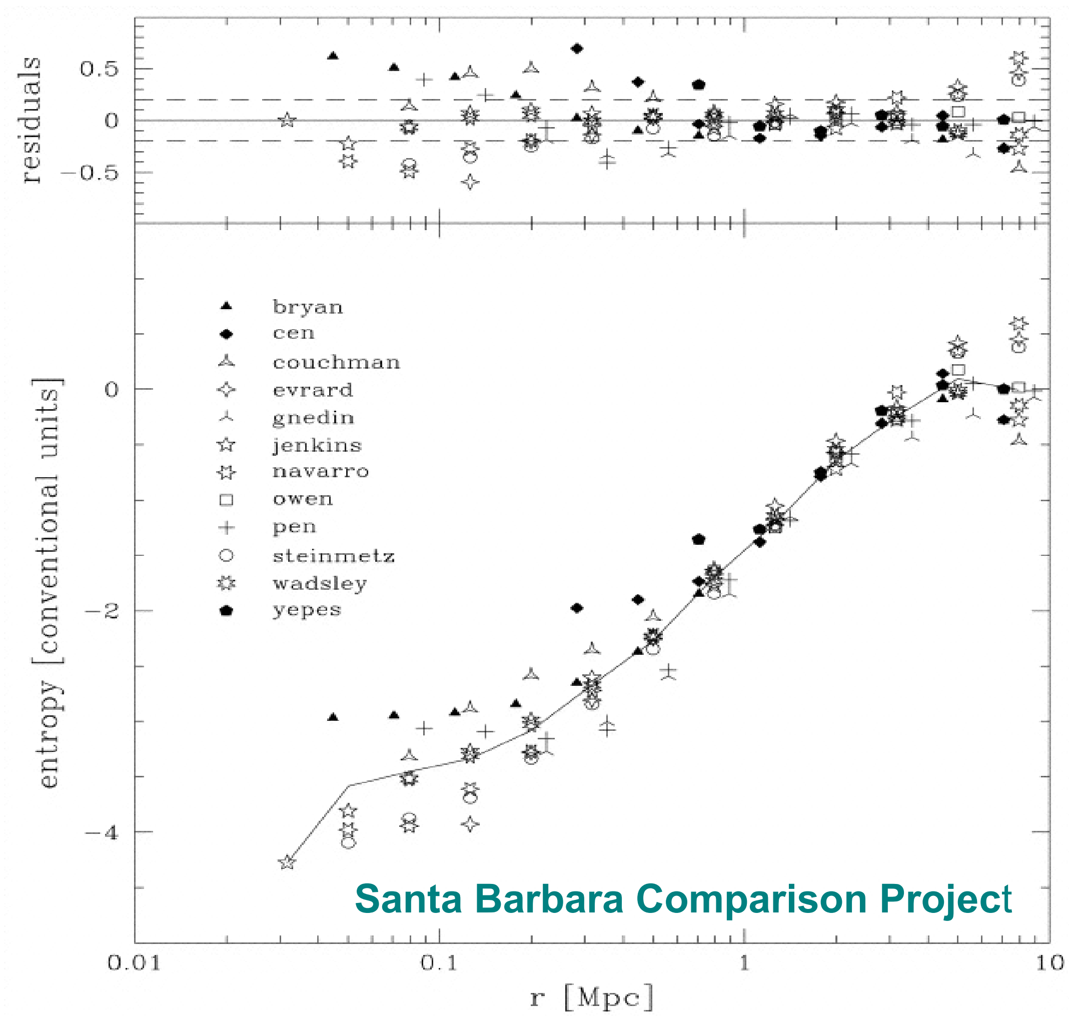
THE SANTA BARBARA CLUSTER COMPARISON PROJECT

Frenk, White & 23 co-authors (1999)



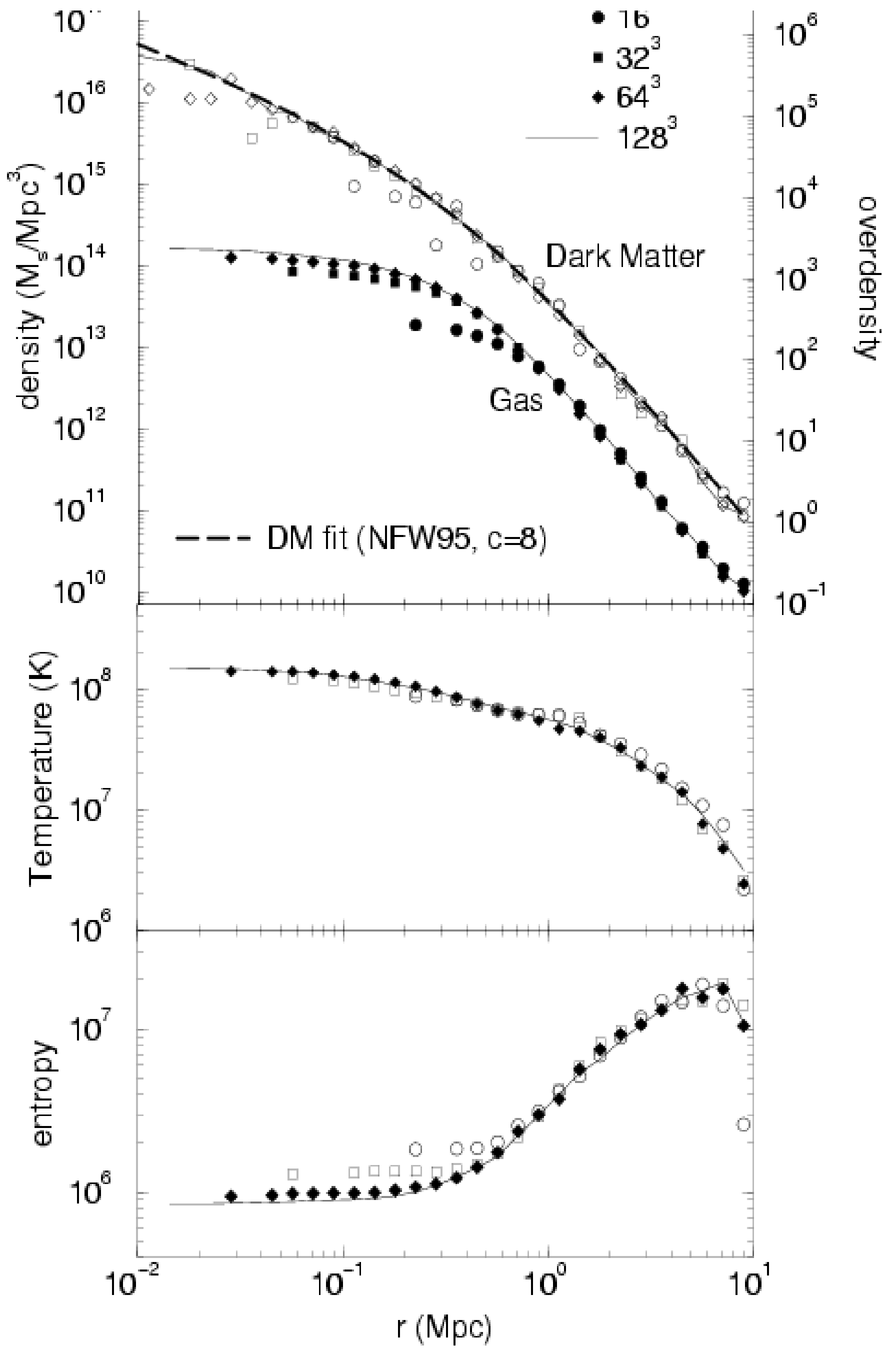
Mesh codes appear to produce higher entropy in the cores of clusters

RADIAL ENTROPY PROFILE



Santa Barbara Comparison Project

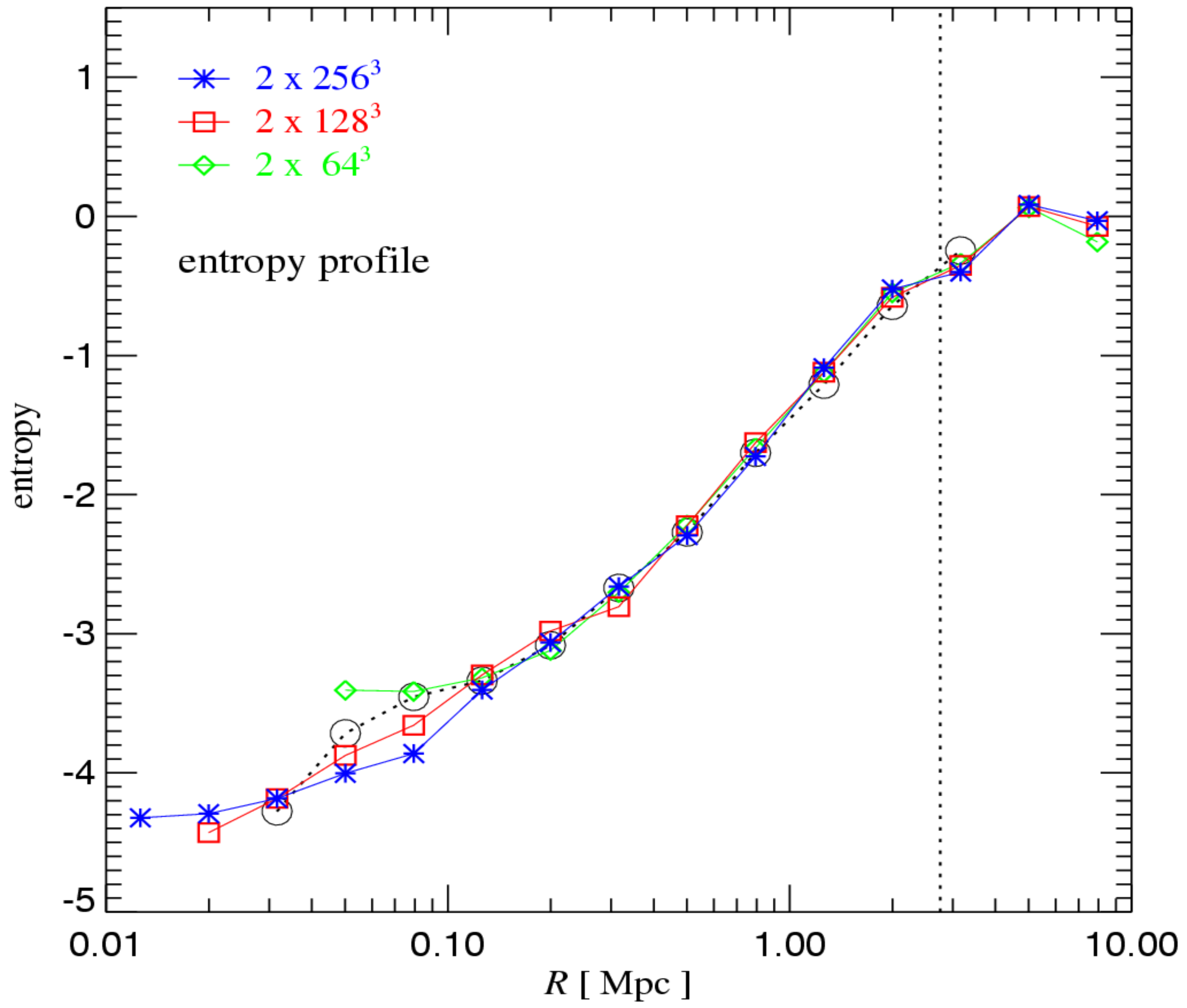
Bryan & Norman 1997



Ascasibar, Yepes, Müller & Gottlöber (2003):
Entropy formulation of SPH also gives somewhat higher core entropy

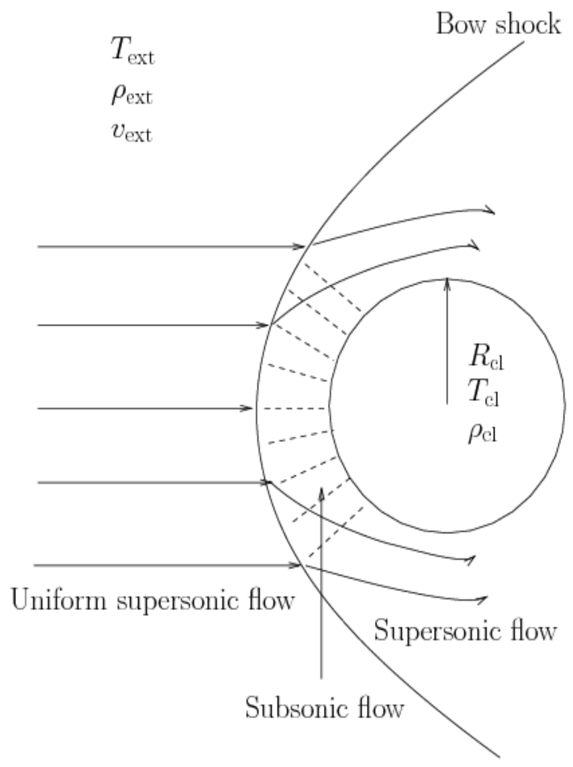
The entropy profile of the Santa Barbara cluster appears to converge well with SPH, yielding a lower level in the center than found with mesh codes

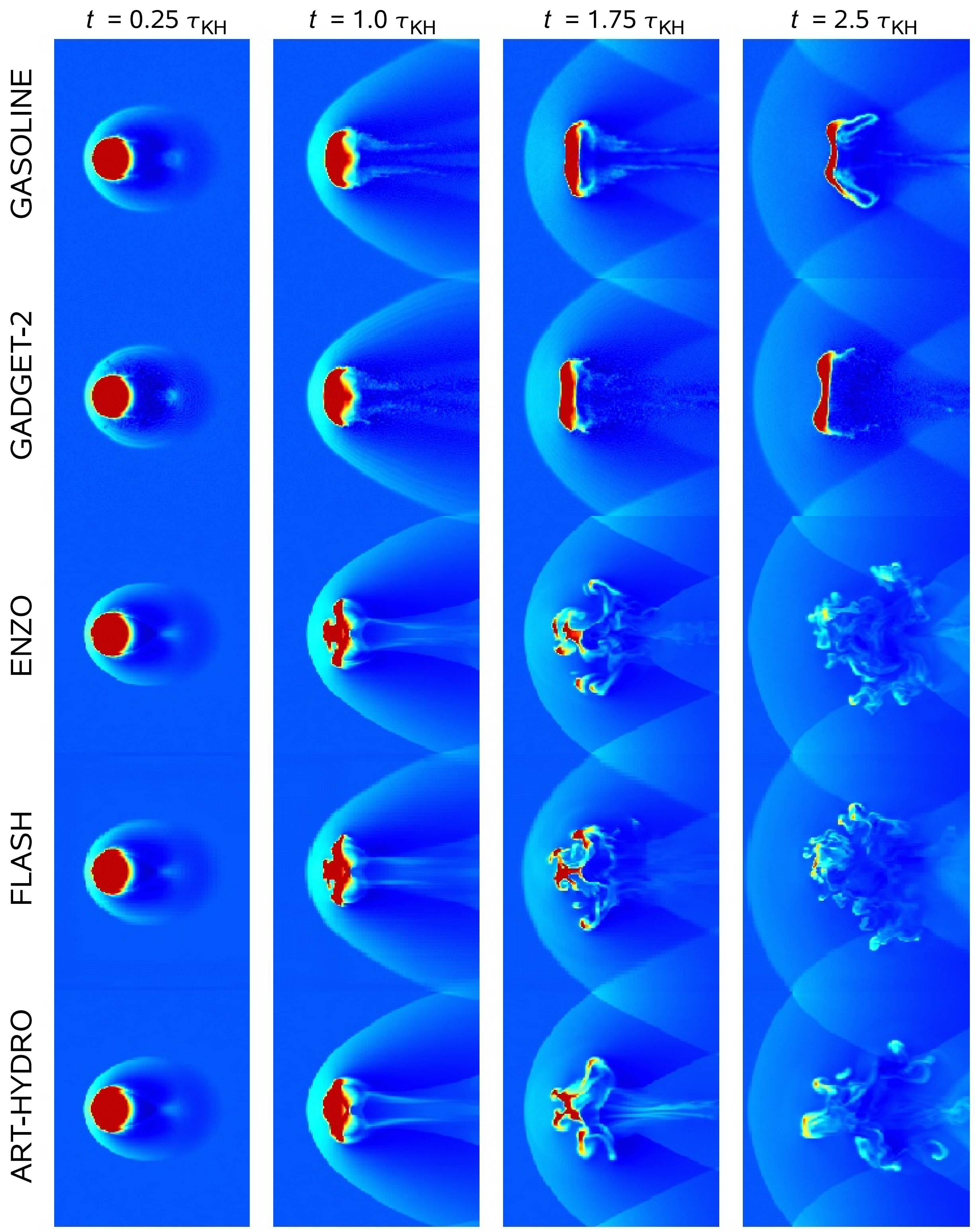
ENTROPY PROFILES OBTAINED WITH GADGET2 AT DIFFERENT RESOLUTION



A cloud moving through ambient gas shows markedly different long-term behavior in SPH and Eulerian mesh codes

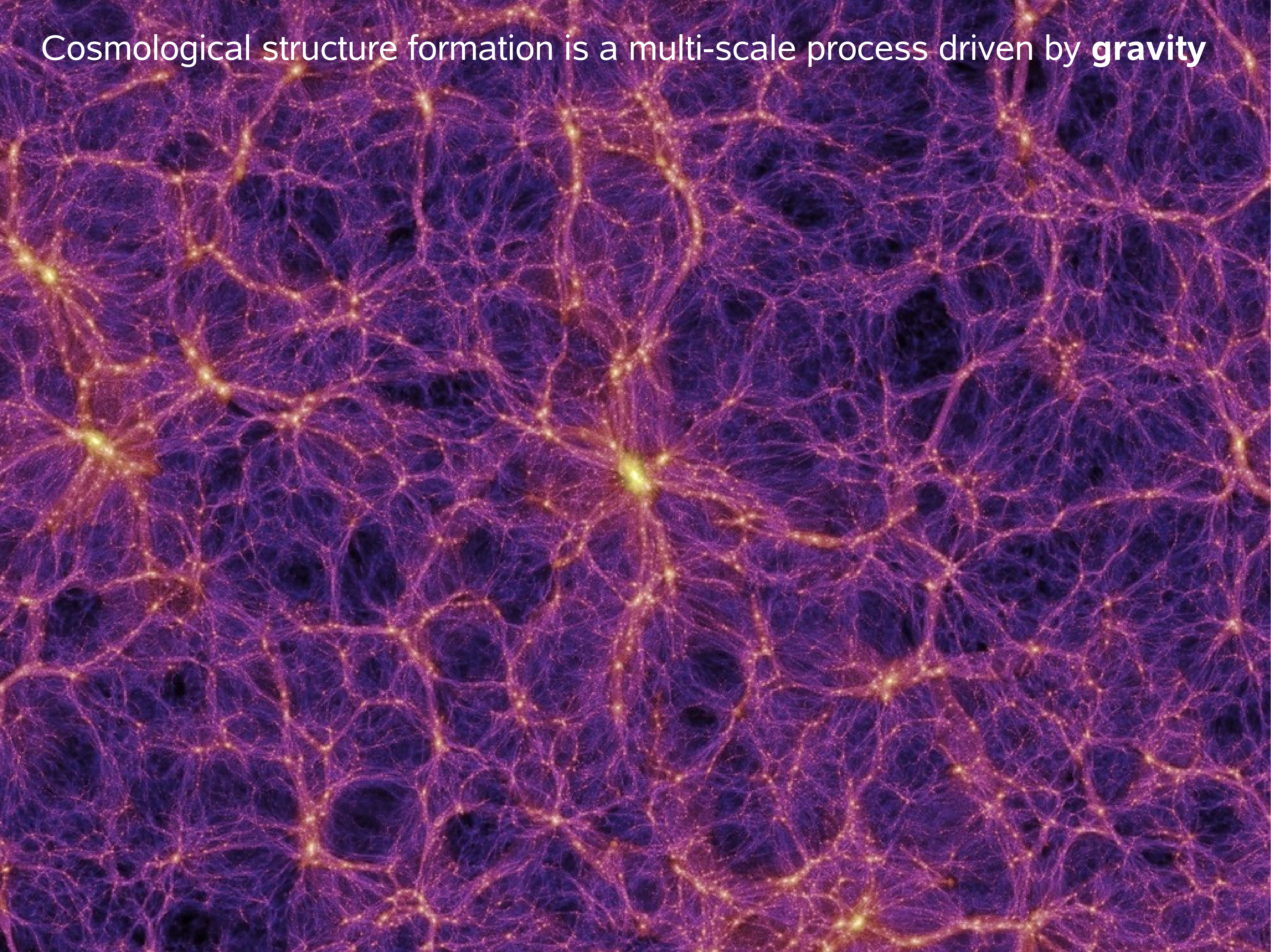
DISRUPTION OF A CLOUD BY KELVIN-HELMHOLTZ INSTABILITIES





Agertz et al. (2007)

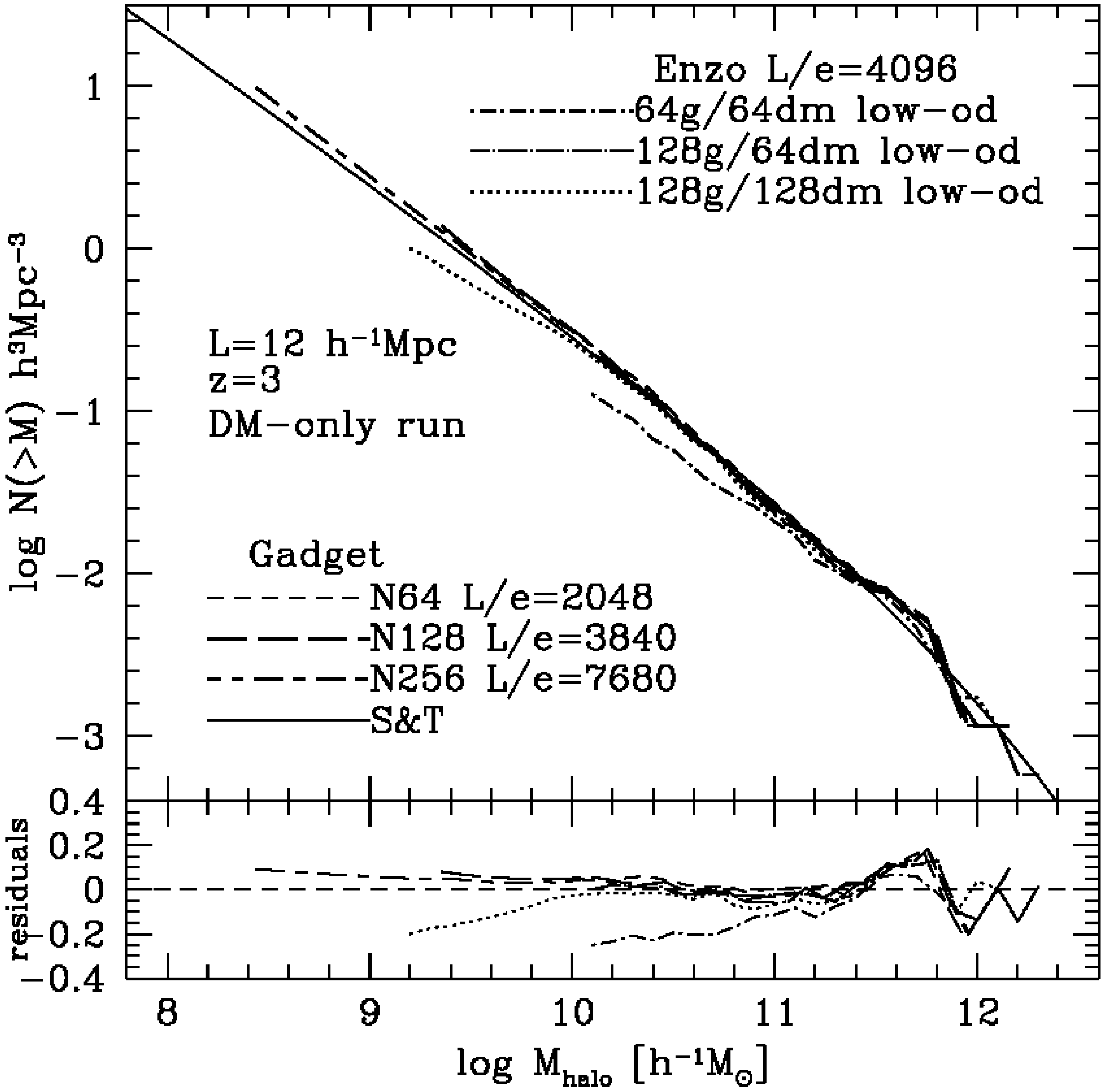
Cosmological structure formation is a multi-scale process driven by **gravity**



AMR codes have difficulty resolving the halo mass function

COMPARISON OF THE DARK MATTER HALO MASS FUNCTIONS IN ENZO AND GADGET

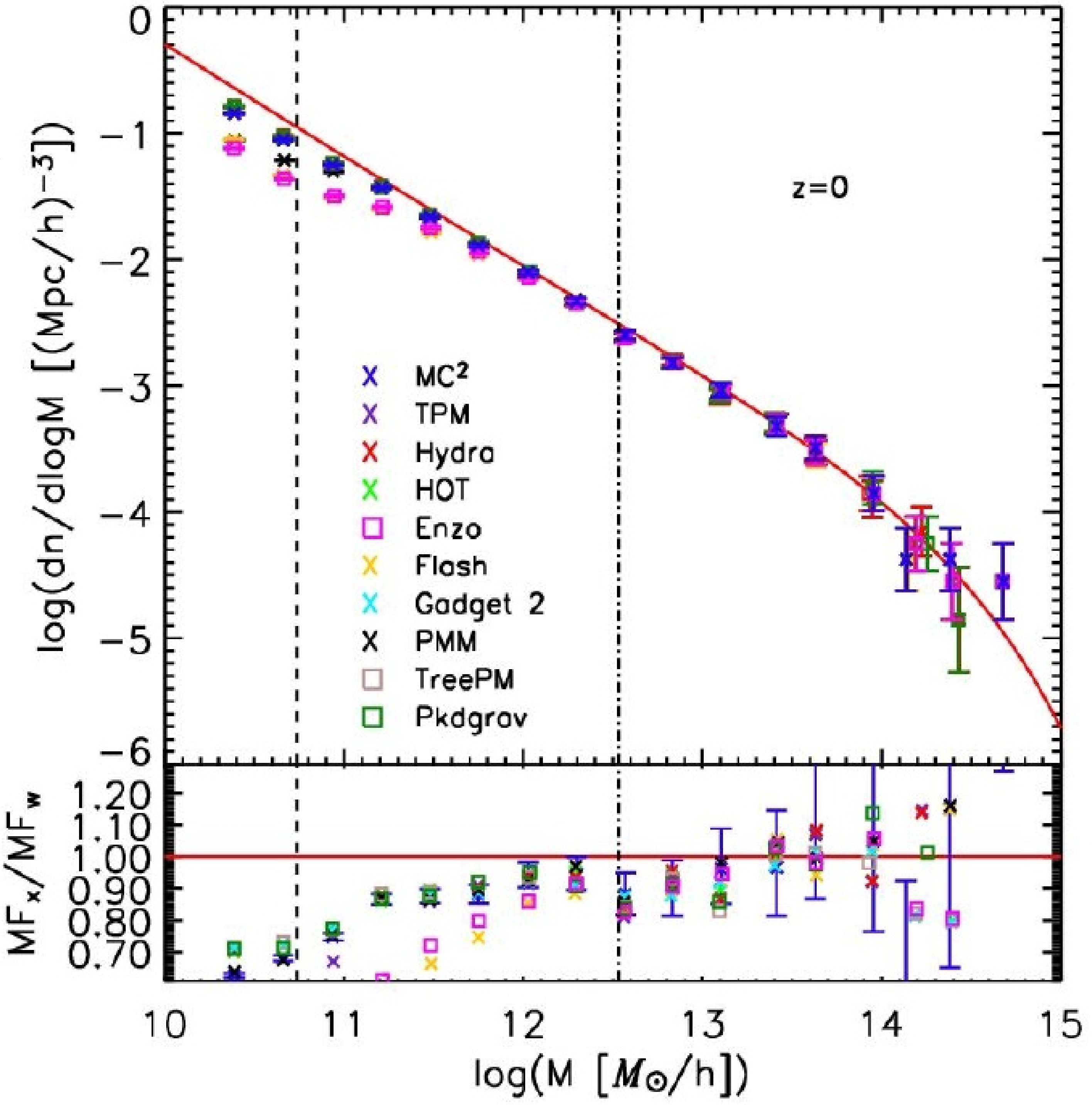
O'Shea et al. (2005)



Problems at the faint-end of the mass function appear to be generic for AMR codes and can only be avoided with very fine base meshes and aggressive refinement criteria

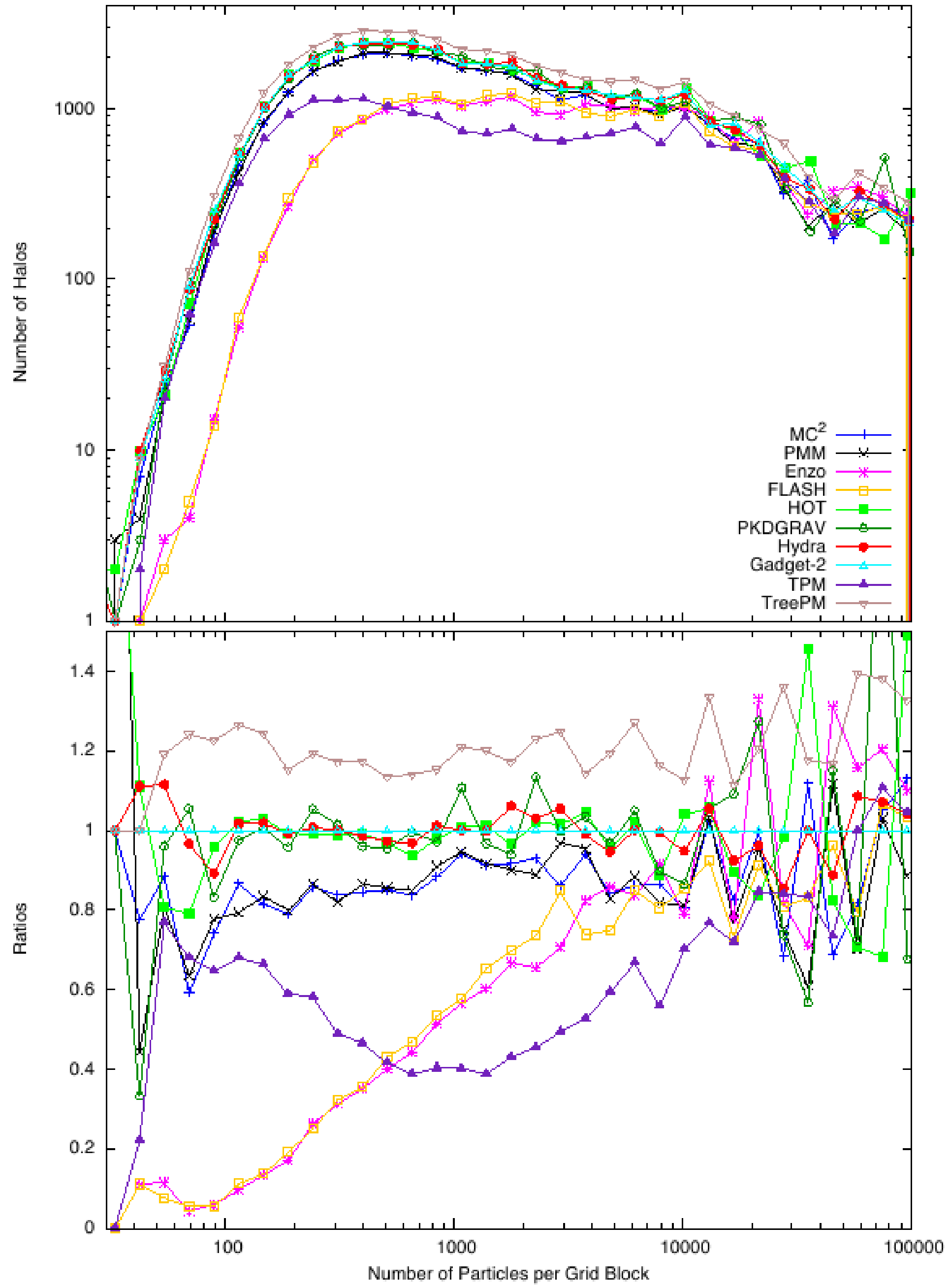
LOS ALAMOS CODE COMPARISON PROJECT

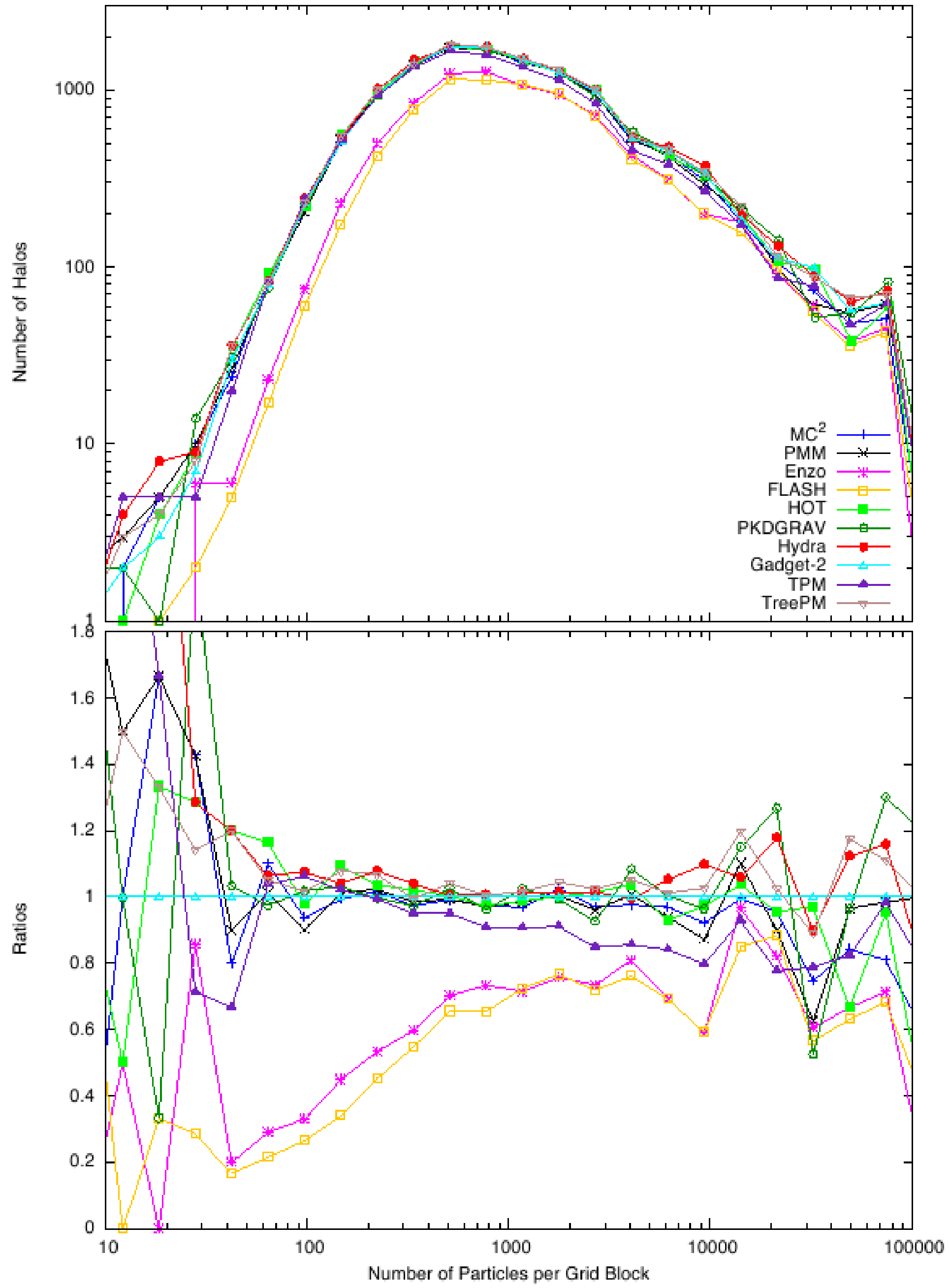
Heitmann et al. (2007)



The codes also show interesting differences in the number of halos as a function of density...

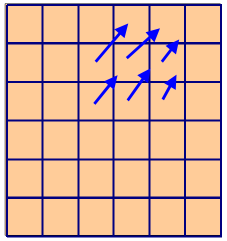
Heitmann et al. (2007)

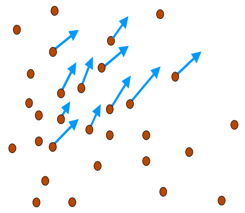




There are principal differences between SPH and Eulerian schemes

SOME FUNDAMENTAL DIFFERENCE BETWEENS SPH AND MESH-HYDRODYNAMICS



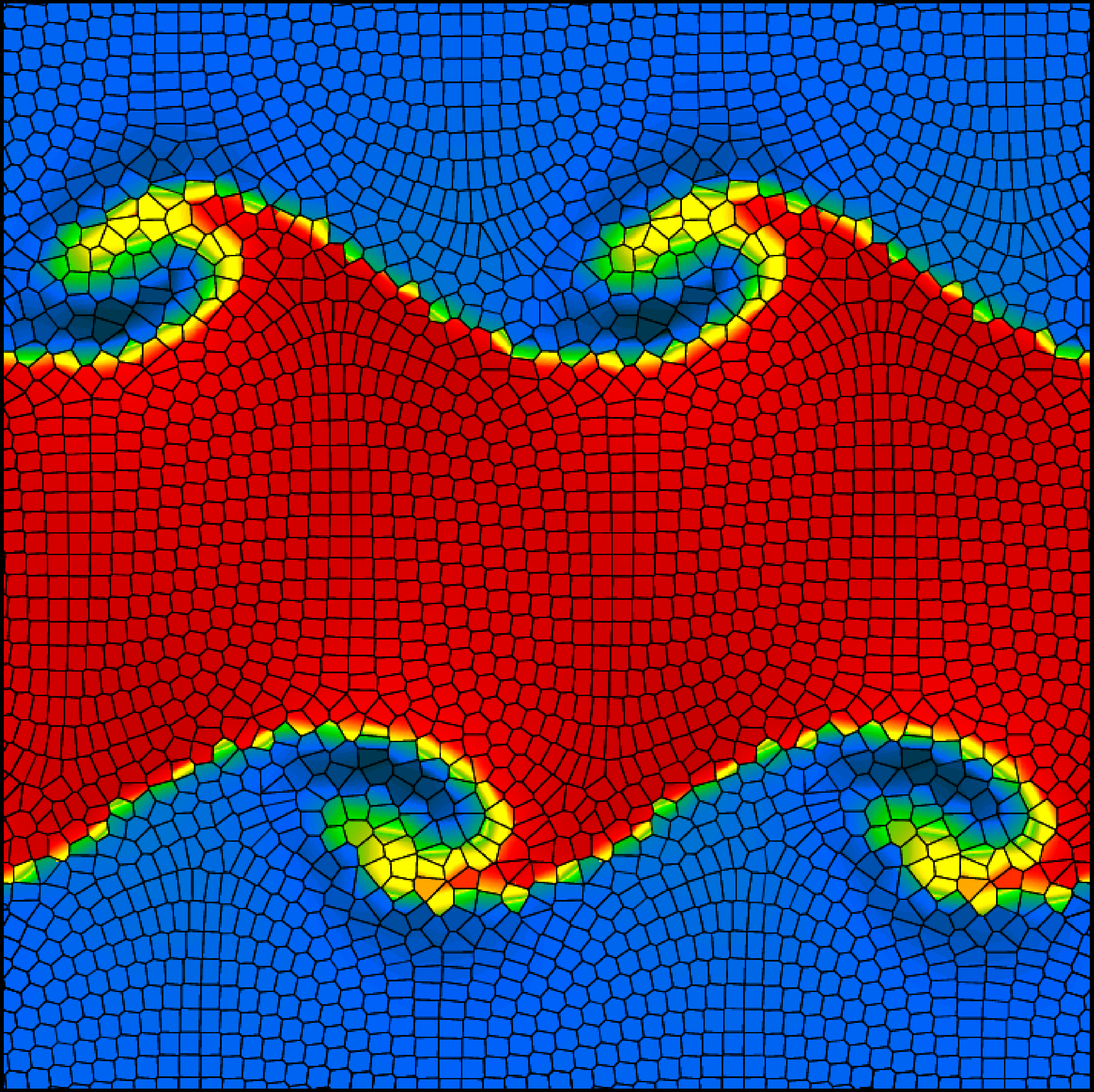


Eulerian	Lagrangian
sharp shocks, somewhat less sharp contact discontinuities (best schemes resolve fluid discontinuities it in one cell)	**shocks broadened over roughly 2-3 smoothing lengths** (post-shock properties are correct though)
mixing happens implicitly at the cell level (but advection adds numerical diffusivity and may provide a source of spurious entropy)	**mixing entirely suppressed at the particle-level** (no spurious entropy production, but fluid instabilities may be suppressed)
no need for artificial viscosity (in Godunov schemes)	**requires artificial viscosity**
Truncation error not Galilean invariant ***("high Mach number problem")***	**Galilean invariant**
self-gravity of the gas done on a mesh (but dark matter must still be represented by particles) **no explicit conservation of total energy when self-gravity is included**	**self-gravity of the gas naturally treated with the same accuracy as the dark matter, total energy conserved**

Moving-mesh hydrodynamics with AREPO

Volker Springel

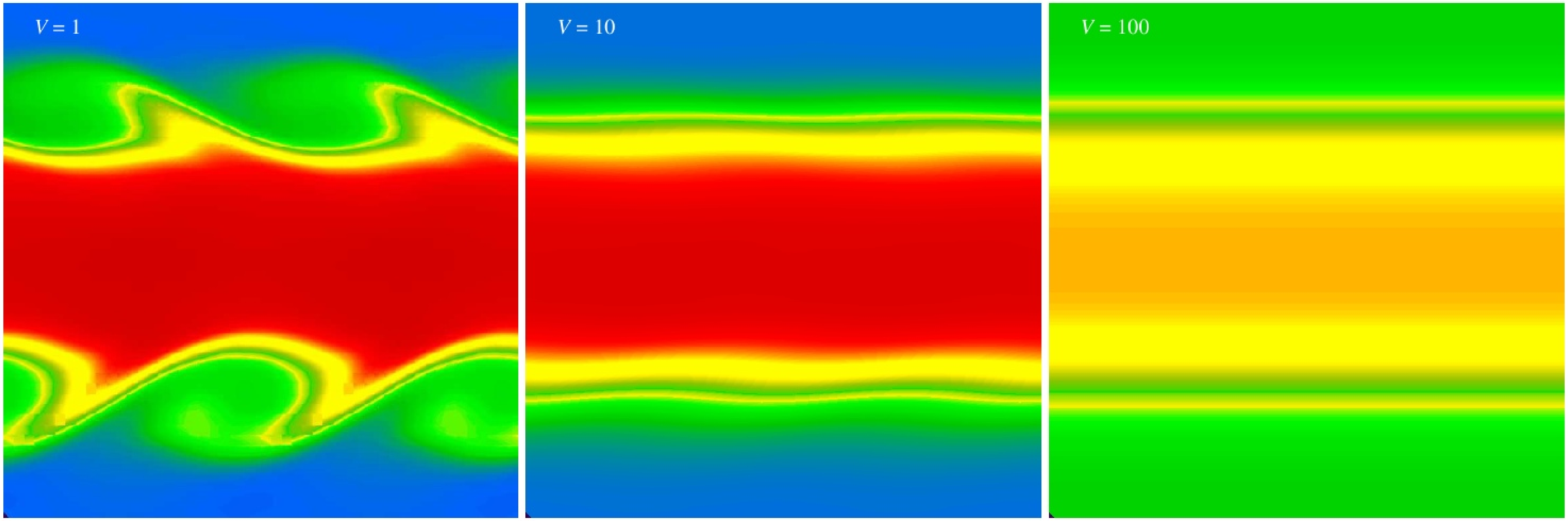
Max-Planck-Institute for Astrophysics



When the mesh is fixed, the results may change if a bulk velocity is imposed

KELVIN-HELMHOLTZH INSTABILITY AT 50 x 50 RESOLUTION WITH A FIXED MESH FOR DIFFERENT GALILEI BOOSTS

This was started from a sharp initial contact discontinuity.



Boost both in x- and y- directions

The truncation error in Eulerian codes is not Galilean invariant.

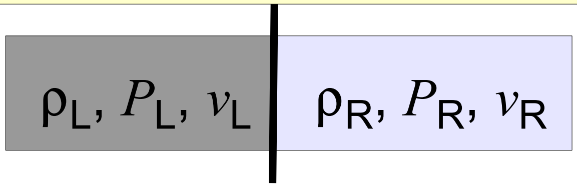
With enough cells, the truncation error can always be reduced, so that for properly resolved initial conditions, effective Galilean invariance is reached.

Nevertheless, this is an unwanted feature that is problematic for simulations of cosmological structure formation. Here the accuracy with which individual galaxies are modeled depends on their velocity magnitude.

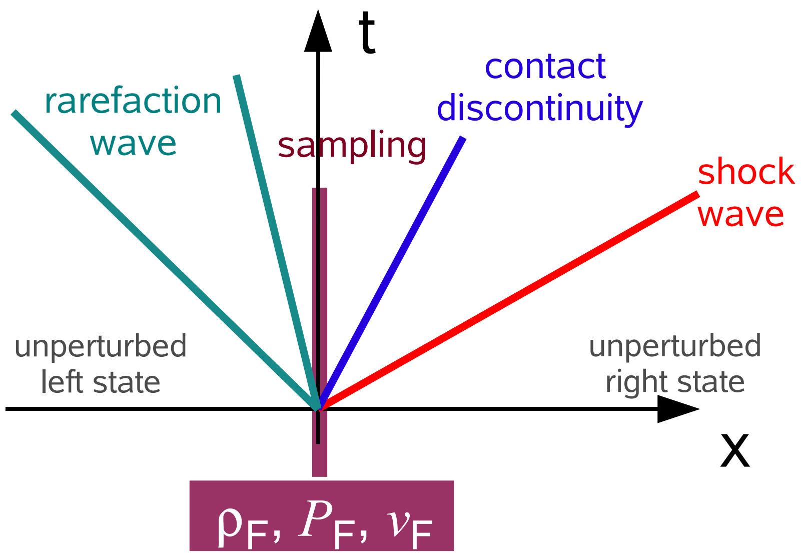
The Riemann problem as basis for high-accuracy Godunov schemes

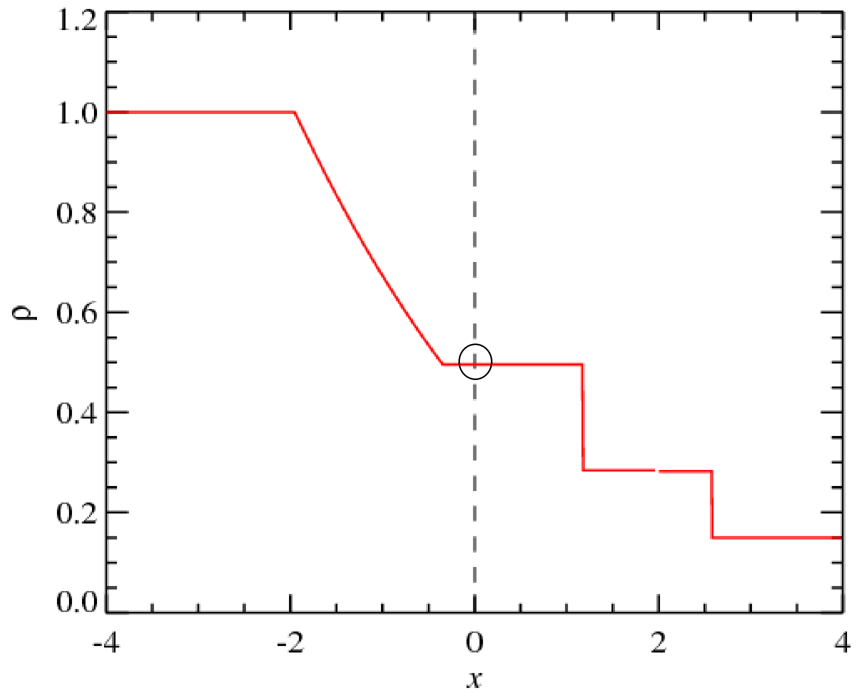
CALCULATION OF THE GODUNOV FLUX

Assume piece-wise constant left and right states for the fluid



Calculate the self-similar time evolution (Riemann problem)

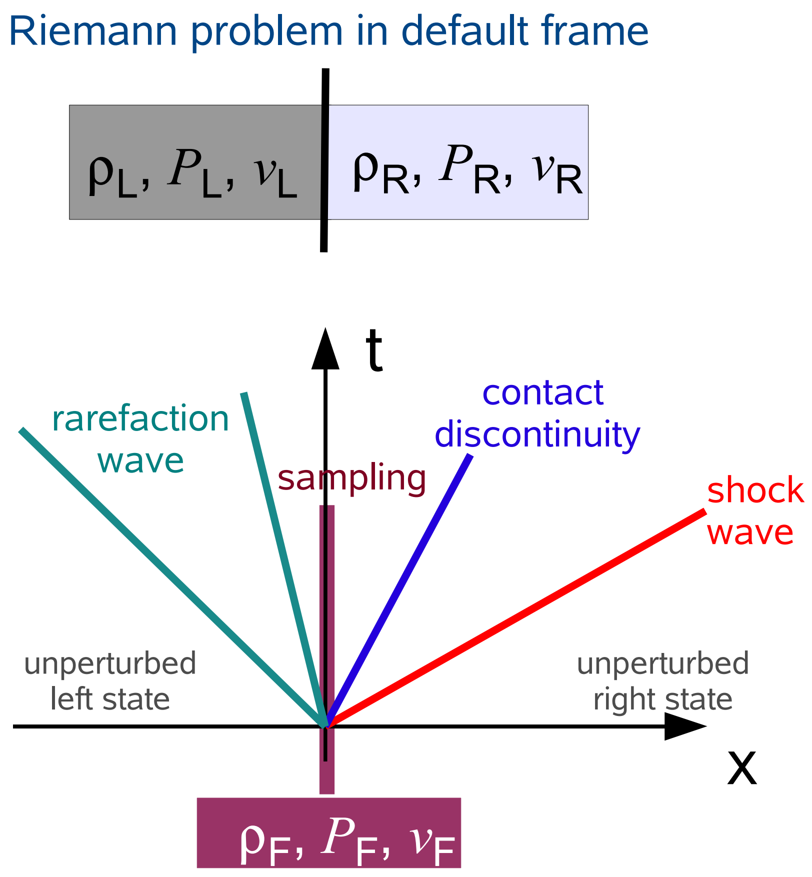


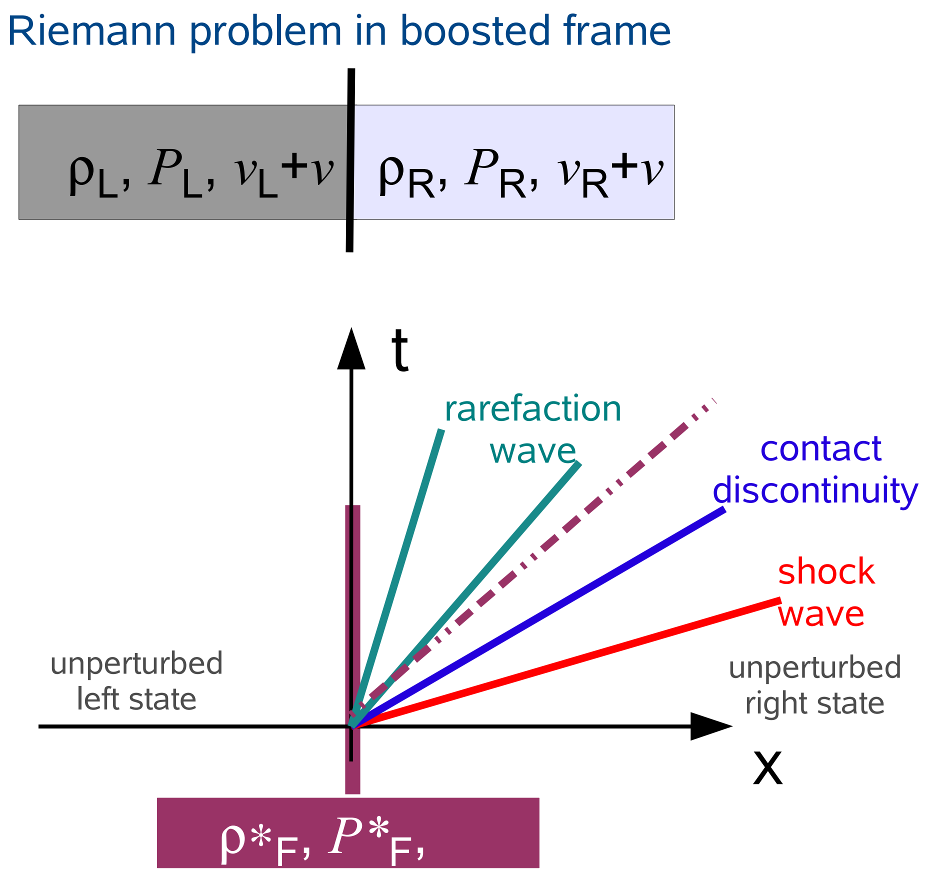


Sample the solution along x/ t=0, which yields the Godunov flux

The “upwind side” of the flow depends on the frame of reference

THE GODUNOV FLUX IN DIFFERENT REFERENCE FRAMES





expected mass flux in boosted frame:

$$\rho_F\,(v_F+v)$$

BUT, in general: $\rho_F\,(v_F+v) \neq \rho^*_F\, v^*_F$

$\longrightarrow$ **Numerical scheme not Galilean invariant**

A moving-mesh Lagrangian finite volume code can combine the advantages of SPH and Eulerian methods

KELVIN-HELMHOLTZ INSTABILITY WITH A MOVING MESH CODE

AREPO Code

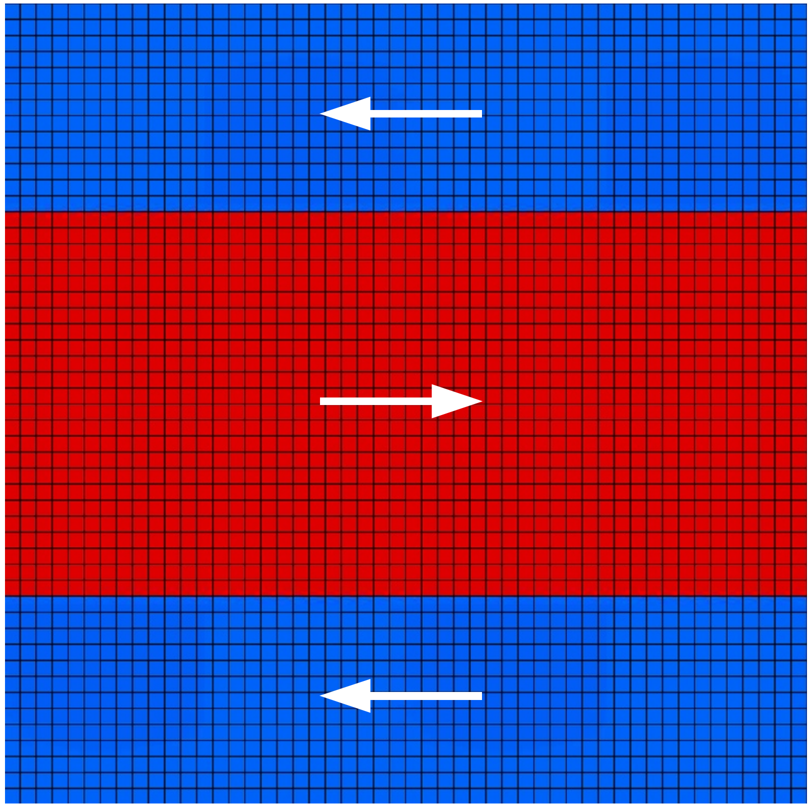
Springel (2008)

$\rho = 1$
$v_x = -0.5$
$P = 2.5$

$\rho = 2$
$v_x = 0.5$
$P = 2.5$

$\rho = 1$
$v_x = -0.5$
$P = 2.5$

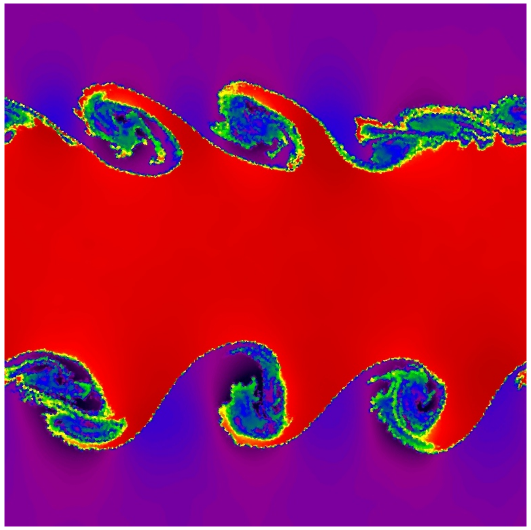
periodic boundaries



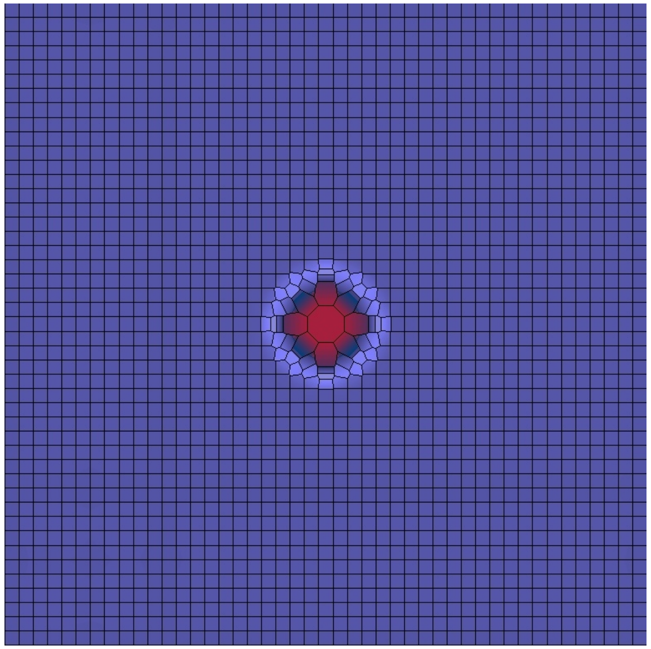
50x50 resolution

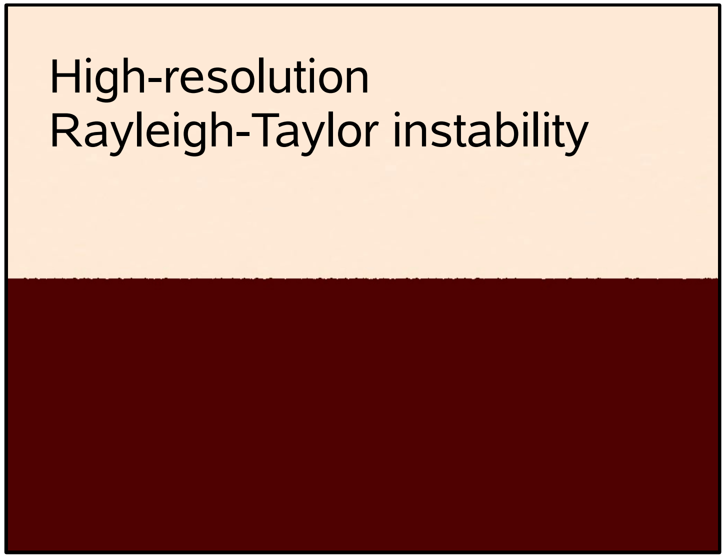
Different examples of test problems with the moving-mesh code

High-resolution
Kelvin-Helmholtz instability

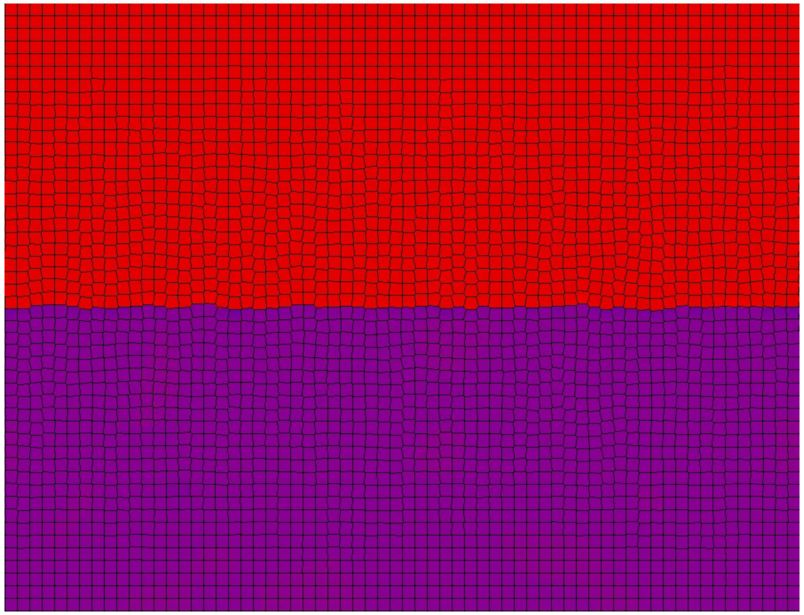


Sedov-Taylor Exposion





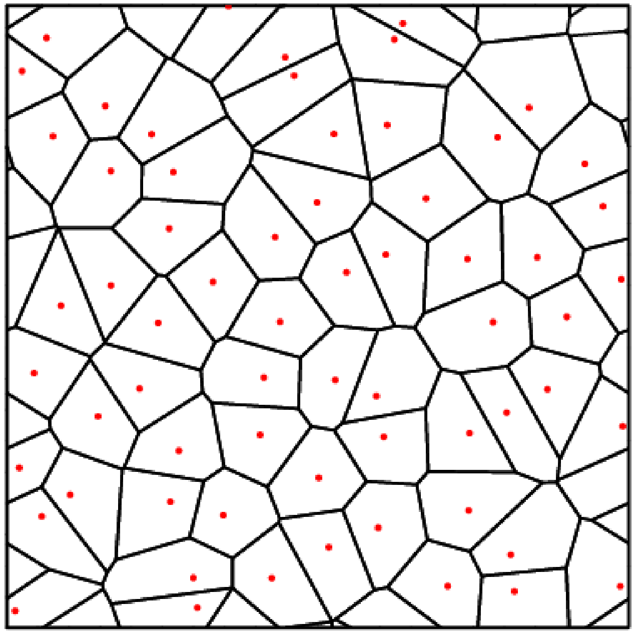
Rayleigh-Taylor (with visible mesh)



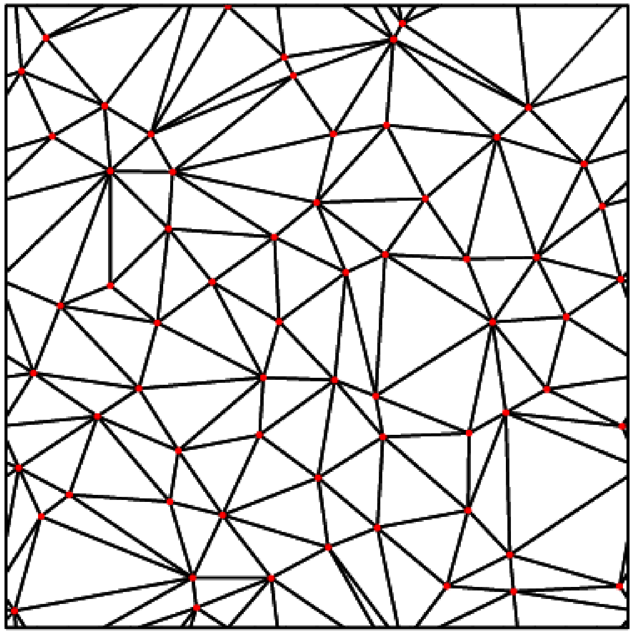
Voronoi and Delaunay tessellations provide unique partitions of space based on a given sample of mesh-generating points

BASIC PROPERTIES OF VORONOI AND DELAUNAY MESHES

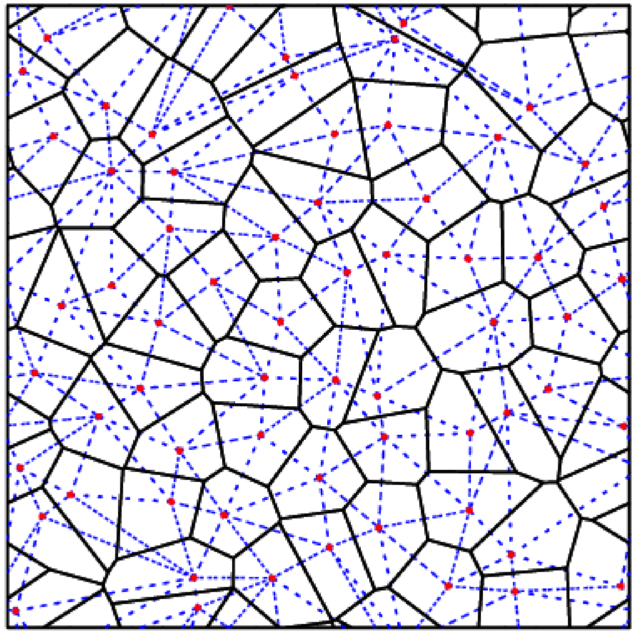
Voronoi mesh



Delaunay triangulation



both shown together



- Each Voronoi cell contains the **space closest** to its generating point
- The Delaunay triangulation contains only triangles with an **empty circumcircle**. The Delaunay tiangulation maximizes the minimum angle occurring among all triangles.
- The centres of the circumcircles of the Delaunay triangles are the vertices of the Voronoi mesh. In fact, the two tessellations are the topological **dual graph** to each other.

A finite volume discretization of the Euler equations on a moving mesh can be readily defined

THE EULER EQUATIONS AS HYPERBOLIC SYSTEM OF CONSERVATION LAWS

Euler equations

$$\boxed{\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = 0}$$

State vector

$$\boxed{\mathbf{U} = \begin{pmatrix} \rho \\ \rho \mathbf{v} \\ \rho e \end{pmatrix}}$$

Flux vector

$$\boxed{\mathbf{F}(\mathbf{U}) = \begin{pmatrix} \rho \mathbf{v} \\ \rho \mathbf{v}\mathbf{v}^T + P \\ (\rho e + P)\mathbf{v} \end{pmatrix}}$$

$$e = u + \mathbf{v}^2/2$$

Equation of state: $P = (\gamma - 1)\rho u$

Discretization in terms of a number of finite volume cells:

Cell averages

$$\boxed{\mathbf{Q}_i = \begin{pmatrix} M_i \\ \mathbf{p}_i \\ E_i \end{pmatrix} = \int_{V_i} \mathbf{U}\, \mathrm{d}V}$$

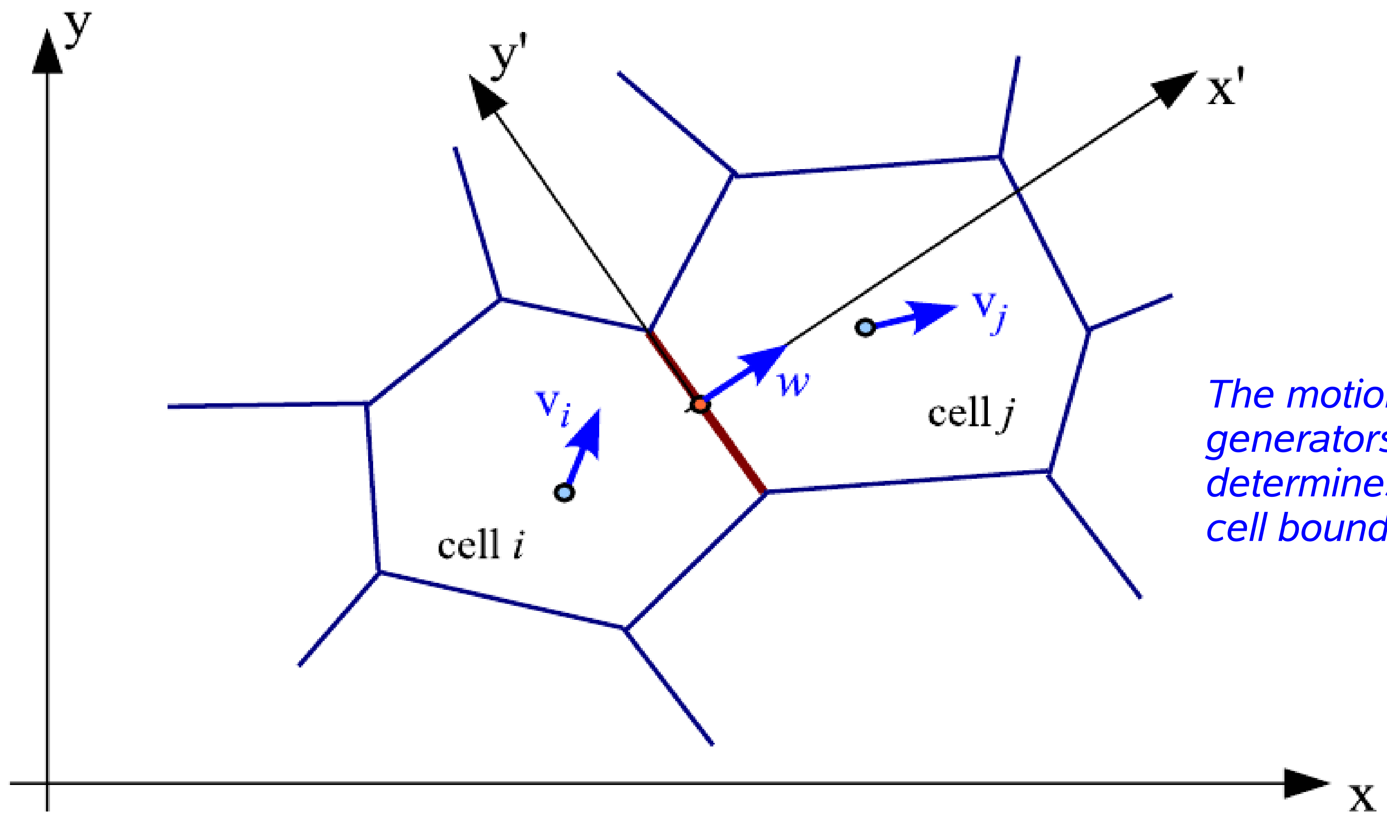
Evolution equation

$$\boxed{\frac{\mathrm{d}\mathbf{Q}_i}{\mathrm{d}t} = - \int_{\partial V_i} \left[\mathbf{F}(\mathbf{U}) - \mathbf{U}\mathbf{w}^T\right] \mathrm{d}\mathbf{n}}$$

But how to compute the fluxes through cell surfaces?

The fluxes are calculated with an exact Riemann solver in the frame of the moving cell boundary

SKETCH OF THE FLUX CALCULATION



The motion of the mesh generators uniquely determines the motion of all cell boundaries

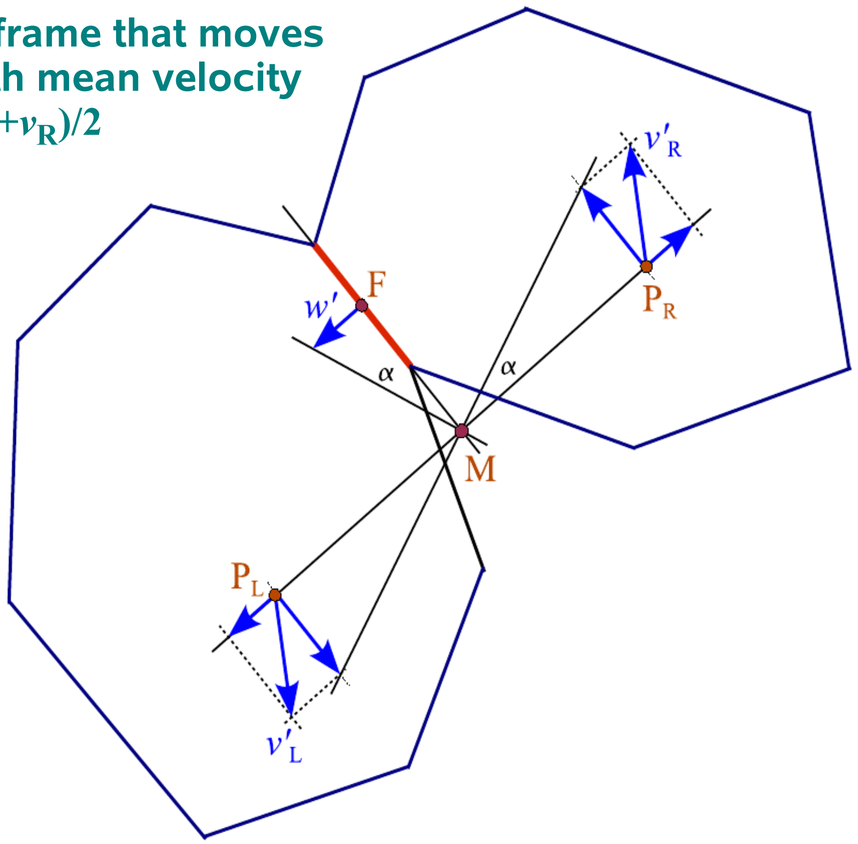
State left of cell face	State right of cell face
$\begin{pmatrix} \rho_L \\ \mathbf{v}_L \\ P_L \end{pmatrix}$	$\begin{pmatrix} \rho_R \\ \mathbf{v}_R \\ P_R \end{pmatrix}$

$$\xrightarrow{\text{Riemann solver (in frame of cell face)}} \begin{pmatrix} \rho \\ \mathbf{v} \\ P \end{pmatrix} \longrightarrow \mathbf{F}(\mathbf{U})$$

The velocities of the mesh-generating points uniquely determine the motion of all Voronoi faces

CHANGE OF VORONOI CELLS AS A FUNCTION OF TIME

in frame that moves with mean velocity $(v_L+v_R)/2$



rate of change of volume of a cell

$$\boxed{\frac{\mathrm{d}V_i}{\mathrm{d}t} = -\sum_{j\neq i} A_{ij}\left[\frac{\boldsymbol{c}_{ij}}{r_{ij}}(\boldsymbol{v}_j - \boldsymbol{v}_i) + \frac{\boldsymbol{r}_{ij}}{2r_{ij}}(\boldsymbol{v}_j + \boldsymbol{v}_i)\right]}$$

$$\boldsymbol{r}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$$

$$\boldsymbol{c}_{ij} = \boldsymbol{f}_{ij} - (\boldsymbol{x}_i + \boldsymbol{x}_j)/2$$

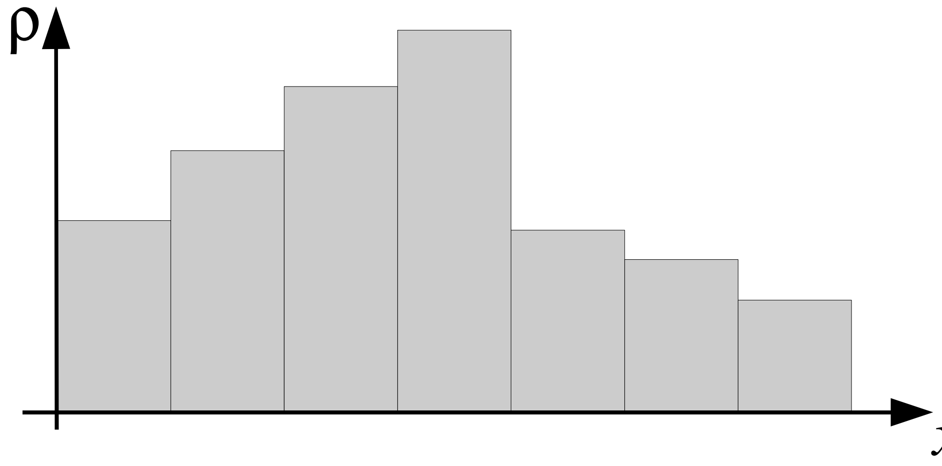
$$\boldsymbol{w}' = \frac{(\boldsymbol{v}_\mathrm{L} - \boldsymbol{v}_\mathrm{R})\cdot[\boldsymbol{f} - (\boldsymbol{x}_\mathrm{R} + \boldsymbol{x}_\mathrm{L})/2]}{|\boldsymbol{x}_\mathrm{R} - \boldsymbol{x}_\mathrm{L}|}\,\frac{(\boldsymbol{x}_\mathrm{R} - \boldsymbol{x}_\mathrm{L})}{|\boldsymbol{x}_\mathrm{R} - \boldsymbol{x}_\mathrm{L}|}$$

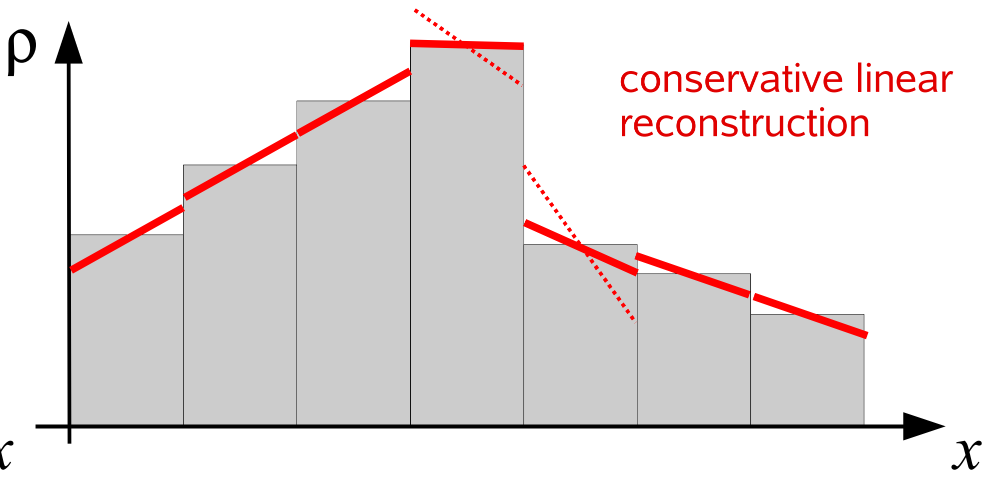
$$\boldsymbol{w} = \frac{\boldsymbol{v}_\mathrm{R} + \boldsymbol{v}_\mathrm{L}}{2} + \boldsymbol{w}'$$

(see also Serrano & Espanol 2001)

To achieve second-order accuracy, we use a **piece-wise linear** reconstruction

GRADIENT ESTIMATION AND LINEAR RECONSTRUCTION





Green-Gauss gradient estimation:

$$\int_{\partial V} \phi \, \mathrm{d}\boldsymbol{n} = \int_V \boldsymbol{\nabla} \phi \, \mathrm{d}V.$$

Leads to:

$$\langle \boldsymbol{\nabla} \phi \rangle_i = \frac{1}{V_i} \sum_{j \neq i} A_{ij} \left([\phi_j - \phi_i] \frac{\boldsymbol{c}_{ij}}{r_{ij}} - \frac{\phi_i + \phi_j}{2} \frac{\boldsymbol{r}_{ij}}{r_{ij}} \right)$$

Slope limiting procedure:

$$\langle \boldsymbol{\nabla} \phi \rangle_i' = \alpha_i \langle \boldsymbol{\nabla} \phi \rangle_i$$

$$\alpha_i = \min(1, \psi_{ij})$$

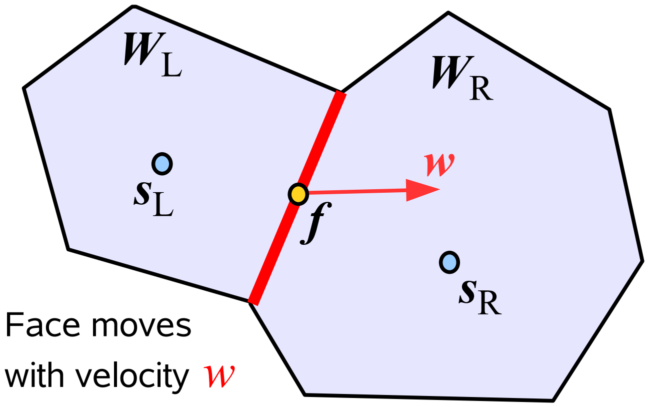
$$\psi_{ij} = \begin{cases} (\phi_i^{\max} - \phi_i)/\Delta\phi_{ij} & \text{for} \quad \Delta\phi_{ij} > 0 \\ (\phi_i^{\min} - \phi_i)/\Delta\phi_{ij} & \text{for} \quad \Delta\phi_{ij} < 0 \\ 1 & \text{for} \quad \Delta\phi_{ij} = 0 \end{cases}$$

$$\Delta\phi_{ij} = \langle \boldsymbol{\nabla} \phi \rangle_i \cdot (\boldsymbol{f}_{ij} - \boldsymbol{s}_i)$$

$$\phi_i^{\max} = \max(\phi_j) \qquad \phi_i^{\min} = \max(\phi_j)$$

Our second-order time integration scheme uses a half-step prediction in primitive variable formulation

A MUSCL-LIKE SCHEME



And finally…

Update the conserved variables of each cell:

$$Q_i^{(n+1)} = Q_i^{(n)} - \Delta t \sum_j A_{ij} \hat{F}_{ij}^{(n+1/2)}$$

This scheme is **Galilean invariant** if w is tied to the fluid velocity.

Transform left and right fluid states into rest frame of face

$$W'_{L,R} = W_{L,R} - \begin{pmatrix} 0 \\ w \\ 0 \end{pmatrix}$$

Linearly predict the states to the midpoint of the face, and evolve them forward in time by half a timestep:

$$W''_{L,R} = W'_{L,R} + \left.\frac{\partial W}{\partial r}\right|_{L,R} (f - s_{L,R}) + \left.\frac{\partial W}{\partial t}\right|_{L,R} \frac{\Delta t}{2}$$

The prediction in time can be done with the Euler equations:

$$\frac{\partial W}{\partial t} + A(W)\frac{\partial W}{\partial r} = 0 \qquad A(W) = \begin{pmatrix} v & \rho & 0 \\ 0 & v & 1/\rho \\ 0 & \gamma P & v \end{pmatrix}$$

Rotate the states such that one coordinate is normal to the face

$$W'''_{L,R} = \Lambda\, W''_{L,R} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \Lambda_{3D} & 0 \\ 0 & 0 & 1 \end{pmatrix} W''_{L,R}$$

Solve the Riemann problem

$$W = R_{\text{iemann}}(W'''_L, W'''_R)$$

Transform the solution back to the calculational frame

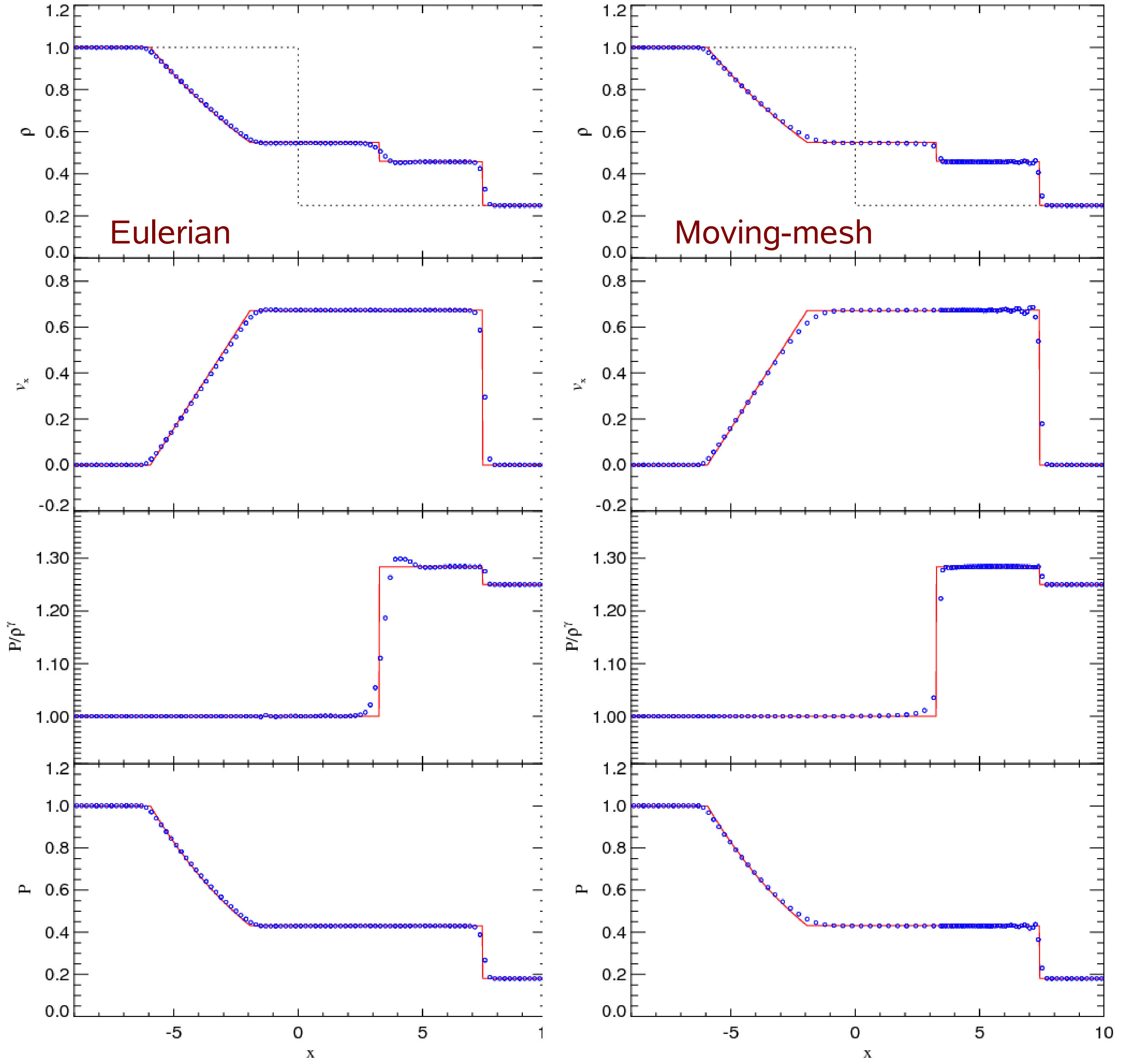
$$W_{\text{lab}} = \begin{pmatrix} \rho \\ v_{\text{lab}} \\ P \end{pmatrix} = \Lambda^{-1} W + \begin{pmatrix} 0 \\ w \\ 0 \end{pmatrix}$$

Calculate the net flux in the calculational frame

$$\hat{F} = F(U) - U w^T = \begin{pmatrix} \rho(v_{\text{lab}} - w) \\ \rho v_{\text{lab}}(v_{\text{lab}} - w)^T + P \\ \rho e_{\text{lab}}(v_{\text{lab}} - w) + P v_{\text{lab}} \end{pmatrix}$$

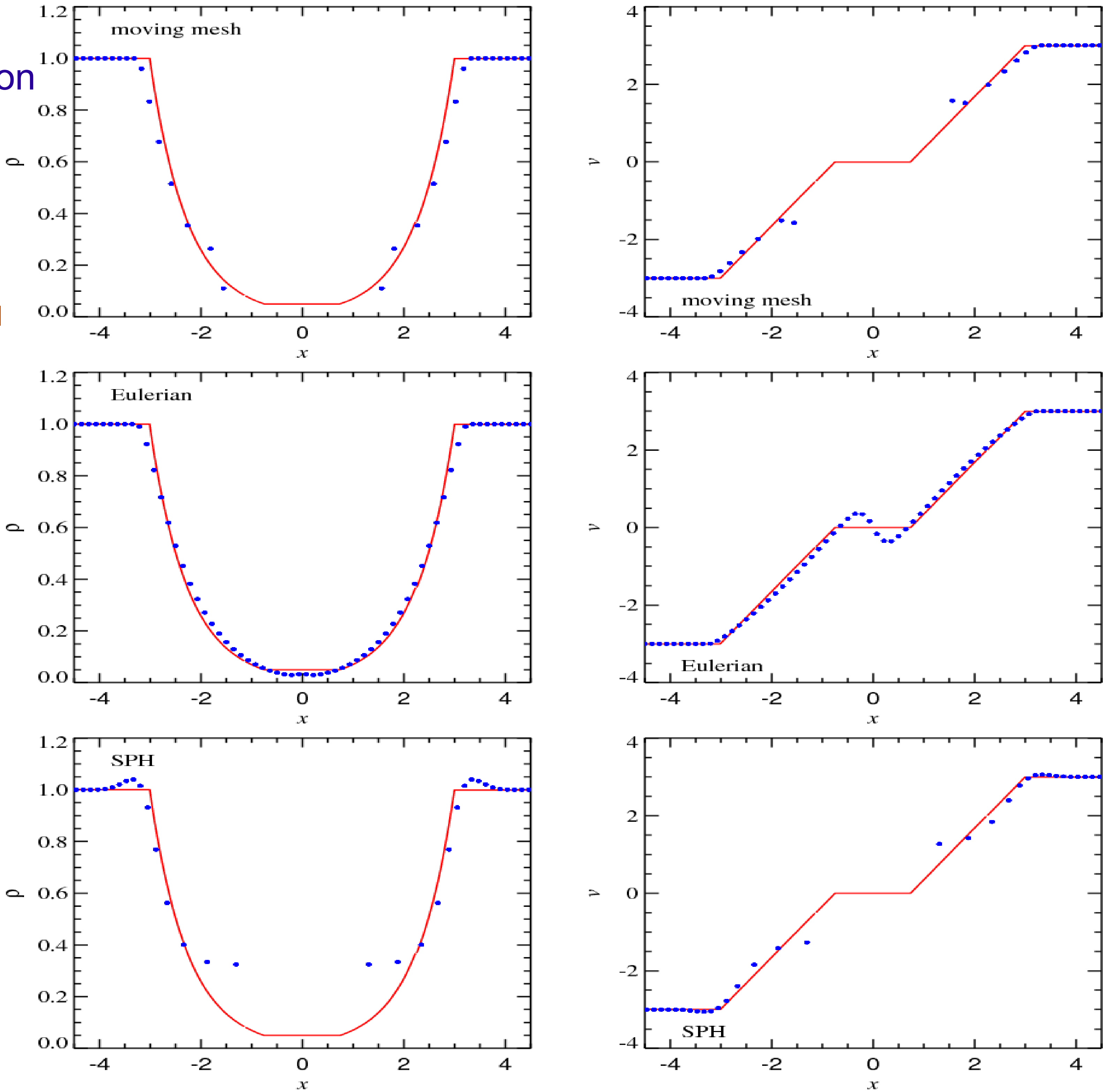
Eulerian and moving-mesh code are similar in shock-capturing, but differ in the treatment of contact discontinuities

COMPARISON OF SOD SOCK TUBES



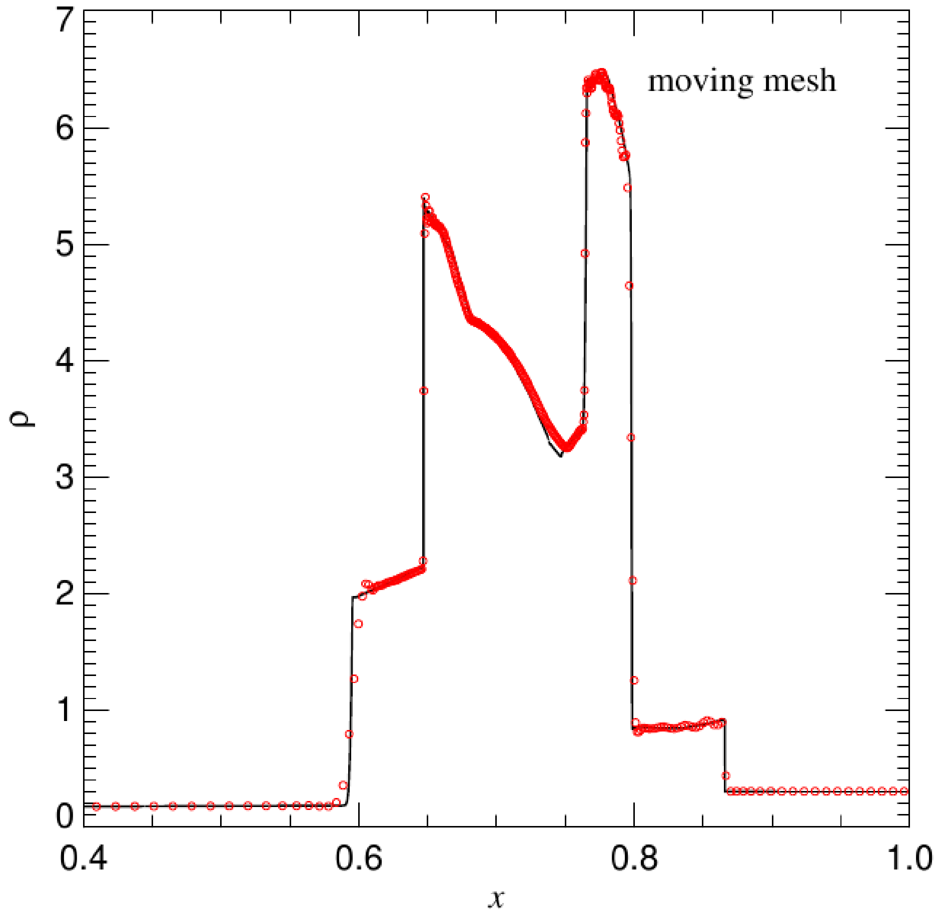
Very strong isothermal rarefaction waves are better represented by the Eulerian approach

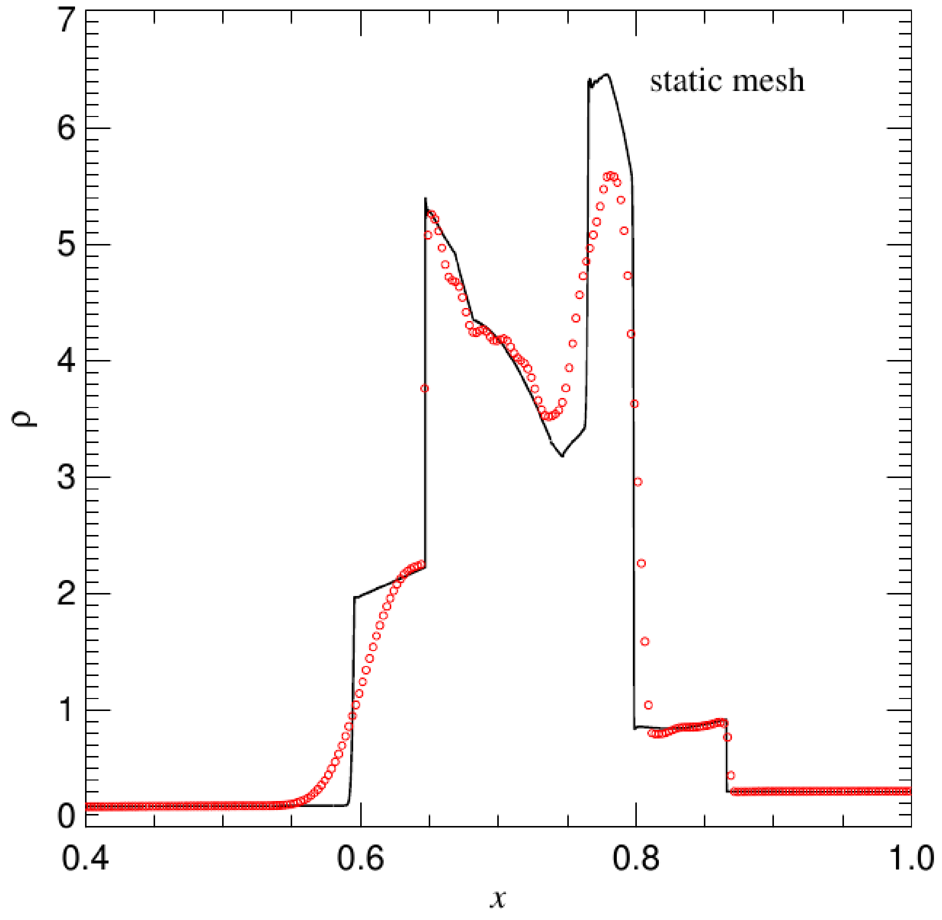
COMPARISON OF A STRONG ISOTHERMAL DOUBLE RAREFACTION



The moving-mesh code deals well will problems that involve complicated shock interactions

WOODWARD & COLELLA'S INTERACTING DOUBLE BLAST PROBLEM





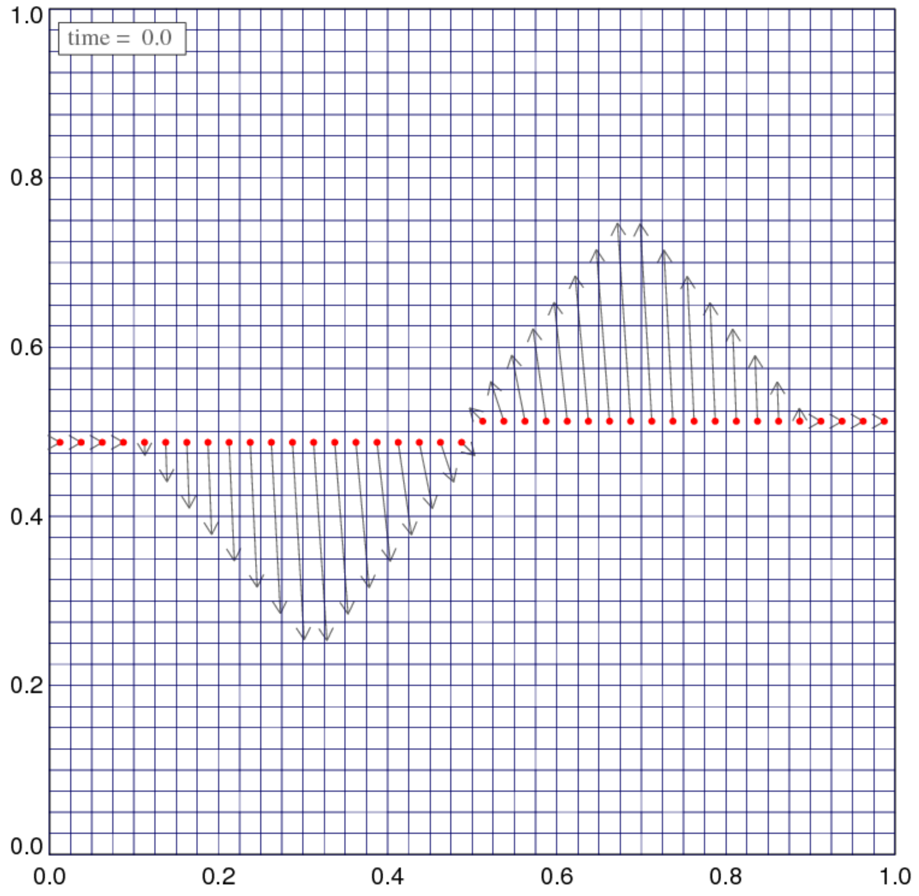
The Gresho vortex test in two dimensions

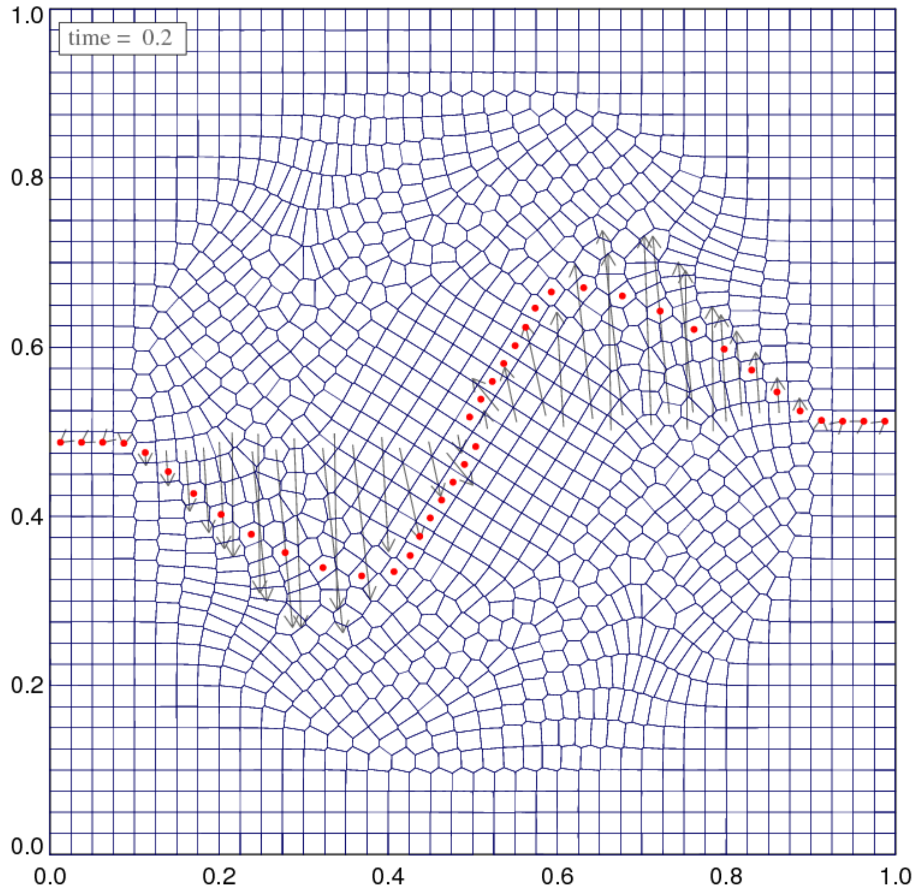
EVOLUTION OF A STATIONARY VORTEX FLOW

Initial conditions:

$$v_\phi(r) = \begin{cases} 5r & \text{for} \quad 0 \le r < 0.2 \\ 2 - 5r & \text{for} \quad 0.2 \le r < 0.4 \\ 0 & \text{for} \quad r \ge 0.4 \end{cases}$$

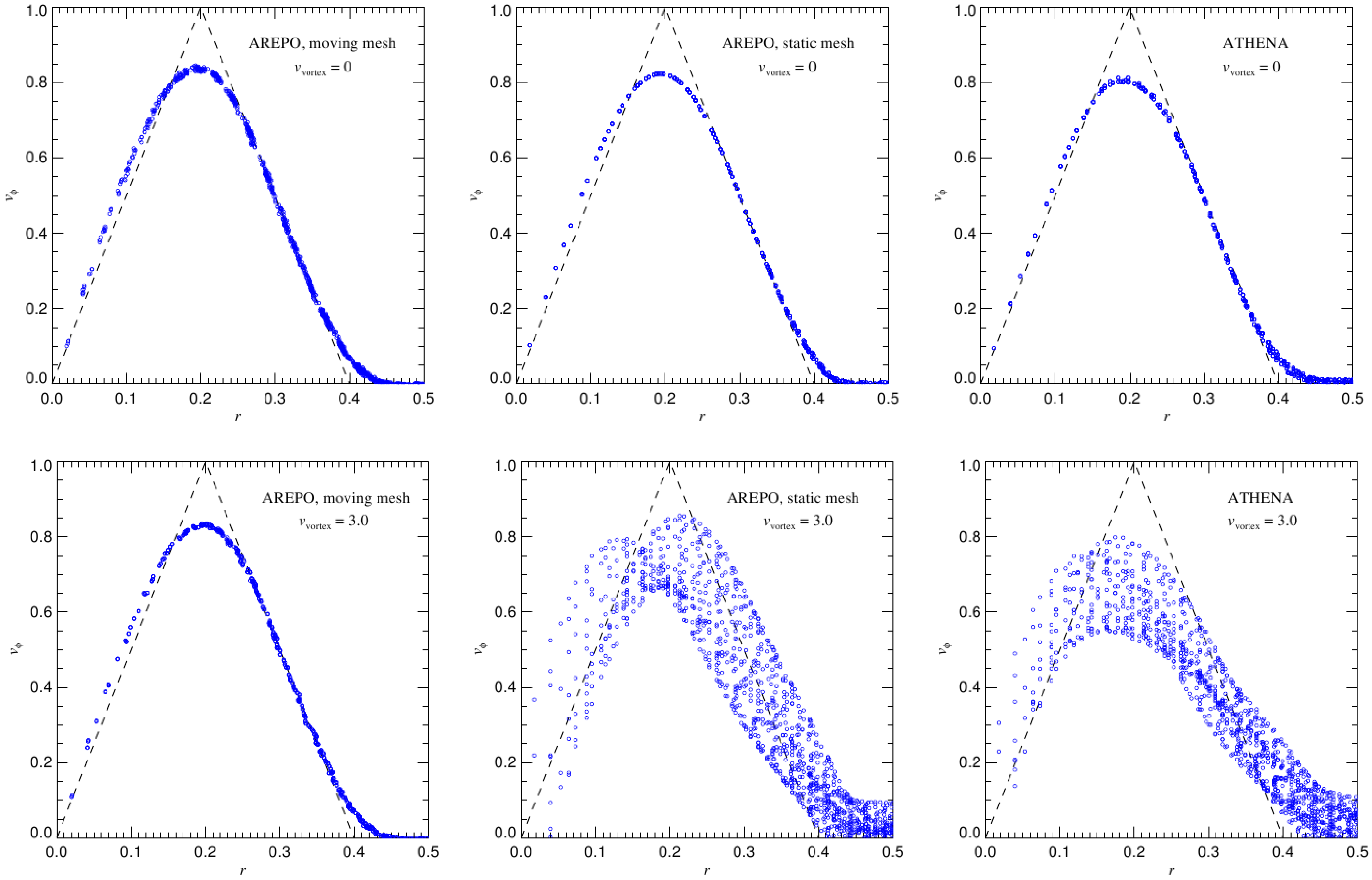
$$P(r) = \begin{cases} 5 + 25/2r^2 & \text{for} \quad 0 \le r < 0.2 \\ 9 + 25/2r^2 - & \\ \qquad 20r + 4\ln(r/0.2) & \text{for} \quad 0.2 \le r < 0.4 \\ 3 + 4\ln 2 & \text{for} \quad r \ge 0.4 \end{cases}$$





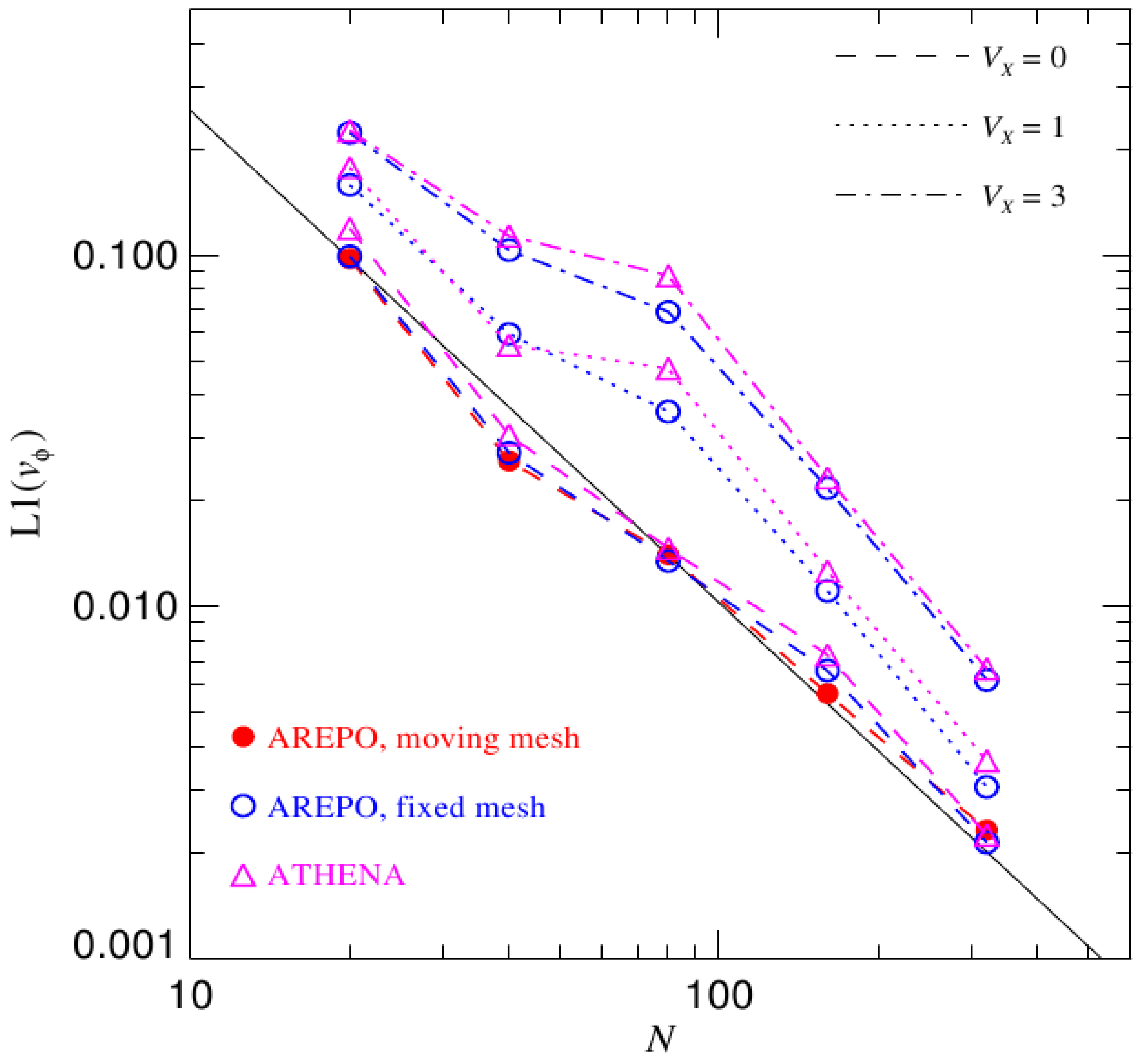
The Gresho vortex test in two dimensions

EVOLVED AZIMUTHAL VELOCITY PROFILE FOR DIFFERENT CODES AND BOOSTS



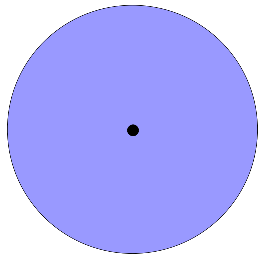
The Gresho vortex test in two dimensions

CONVERGENCE RATE AGAINST ANALYTIC SOLUTION



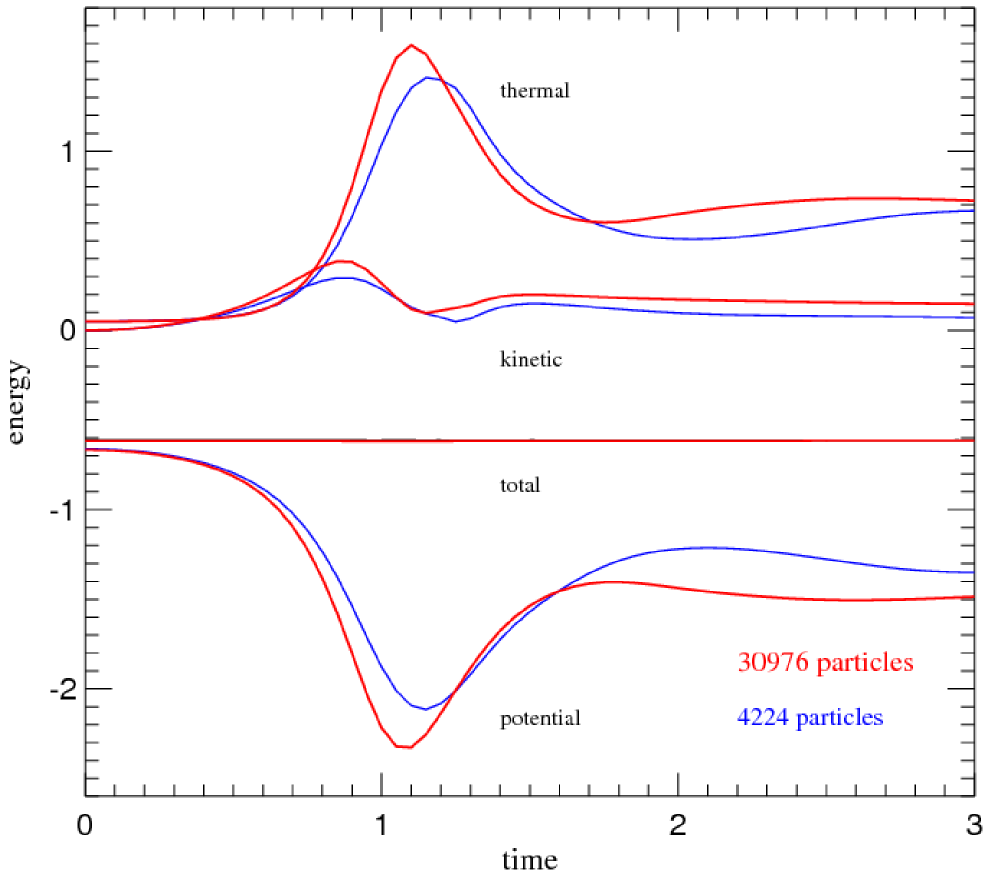
The “Evrard-Collapse” is a popular standard test problem for SPH codes

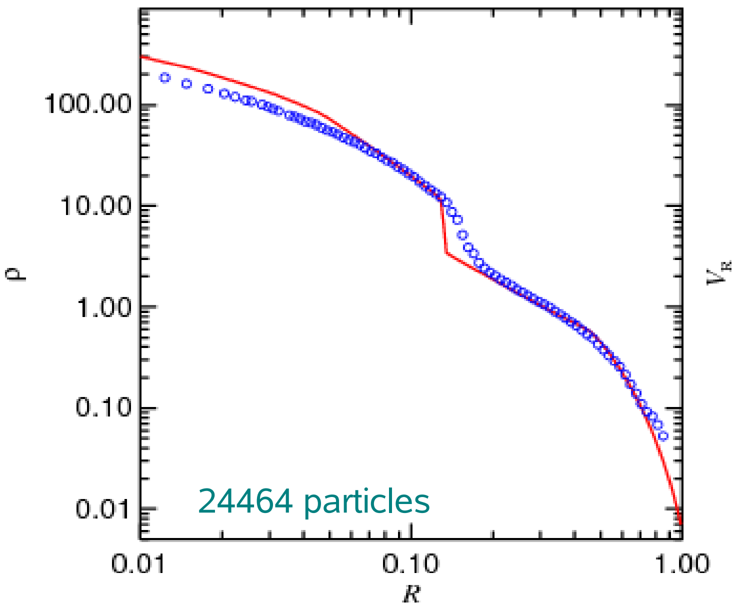
PROBLEM DEFINITION AND BASIC RESULTS WITH SPH

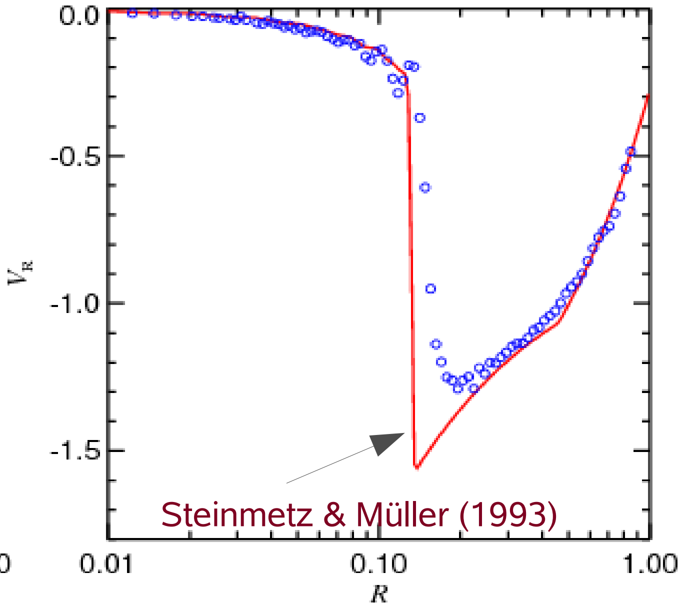


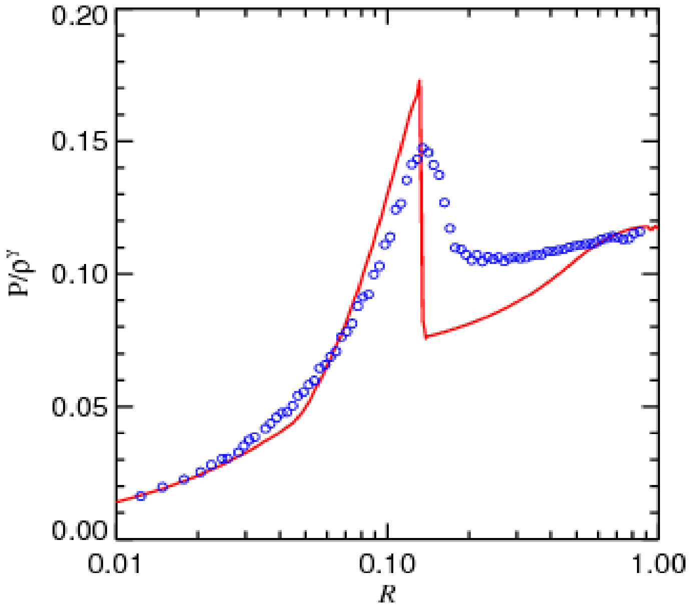
$$M = 1$$
$$R = 1$$
$$u = 0.05$$
$$G = 1$$

$$\rho(r) = \begin{cases} M/(2\pi R^2 r) & \text{for } r \leq R \\ 0 & \text{for } r > R \end{cases}$$





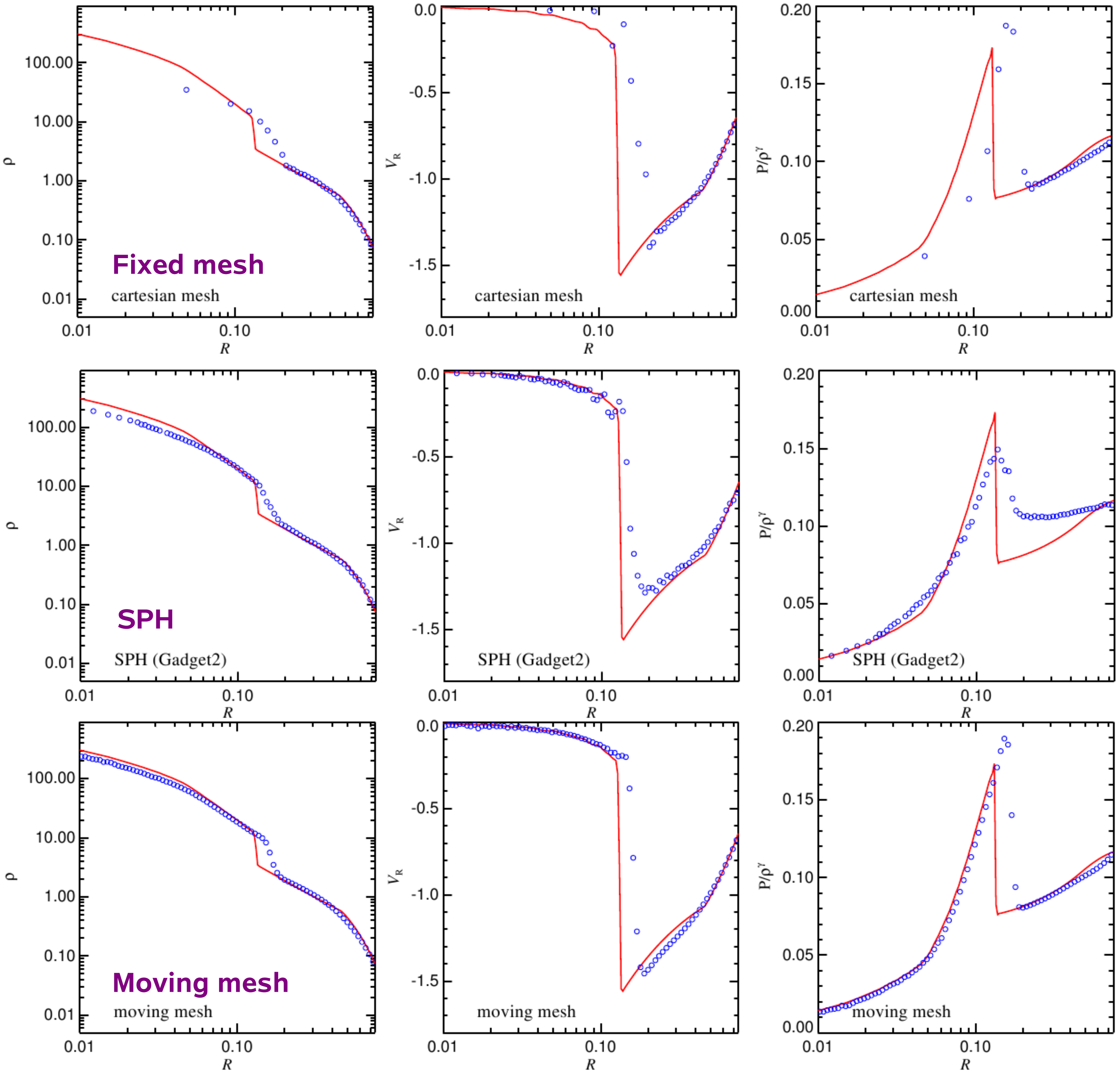




The “Evrard-Collapse” problem is calculated more accurately by the moving-mesh code than by SPH

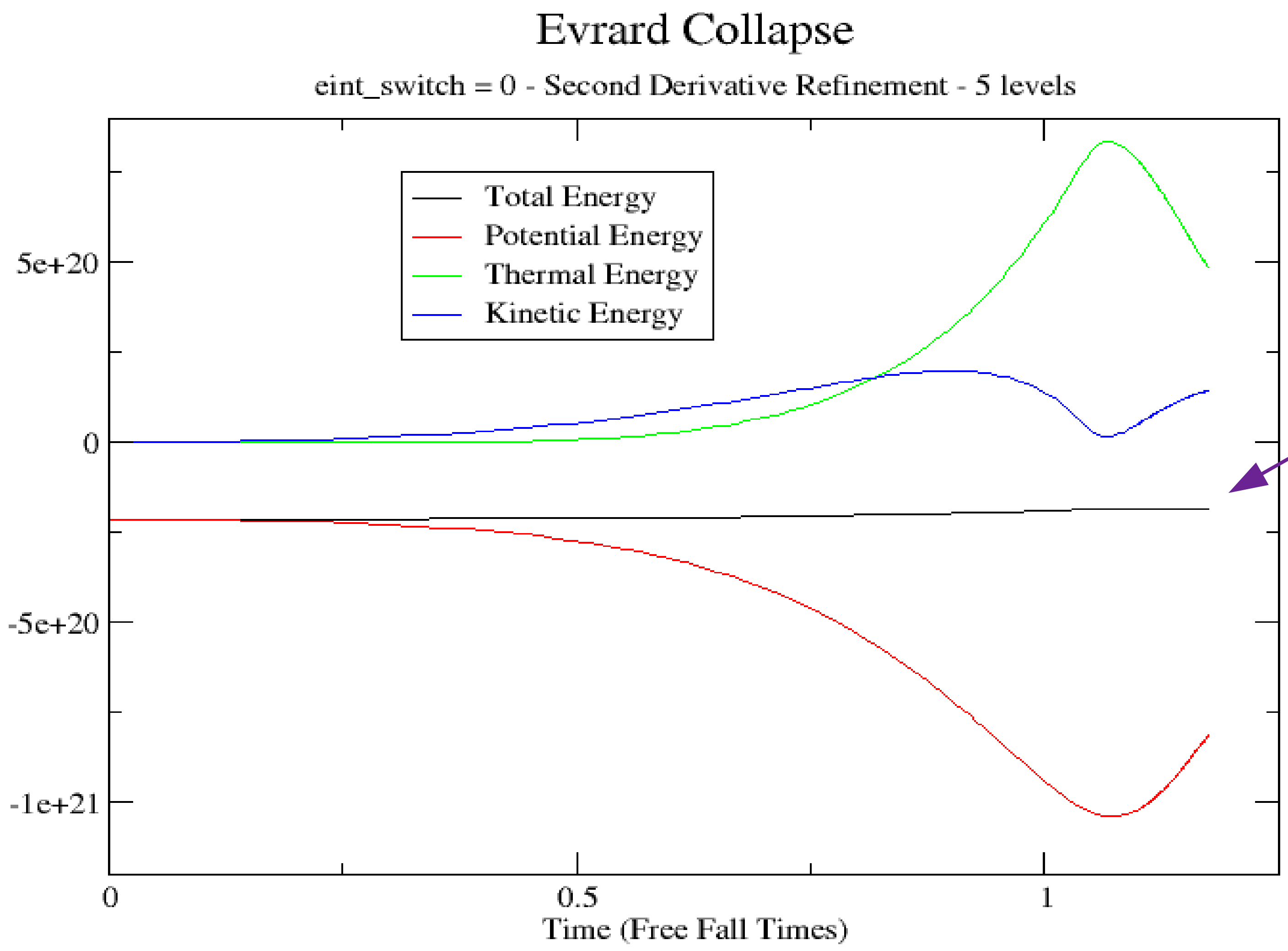
COMPARISON OF SPH WITH A FIXED AND A MOVING MESH

low resolution, equal number of cells/particles



Eulerian finite volume codes have problems to accurately conserve total energy in the “Evrard Collapse” problem

EVRARD COLLAPSE WITH **FLASH-2.5**

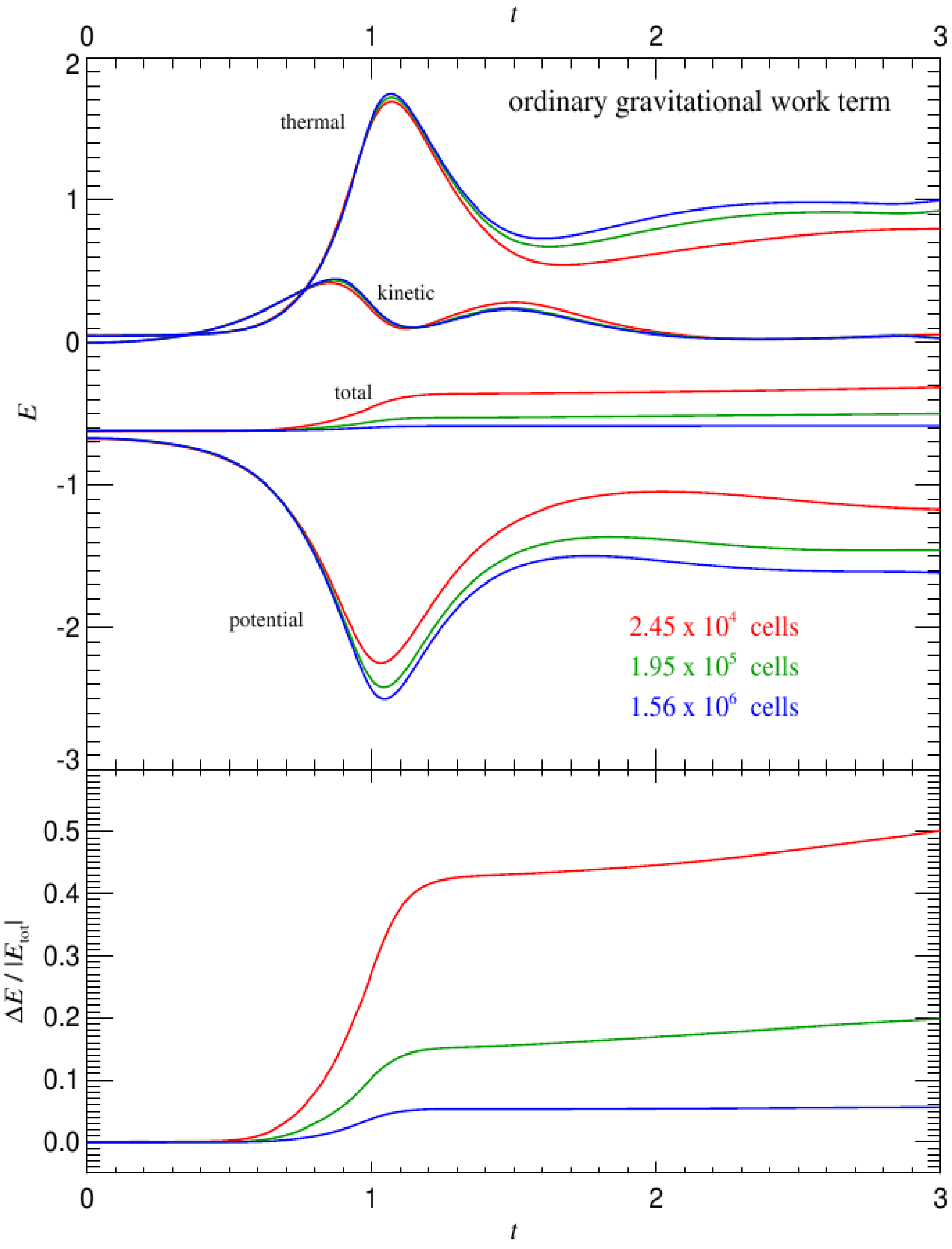


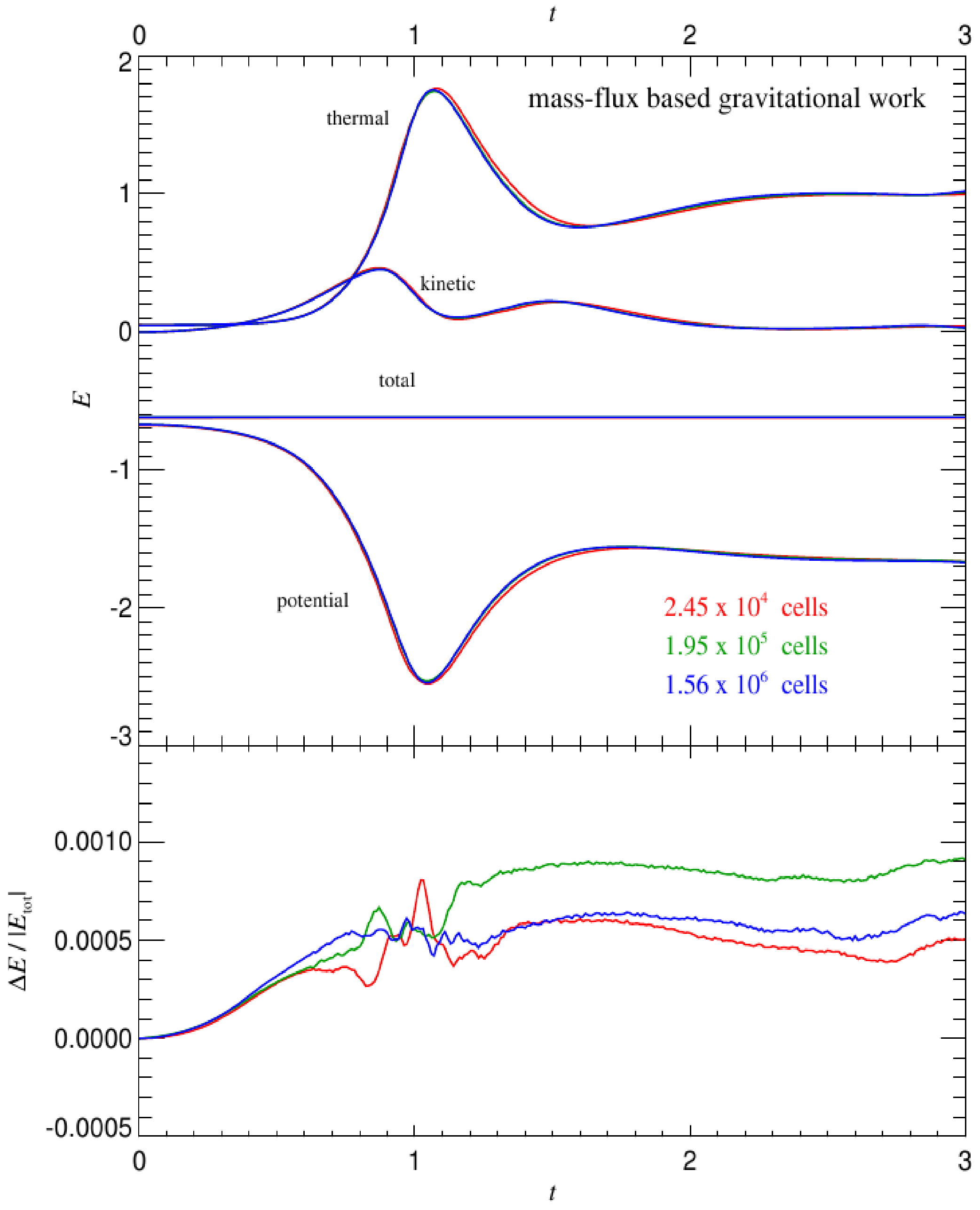
7% energy error

Colin McNally (2002, McMaster University)

The error in the total energy can be substantial in standard treatments to include self-gravity in Eulerian codes

ENERGY ERROR FOR DIFFERENT RESOLUTIONS AND DIFFERENT DISCRETIZATION SCHEMES





How to construct the Voronoi mesh

Construction of the Voronoi diagram is most efficiently done by constructing it as dual of the Delaunay tessellation

A FEW ALGORITHMS FOR DELAUNAY TRIANGULATIONS

2D
- Divide & Conquer (fastest)
- **Sequential insertion**
- Sweepline algorithm
- Projection of 3D convex hull to 3D

3D
- **Sequential insertion**
- Projection of 4D convex hull to 3D
- Incremental construction

Sequential insertion:

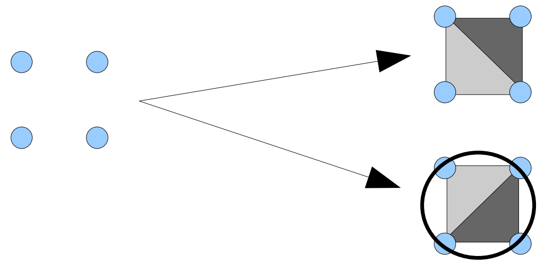
(1) **Point location:** Find triangle/tetrahedron that contains point

(2) **Point insertion**: Split enclosing triangle/tetrahedron into several simplices

(3) **Flips to restore Delaunayhood:** Replace edges/facets around the inserted point if they violate the Delaunay condition (empty circumcircle)

Most algorithms assume the **general position assumption**

Unfortunately, **degenerate cases** do occur in practice, and induce numerical difficulties due to numerical round-off

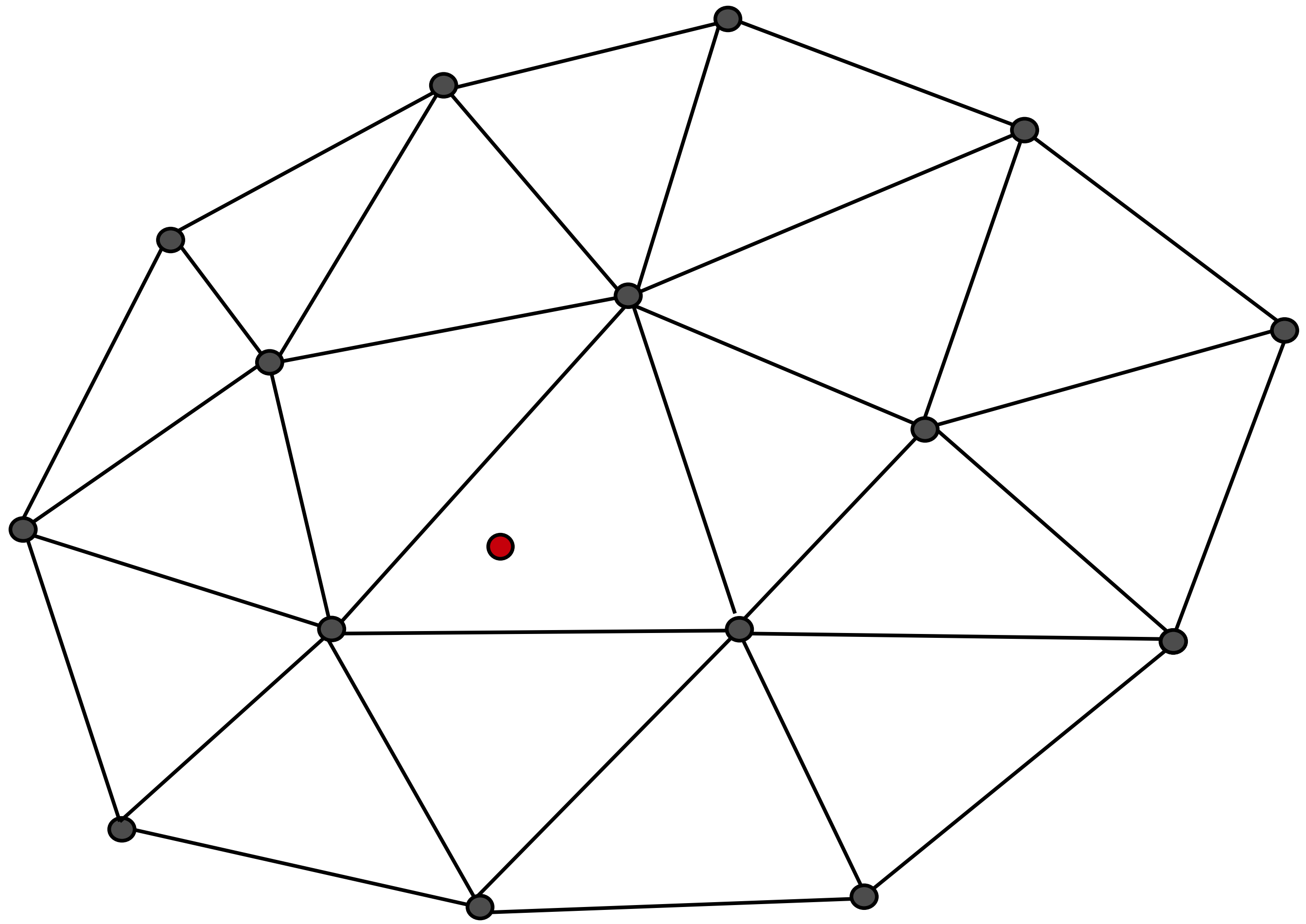


4 cocircular points !

How can we consistently break ties?

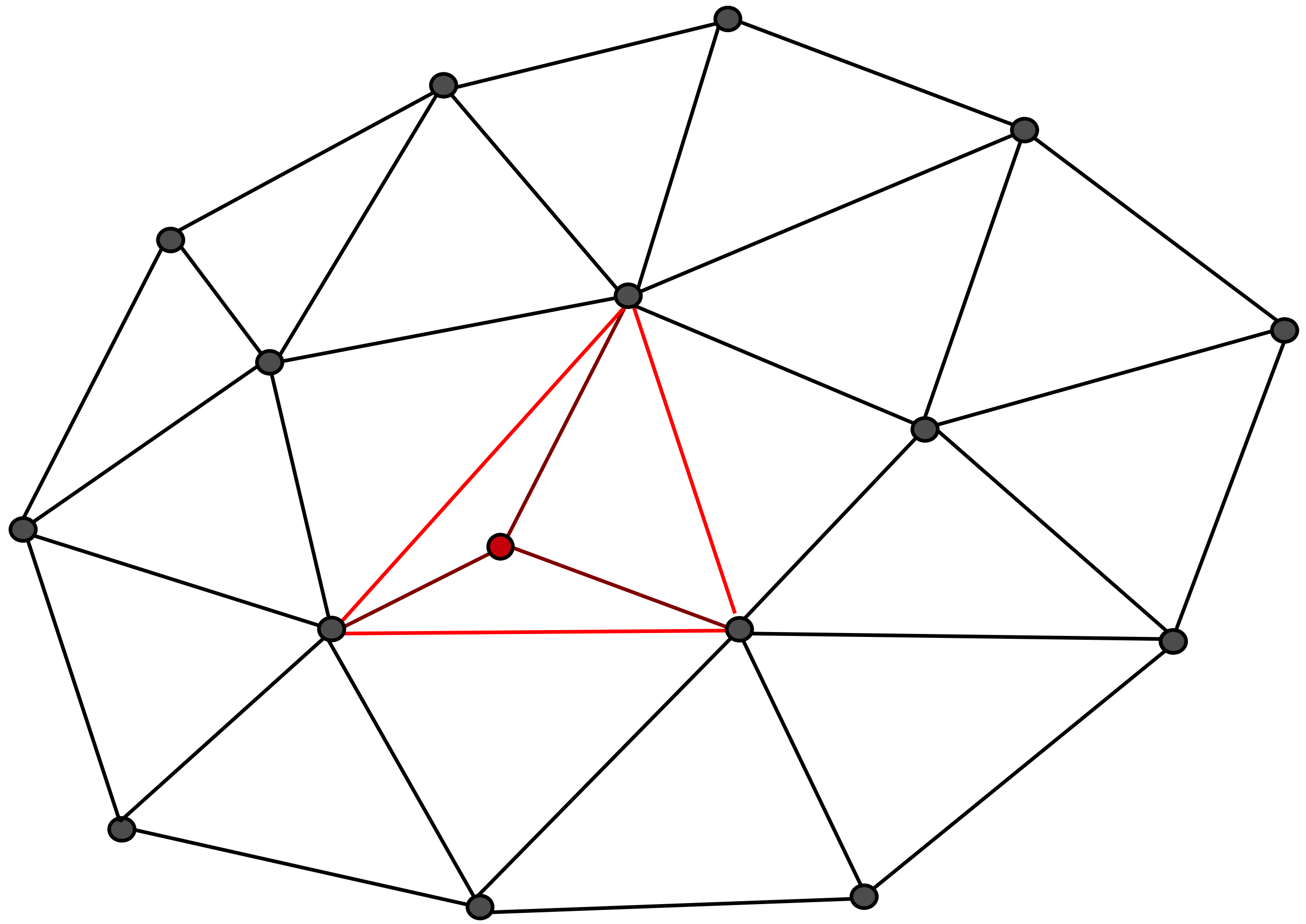
Adding a point by sequential insertion

1. Step: Locate the triangle that contains the point



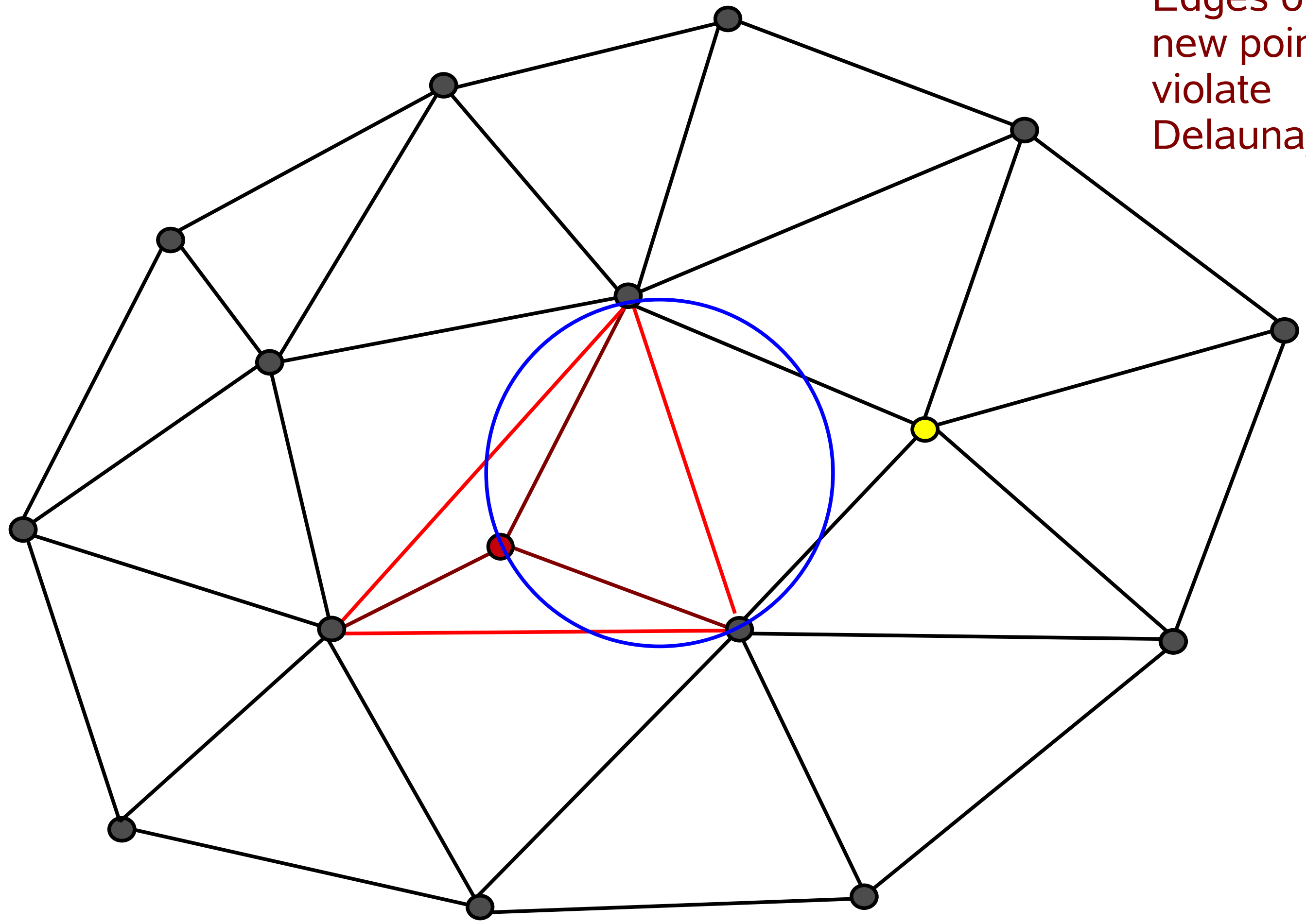
Adding a point by sequential insertion

2. Step: Split the triangle into three triangles



Adding a point by sequential insertion

3. Step: Legalize the new triangles

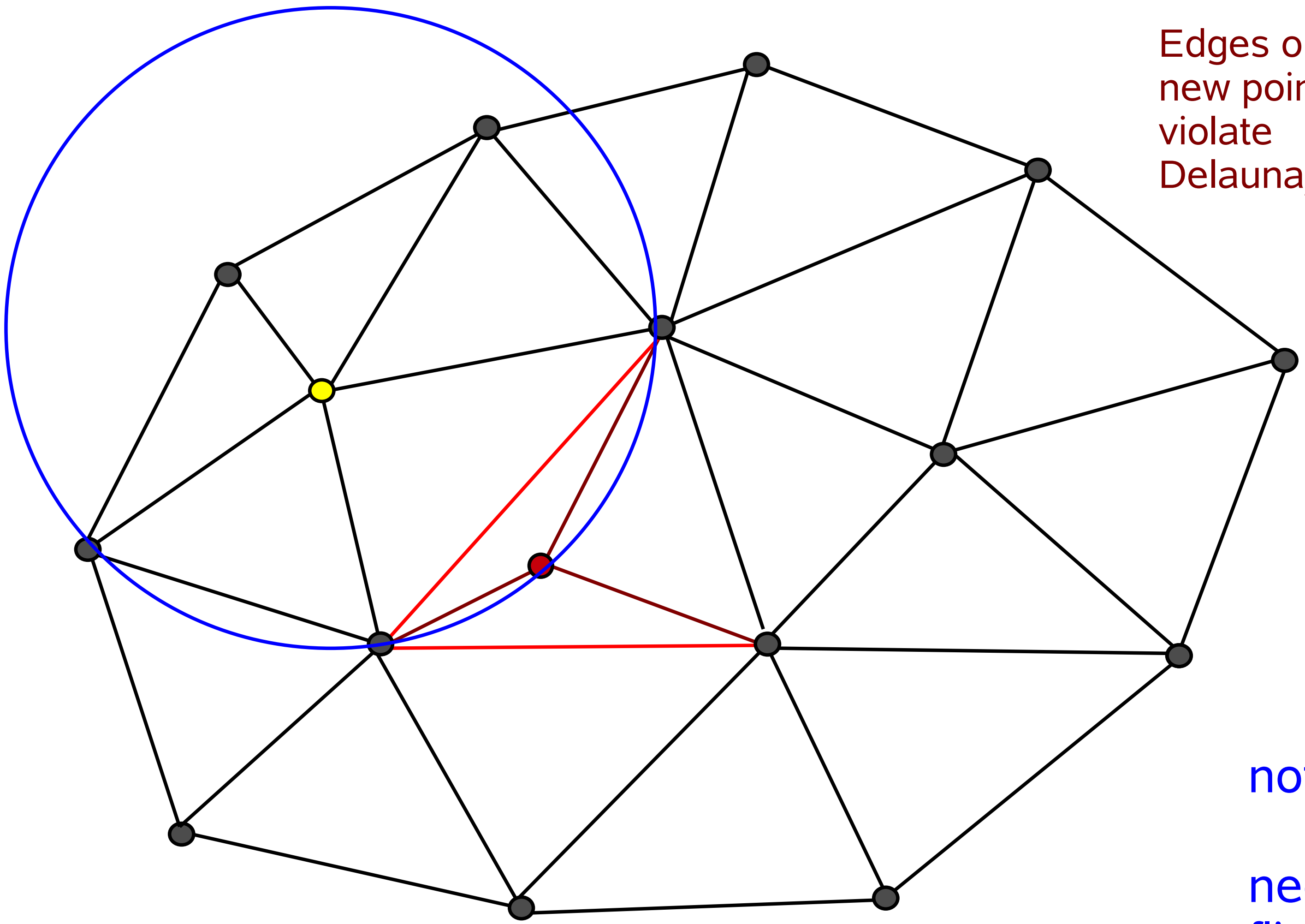


Edges opposite of new point my violate Delaunayhood

Ok!

Adding a point by sequential insertion

3. Step: Legalize the new triangles



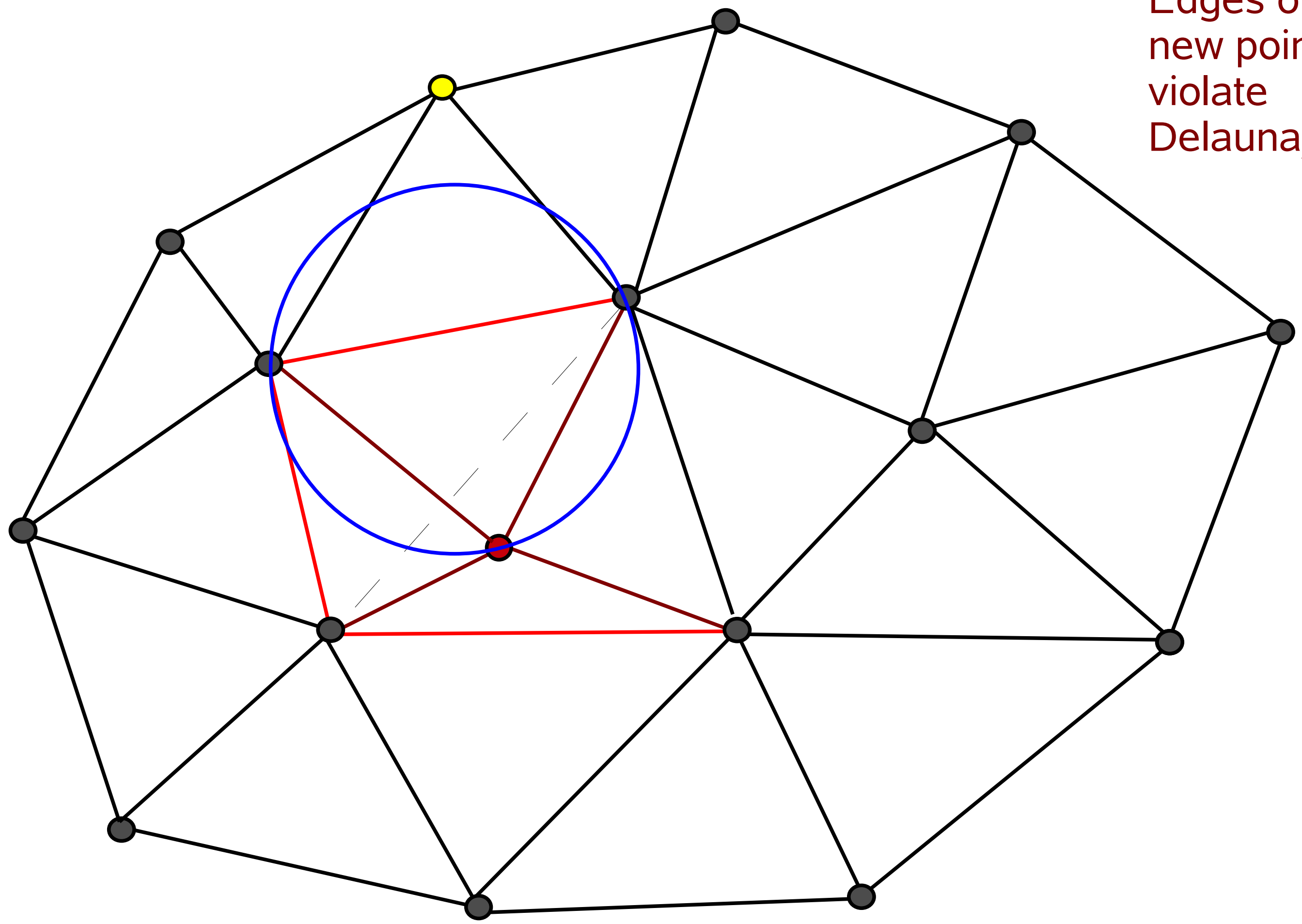
Edges opposite of new point my violate Delaunayhood

not ok...!

need to flip edge

Adding a point by sequential insertion

3. Step: Legalize the new triangles

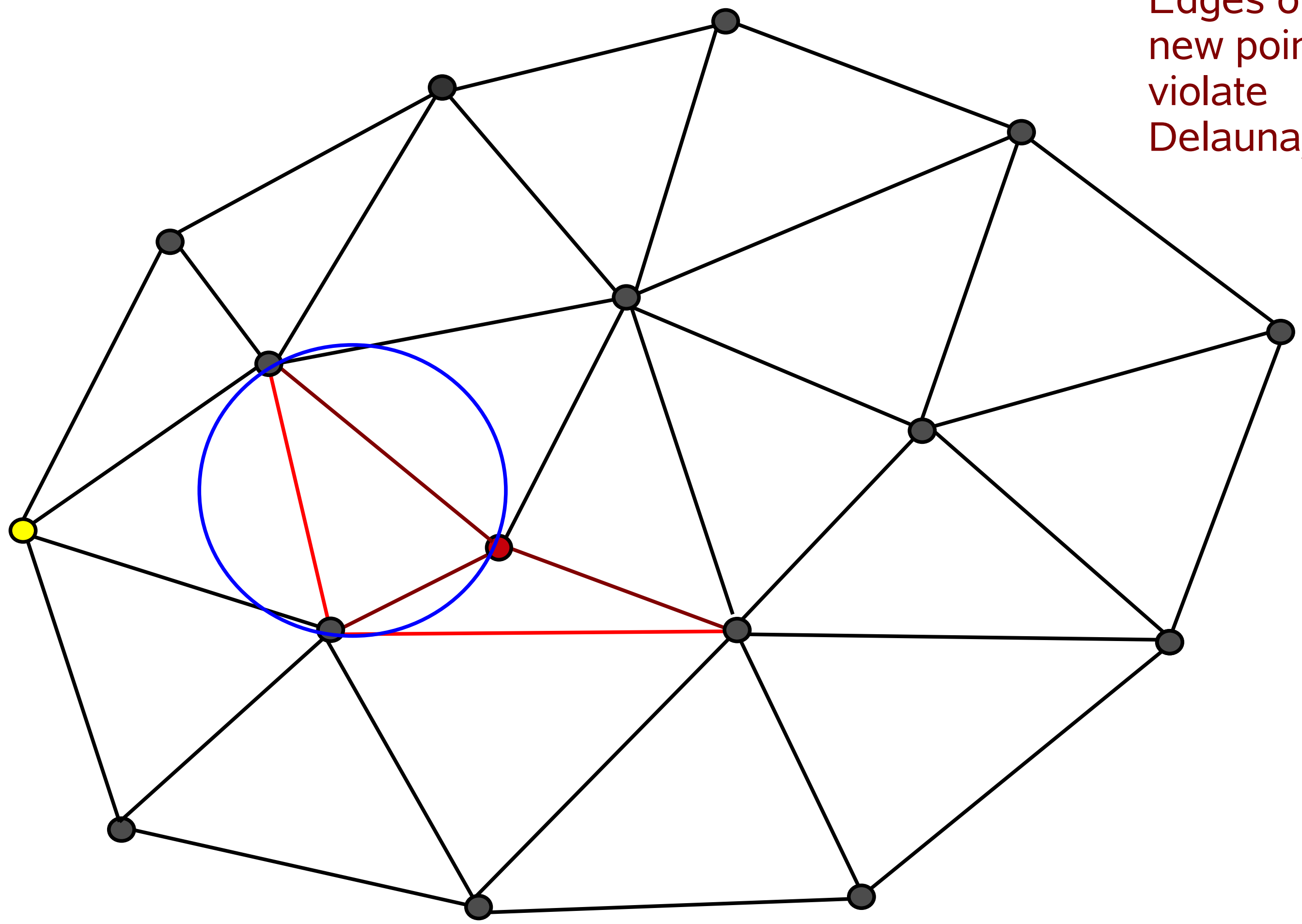


Edges opposite of new point my violate Delaunayhood

Ok!

Adding a point by sequential insertion

3. Step: Legalize the new triangles

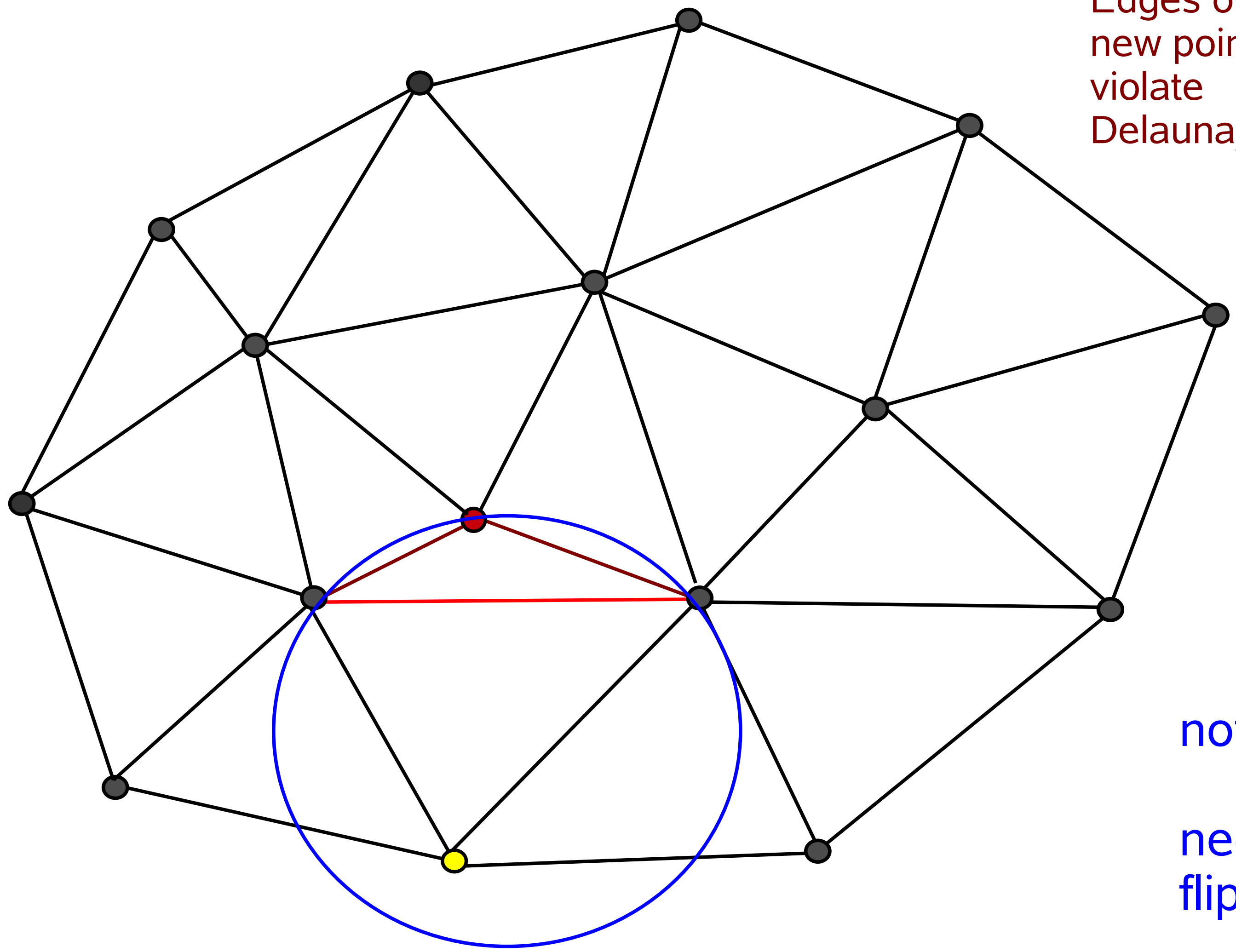


Edges opposite of new point my violate Delaunayhood

Ok!

Adding a point by sequential insertion

3. Step: Legalize the new triangles



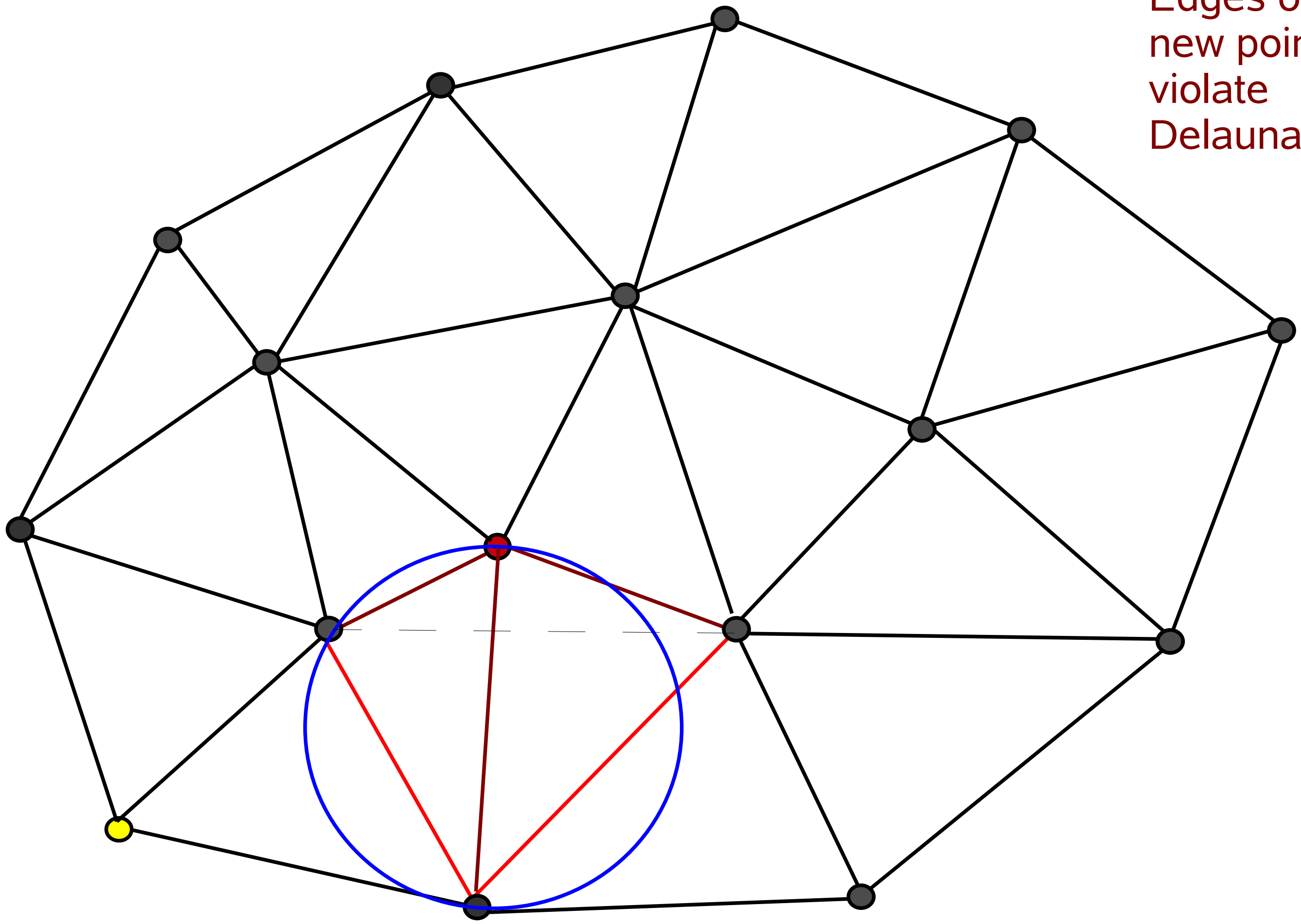
Edges opposite of new point my violate Delaunayhood

not ok...!

need to flip edge

Adding a point by sequential insertion

3. Step: Legalize the new triangles

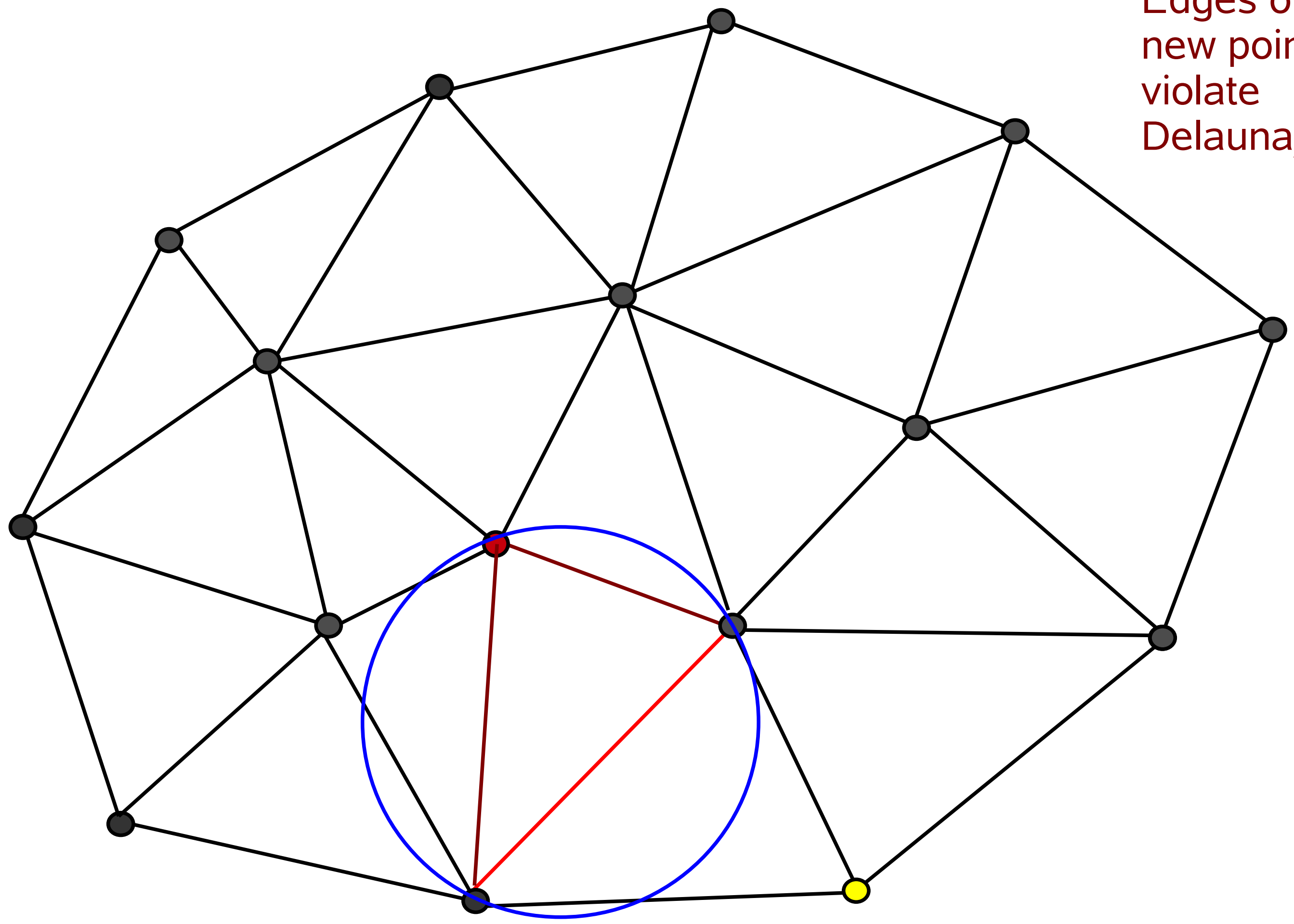


Edges opposite of new point my violate Delaunayhood

Ok!

Adding a point by sequential insertion

3. Step: Legalize the new triangles

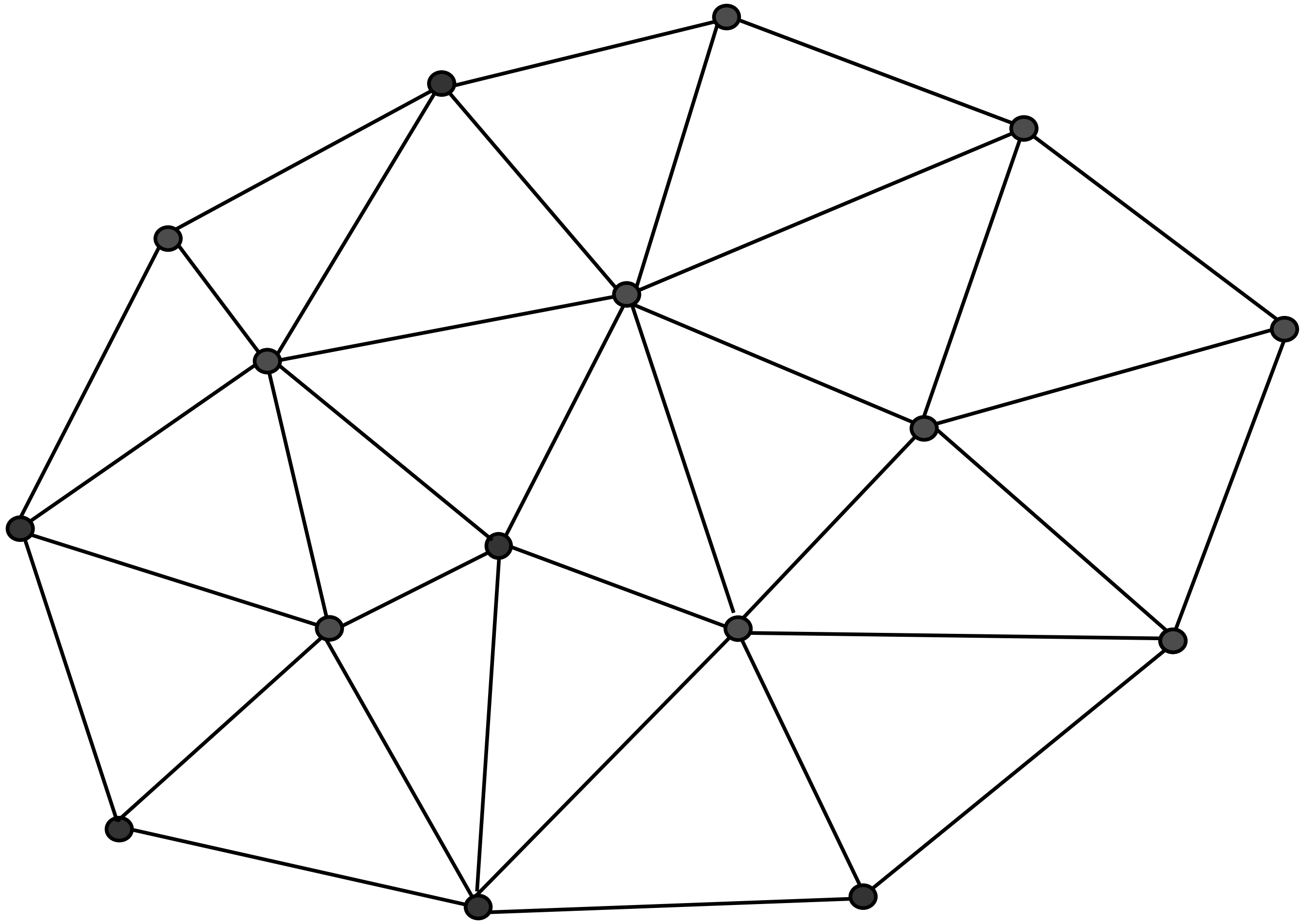


Edges opposite of new point my violate Delaunayhood

Ok!

Adding a point by sequential insertion

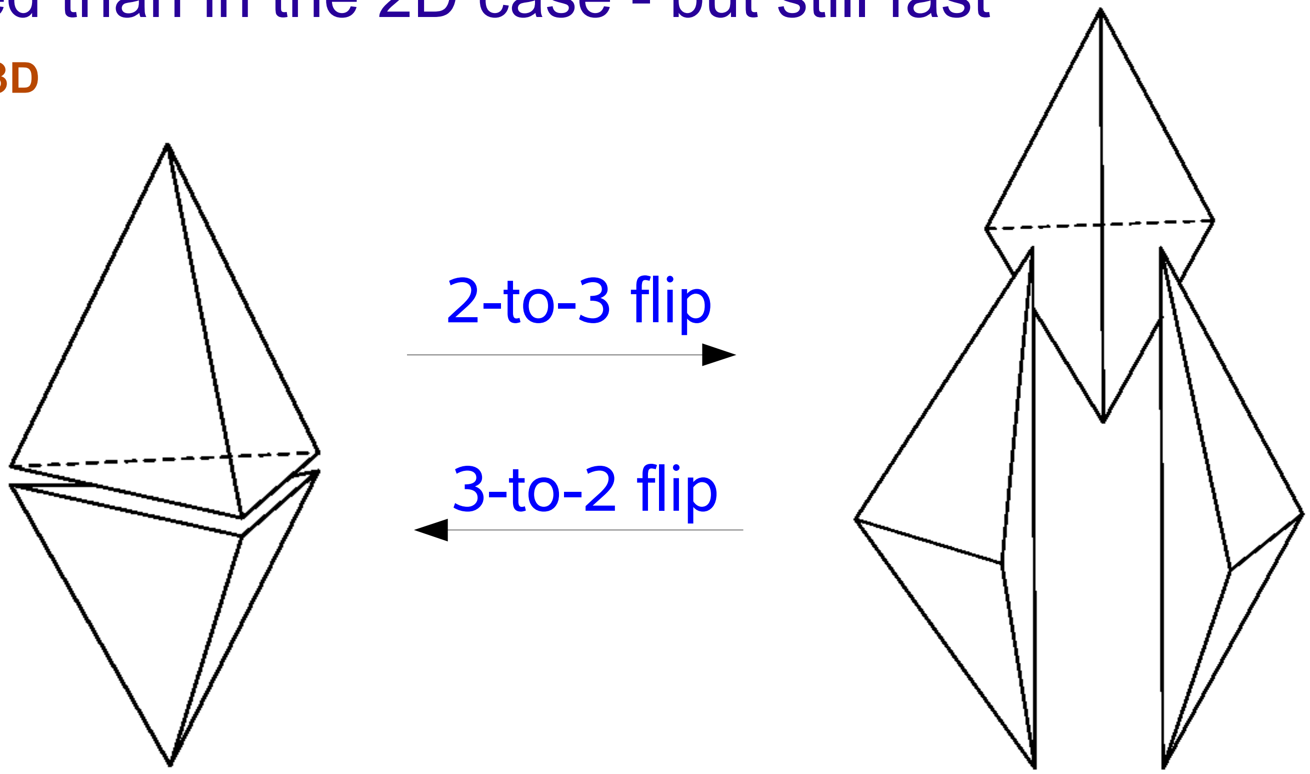
4. Step: Finished! (Or insert next point)



The construction of the 3D Delaunay tessellation is significantly more complicated than in the 2D case - but still fast

FLIP OPERATIONS IN 3D

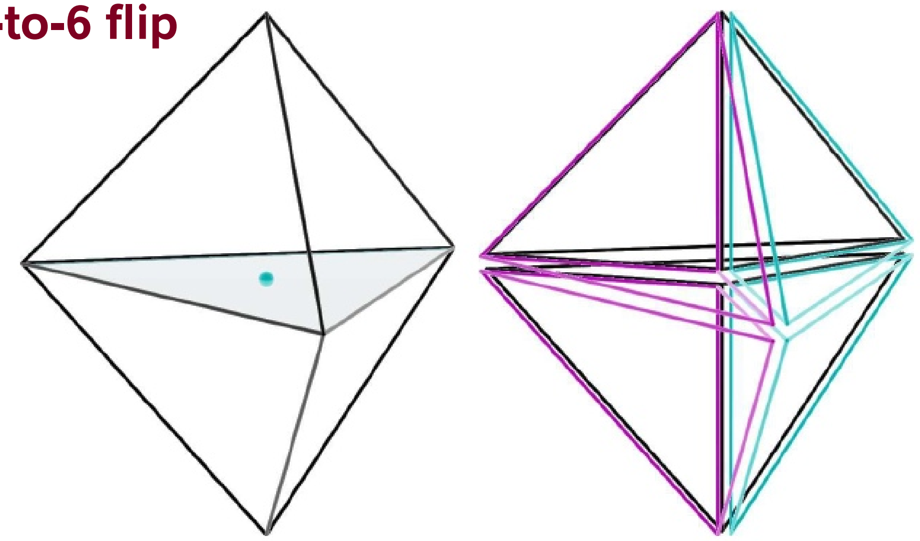
1-to-4 flip
(point insertion)



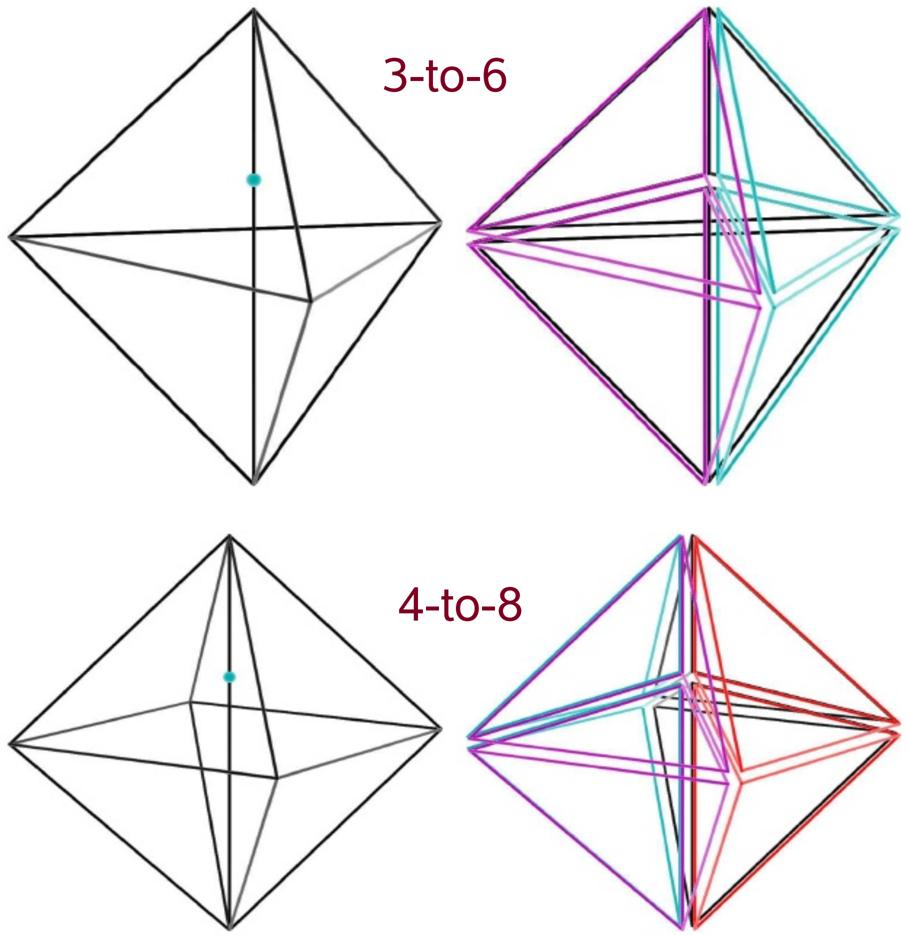
If the **general position assumption** is not fulfilled, degenerate cases can occur. This makes thinks a lot more complicated. One then needs:

- **1-to-N flips** for point insertion when the point lies on an edge
- **2-to-6 flips** if the point lies on a face
- **4-to-4 flips** for reestablishing Delaunayhood
- Accurate geometric predicates required (difficult! Occasionally requires *exact* arithmetic)

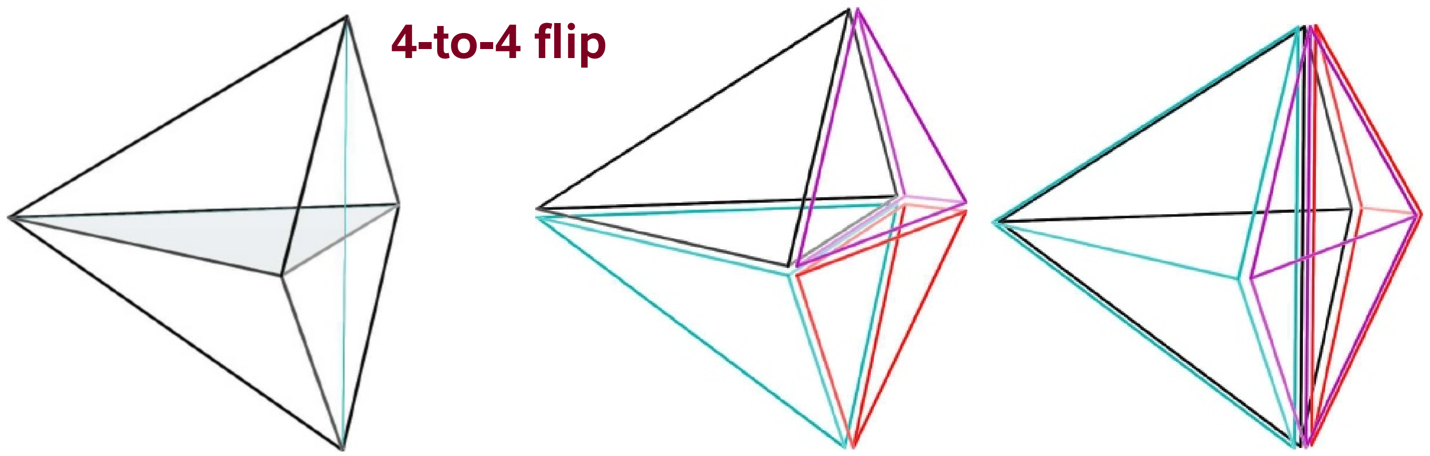
2-to-6 flip



n-to-2n flips

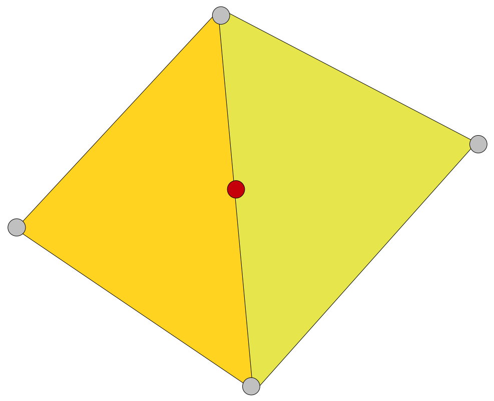


4-to-4 flip



Degenerate point configurations cause trouble – exact arithmetic is required to guarantee robustness

USE OF EXACT ARITHMETIC TO DEAL WITH POINTS IN NON-GENERAL POSITION



Is the point in the left or right triangle?

Or is it *exactly* on the line?

(boils down to evaluating the sign of geometric tests)

$$T_{\mathrm{InCircle}}(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d}) = \begin{vmatrix} 1 & a_x & a_y & a_x^2 + a_y^2 \\ 1 & b_x & b_y & b_x^2 + b_y^2 \\ 1 & c_x & c_y & c_x^2 + c_y^2 \\ 1 & d_x & d_y & d_x^2 + d_y^2 \end{vmatrix}$$

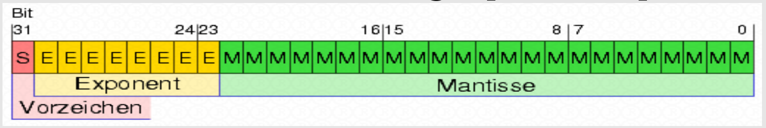
Delaunay algorithms tend to crash if wrong decisions are made!

Solution

- **Calculate maximum round-off error** in geometric tests, and check whether result could be incorrect
- If the decision is ambiguous due to floating point round-off, **use exact arithmetic** instead

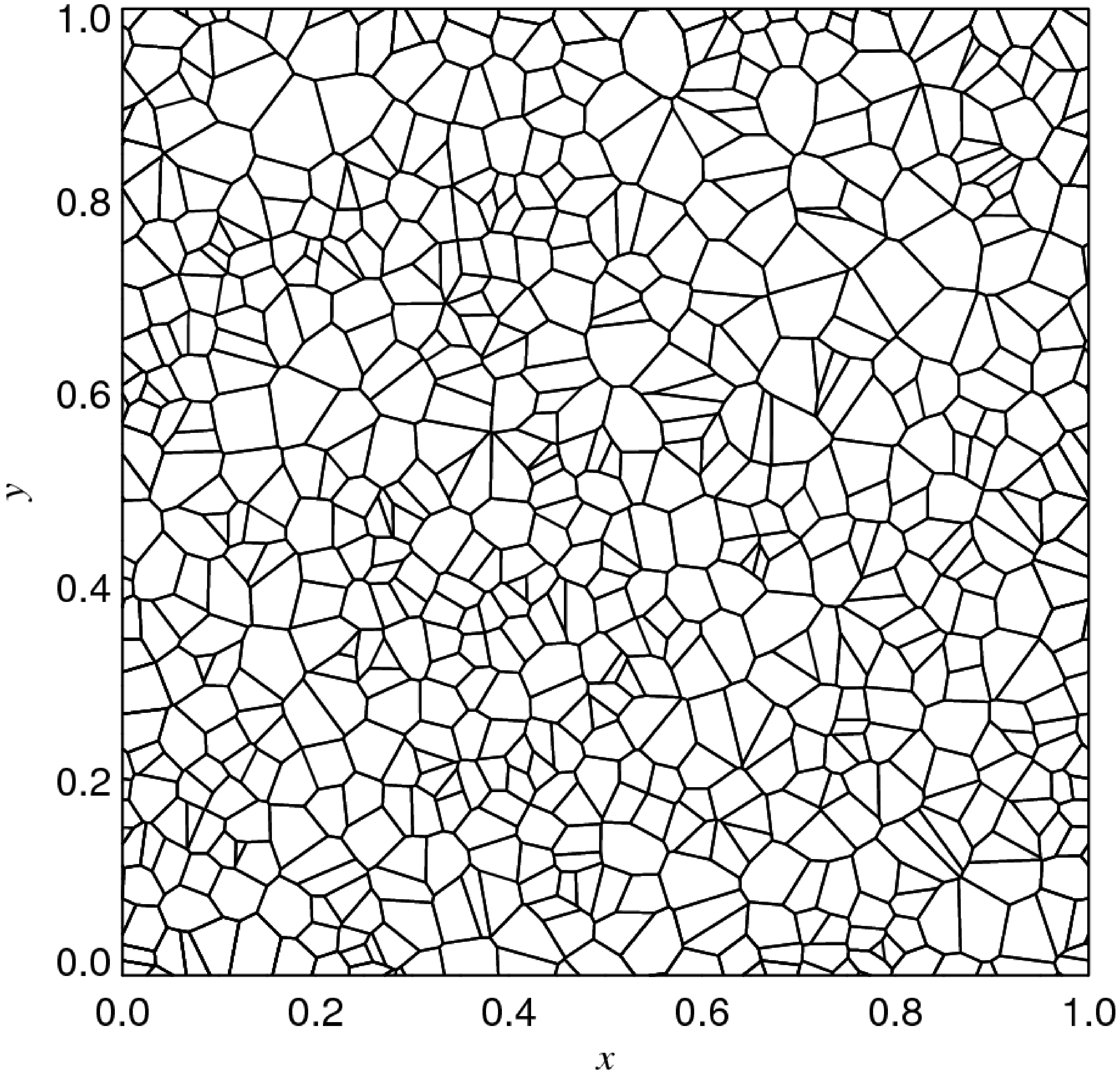
We use **exact integer arithmetic** if needed:

- Domain is mapped to floating point numbers in the range [1.0, 2.0]

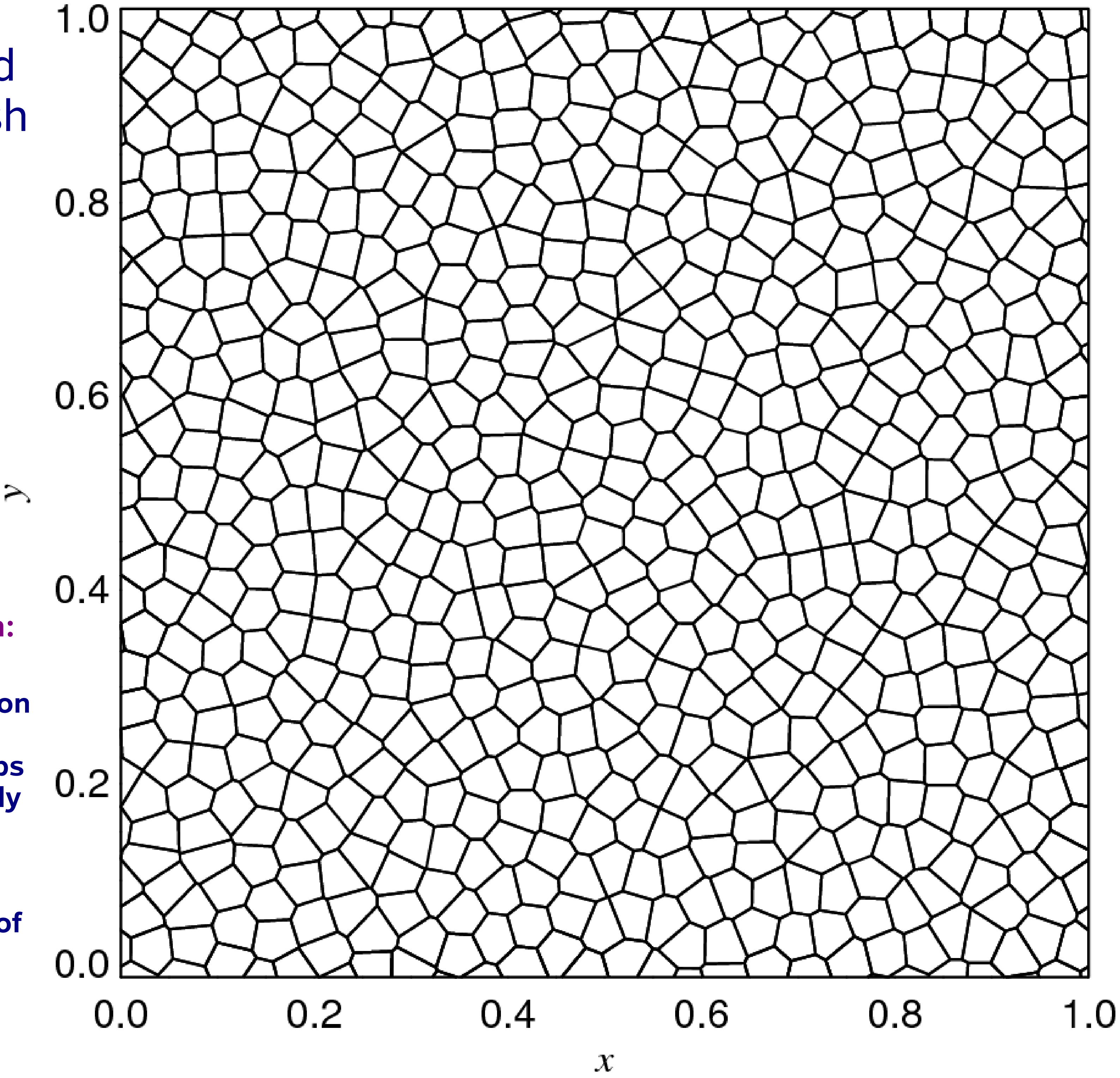

Bit
31
24|23
16|15
8|7
0
S E E E E E E E E M M M M M M M M M M M M M M M M M M M M M M M
Exponent
Mantisse
Vorzeichen

- Mantissa provides a 53-bit integer with a unique one-to-one mapping to the floating point numbers
- Carry out the geometric test with the GMP-library using long integers

The Voronoi mesh of 625 randomly distributed points



A regularized Voronoi mesh



We would like to have such a mesh:

- **Optimal use of available resolution**
- **No small timesteps due to accidentally small cell dimensions**
- **Higher accuracy of spatial reconstruction**

Lloyd's algorithm is a simple iterative scheme to create a centroidal Voronoi tessellation

MAKING ROUND VORONOI CELLS

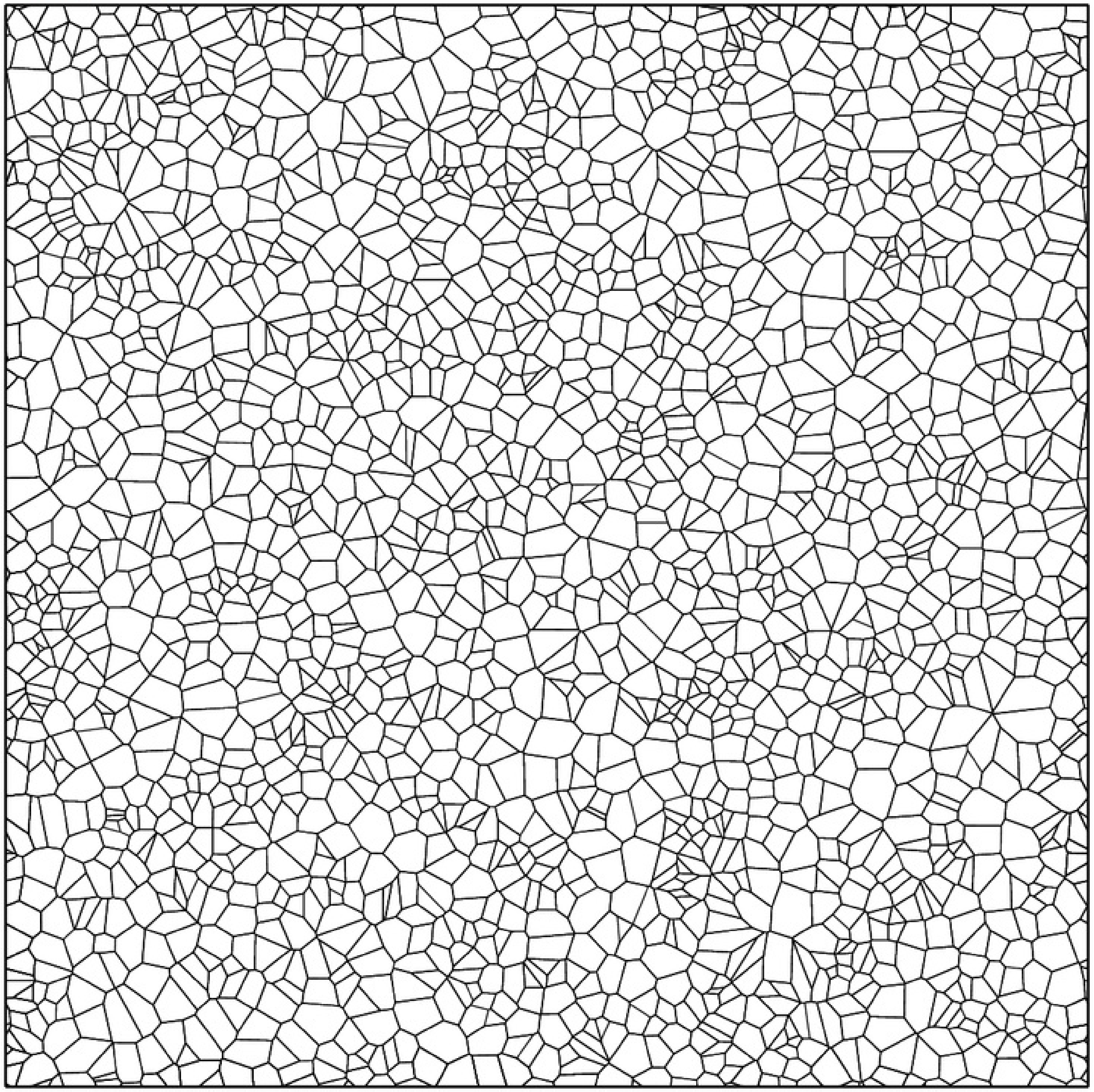
Lloyd's algorithm:

(1) Reposition each mesh-generating point of a Voronoi tessellation to the center-of-mass of its associated cell.

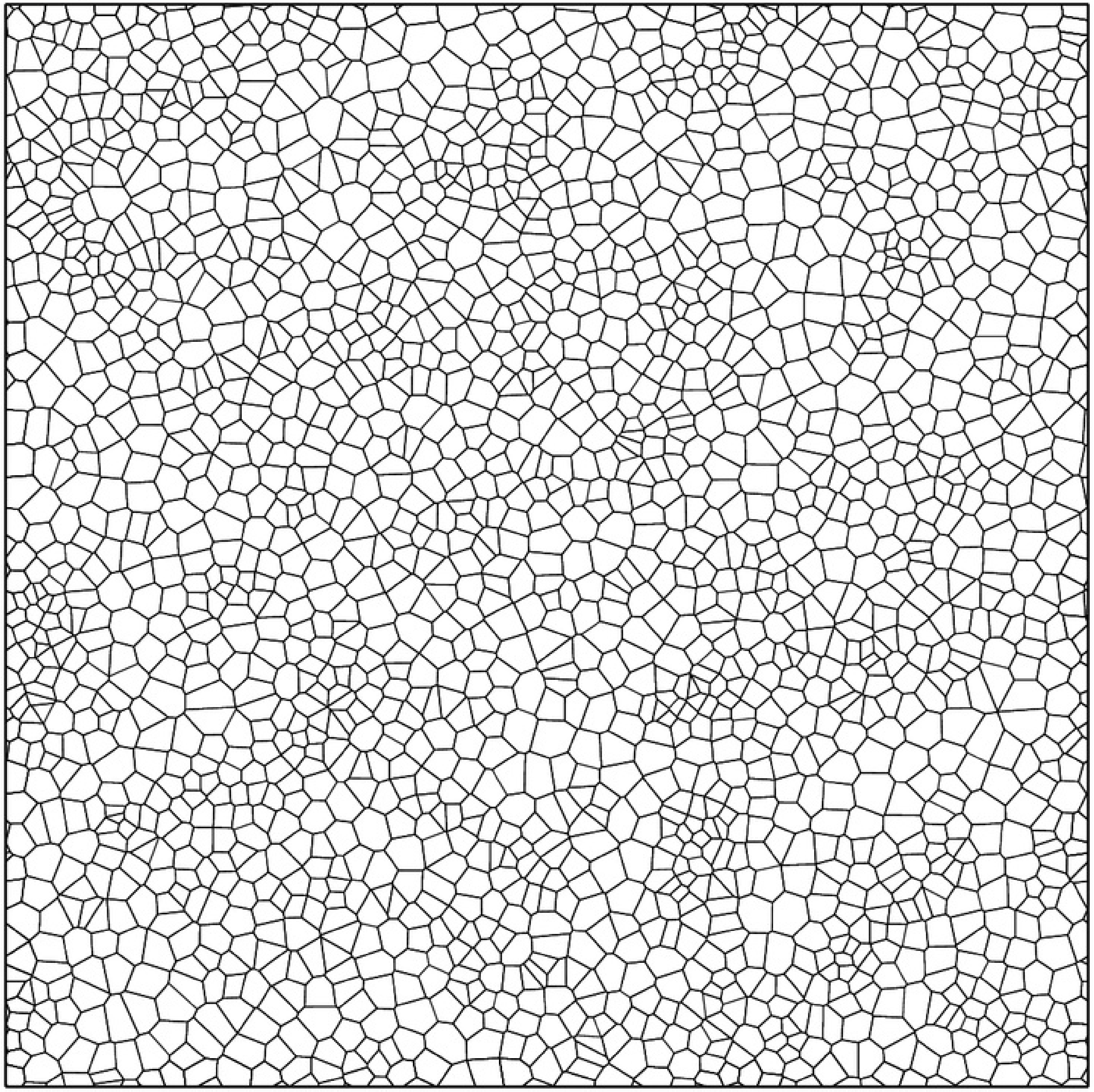
(2) Reconstruct the tessellation, and repeat step (1).

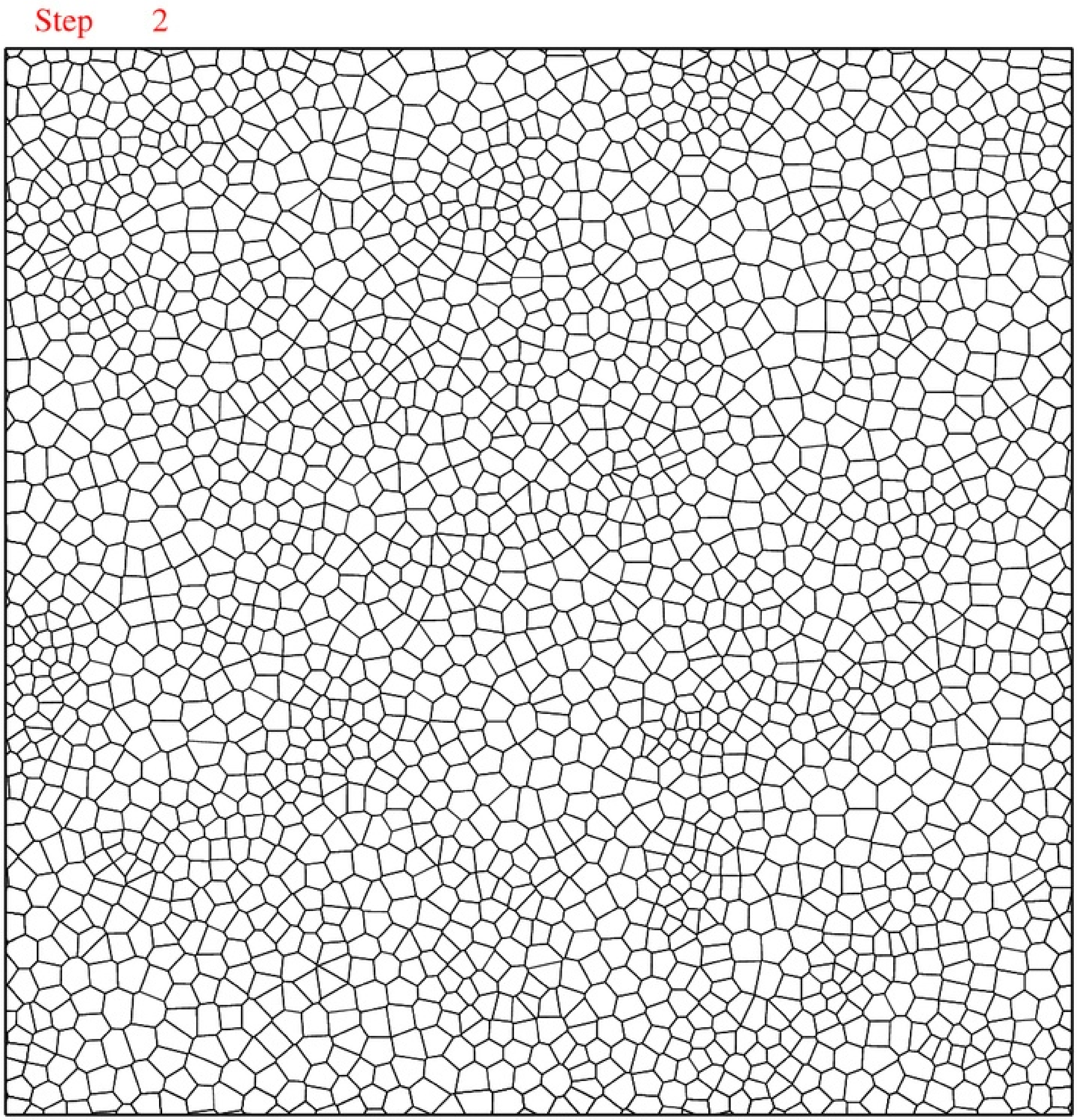
$\longrightarrow$ Converges slowly to **centroidal Voronoi tessellation**.

Step 0

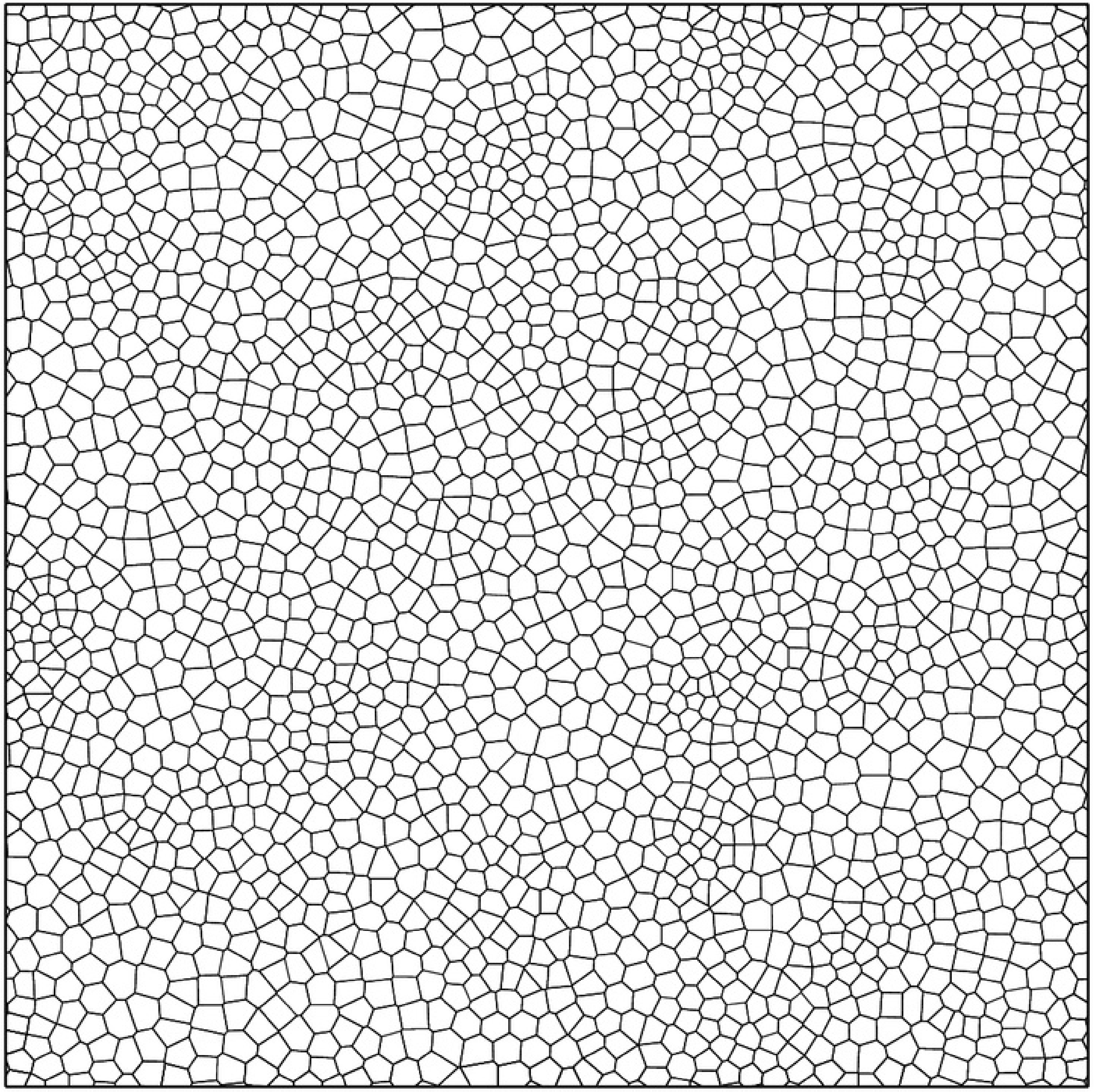


Step 1

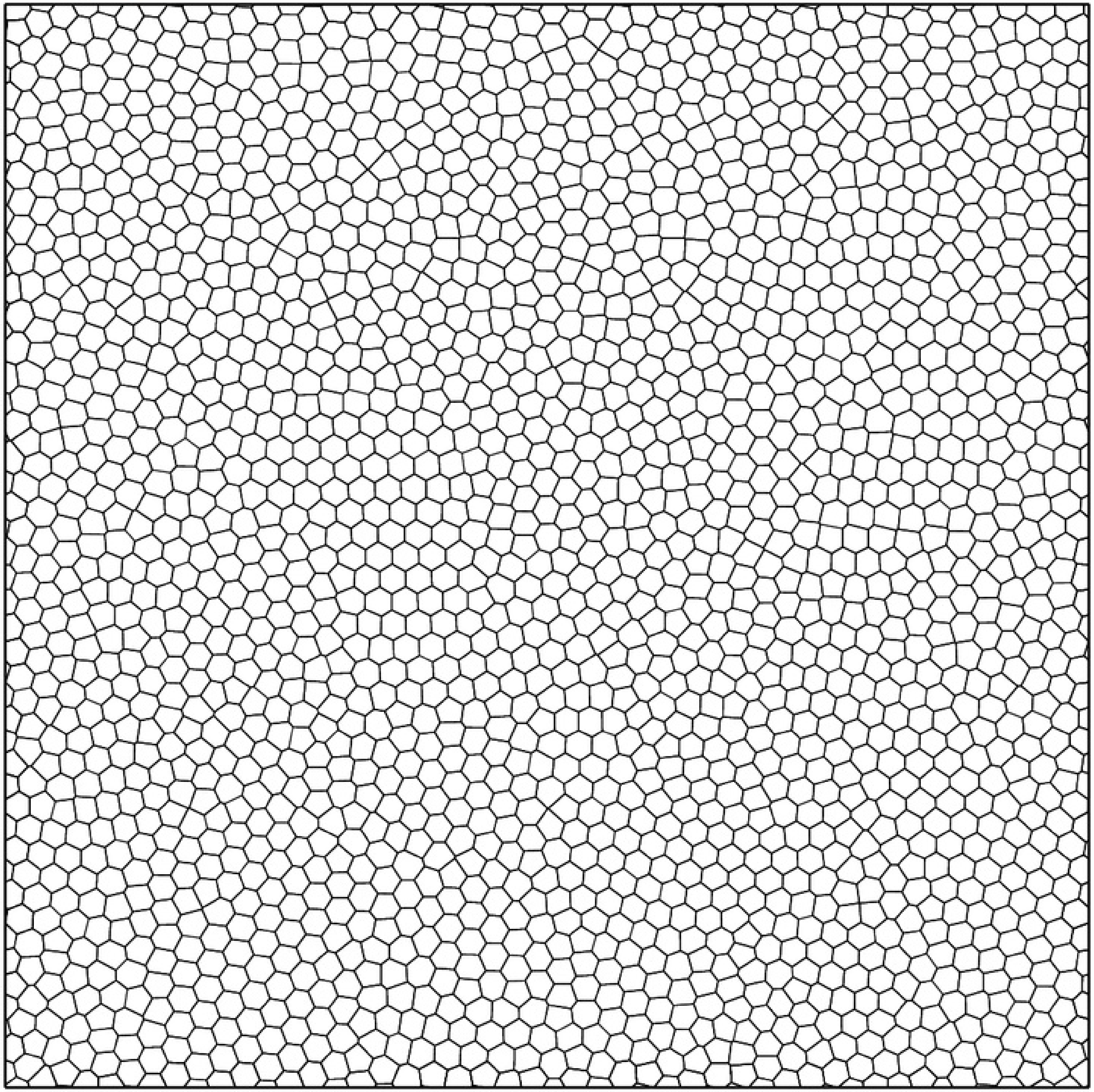


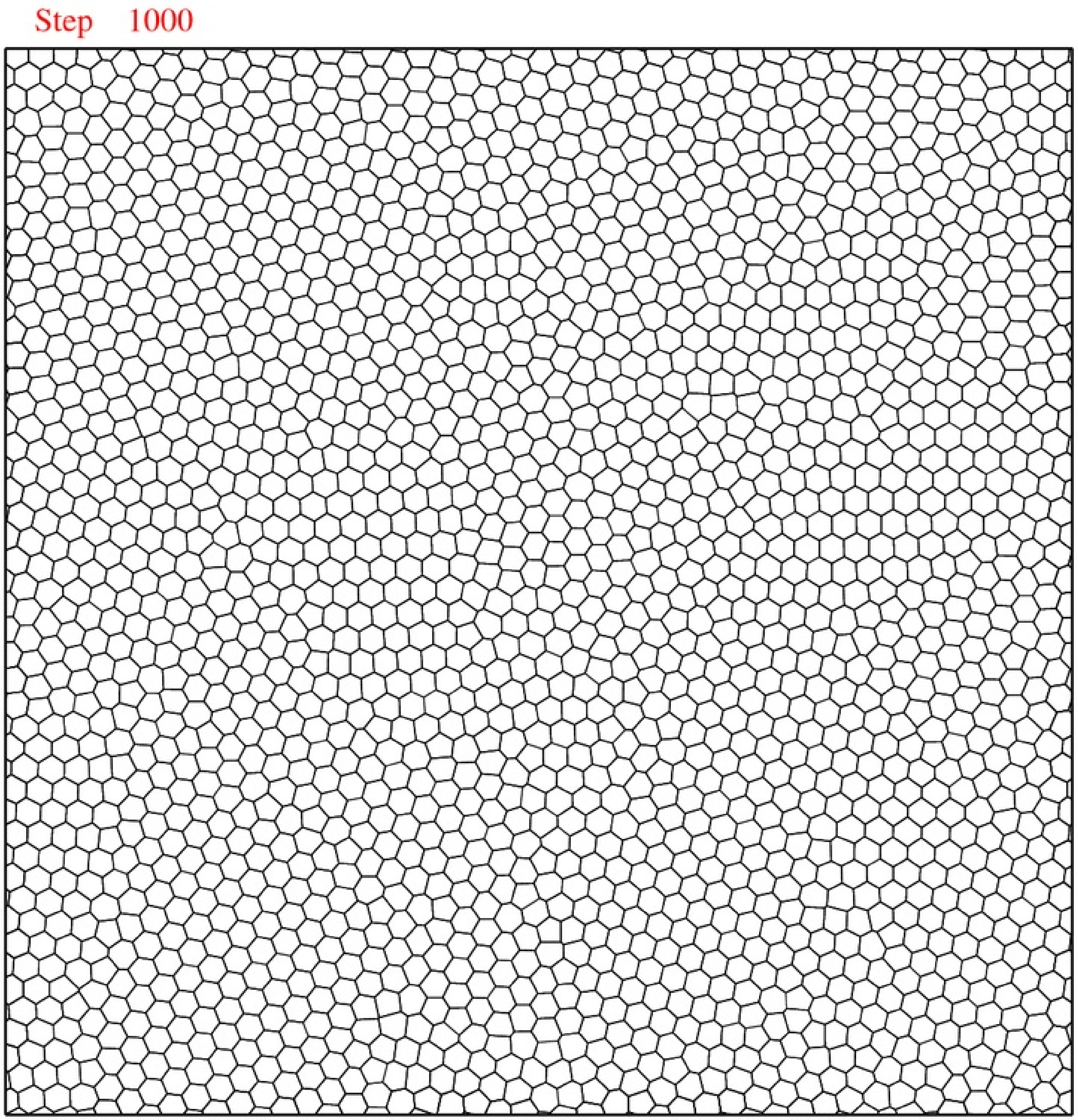

Step 2

Step 3



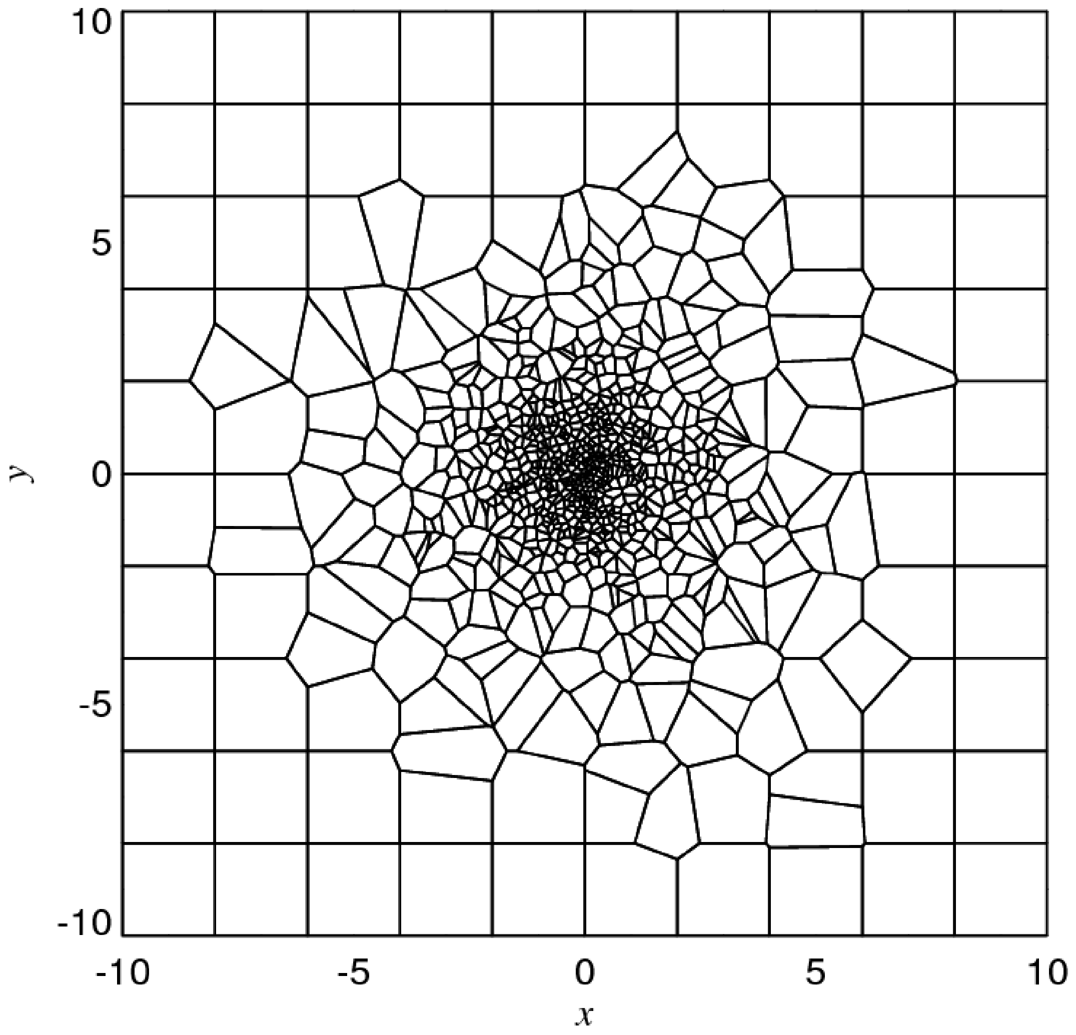
Step 100

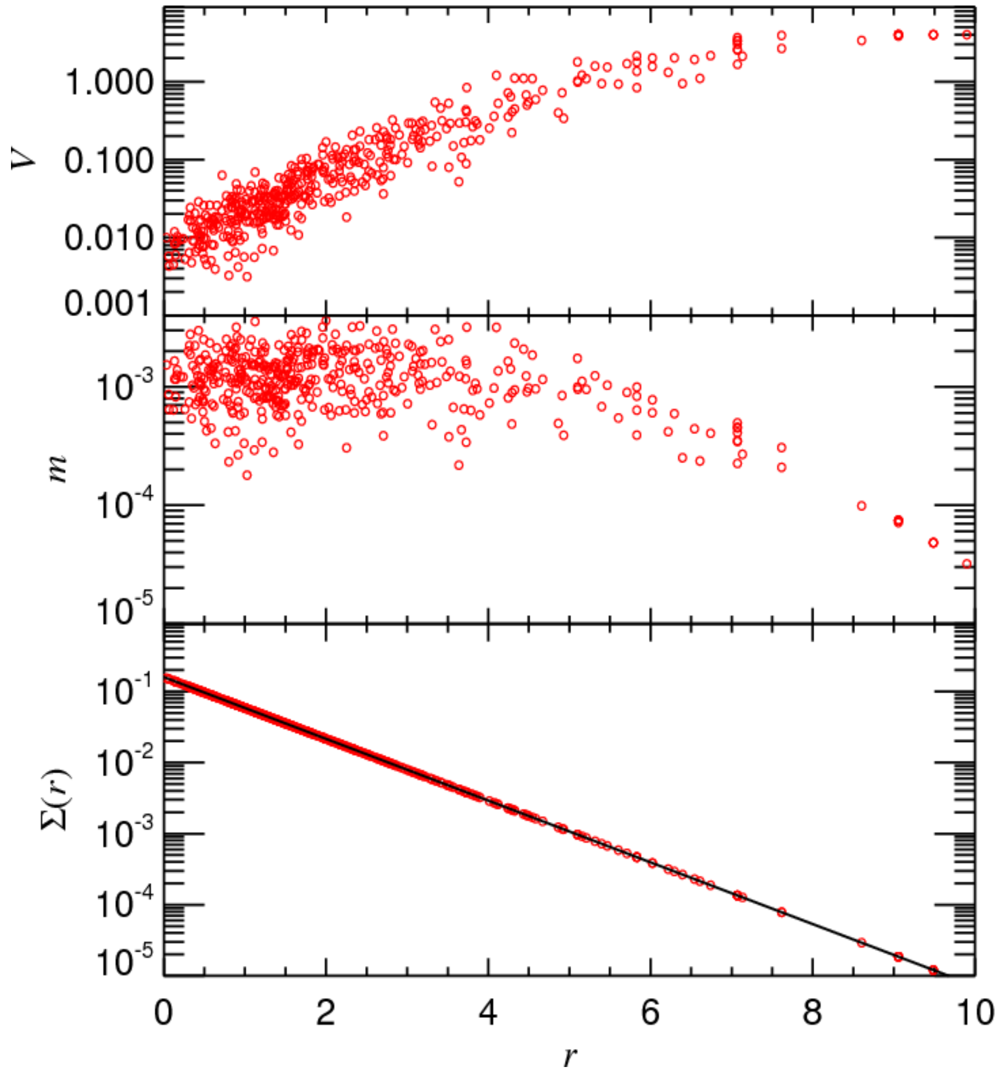



Step 1000

Mesh relaxation would also be very useful for creating quiet initial conditions

AN EXPONENTIAL DISK INITIALIZED WITH A POISSOIN DISTRIBUTION





How should we shift the points to obtain the desired mesh properties?

MAKING OR PRESERVING A "NICE" MESH

Current distribution of points $n(\boldsymbol{x})$

Ideal (desired) distribution of points $n_0(\boldsymbol{x})$

Lets assume the displacement field from the current coordinate can be described by:

$$\boldsymbol{x}_i = \boldsymbol{q}_i + \epsilon\, \boldsymbol{d}_i \qquad \boldsymbol{d} = -\boldsymbol{\nabla}\Psi$$

Then to linear order:

$$\boldsymbol{v}_i = \mathrm{d}\boldsymbol{x}_i/\mathrm{d}\epsilon = \boldsymbol{d}_i \qquad n_\epsilon(\boldsymbol{x}+\epsilon\boldsymbol{d}) = \epsilon\, n(\boldsymbol{x}+\boldsymbol{d}) + (1-\epsilon)\, n_0(\boldsymbol{x}) \qquad \frac{\mathrm{d}n}{\mathrm{d}\epsilon} + n\boldsymbol{\nabla}\cdot\boldsymbol{v} = 0$$

Yields Poisson equation for displacement:

$$\nabla^2\Psi = \frac{n(\boldsymbol{x}+\boldsymbol{d})}{n_0(\boldsymbol{x})} - 1 \simeq \frac{n(\boldsymbol{x})}{n_0(\boldsymbol{x})} - 1$$

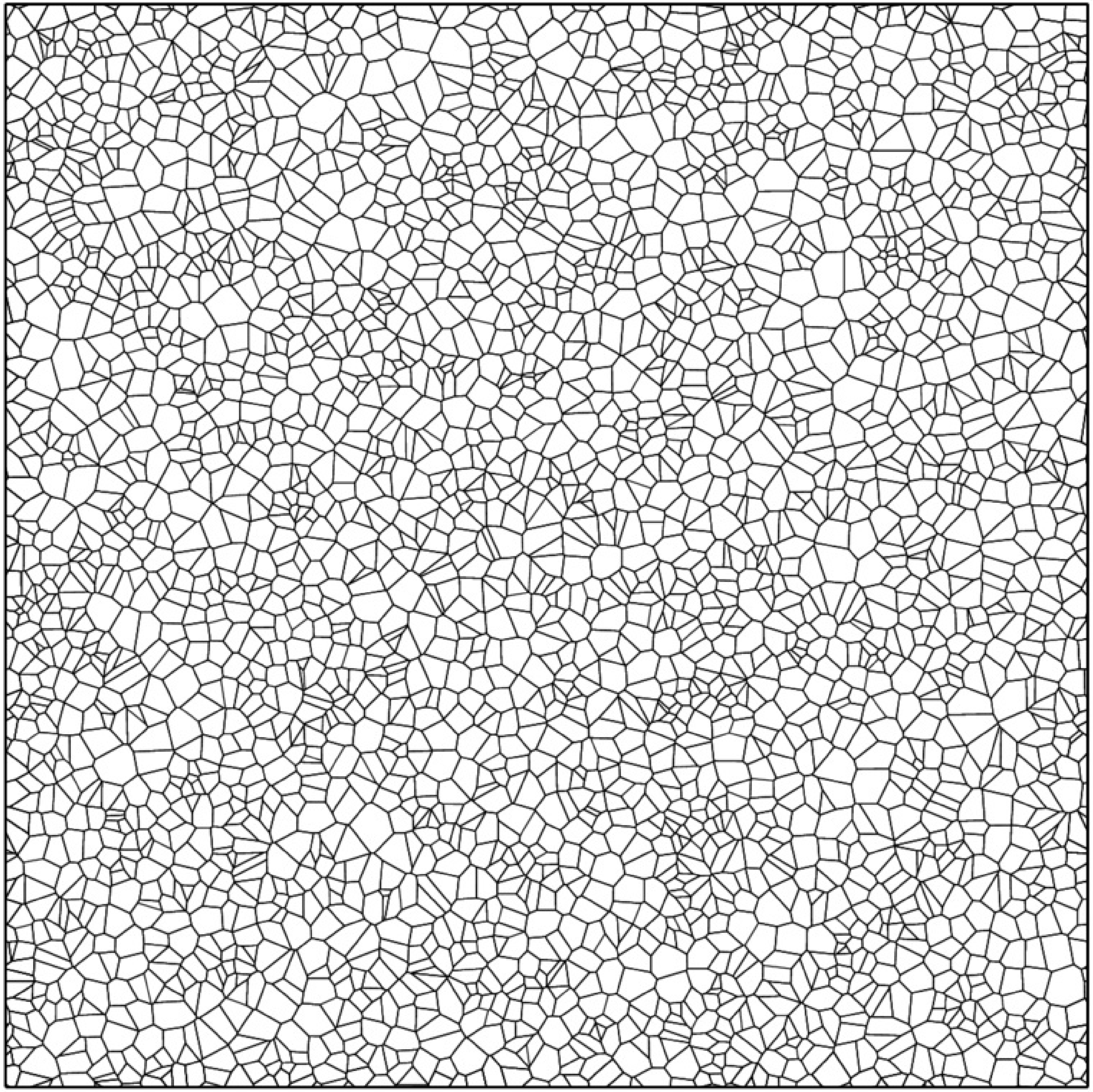
How to set the desired distributionof points?

$$A_i \equiv \frac{m_i}{\tilde{m}} + \frac{V_i}{\tilde{V}} \qquad A_i = \tilde{A} \quad \longrightarrow \quad n_0(\boldsymbol{x}) = \frac{1}{\tilde{A}}\left(\frac{\rho(\boldsymbol{x})}{\tilde{m}} + \frac{1}{\tilde{V}}\right)$$

At the end we have:

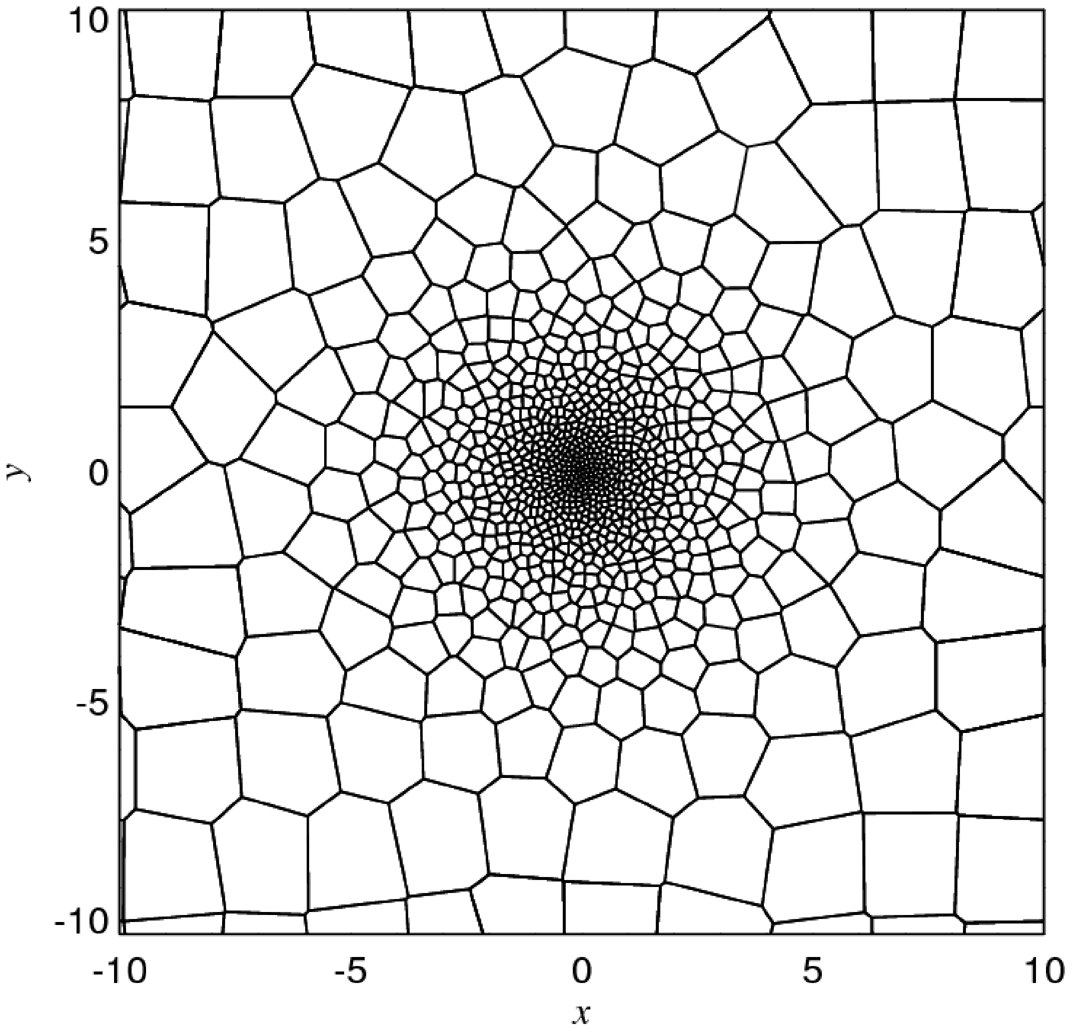
$$\nabla^2\Psi = \frac{\tilde{A}\, n(\boldsymbol{x})}{\rho(\boldsymbol{x})/\tilde{m} + 1/\tilde{V}} - 1 \qquad \tilde{A} = \frac{V_{\mathrm{tot}}}{\sum_i \left(\rho_i/\tilde{m} + 1/\tilde{V}\right)^{-1}}$$

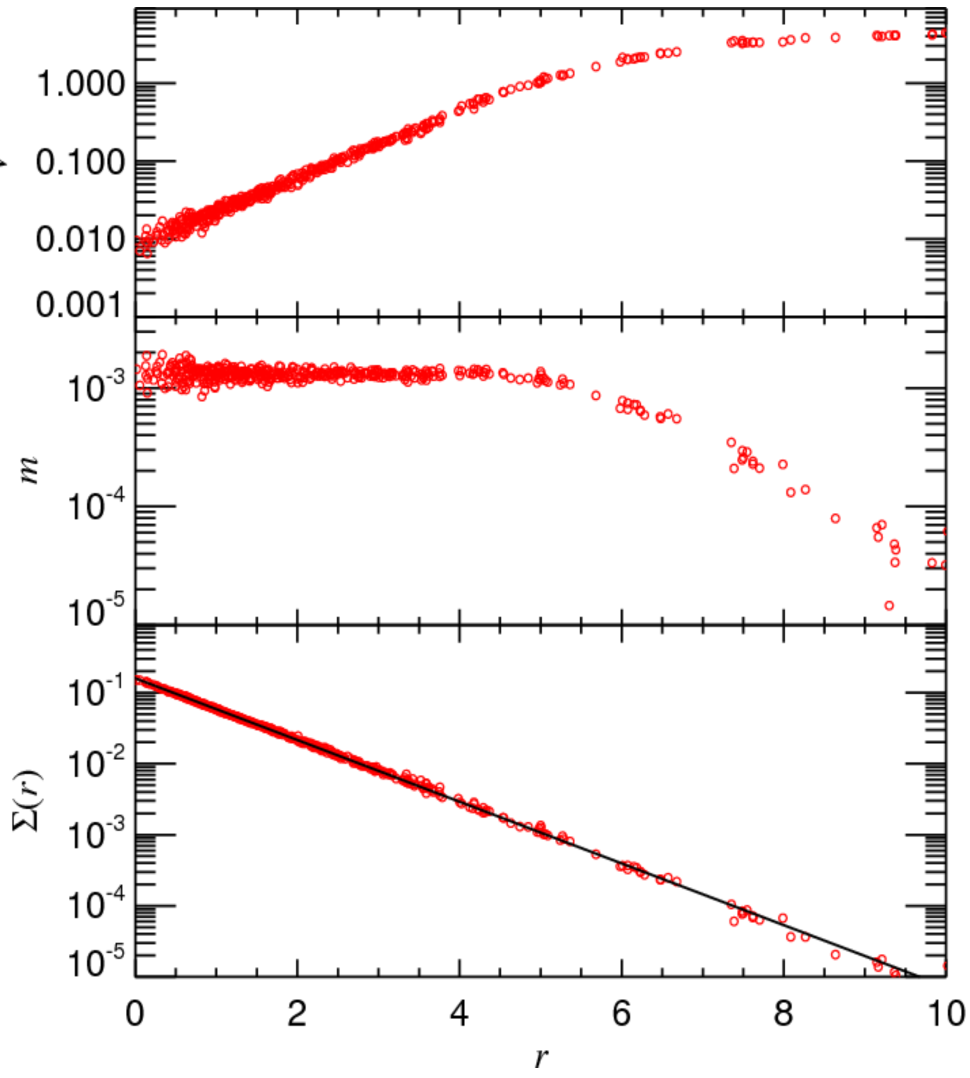
An example...



Our technique for mesh relaxation readily creates good initial conditions for arbitrary initial density fields

REGULARIZED INITIAL CONDITIONS FOR A POISSON REALIZATION OF AN EXPONENTIAL DISK



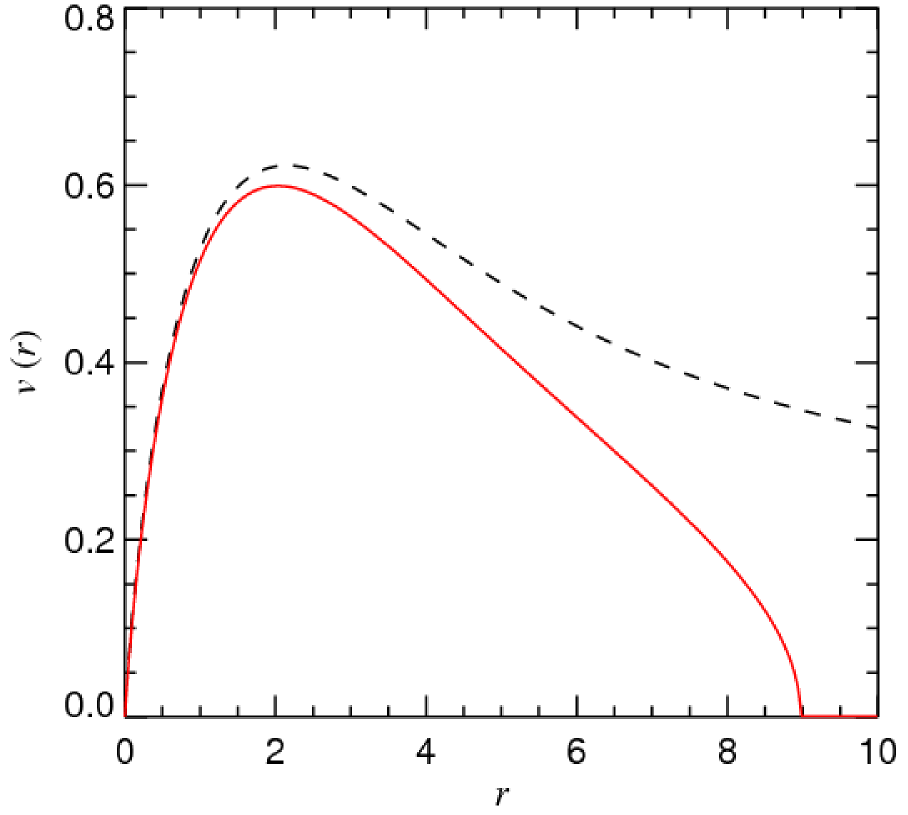


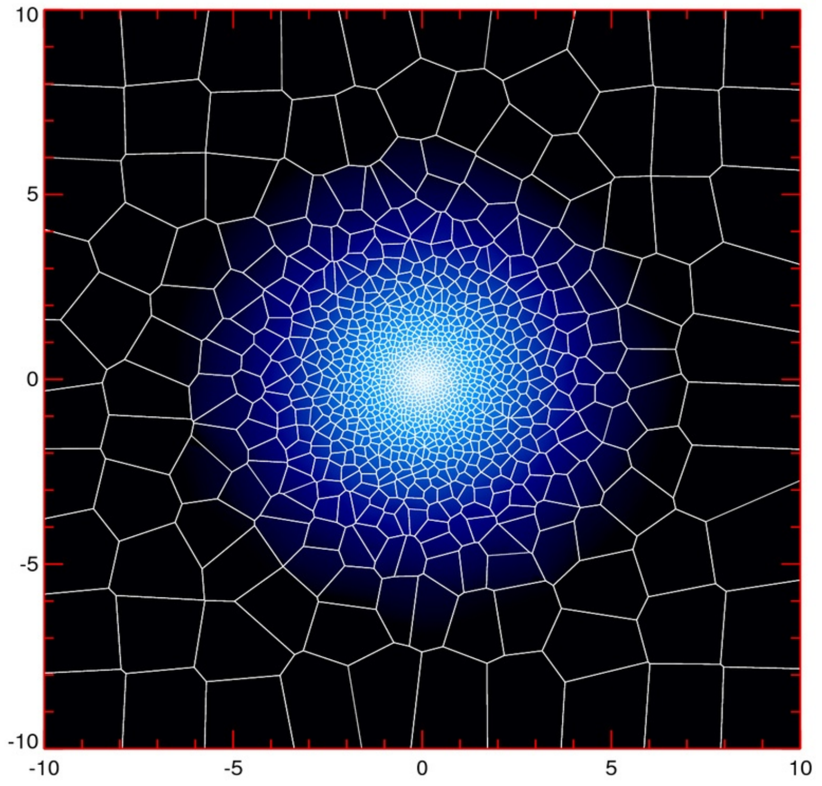
A differentially rotating gaseous disk with strong shear can be simulated well with the moving mesh code

MODEL FOR A CENTRIFUGALLY SUPPORTED, THIN DISK

$$\Sigma(r) = \Sigma_0 \exp(-r/h)$$

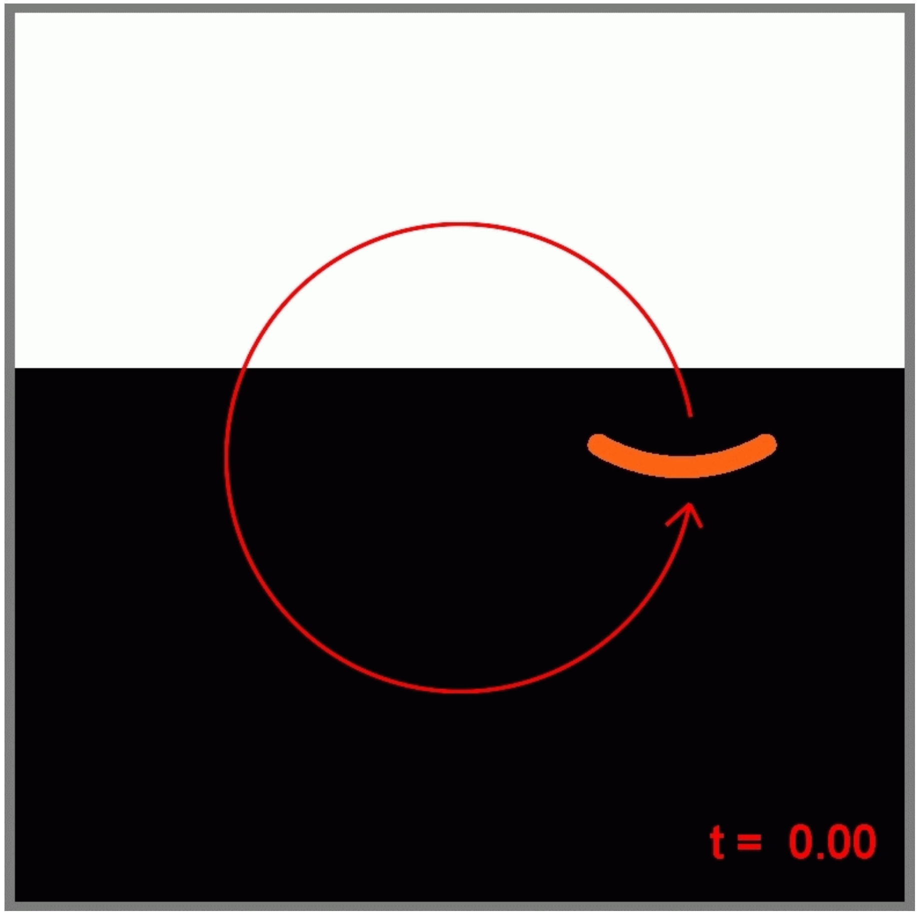
$$v_c^2(r) \equiv r\frac{\partial\Phi}{\partial r} = 2\frac{Gm}{h}y^2\left[I_0(y)K_0(y) - I_1(y)K_1(y)\right]$$

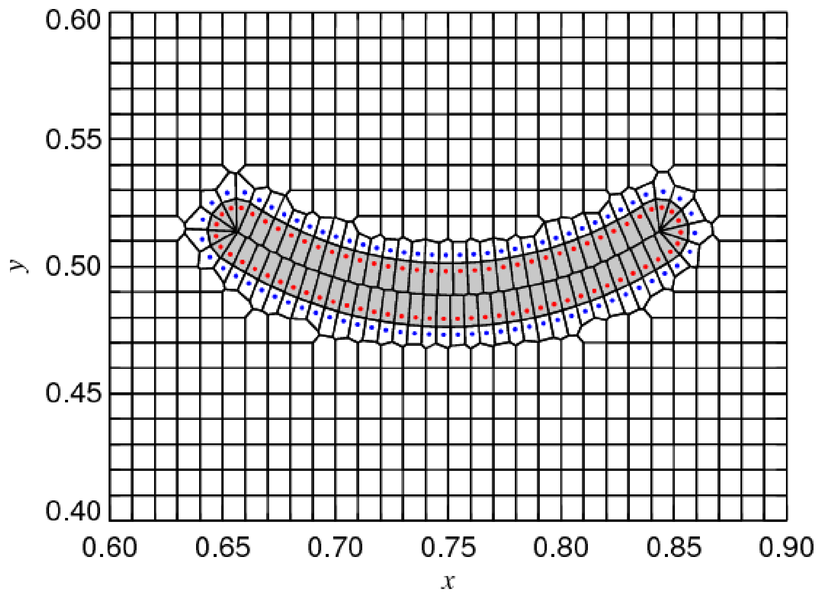


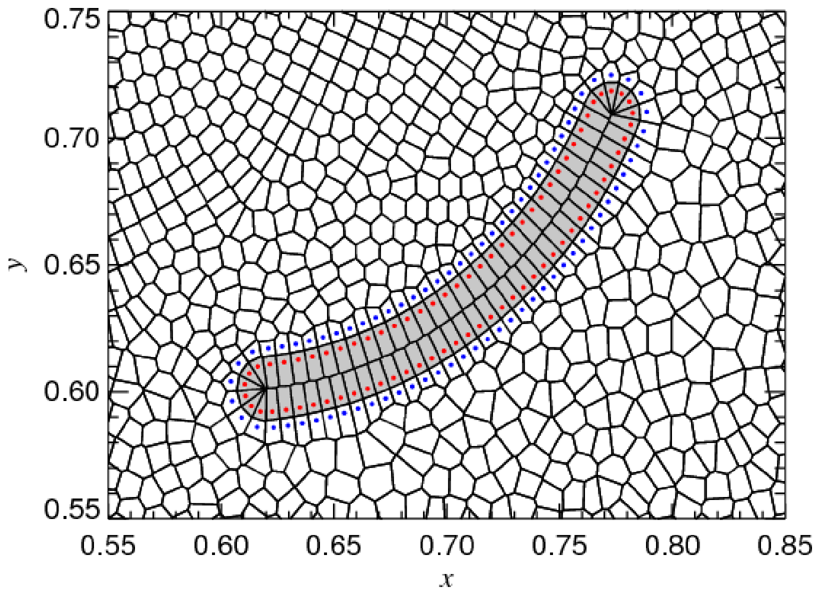


The moving-mesh approach can also be used to realize arbitrarily shaped, moving boundaries

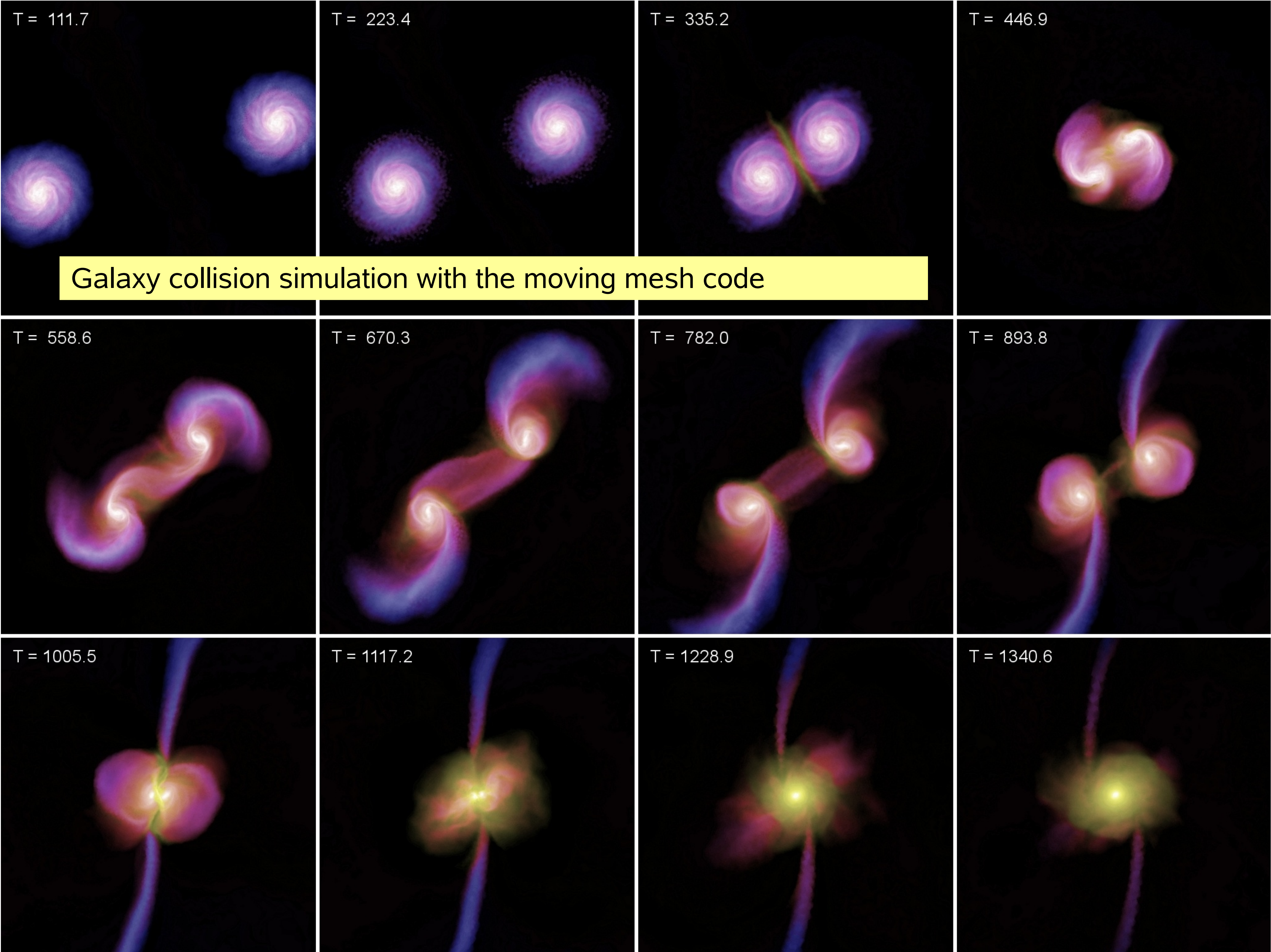
STIRRING A COFFEE MUG







Galaxy collision simulation with the moving mesh code

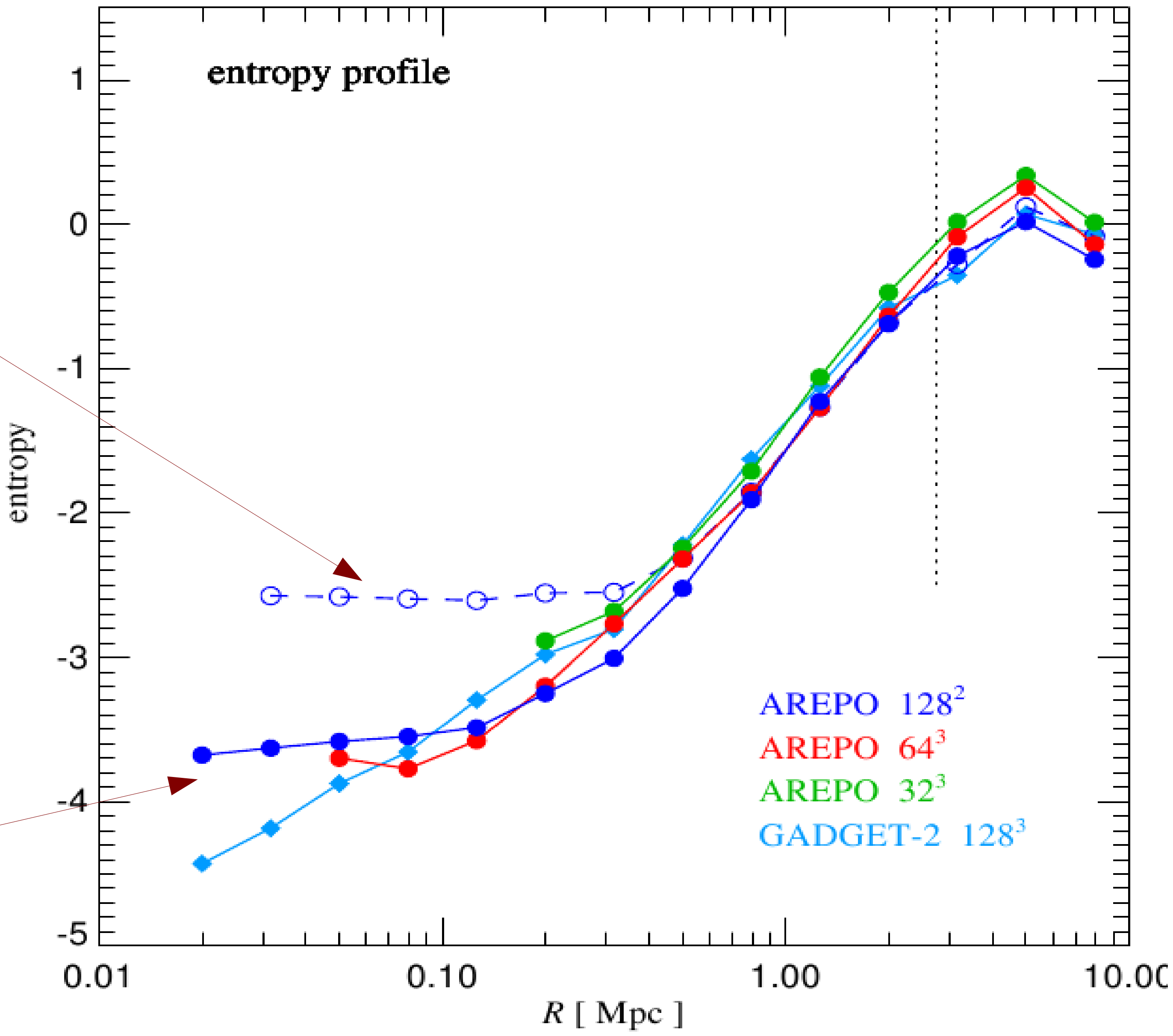


The Santa Barbara cluster develops a high entropy core with the moving mesh scheme, but this is affected by heating from N-body noise

ENTROPY PROFILE IN THE SANTA BARBARA CLUSTER FOR AREPO AND GADGET

plain solve of the total energy equation

Dissipation in very weak shocks suppressed



Conclusions

I described the novel quasi-Lagrangian hydrodynamical code AREPO. It has the following important features:

- Allows Galilean invariant solutions.
- No artificial viscosity, very low numerical diffusivity, and residual diffusivity is invariant in the presence of bulk flows, well behaved for supersonic flows.
- Automatic Lagrangian adaptivity of the mesh. Conveniently gives near ideal resolution improvements in regions that collapse
- Mesh geometry is very flexible, adjusts resolution to primary flow properties, no preferred directions along coordinate axes
- Code parallelized for distributed memory systems both in 2D and 3D, gravity adopted from Gadget-3 code.
- The code can also be used for fixed meshes, both of Cartesian and unstructured type.
- Moving and curved boundary conditions can be implemented comparatively easily.