

# Computational methods for elastic sheets I

2026 Boulder Summer School for Condensed Matter and Materials Physics  
June 6–10, 2026

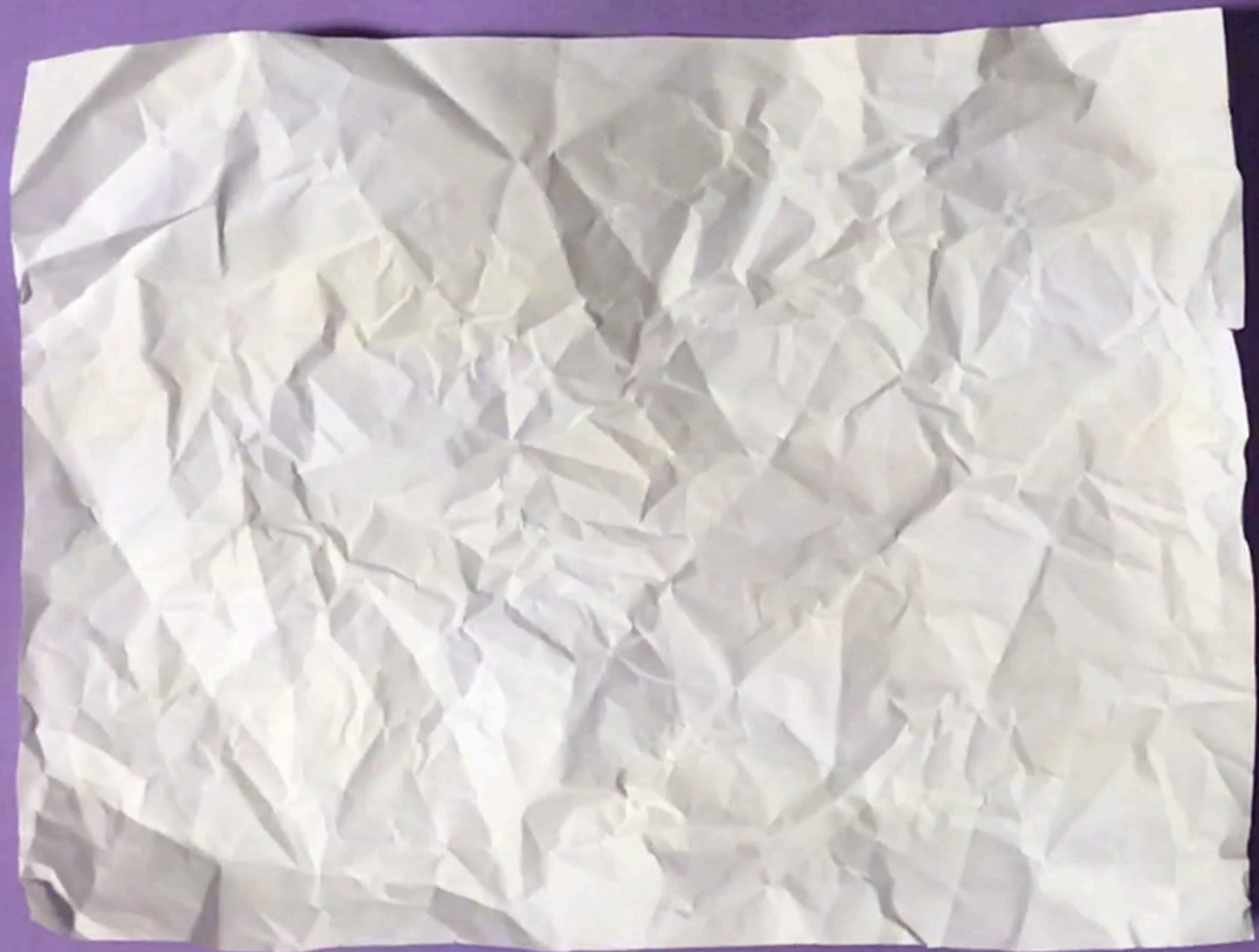
Chris H. Rycroft, University of Wisconsin–Madison  
([rycroft@wisc.edu](mailto:rycroft@wisc.edu))



Motivating example: a  
crumpled thin sheet



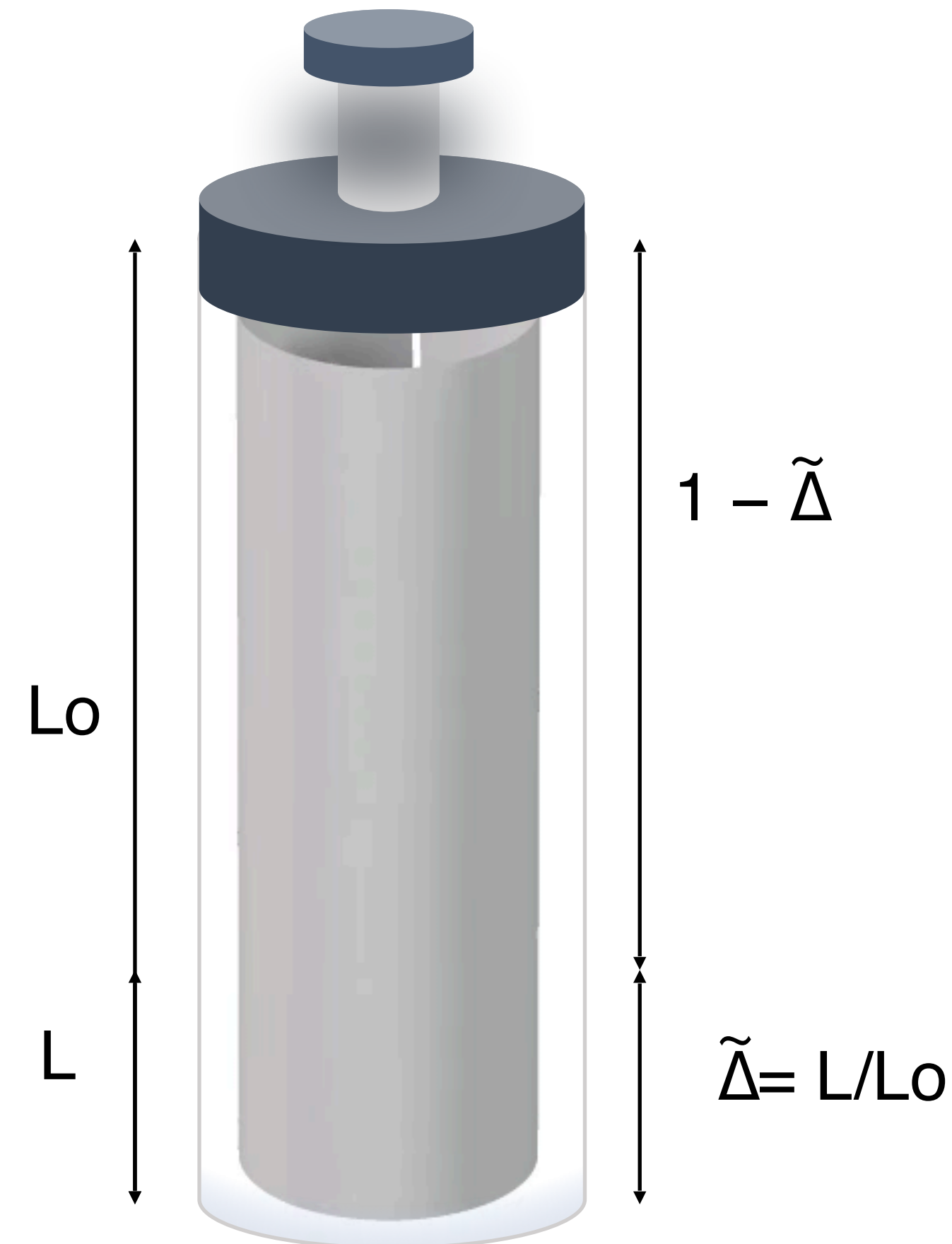




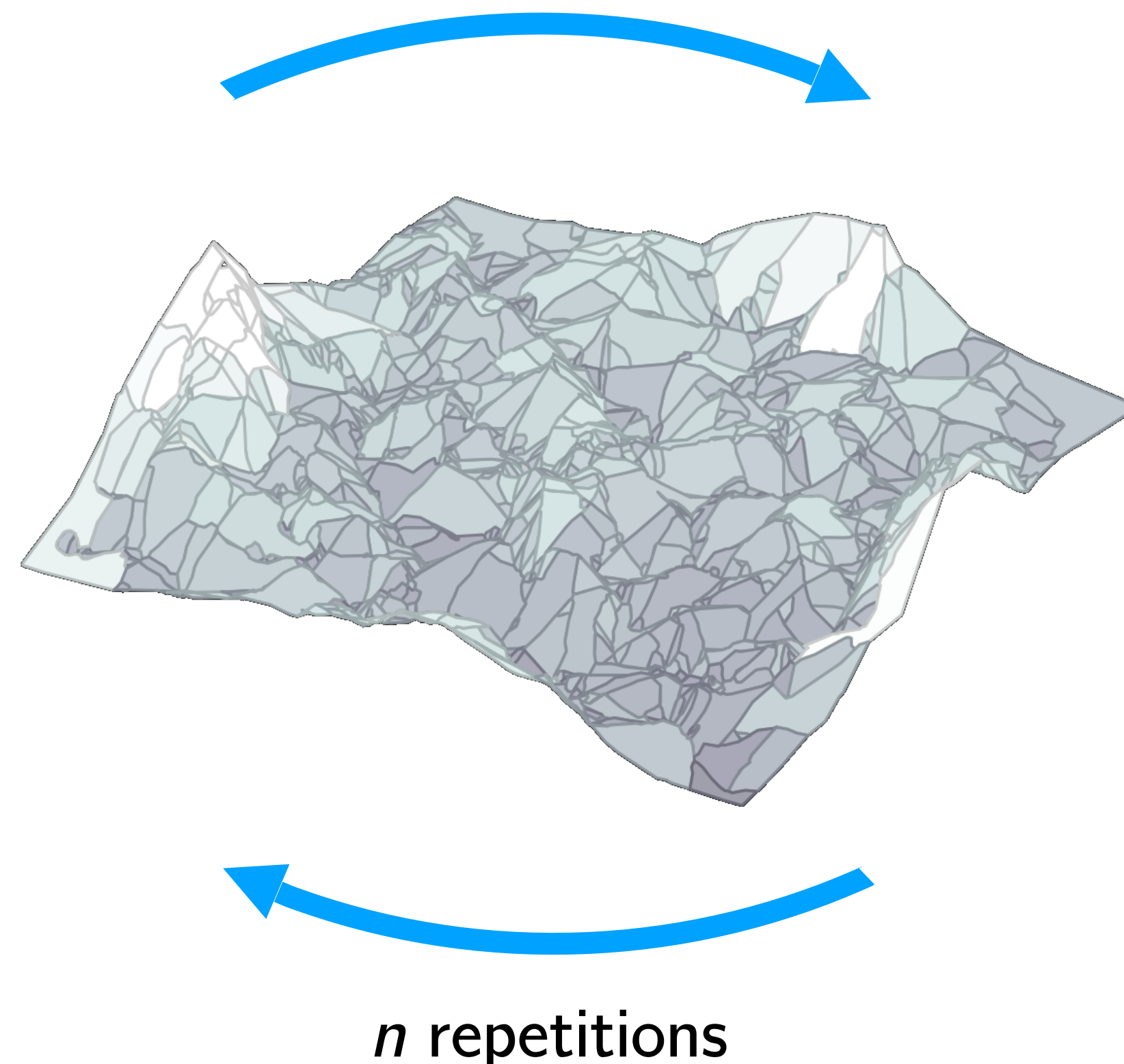


# Experimental procedure

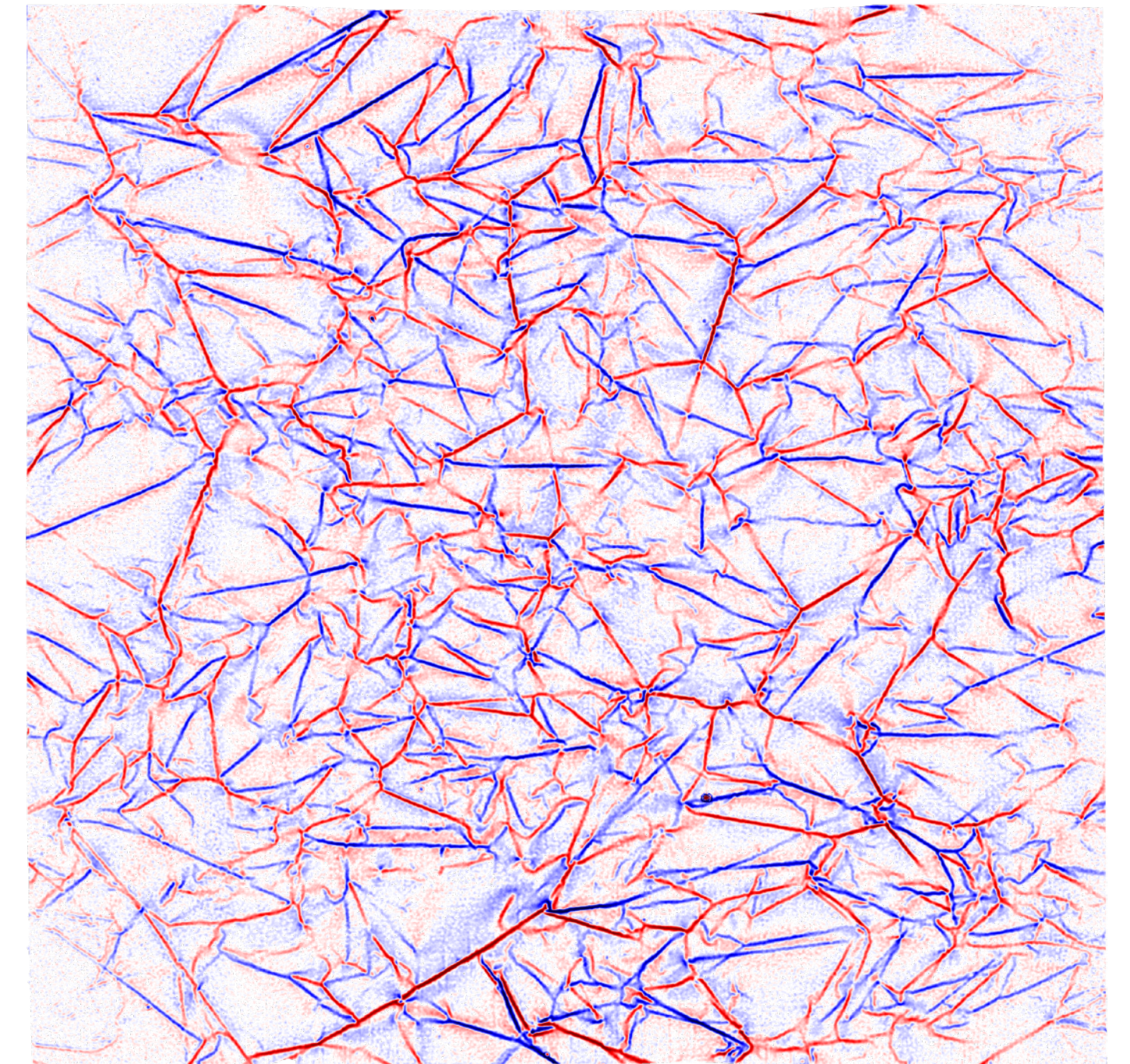
An  $L_0 \times L_0$  Mylar sheet is compressed uniaxially to a compaction ratio  $\tilde{\Delta}$



The local curvature is calculated from the sheet's topography, producing a map of crease locations



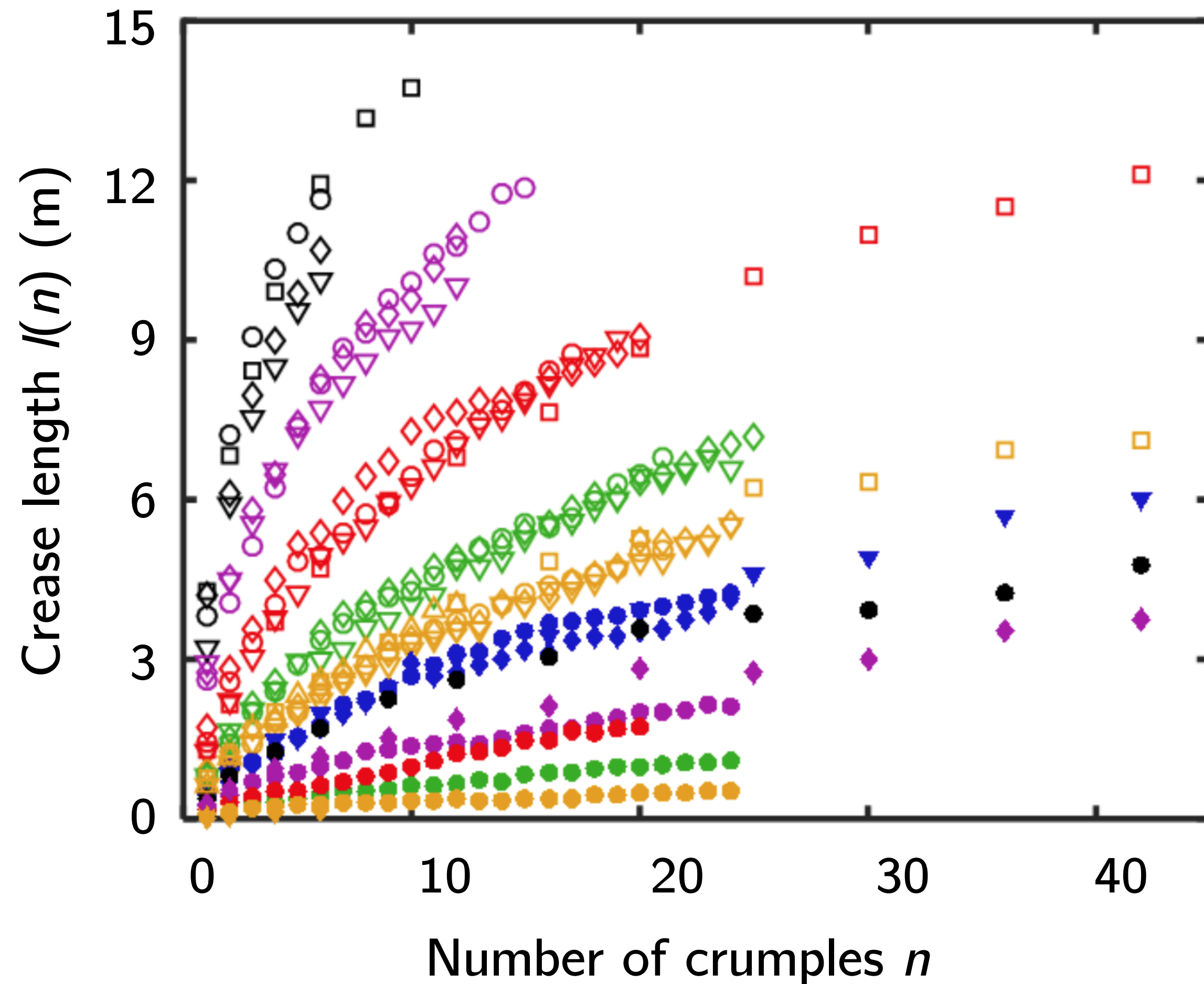
Creases are identified by looking at areas of **positive** and **negative** mean curvature



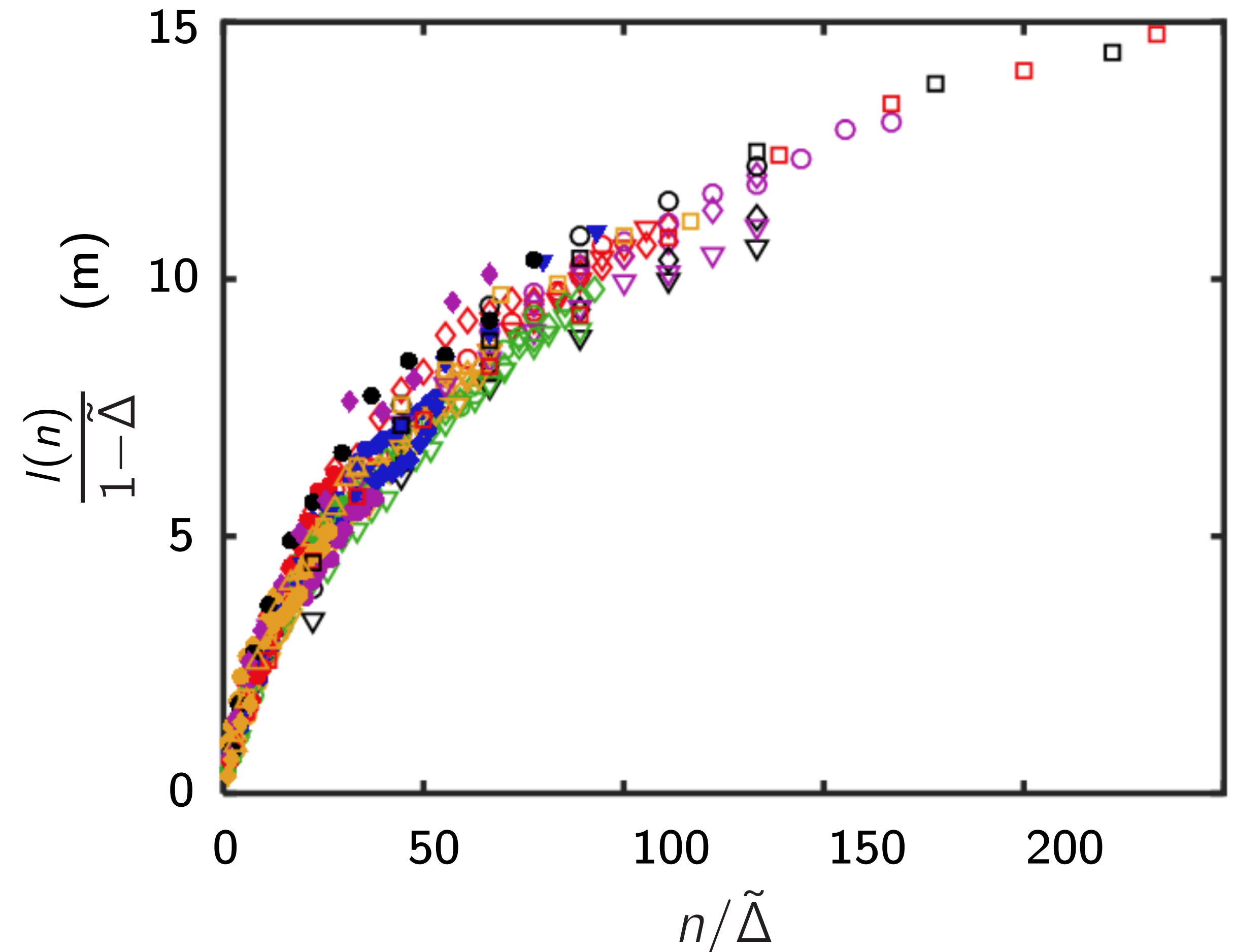
Aim to look at total crease length  $l(n)$  in terms of the repetitions  $n$



# Evolution of crease length



Colors show experiments with different  $\tilde{\Delta}$

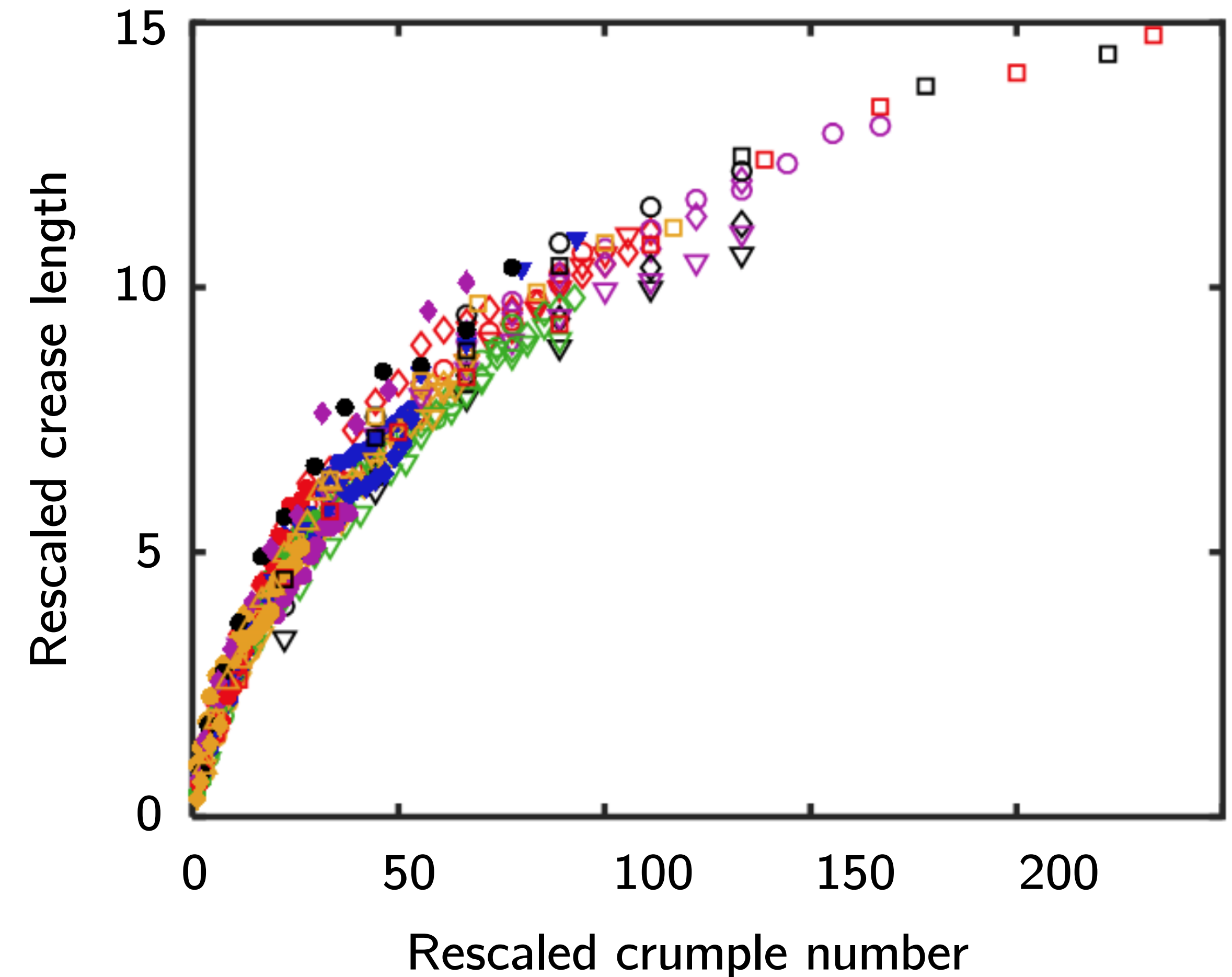
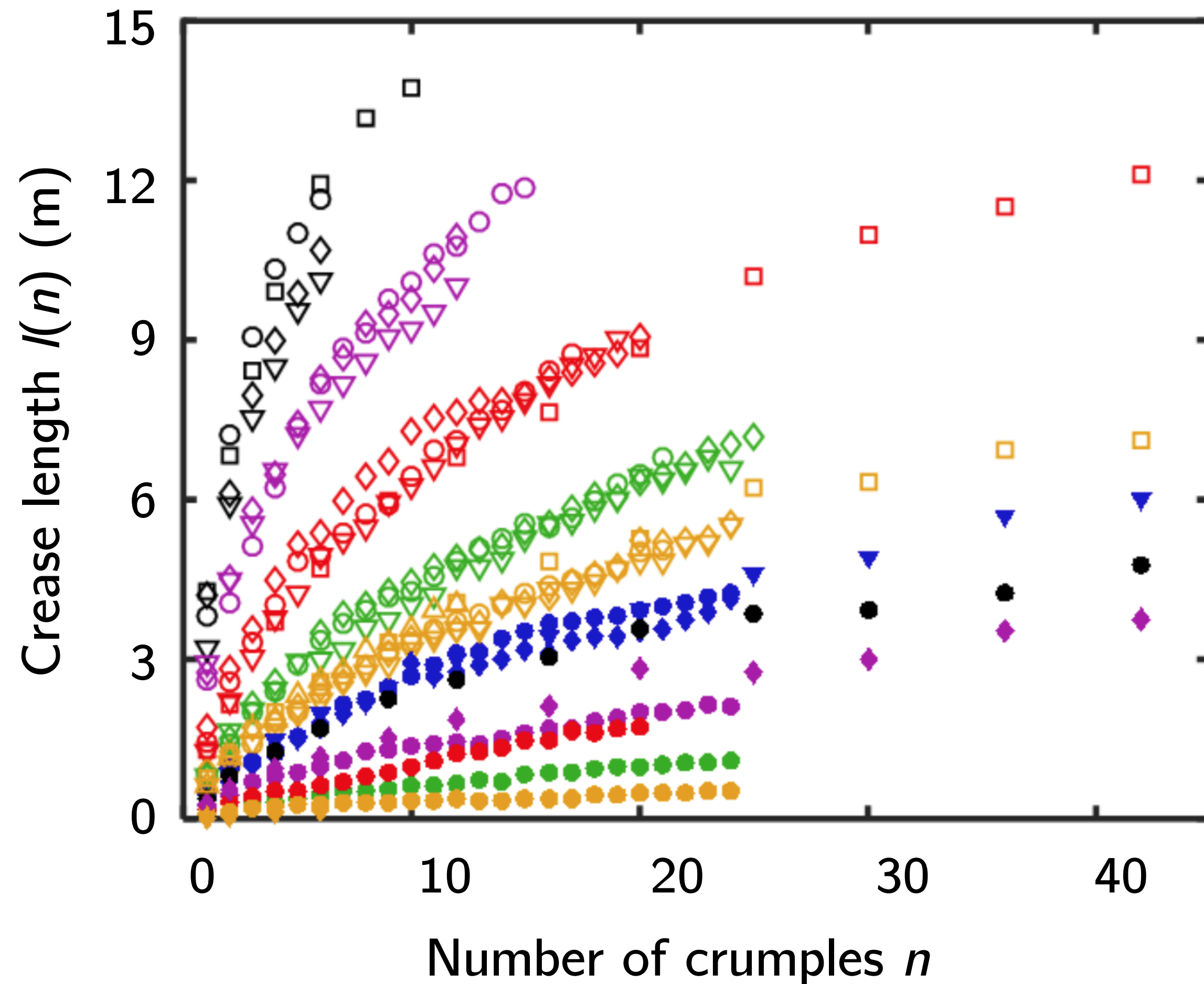


Universal logarithmic scaling

$$l(n) = c_1(1 - \tilde{\Delta}) \log \left( 1 + \frac{c_2 n}{\tilde{\Delta}} \right)$$



# Evolution of crease length



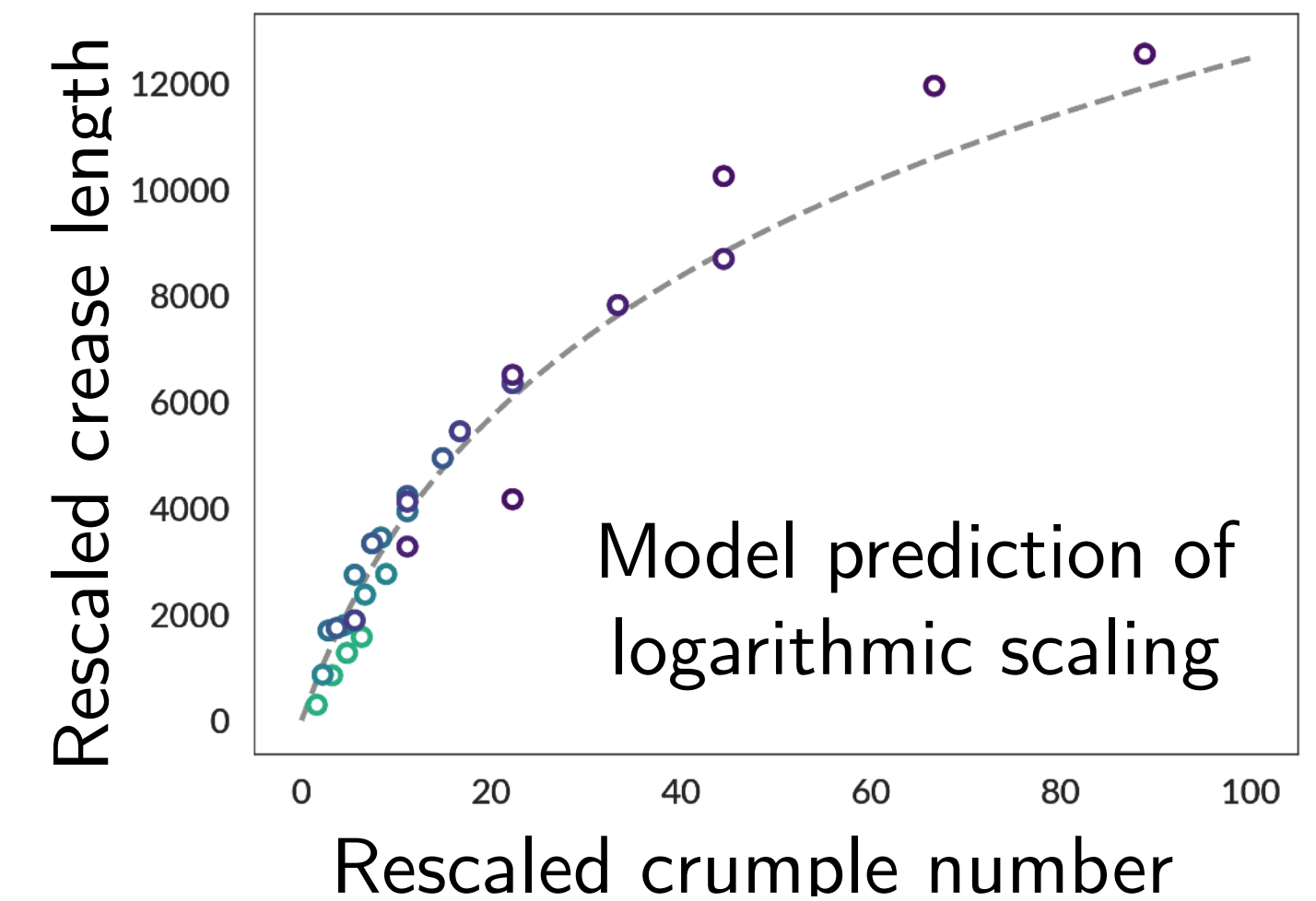
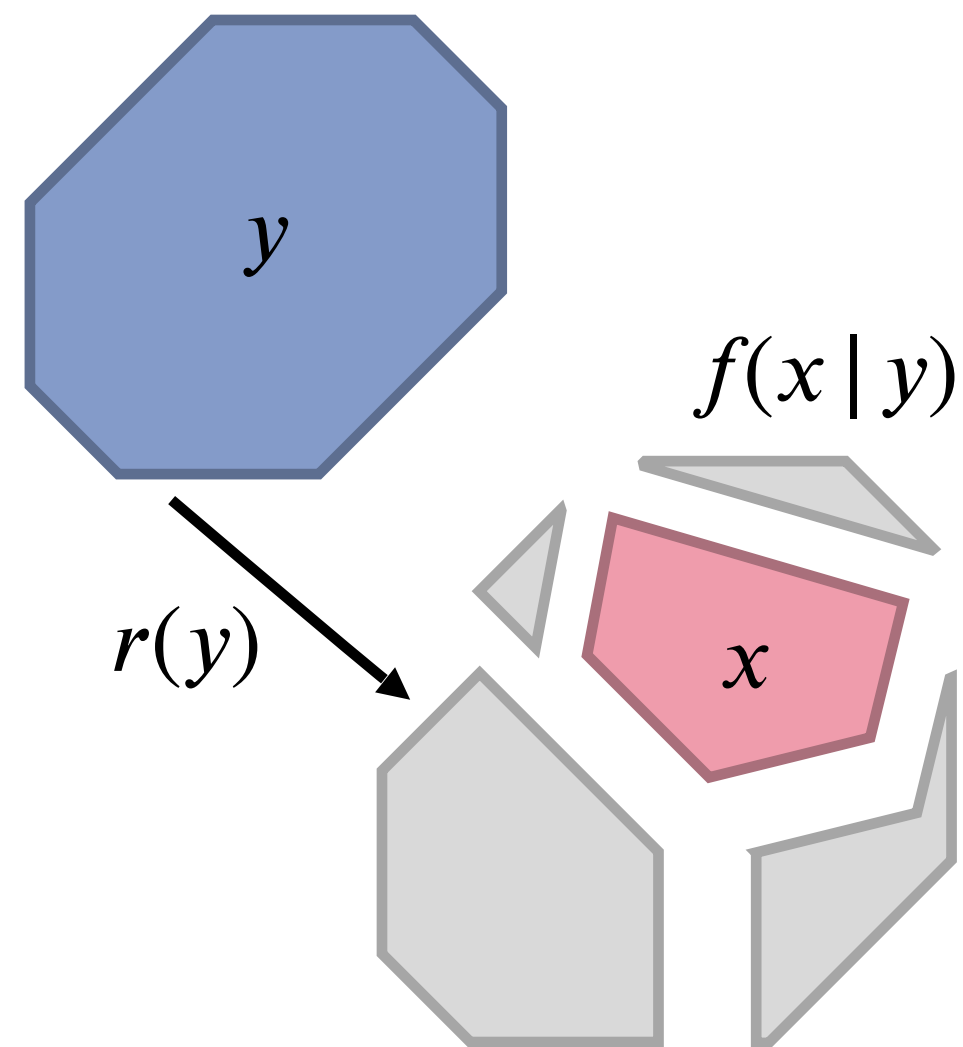
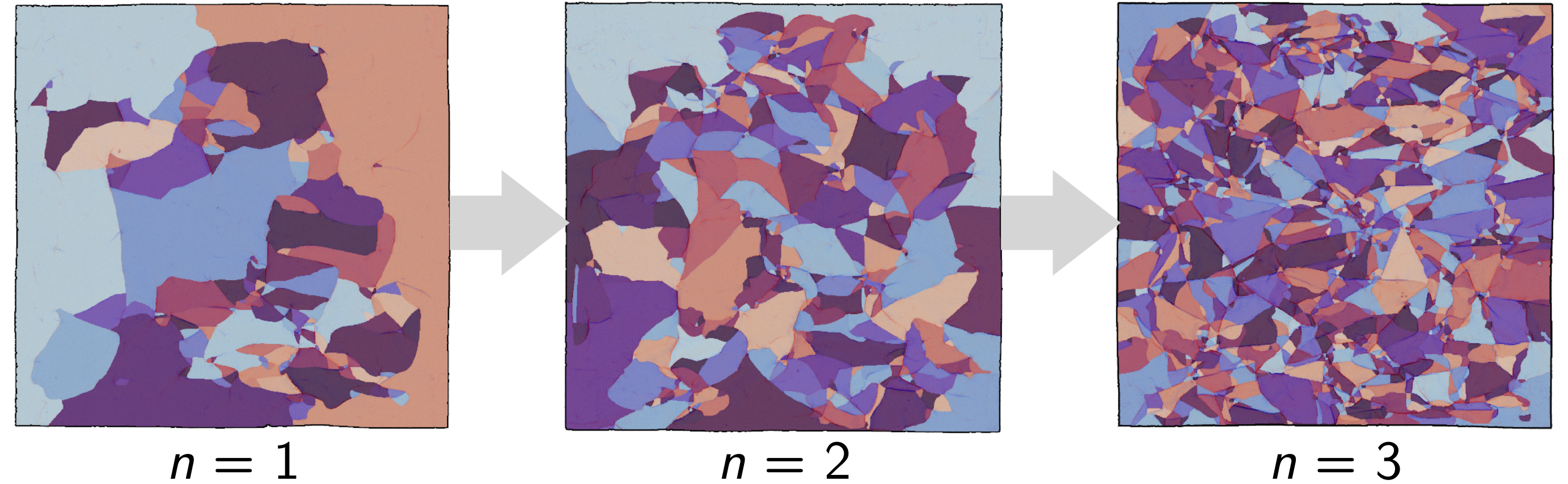
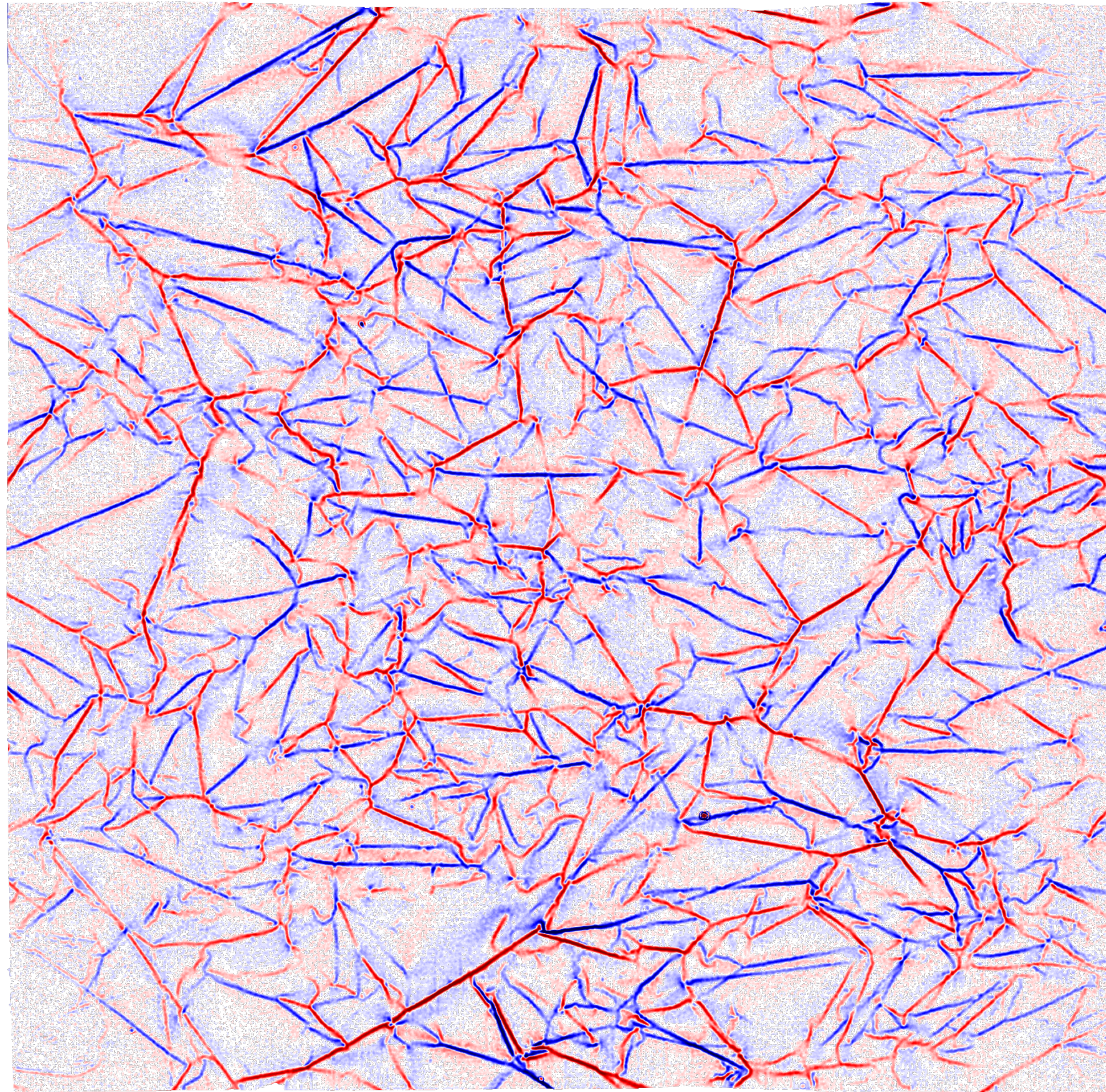
Colors show experiments with different  $\tilde{\Delta}$

Universal logarithmic scaling

$$l(n) = c_1(1 - \tilde{\Delta}) \log \left( 1 + \frac{c_2 n}{\tilde{\Delta}} \right)$$



# A fragmentation model of crumpling

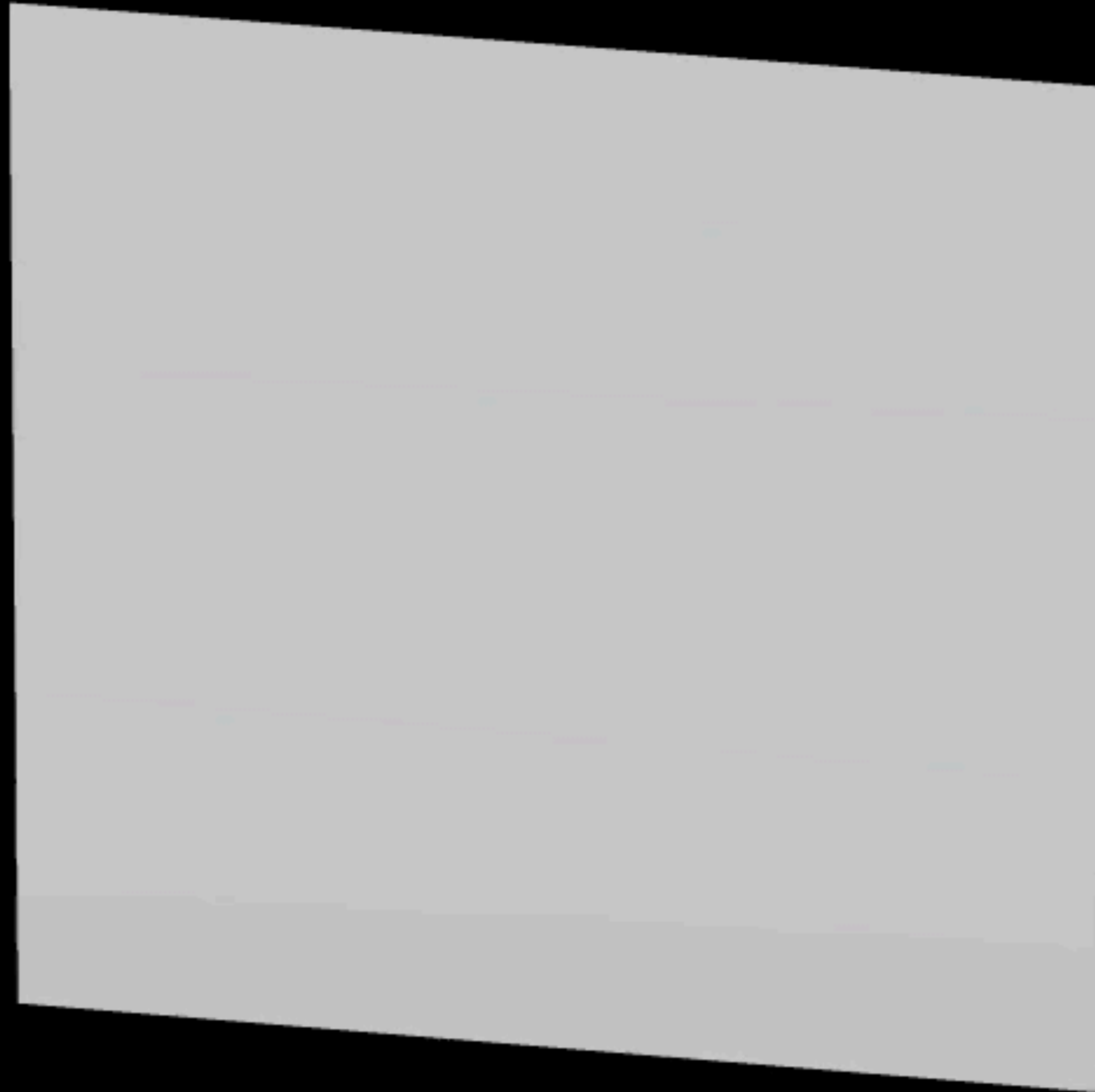


**Experimental data collection was time-consuming!**



# Physical simulation of axial crumpling

(first crumple)



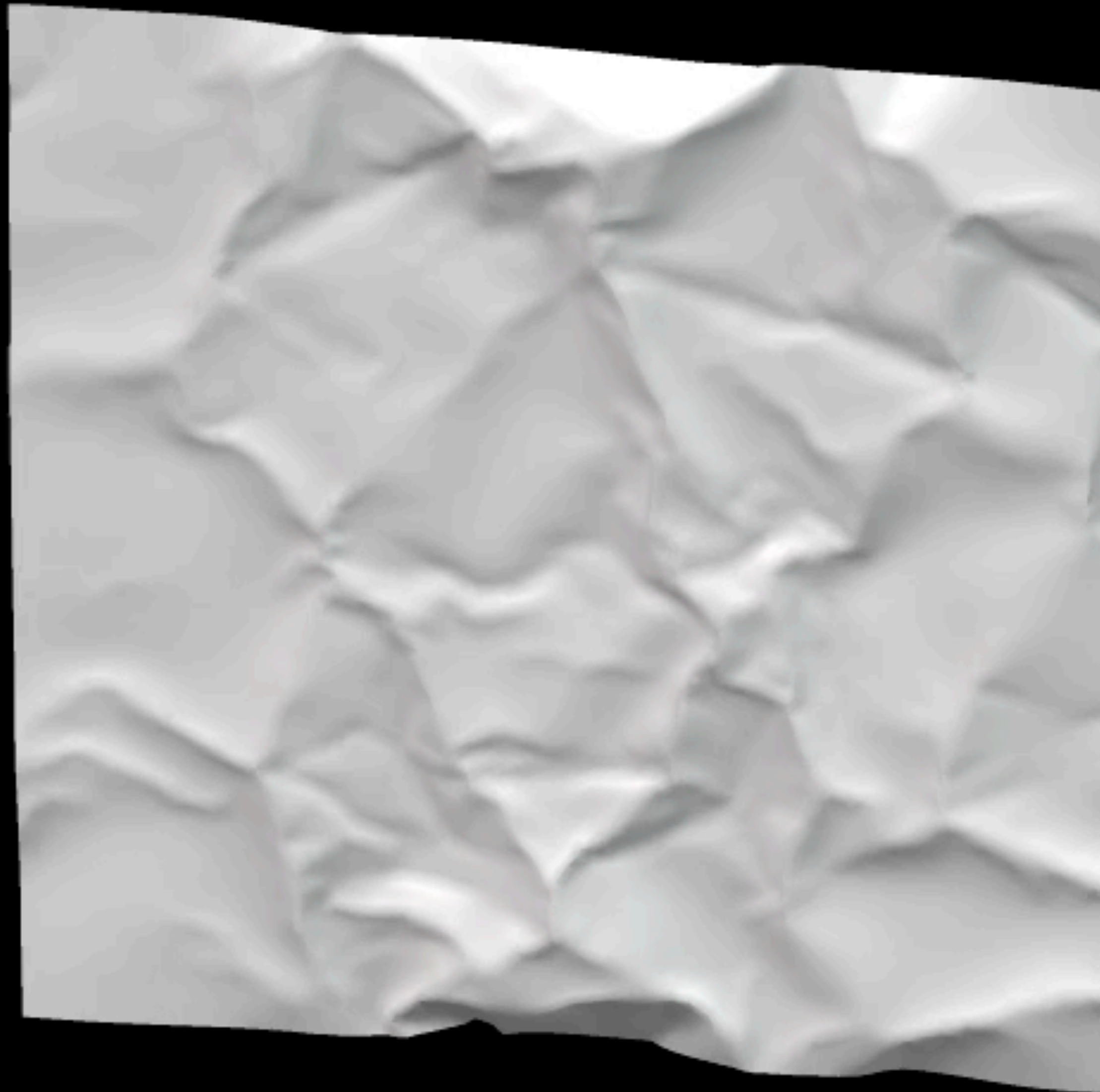
11837 grid points  
23268 triangles  
Compute time: ~5 days on 16 threads

10 cm  $\times$  10 cm sheet  
0.1 mm thick  
Young's modulus 0.5 GPa  
Bending rigidity  $4.7 \times 10^{-5}$  N m  
Physical time per crumple: 3.7 s



# Physical simulation of axial crumpling

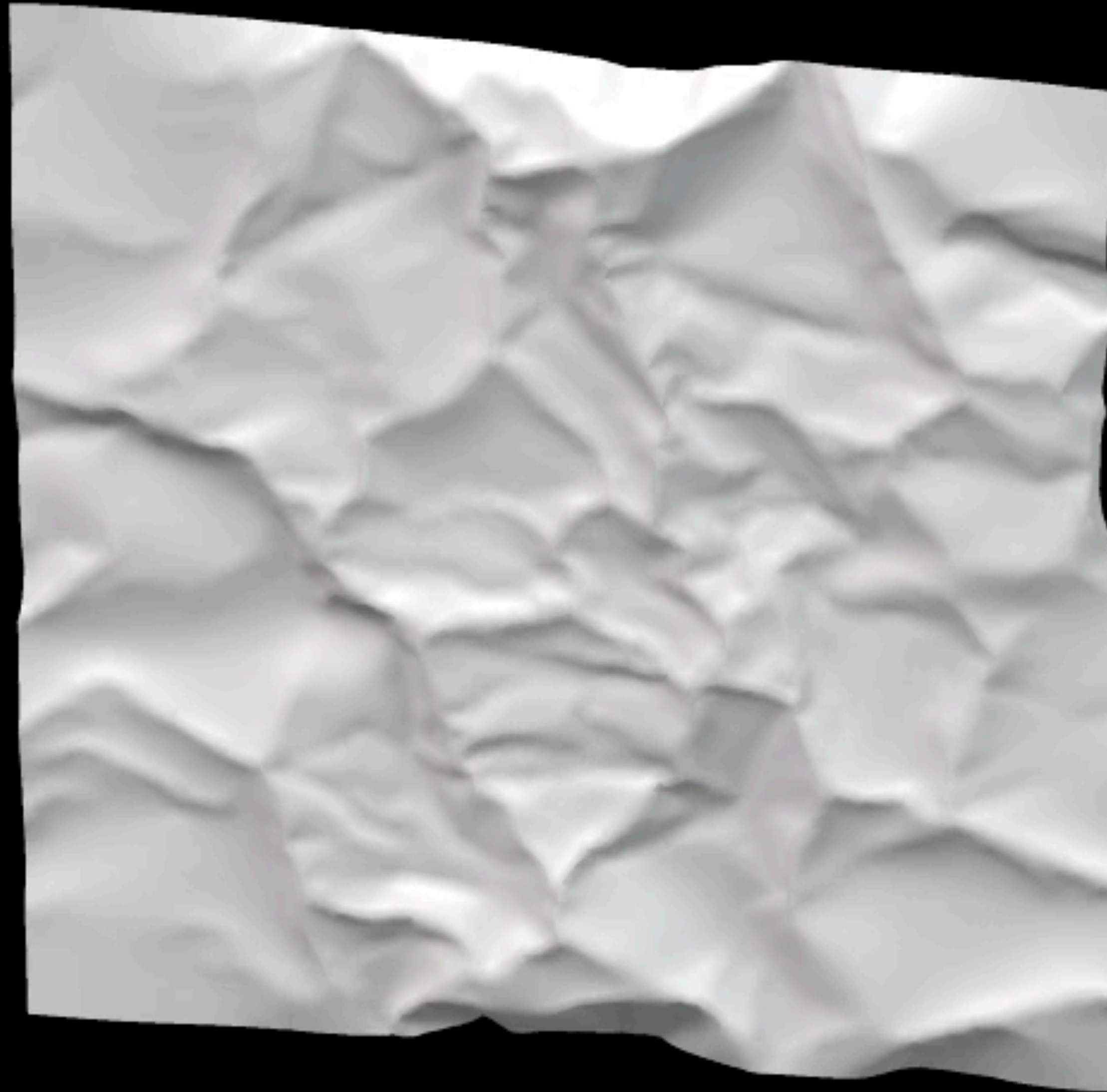
(second crumple)





# Physical simulation of axial crumpling

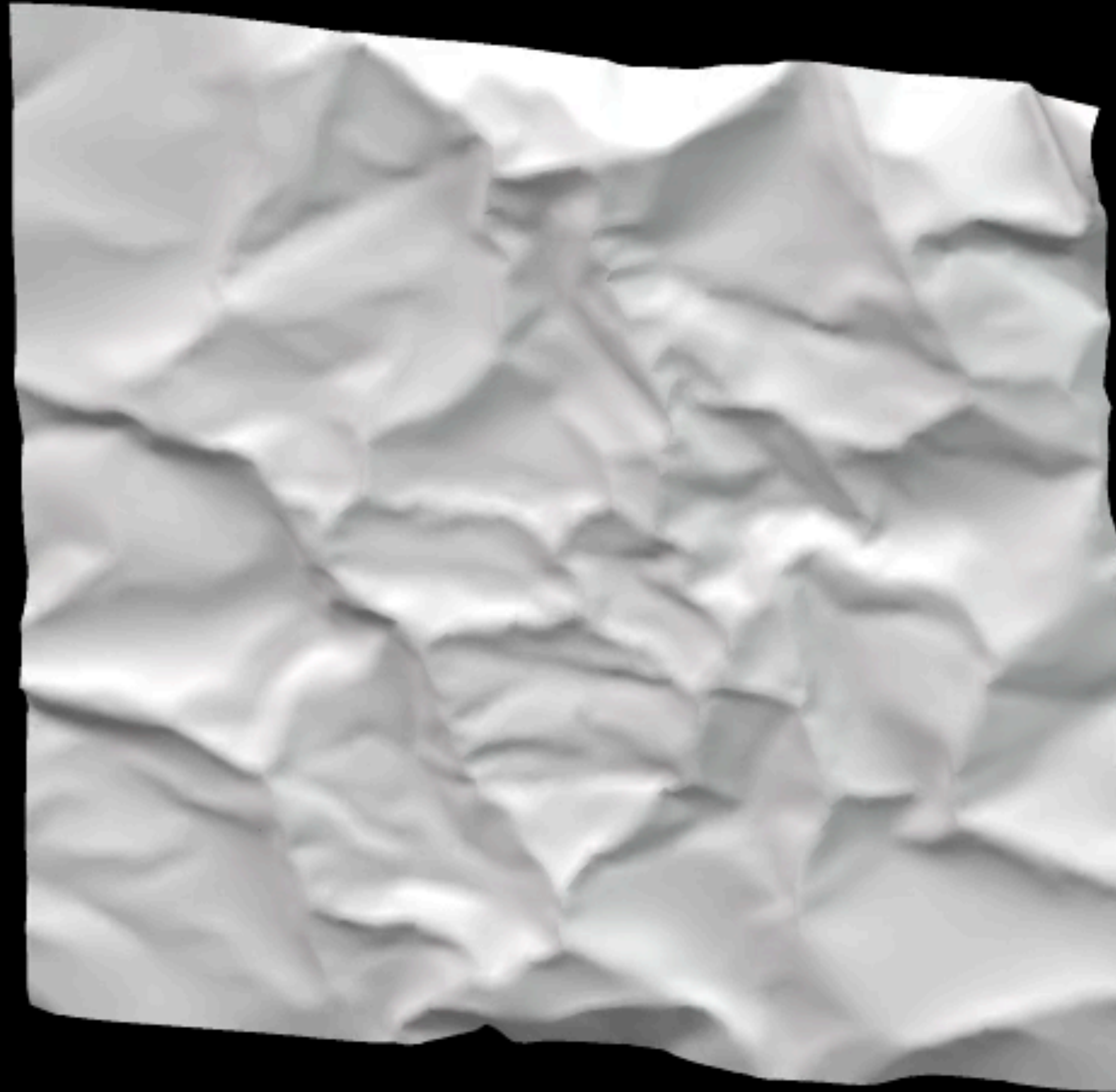
(third crumple)





# Physical simulation of axial crumpling

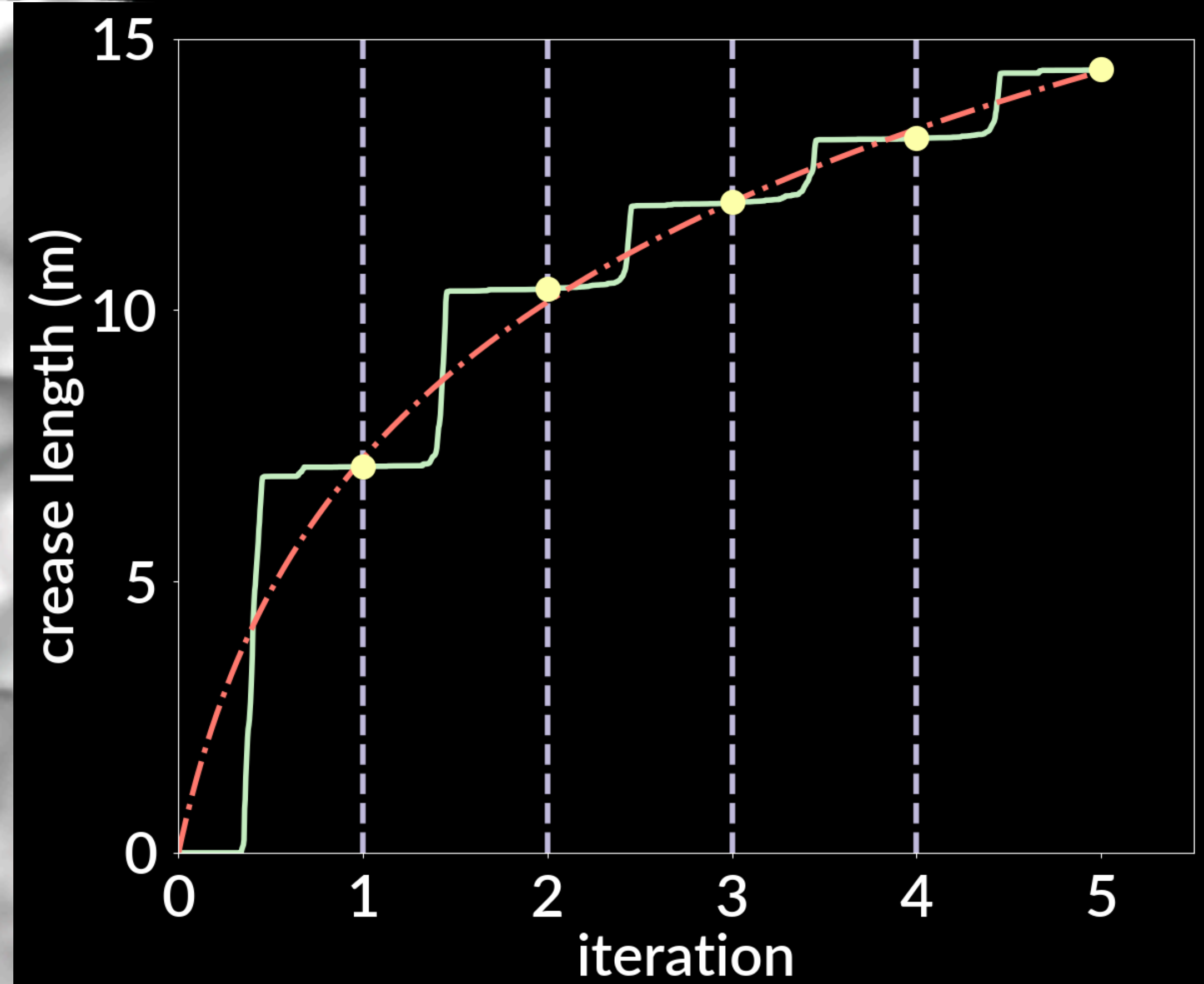
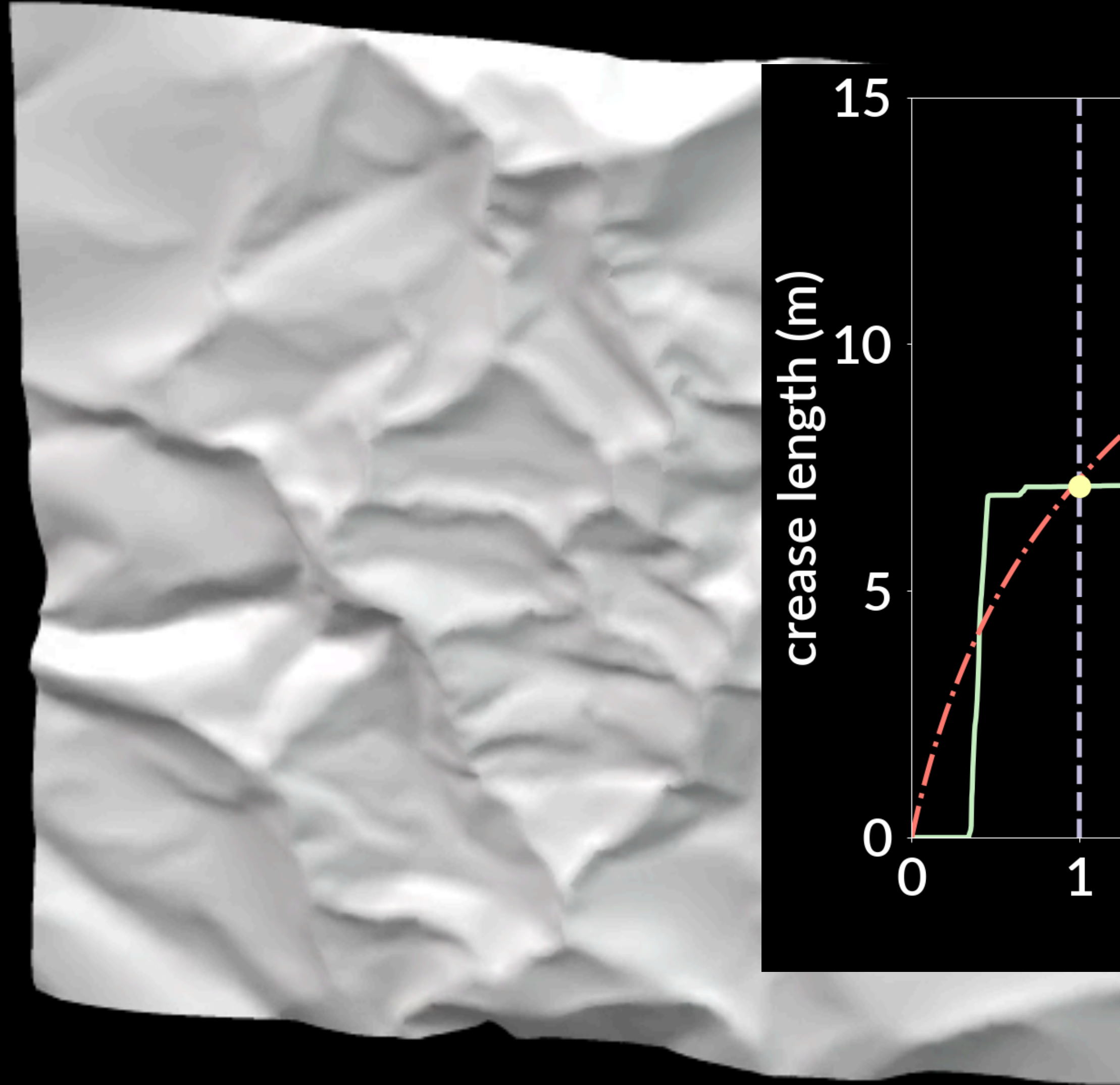
(fourth crumple)





# Physical simulation of axial crumpling

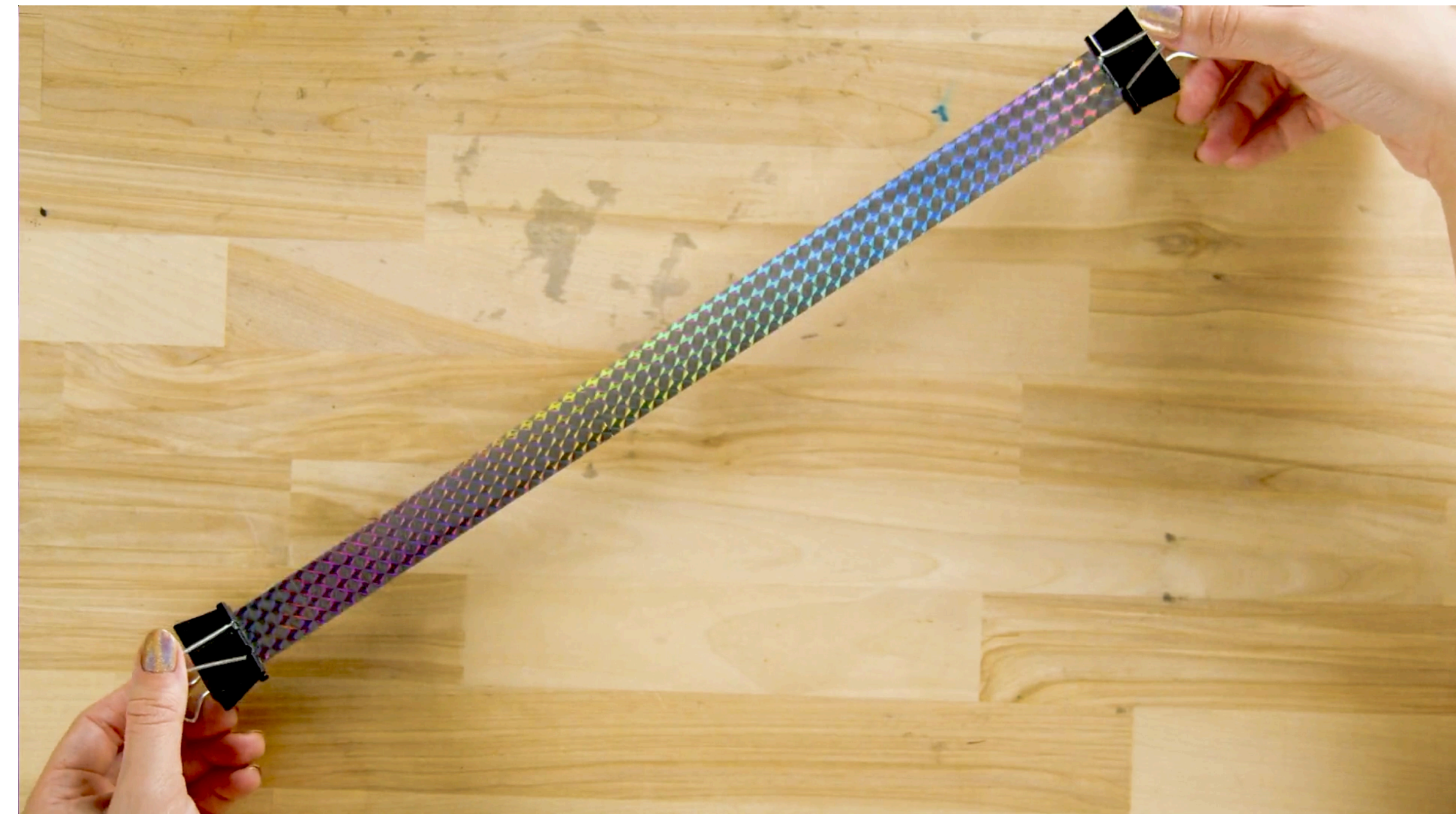
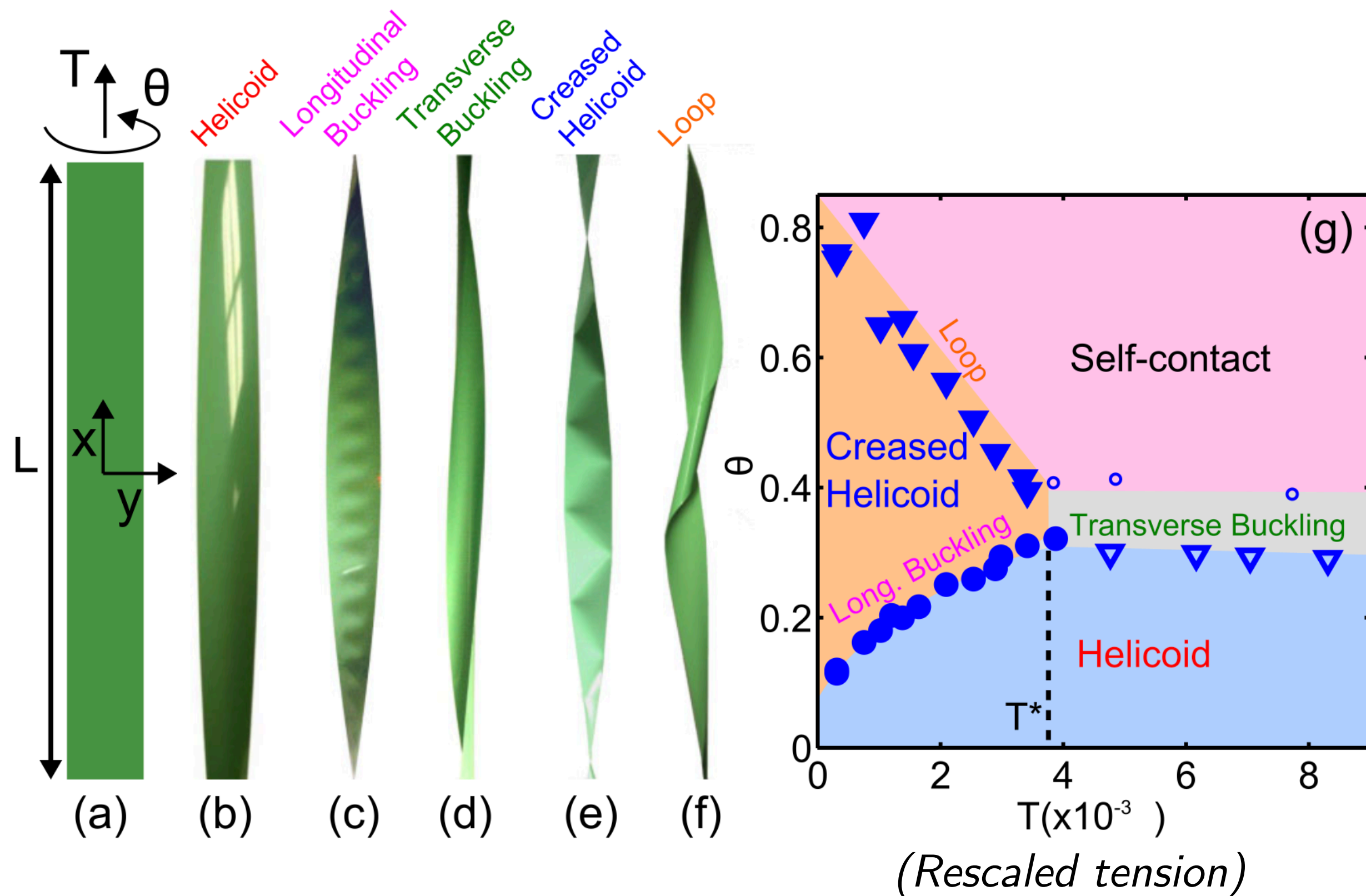
(fifth crumple)





# Twisted ribbons

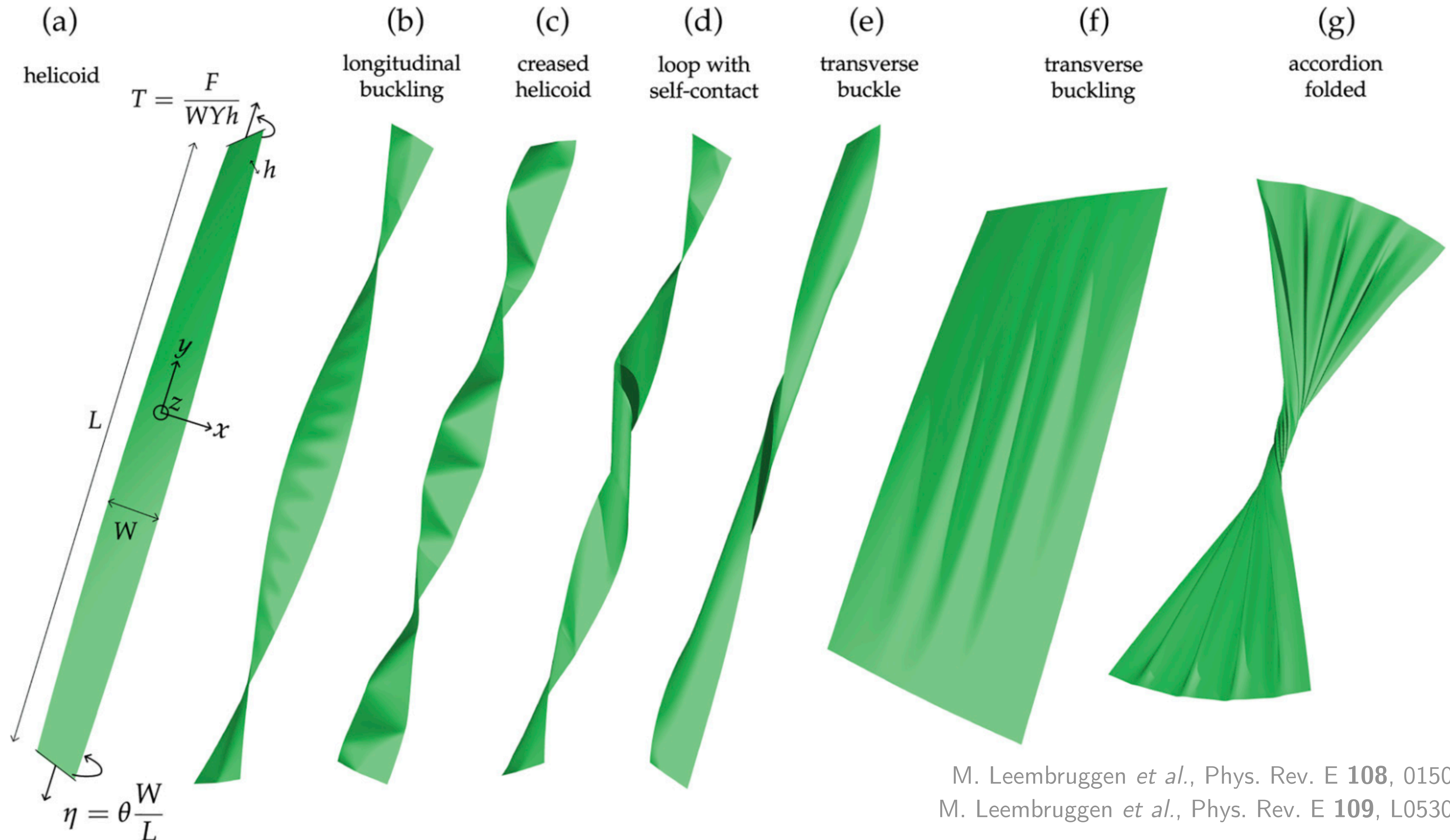
- A simple twisted ribbon has a surprising number of different behaviors, depending on applied twist and tension



Video by Madelyn Leembruggen



# All behaviors can be recreated in simulation





# Outline

*(Italic: Mainly theory)*

(Non-italic: mainly computational)

## Lecture I

(Monday)

## Lecture II

(Tuesday)

## Lecture III

(Friday)

1. *A mass-spring model of a thin sheet*
2. Constructing a triangular mesh:
  - a) The Voronoi tessellation
  - b) Delaunay triangulation
  - c) Lloyd's algorithm and mesh generation
3. *Time-integration of the sheet model*
4. Self-contact
5. Simulating wrinkling with the sheet model
6. Further aspects (*quasi-static simulation, plasticity, numerical linear algebra*)



**Blackboard: mass-spring model of a thin sheet**

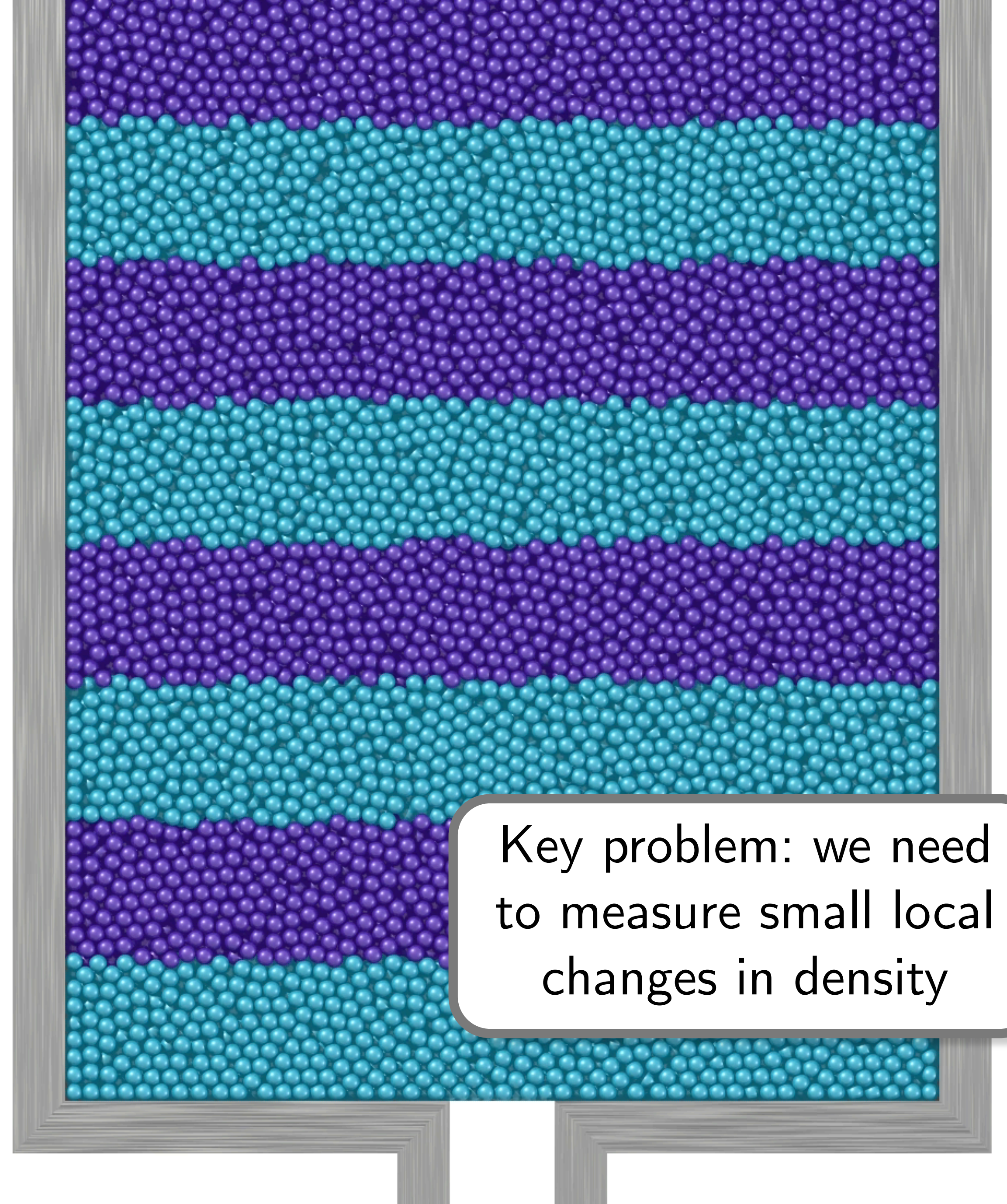


# Motivating example: density changes in dense granular drainage

Consider a dense granular sphere packing flowing out of an orifice by gravity, simulated with the discrete-element method (DEM)

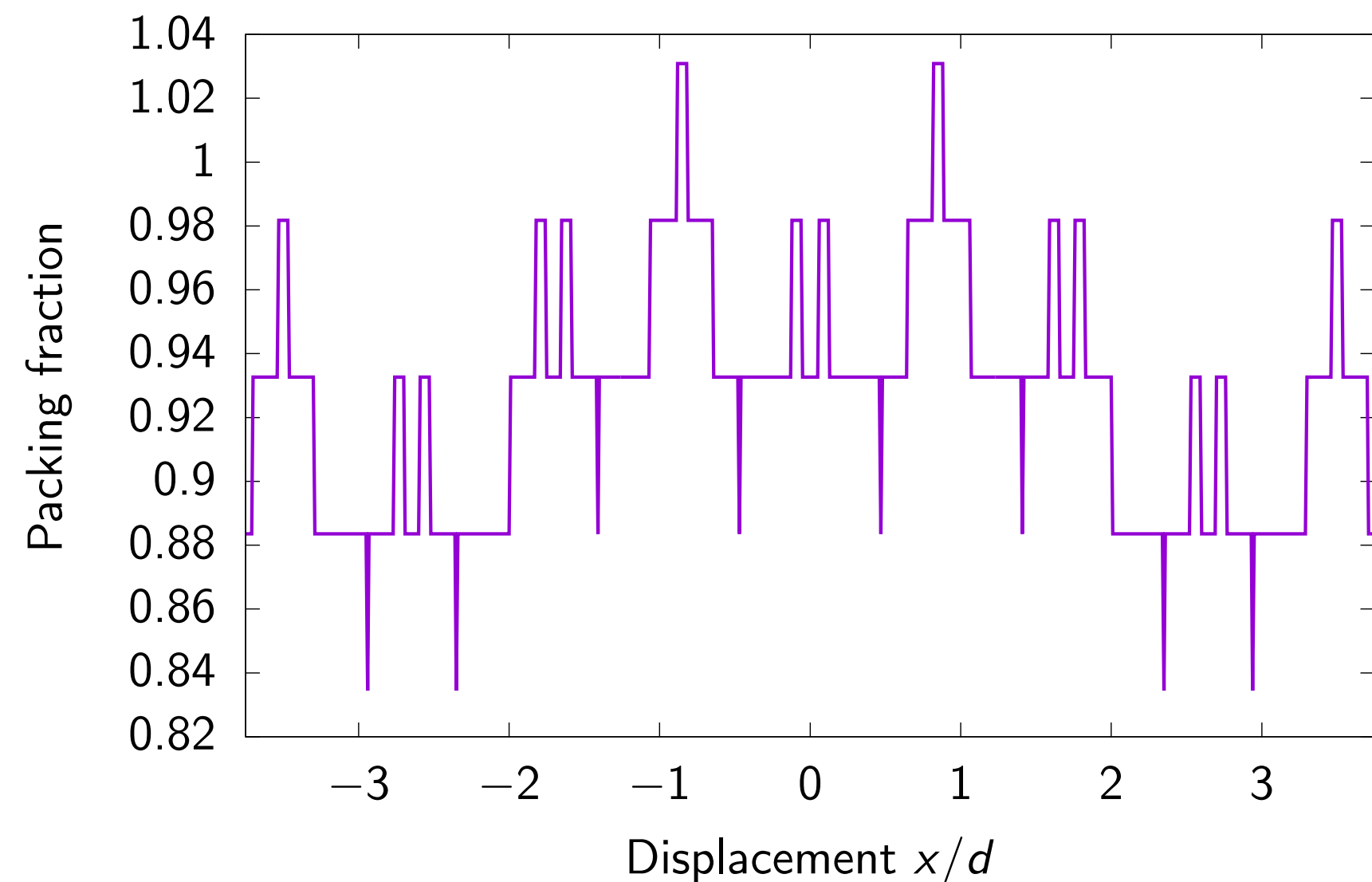
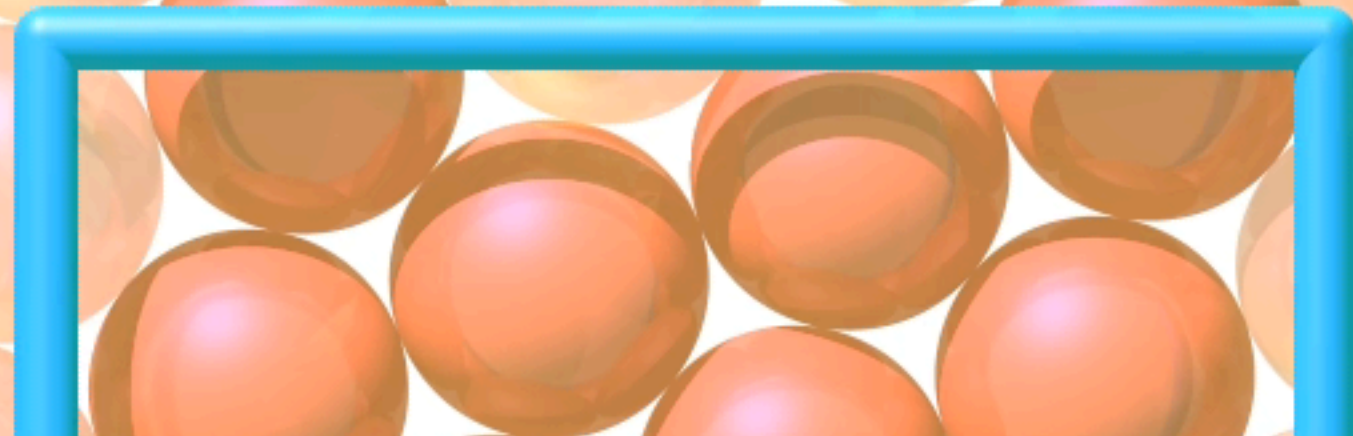
Some models are based on introducing *free volume* at the orifice that propagates upward, moving the particles down.

Can we track these small changes in volume?





# Measuring 2D density (version 1)

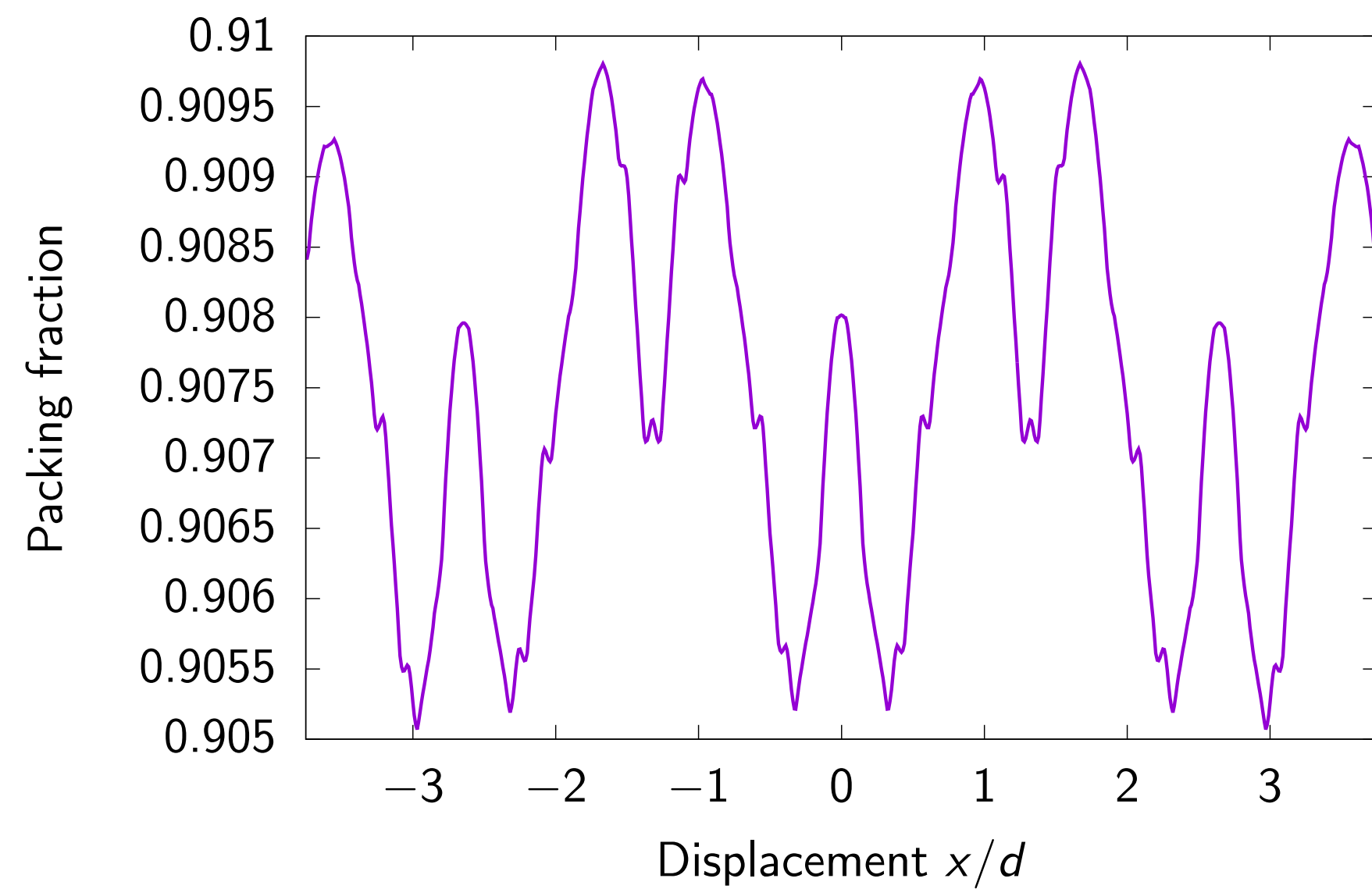
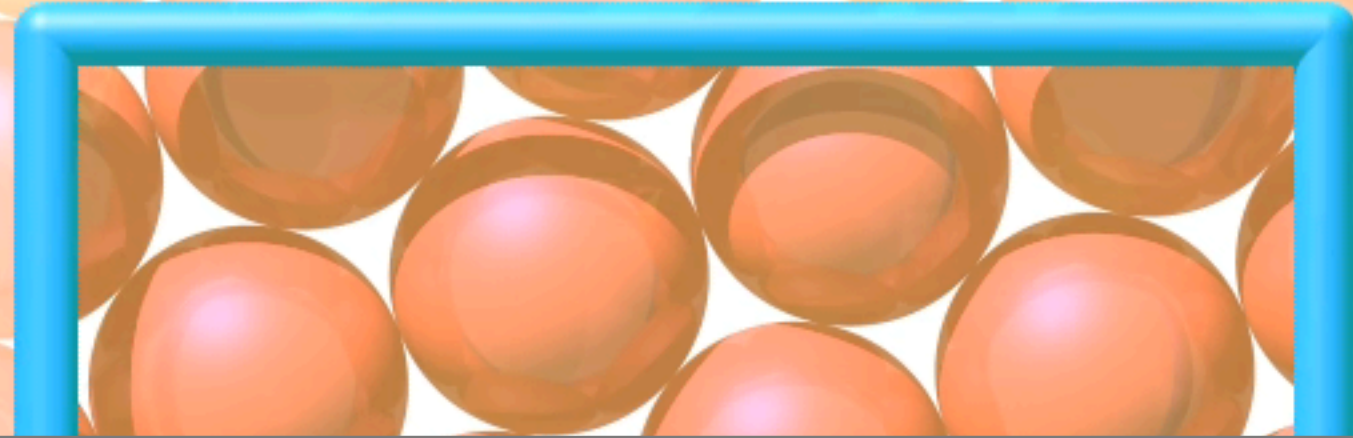


Packing fraction varies by 20%,  
and sometimes exceeds 100%

We use **packing fraction**, the  
proportion of space occupied by  
particles, as a proxy for density



# Measuring 2D density (version 2)



More accurate, but still  
has variations of 0.5%

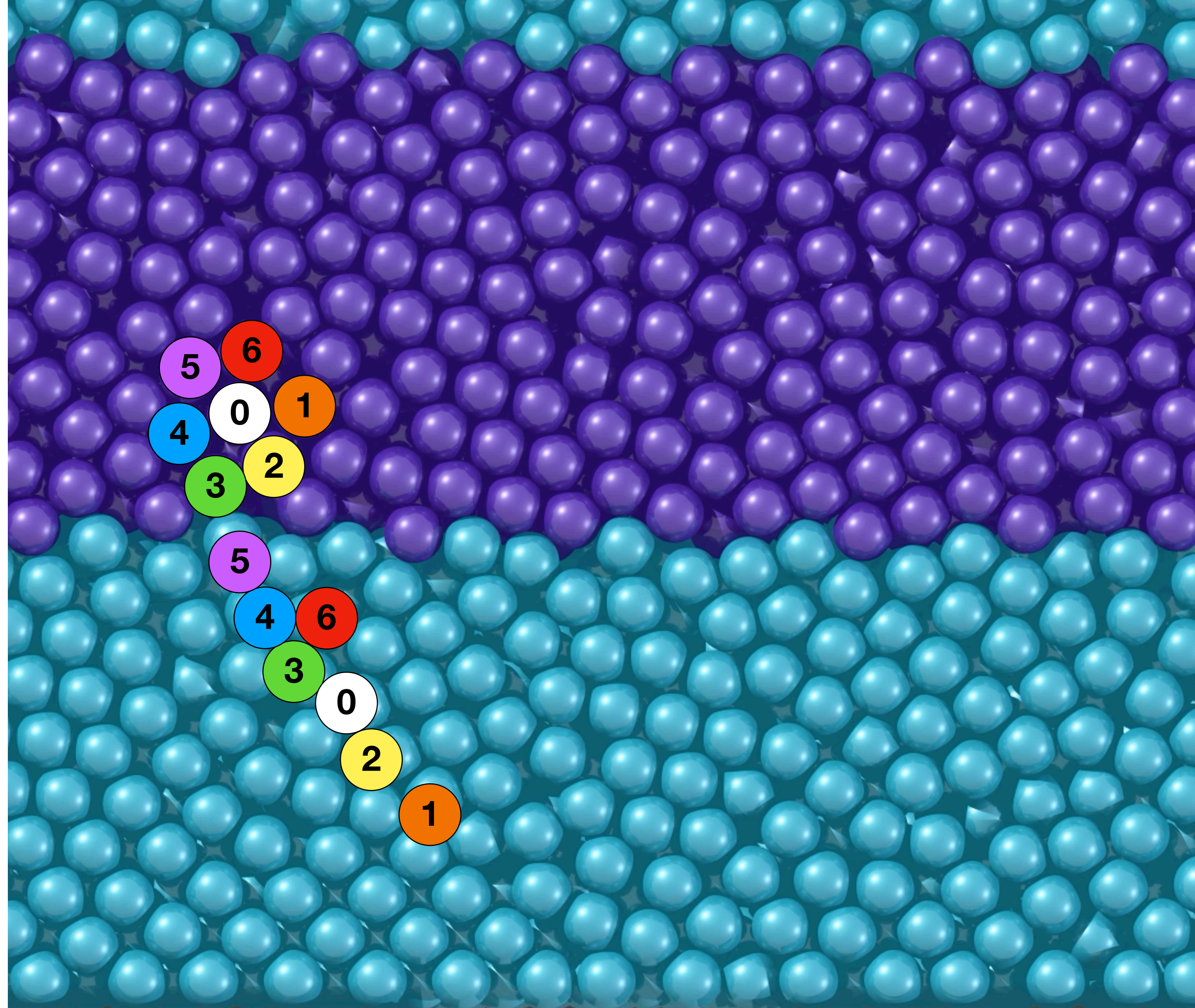


# Another question: packing structure

## Question

The DEM simulation realistically models how particles move past each other\*

How do we track the neighbor relations of particles during flow?



\* C. H. Rycroft *et al.*, *Physical test of a particle simulation model in a sheared granular system*, Phys. Rev. E **80**, 031305 (2009).

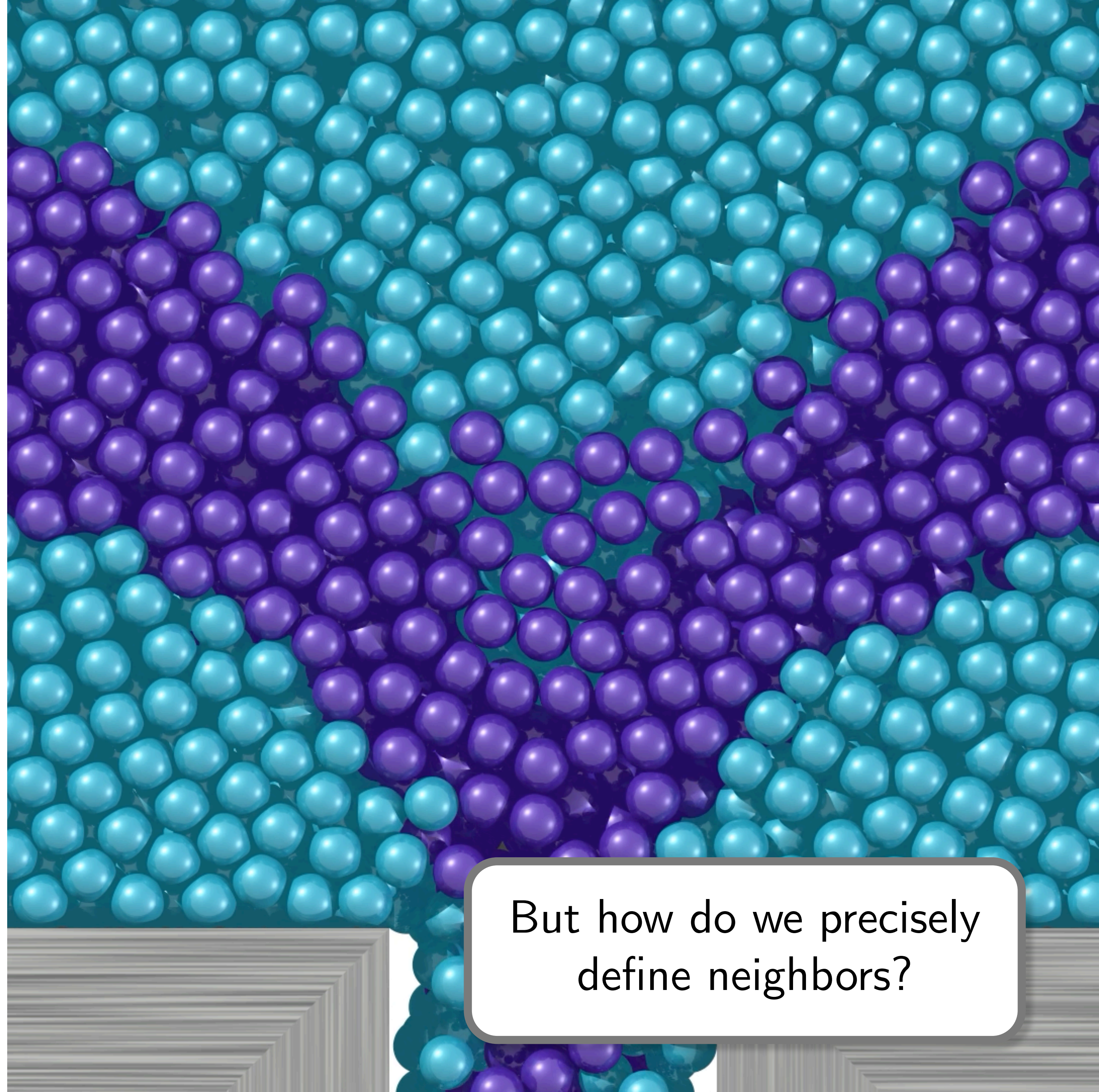


# Another question: packing structure

## Question

The DEM simulation realistically models how particles move past each other\*

How do we track the **neighbor** relations of particles during flow?



But how do we precisely define neighbors?

\* C. H. Rycroft *et al.*, *Physical test of a particle simulation model in a sheared granular system*, Phys. Rev. E **80**, 031305 (2009).



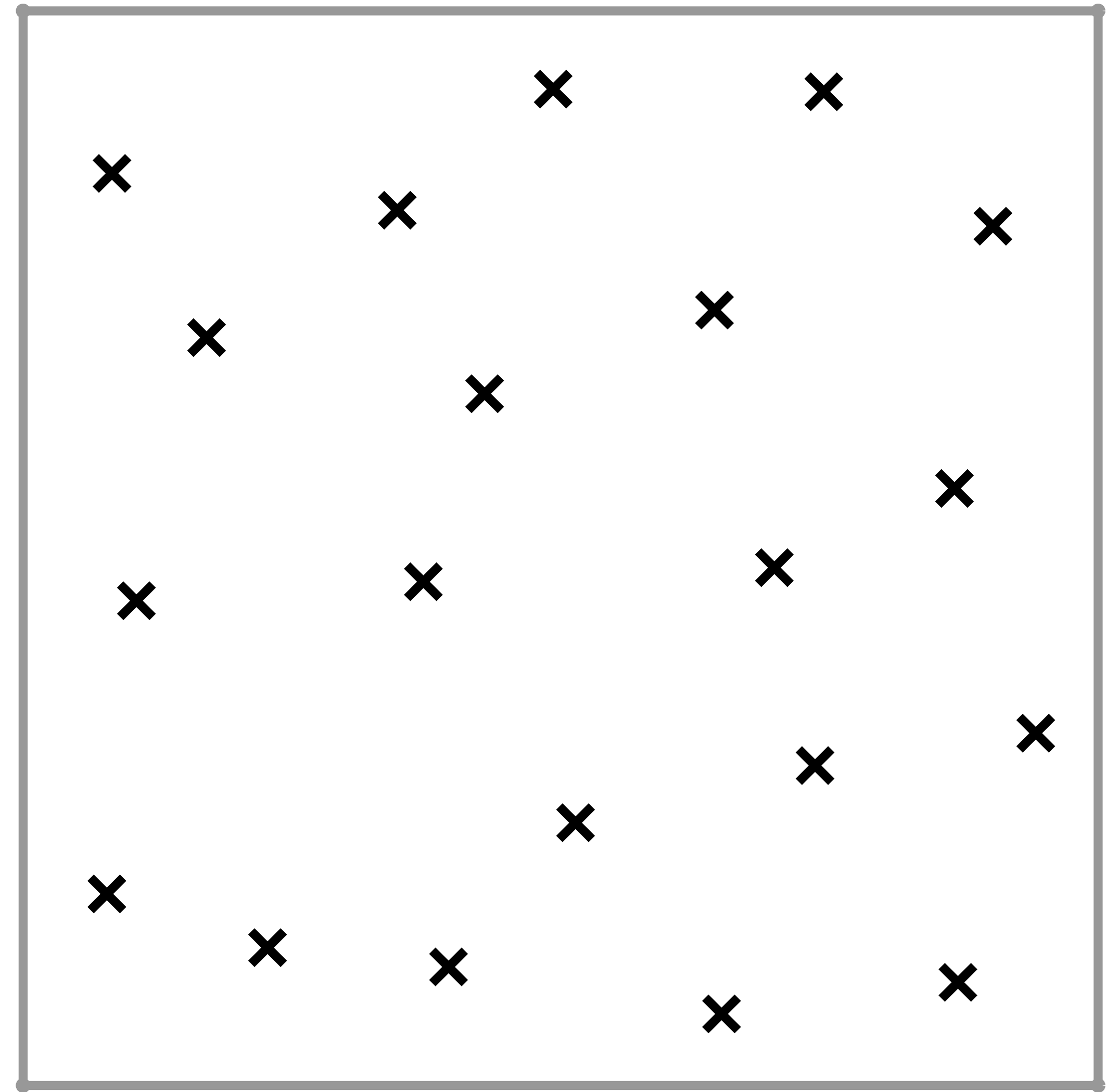
# A technique from computational geometry

- Consider a domain  $\Omega$  with distance metric  $d$ , and introduce a set of points  $x_i$
- The **Voronoi cell** for point\*  $x_i$  is the space  $x$  that satisfies

$$d(x, x_i) < d(x, x_j)$$

for any  $j \neq i$

- Together the Voronoi cells form the **Voronoi tessellation** of  $\Omega$



\* We can think of *point* and *particle* as synonymous in this definition

A. Okabe *et al.*, *Spatial tessellations: Concepts and Applications of Voronoi Diagrams*, Wiley, 2000.

E. A. Lazar *et al.*, *Voronoi cell analysis: the shapes of particle systems*, Am. J. Phys. **90**, 469 (2022).



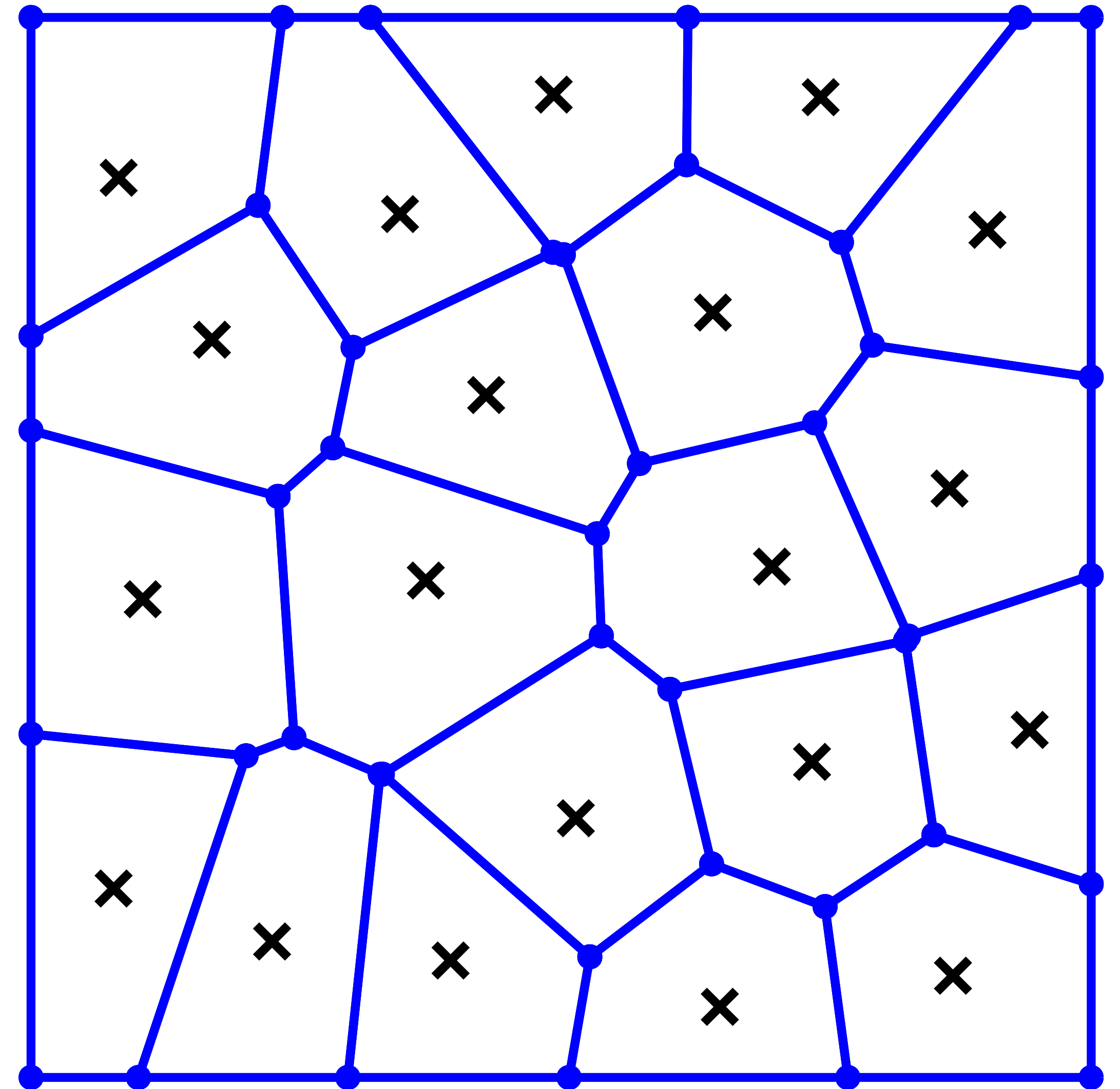
# A technique from computational geometry

- Consider a domain  $\Omega$  with distance metric  $d$ , and introduce a set of points  $x_i$
- The **Voronoi cell** for point\*  $x_i$  is the space  $x$  that satisfies

$$d(x, x_i) < d(x, x_j)$$

for any  $j \neq i$

- Together the Voronoi cells form the **Voronoi tessellation** of  $\Omega$



\* We can think of *point* and *particle* as synonymous in this definition

A. Okabe *et al.*, *Spatial tessellations: Concepts and Applications of Voronoi Diagrams*, Wiley, 2000.

E. A. Lazar *et al.*, *Voronoi cell analysis: the shapes of particle systems*, Am. J. Phys. **90**, 469 (2022).



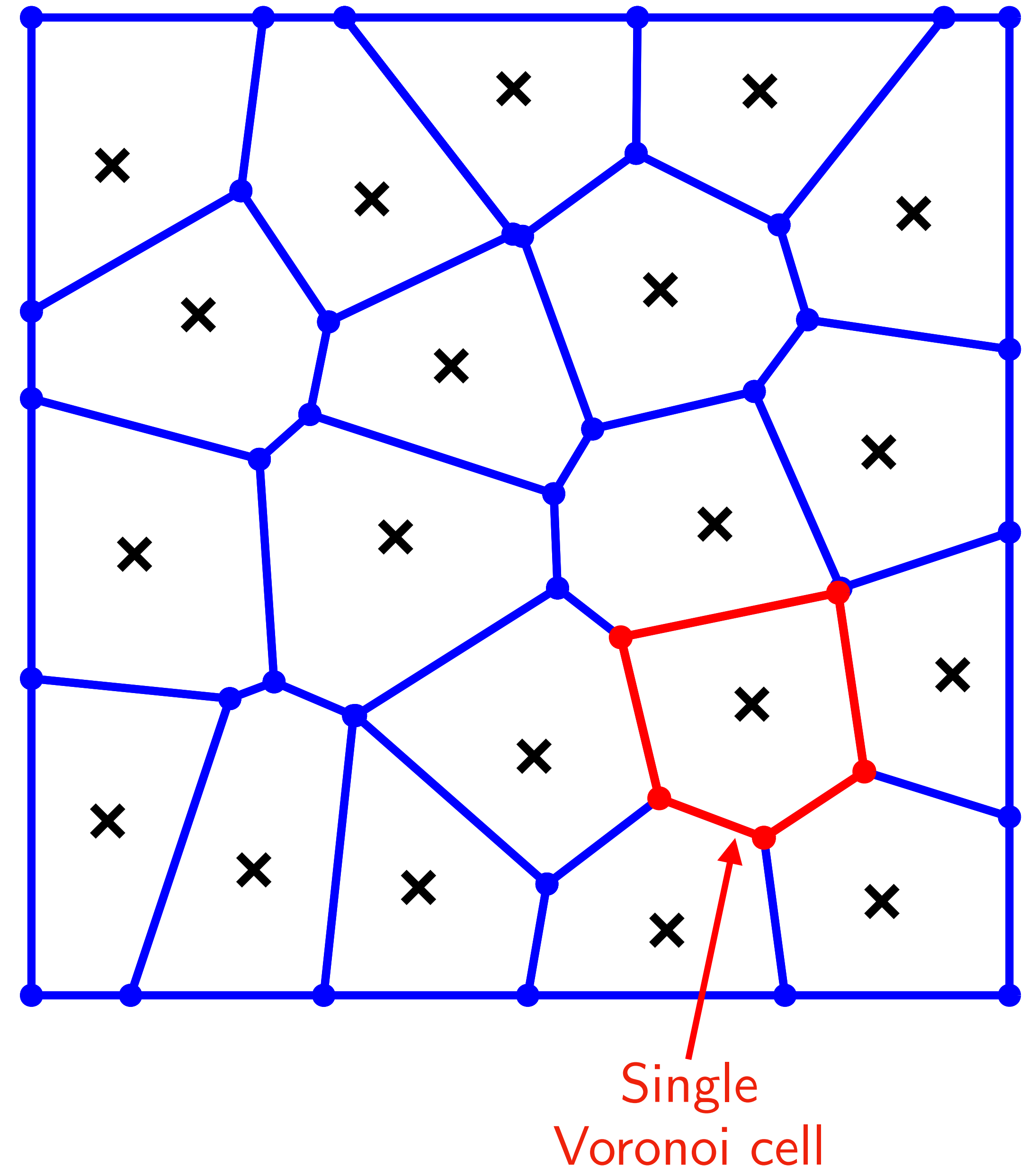
# A technique from computational geometry

- Consider a domain  $\Omega$  with distance metric  $d$ , and introduce a set of points  $x_i$
- The **Voronoi cell** for point\*  $x_i$  is the space  $x$  that satisfies

$$d(x, x_i) < d(x, x_j)$$

for any  $j \neq i$

- Together the Voronoi cells form the **Voronoi tessellation** of  $\Omega$



\* We can think of *point* and *particle* as synonymous in this definition

A. Okabe *et al.*, *Spatial tessellations: Concepts and Applications of Voronoi Diagrams*, Wiley, 2000.

E. A. Lazar *et al.*, *Voronoi cell analysis: the shapes of particle systems*, Am. J. Phys. **90**, 469 (2022).



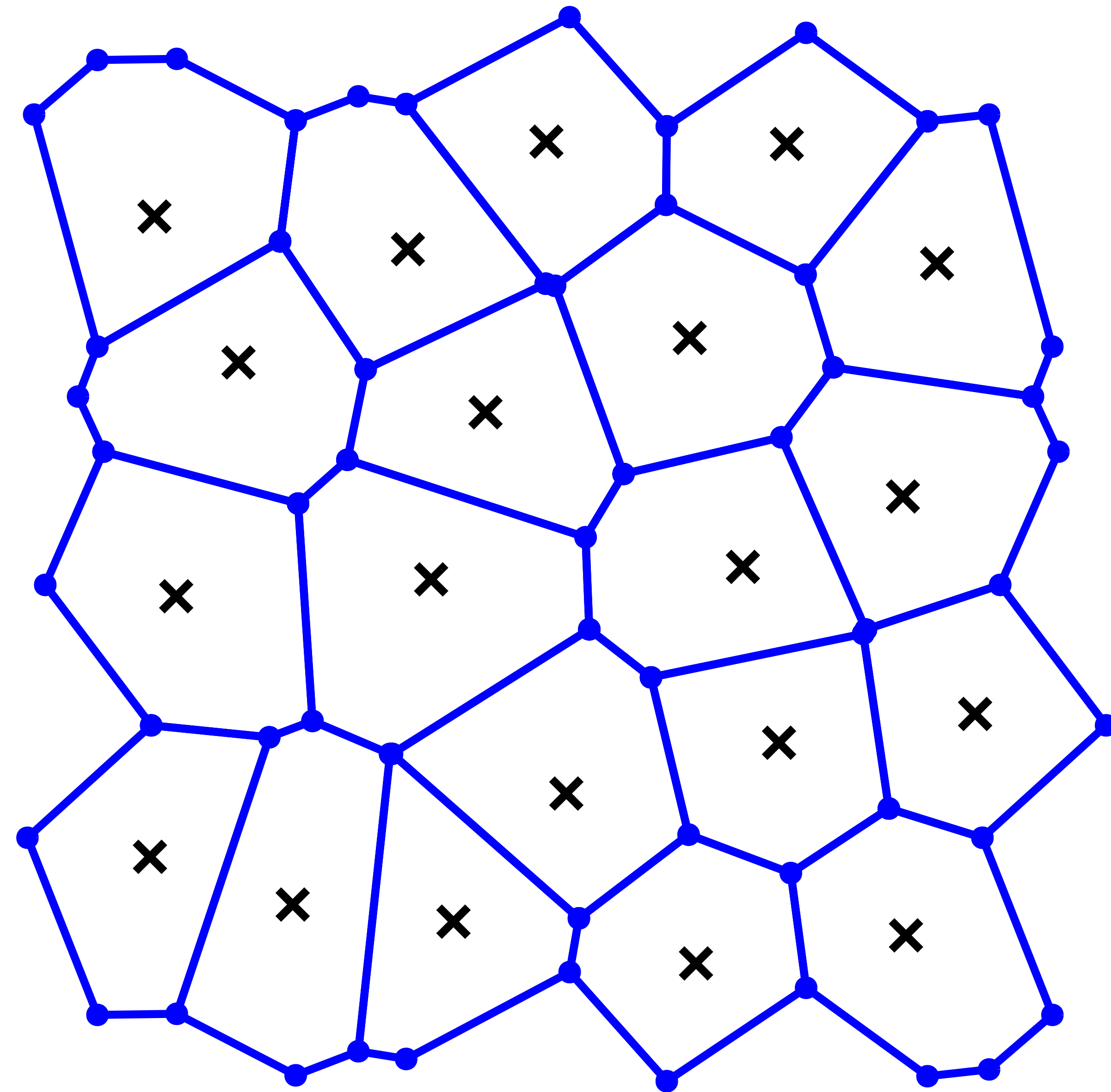
# A technique from computational geometry

- Consider a domain  $\Omega$  with distance metric  $d$ , and introduce a set of points  $x_i$
- The **Voronoi cell** for point\*  $x_i$  is the space  $x$  that satisfies

$$d(x, x_i) < d(x, x_j)$$

for any  $j \neq i$

- Together the Voronoi cells form the **Voronoi tessellation** of  $\Omega$



(Can also use periodic boundary conditions)

\* We can think of *point* and *particle* as synonymous in this definition

A. Okabe *et al.*, *Spatial tessellations: Concepts and Applications of Voronoi Diagrams*, Wiley, 2000.

E. A. Lazar *et al.*, *Voronoi cell analysis: the shapes of particle systems*, Am. J. Phys. **90**, 469 (2022).



# Some history



- Developed by Georgy Voronyi (1868–1908) in Imperial Russia (present day Ukraine)
- Introduced in two pure mathematics papers in 1908
- Commemorated in 2008 in Ukraine with two-hryvnia coin



G. Voronoï, *Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire* ..., J. reine angew. Math. **133**, 97–179 (1908).

G. Voronoï, *Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire* ..., J. reine angew. Math. **134**, 198–287 (1908).

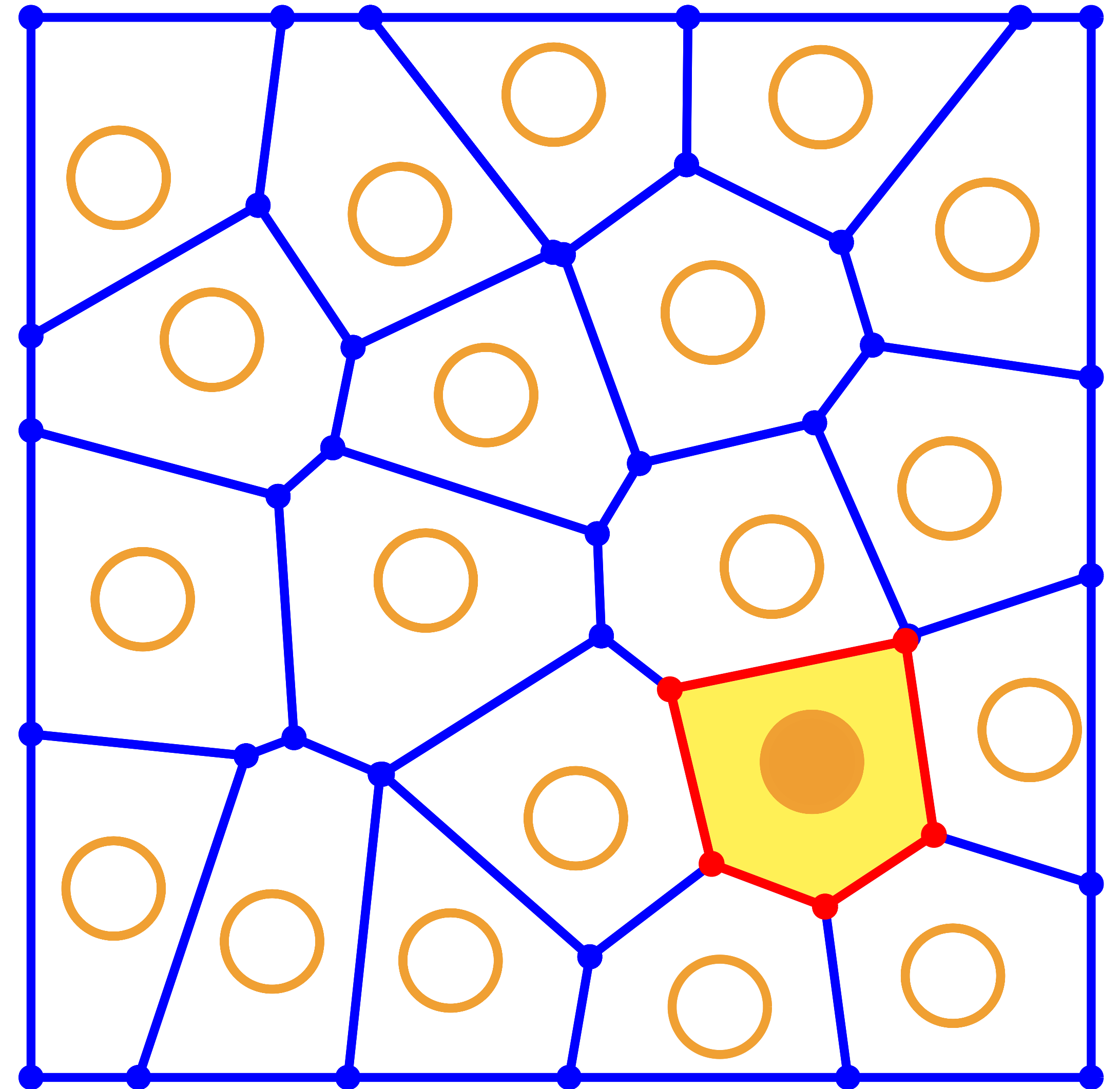


# Accurate density measurements

- The Voronoi tessellation provides method to compute the packing fraction  $\phi$  accurately
- For a single particle, measure packing fraction as

$$\phi = \frac{\text{Particle volume}}{\text{Voronoi cell volume}}$$

- Could also be average over several particles and Voronoi cells





# Revisiting 2D hexagonal packing

Each particle has area

$$V_p = \frac{\pi d^2}{4}$$



# Computing the Voronoi cells

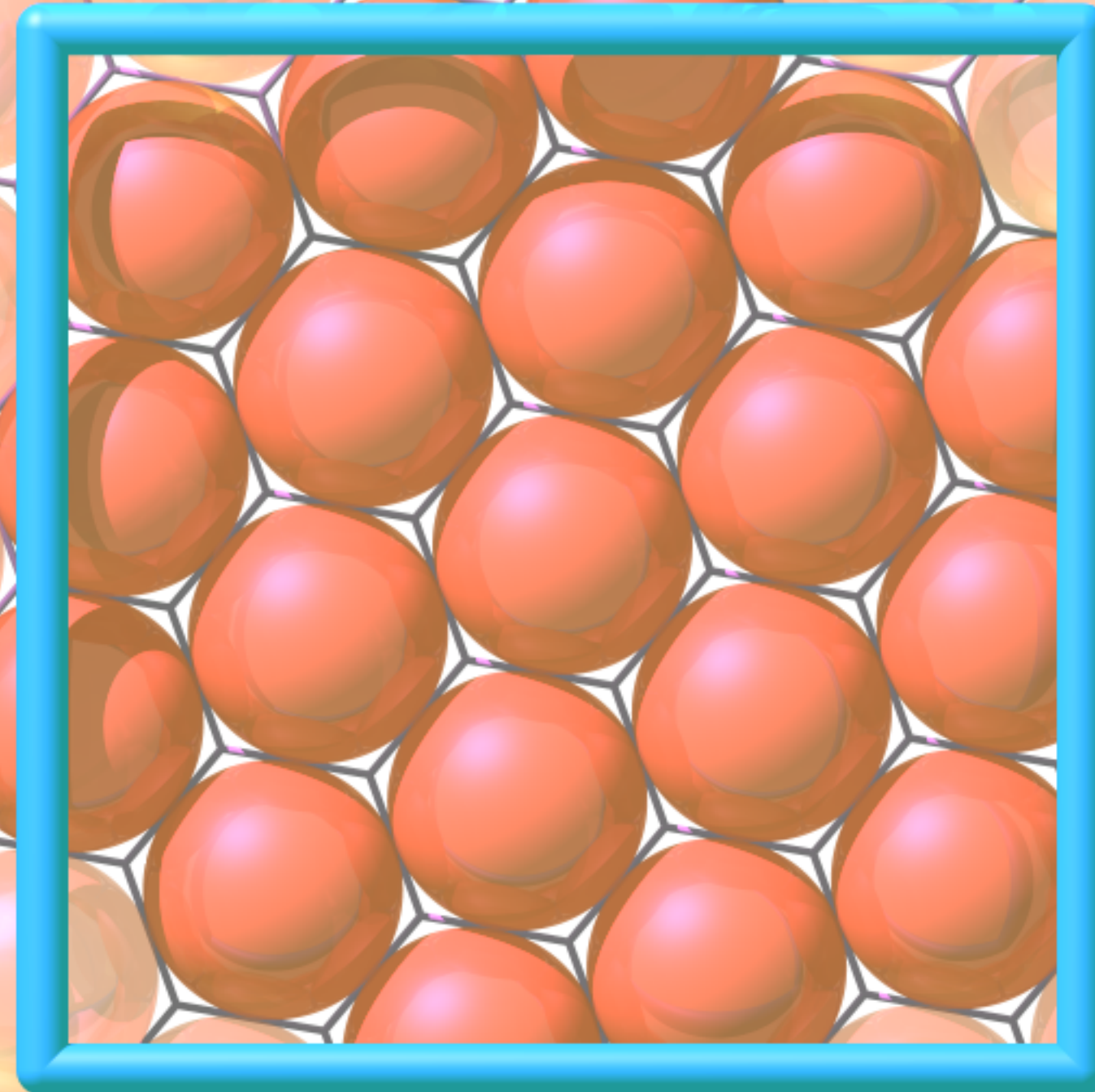
The Voronoi cells will  
all be regular hexagons

Each has area

$$V_c = \frac{d^2 \sqrt{3}}{2}$$



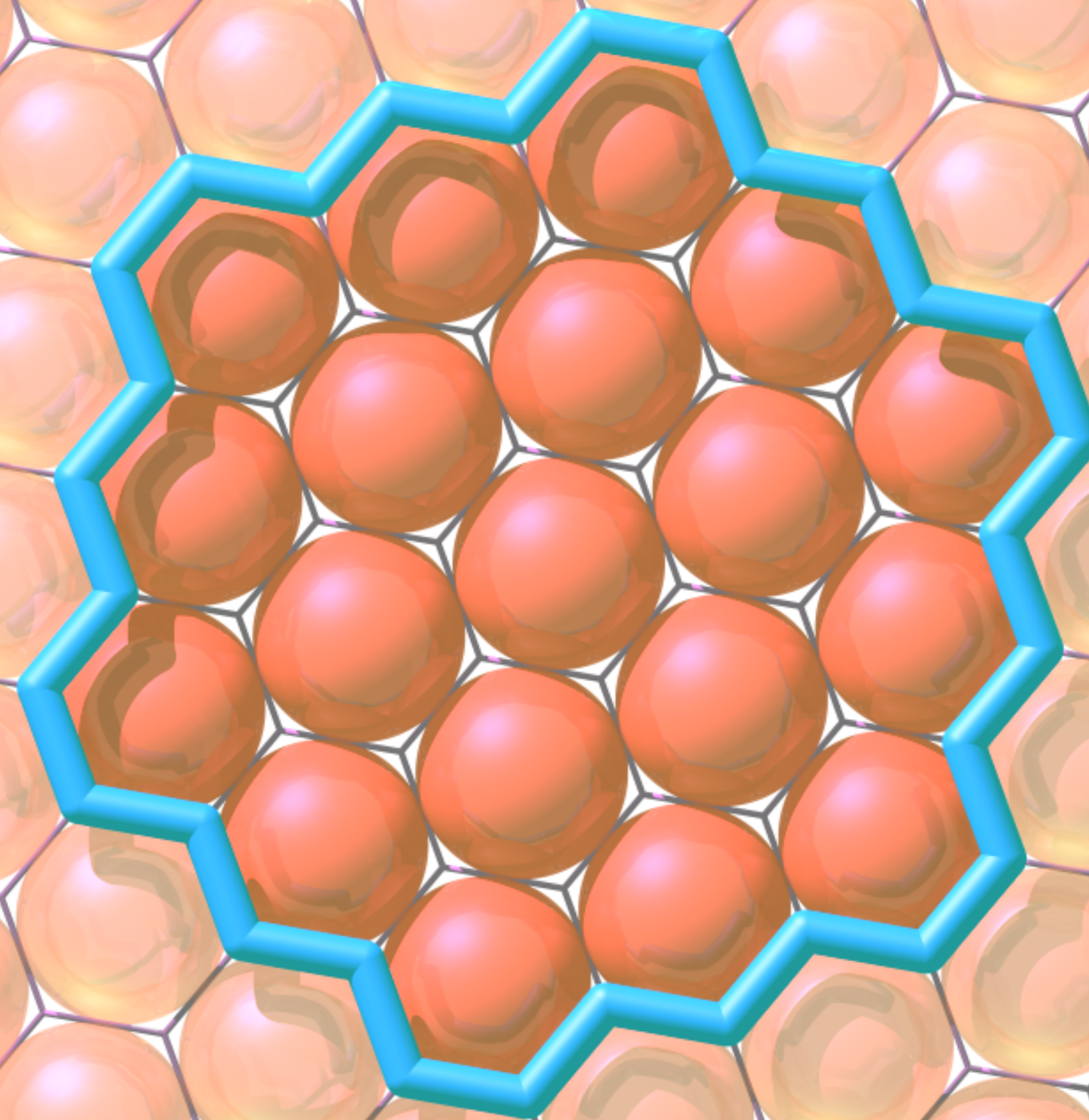
# Examining particles within test box



Consider all of the 19  
particles and Voronoi  
cells within a box



# Examining particles within test box



Calculate packing fraction based on

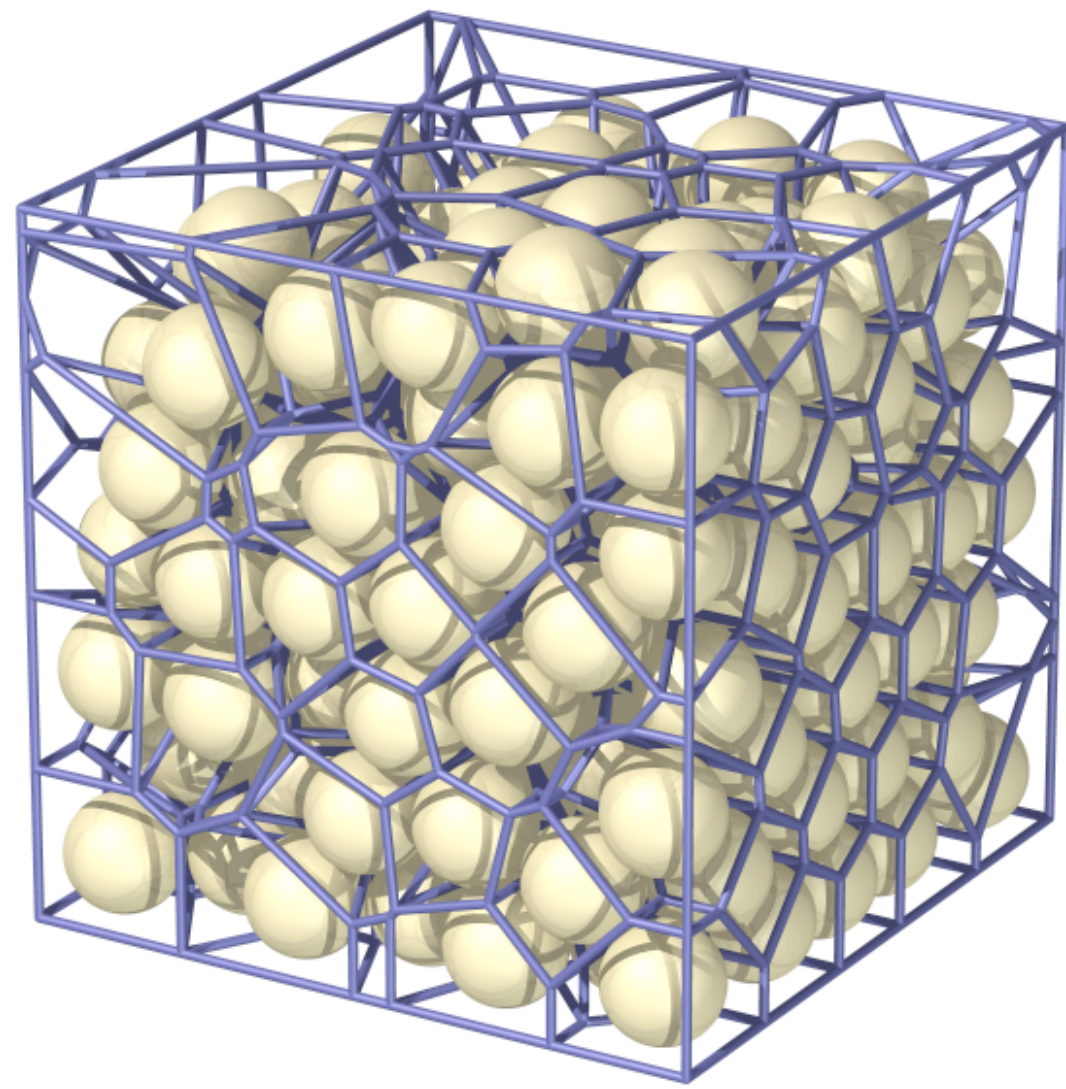
$$\phi = \frac{19V_p}{19V_c} = \frac{\pi}{2\sqrt{3}} = 90.69\%$$

Even based on a single particle, this precisely matches the asymptotic value for large regions

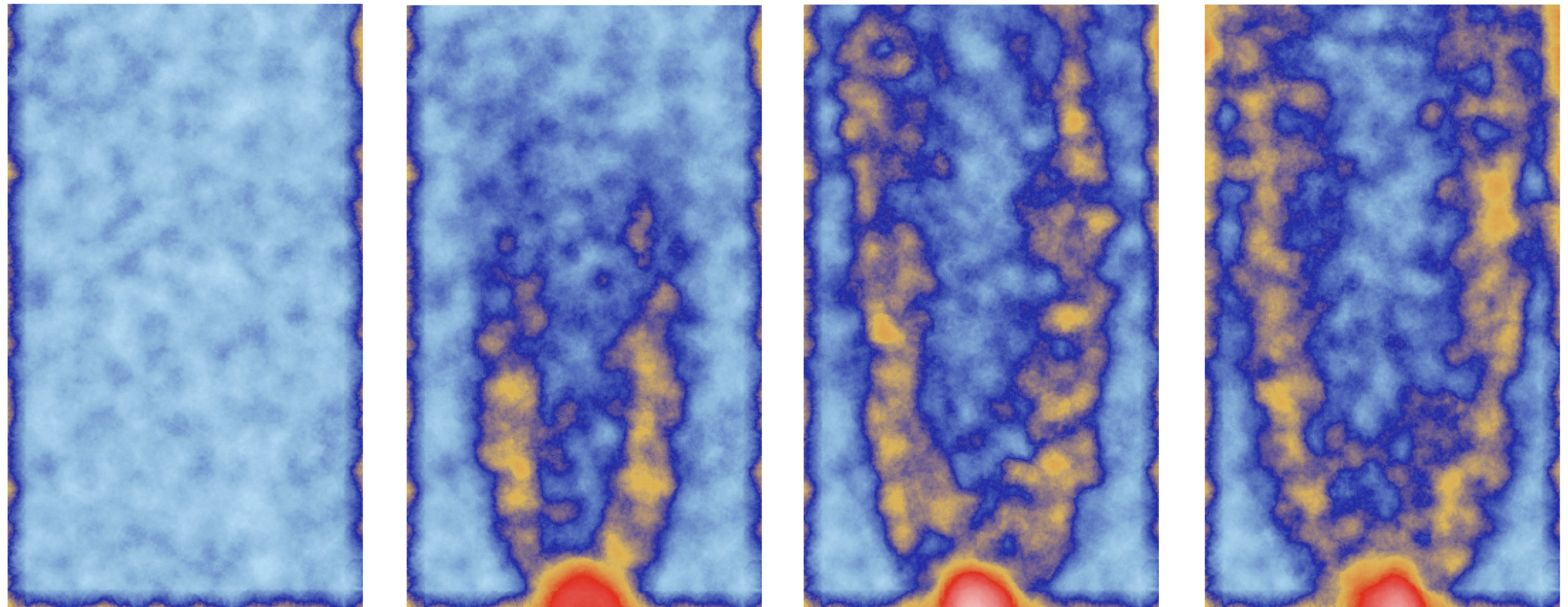
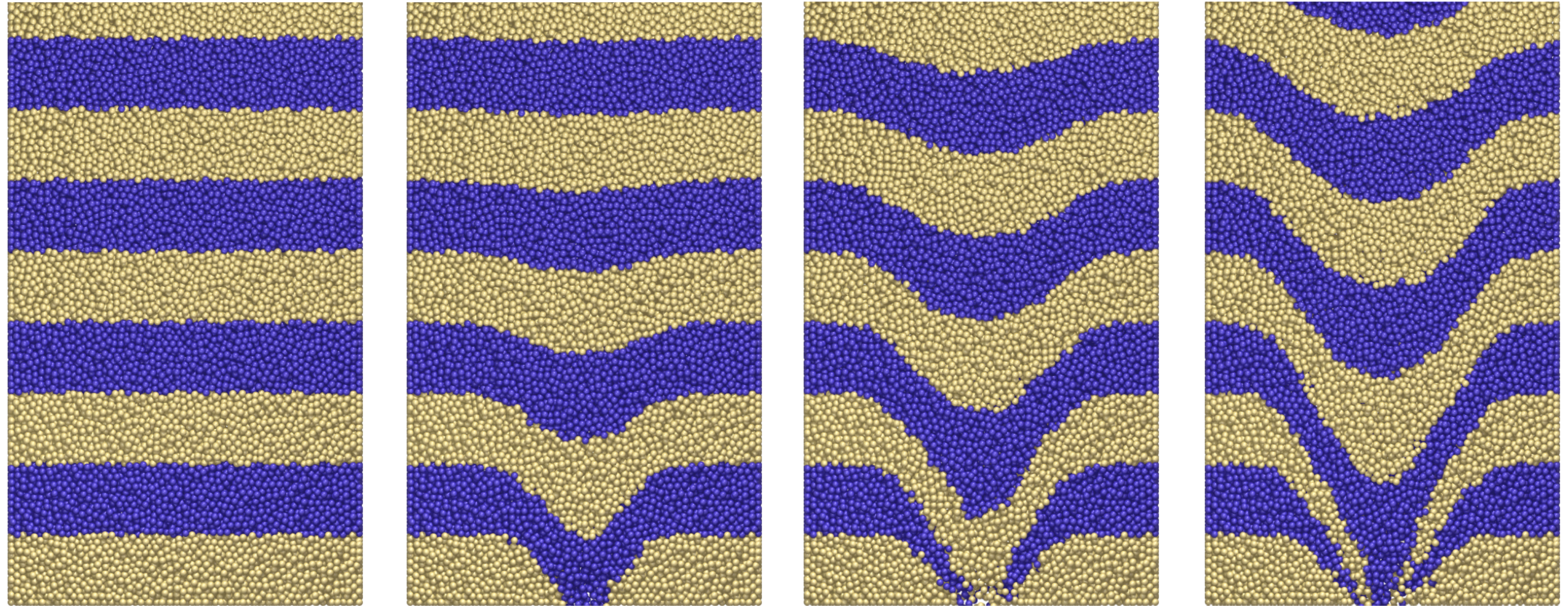


# Density variations

*based on 3D Voronoi cell volumes in local patches*



DEM snapshots through central slice



Local packing fraction

→ increasing time





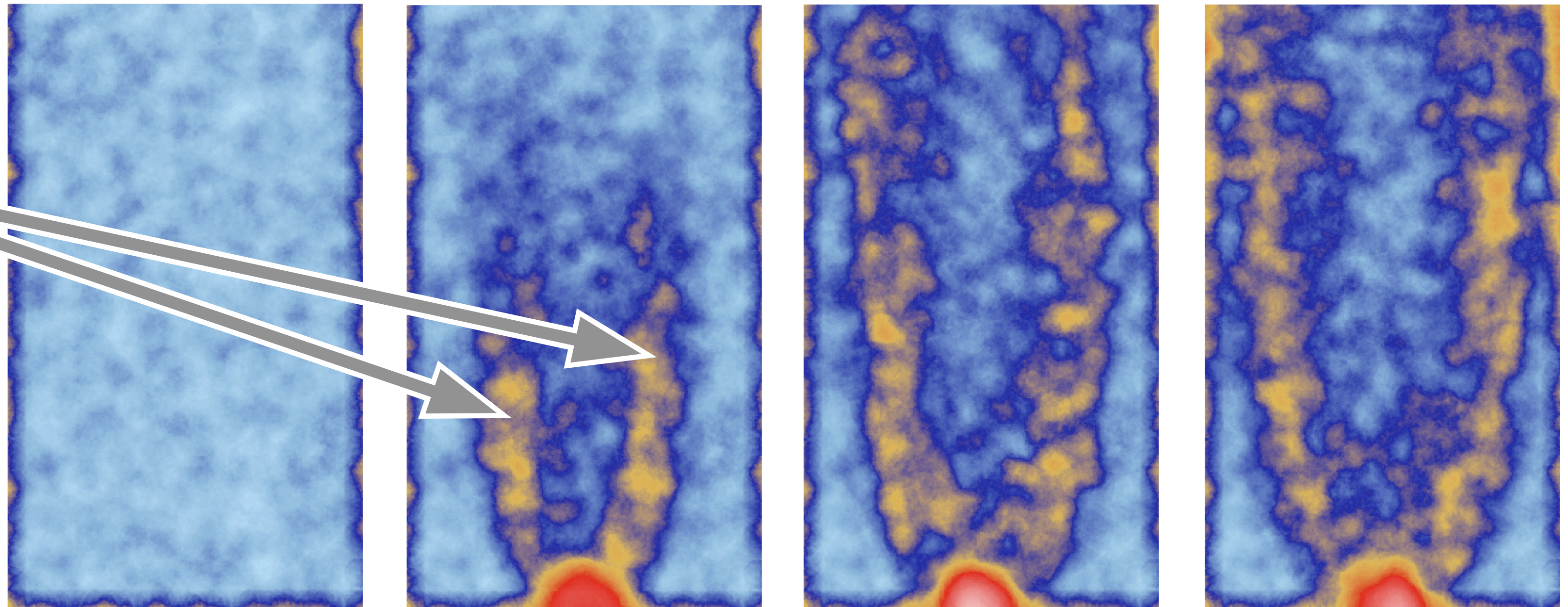
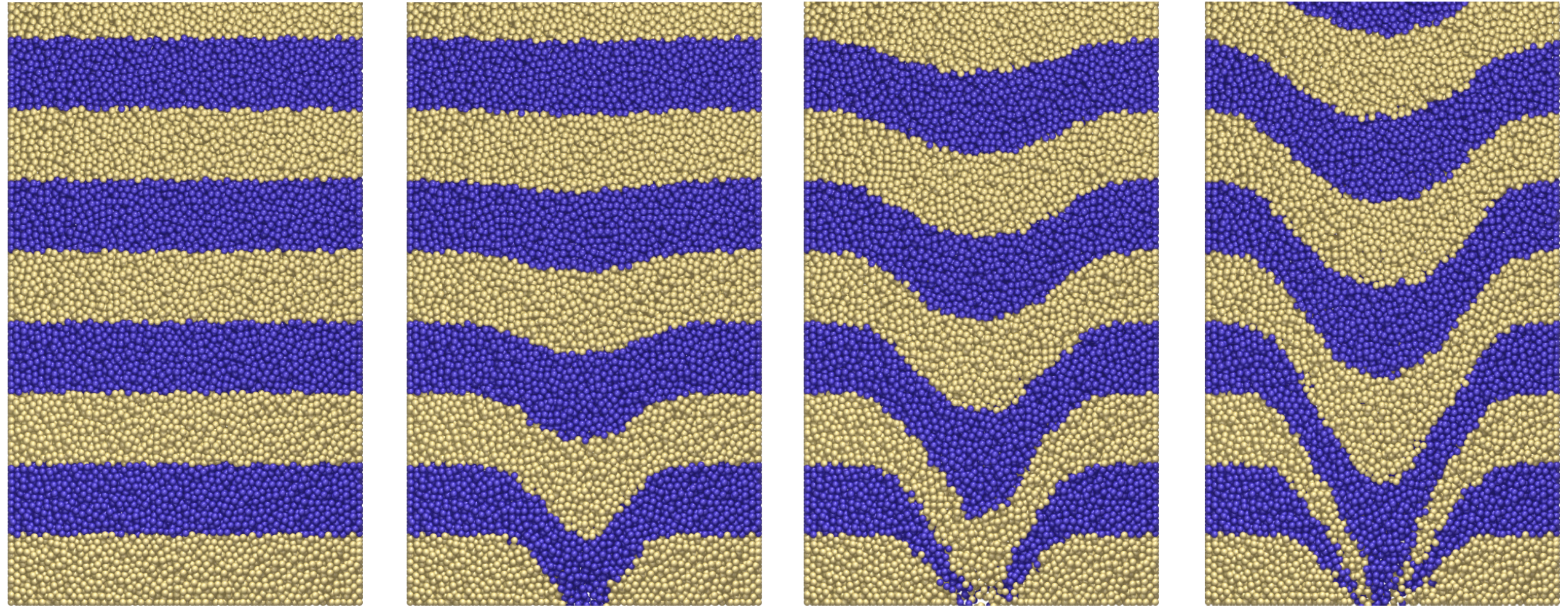
# Density variations

*based on 3D Voronoi cell volumes in local patches*

Packing fraction is reduced in regions of **highest shear**, not highest velocity

Makes physical sense, since particles need room to flow past each other

DEM snapshots through central slice



Local packing fraction

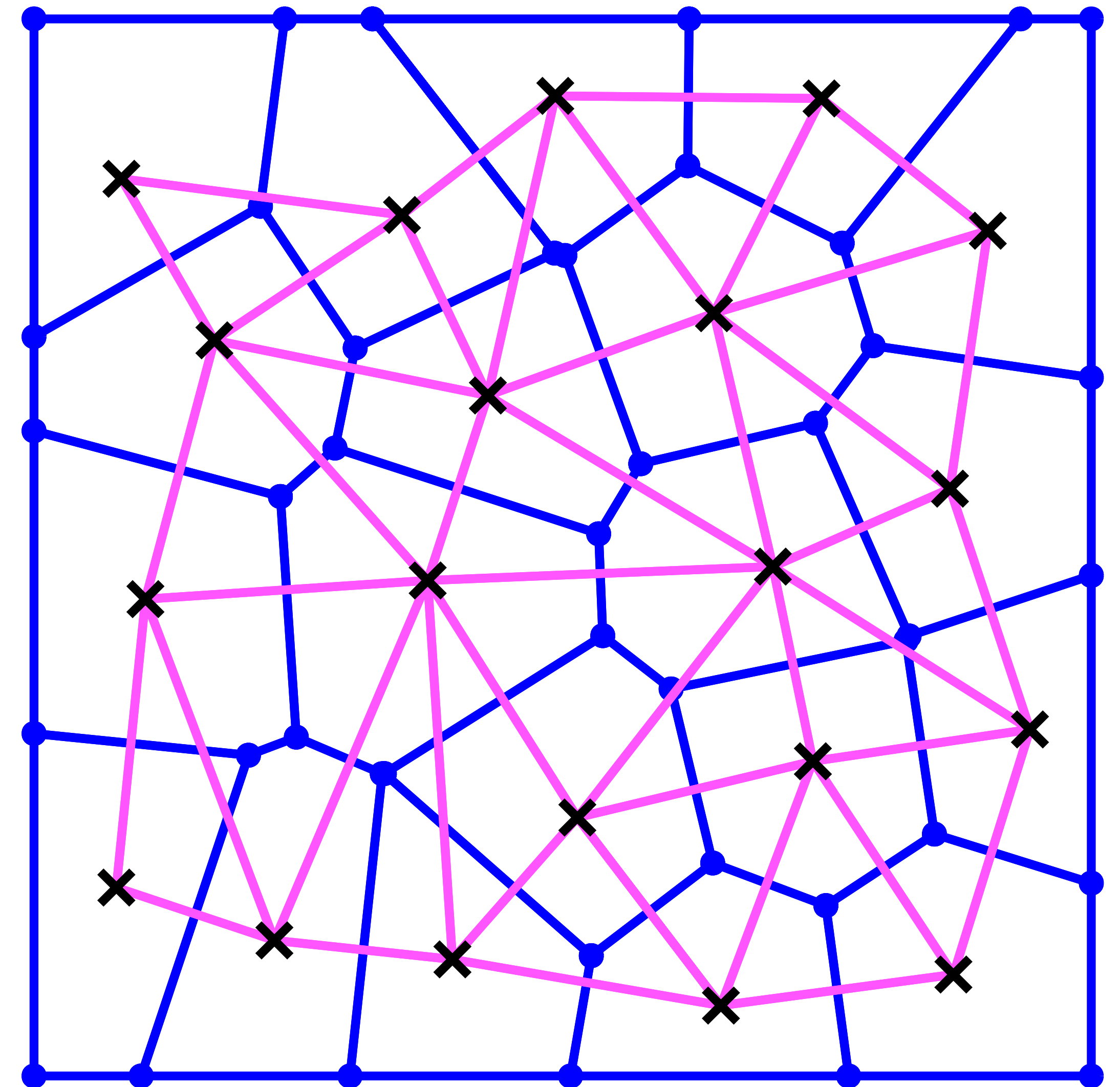
→ increasing time





# A topological definition of neighbors

- We can define two particles as neighbors if their Voronoi cells share a face
- Connecting all these neighbors gives the **Delaunay triangulation**
- Each Delaunay edge corresponds to a Voronoi face; this is called **Voronoi–Delaunay duality**

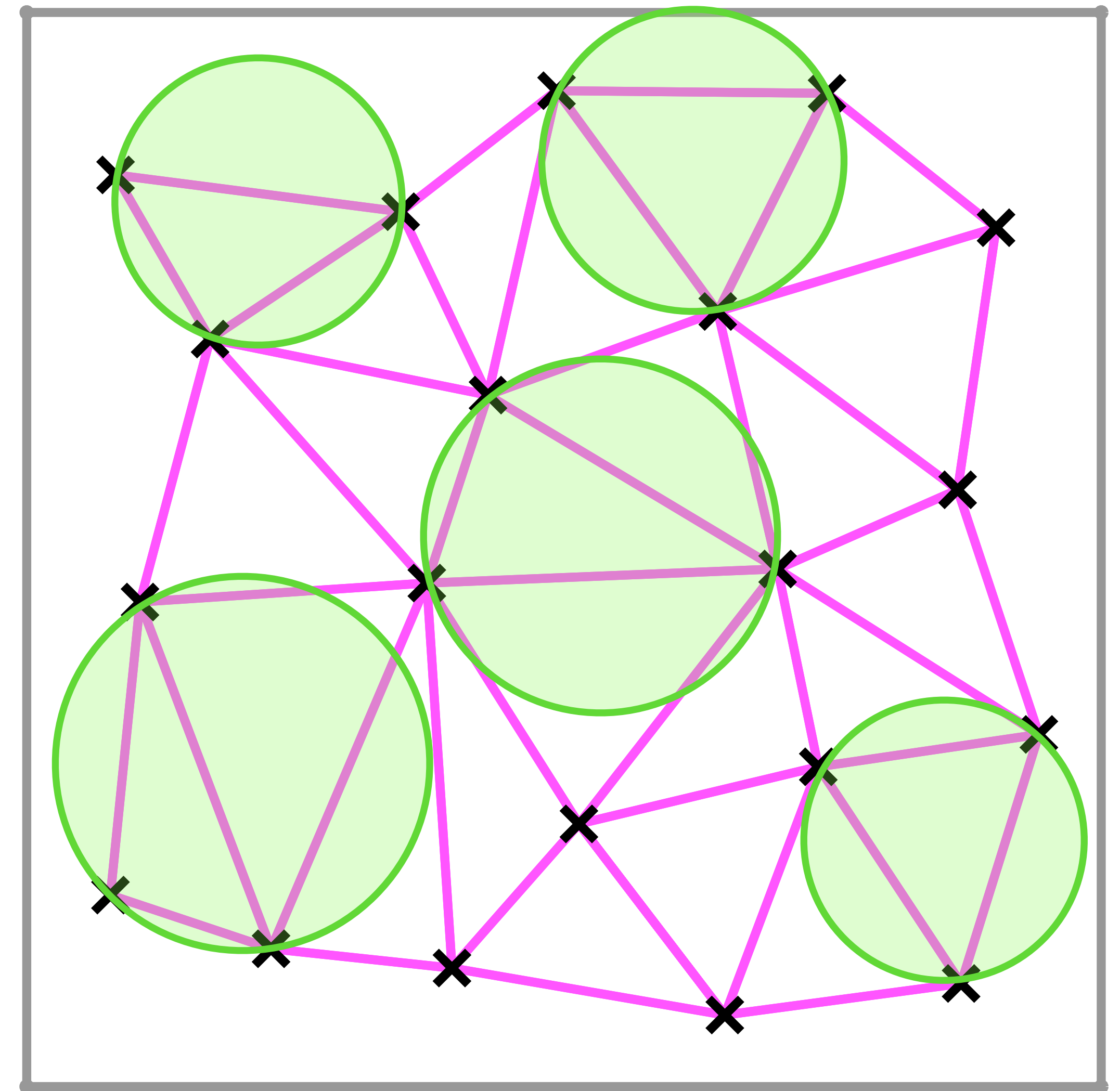


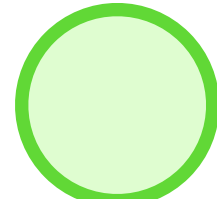
— Delaunay triangulation



# The Delaunay triangulation

- The Delaunay triangulation is interesting in its own right, and can be defined independently of the Voronoi tessellation
- It is the unique triangulation where the circumcircle of each triangle contains no other particle
- Useful basis for triangulation

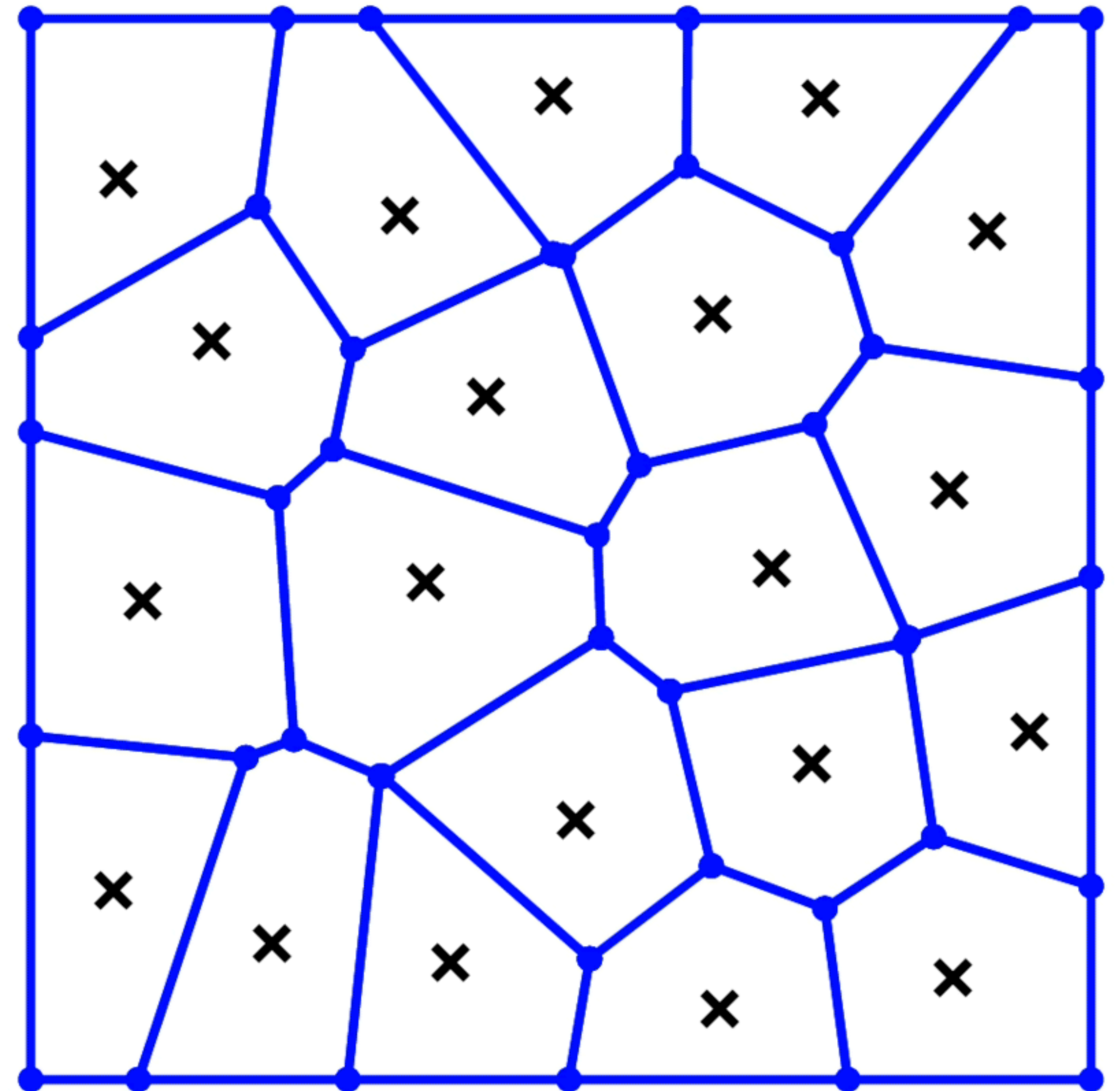


 Example circumcircles



# Calculating an individual Voronoi cell

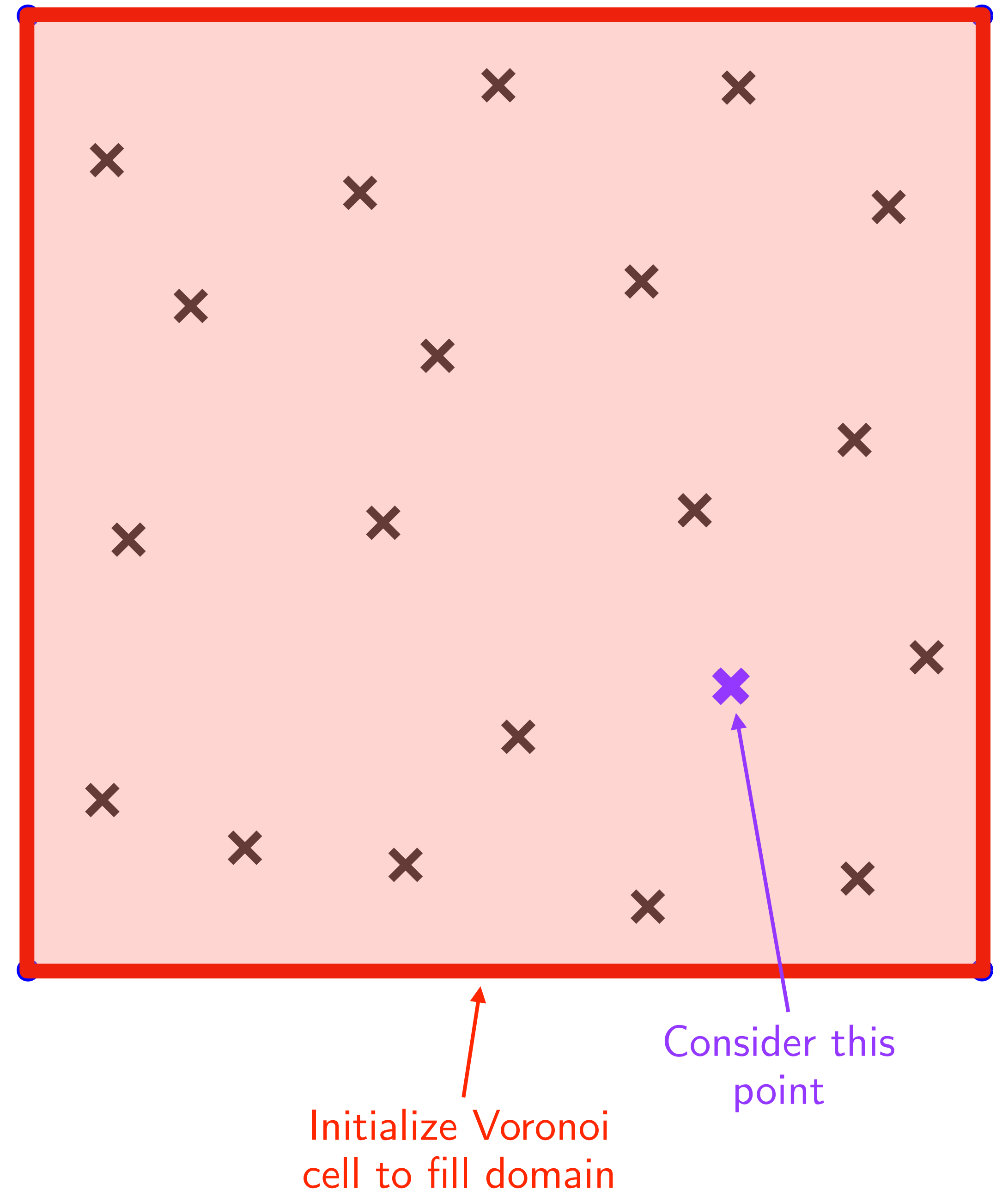
- There are a number of algorithms for computing the Voronoi tessellation:
  - **Incremental approach:** add particles one-by-one and recompute the vertices each time
  - **Fortune's sweeping algorithm:** builds the tessellation by sweeping an advancing front across the domain
  - **Quickhull algorithm:** finding the tessellation in  $\mathbb{R}^n$  can be converted into computing a convex hull in  $\mathbb{R}^{n+1}$ ; used by the *Qhull* software package





# Calculating an individual Voronoi cell

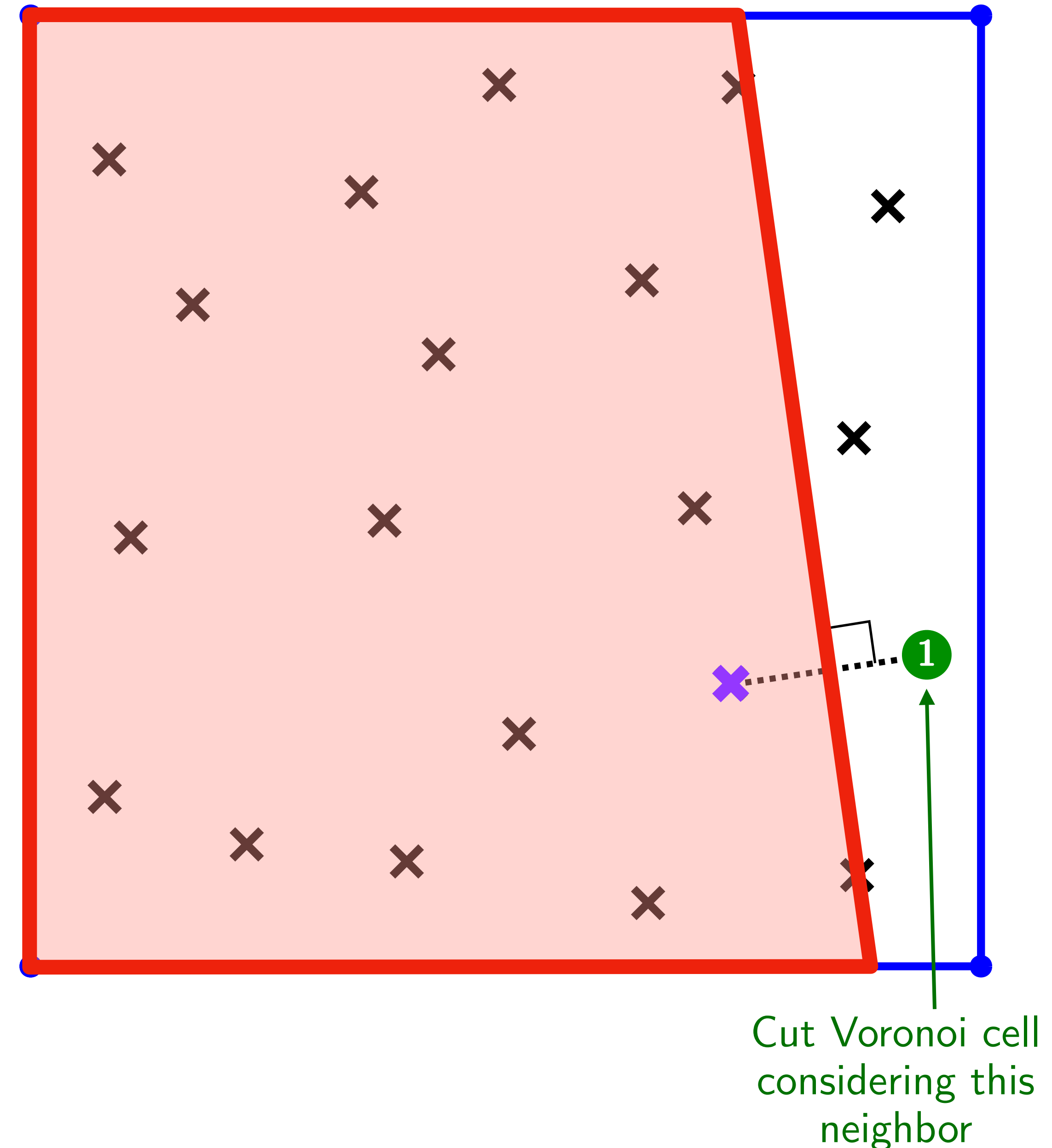
- An alternative method is to compute each Voronoi cell separately
- Initialize Voronoi cell to fill  $\Omega$
- Each neighboring particle cuts the Voronoi cell by a plane that is a perpendicular bisector
- After enough nearby particles are considered, the Voronoi cell is complete





# Calculating an individual Voronoi cell

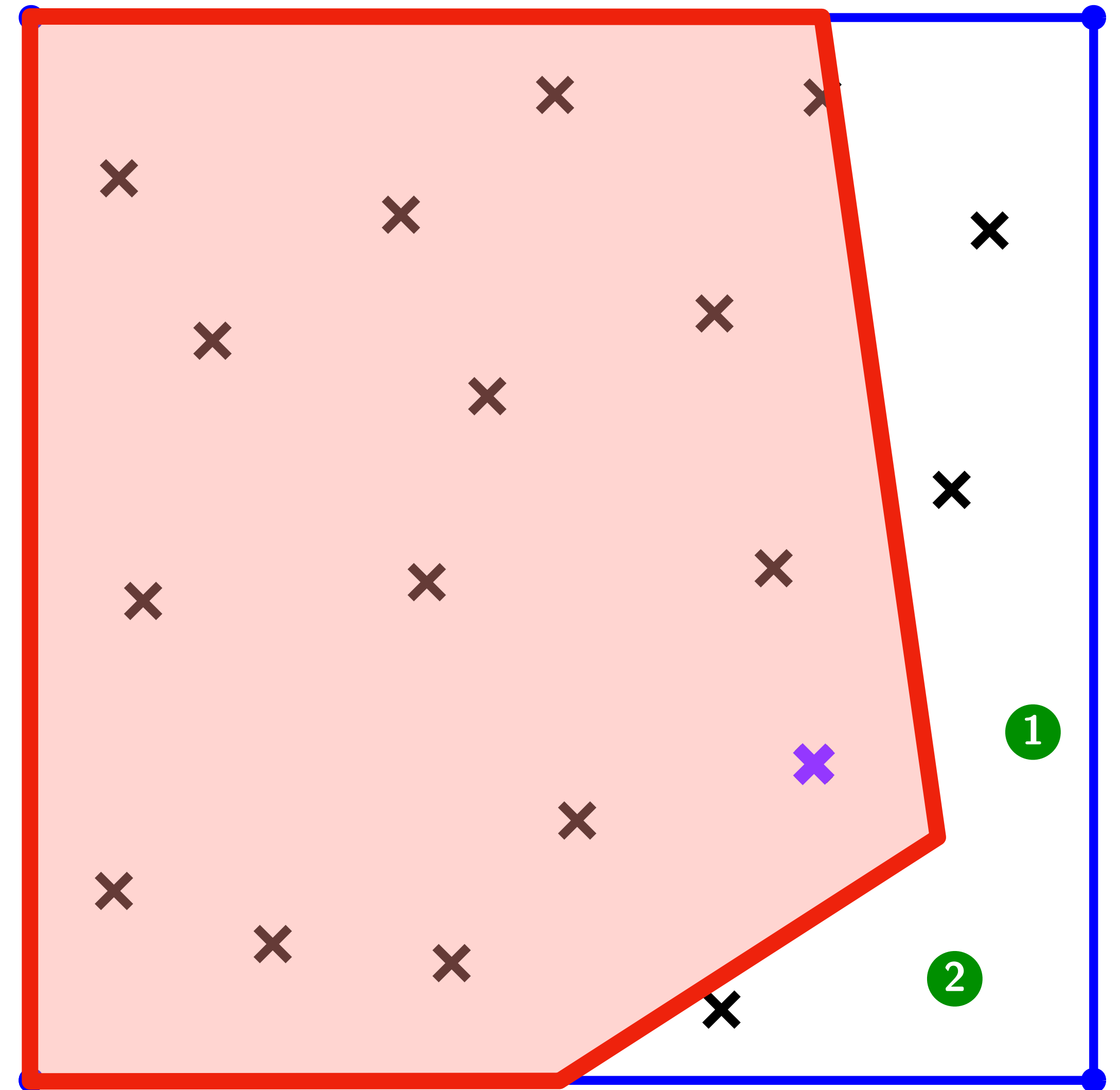
- An alternative method is to compute each Voronoi cell separately
- Initialize Voronoi cell to fill  $\Omega$
- Each neighboring particle cuts the Voronoi cell by a plane that is a perpendicular bisector
- After enough nearby particles are considered, the Voronoi cell is complete





# Calculating an individual Voronoi cell

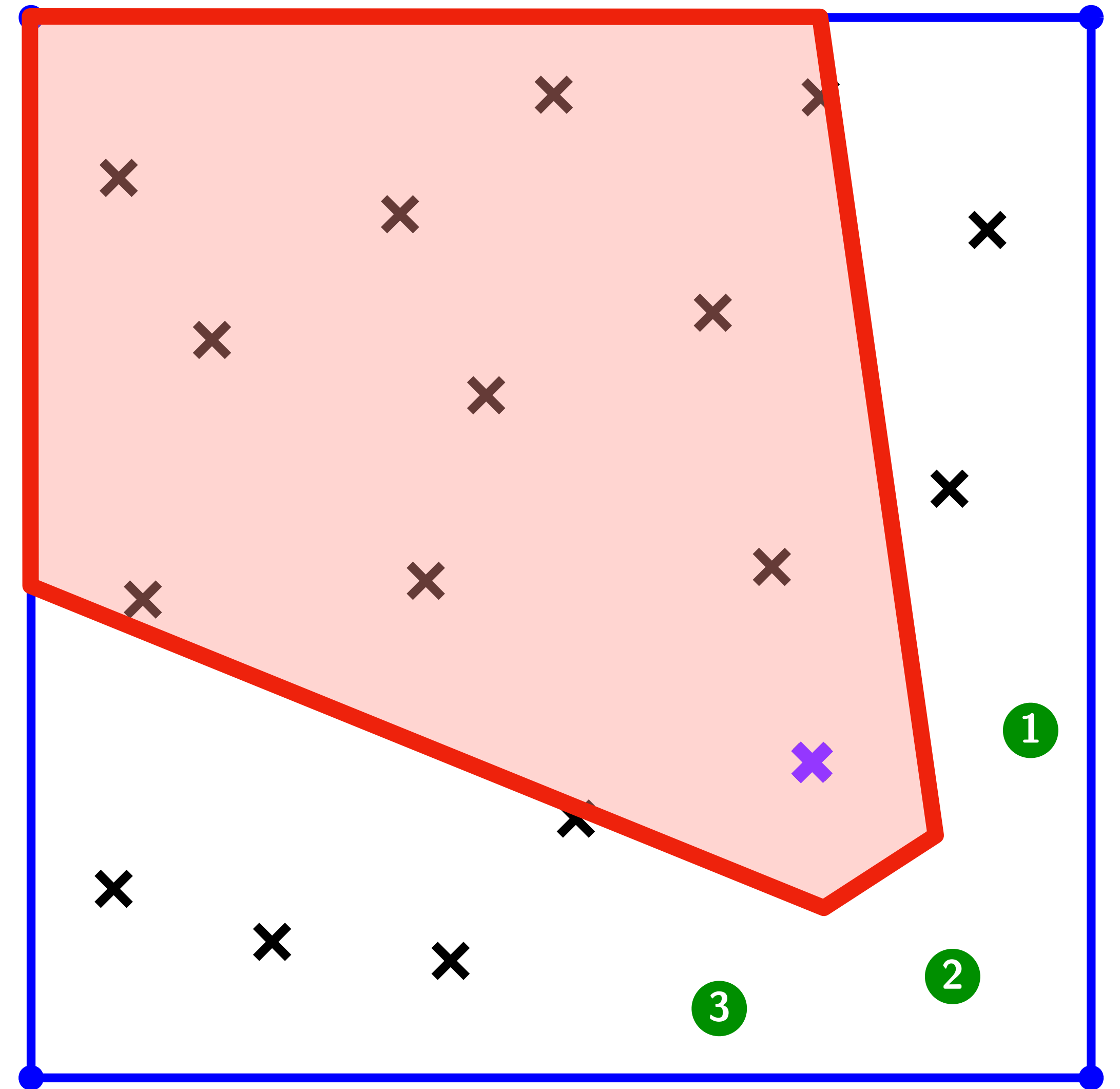
- An alternative method is to compute each Voronoi cell separately
- Initialize Voronoi cell to fill  $\Omega$
- Each neighboring particle cuts the Voronoi cell by a plane that is a perpendicular bisector
- After enough nearby particles are considered, the Voronoi cell is complete





# Calculating an individual Voronoi cell

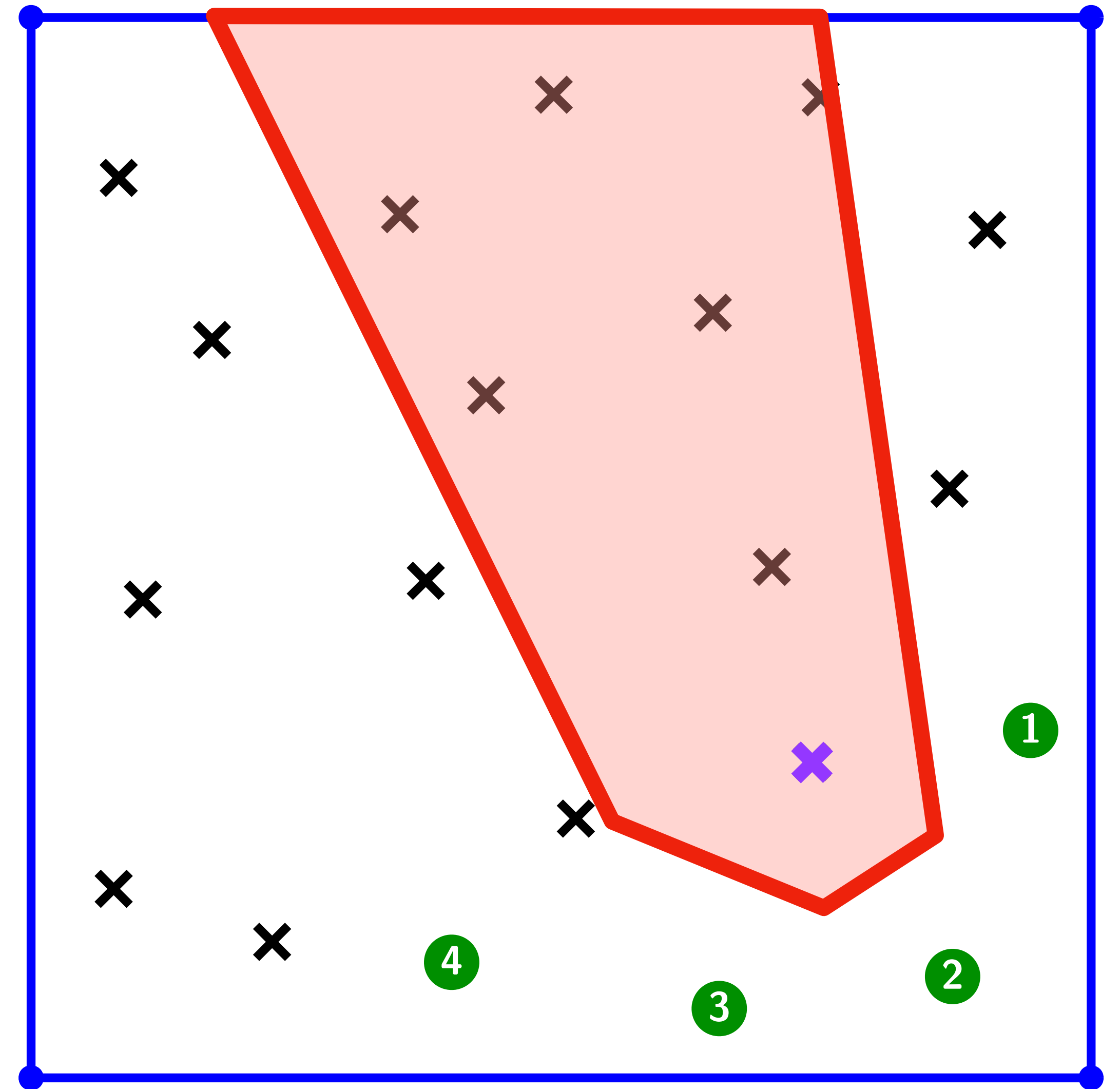
- An alternative method is to compute each Voronoi cell separately
- Initialize Voronoi cell to fill  $\Omega$
- Each neighboring particle cuts the Voronoi cell by a plane that is a perpendicular bisector
- After enough nearby particles are considered, the Voronoi cell is complete





# Calculating an individual Voronoi cell

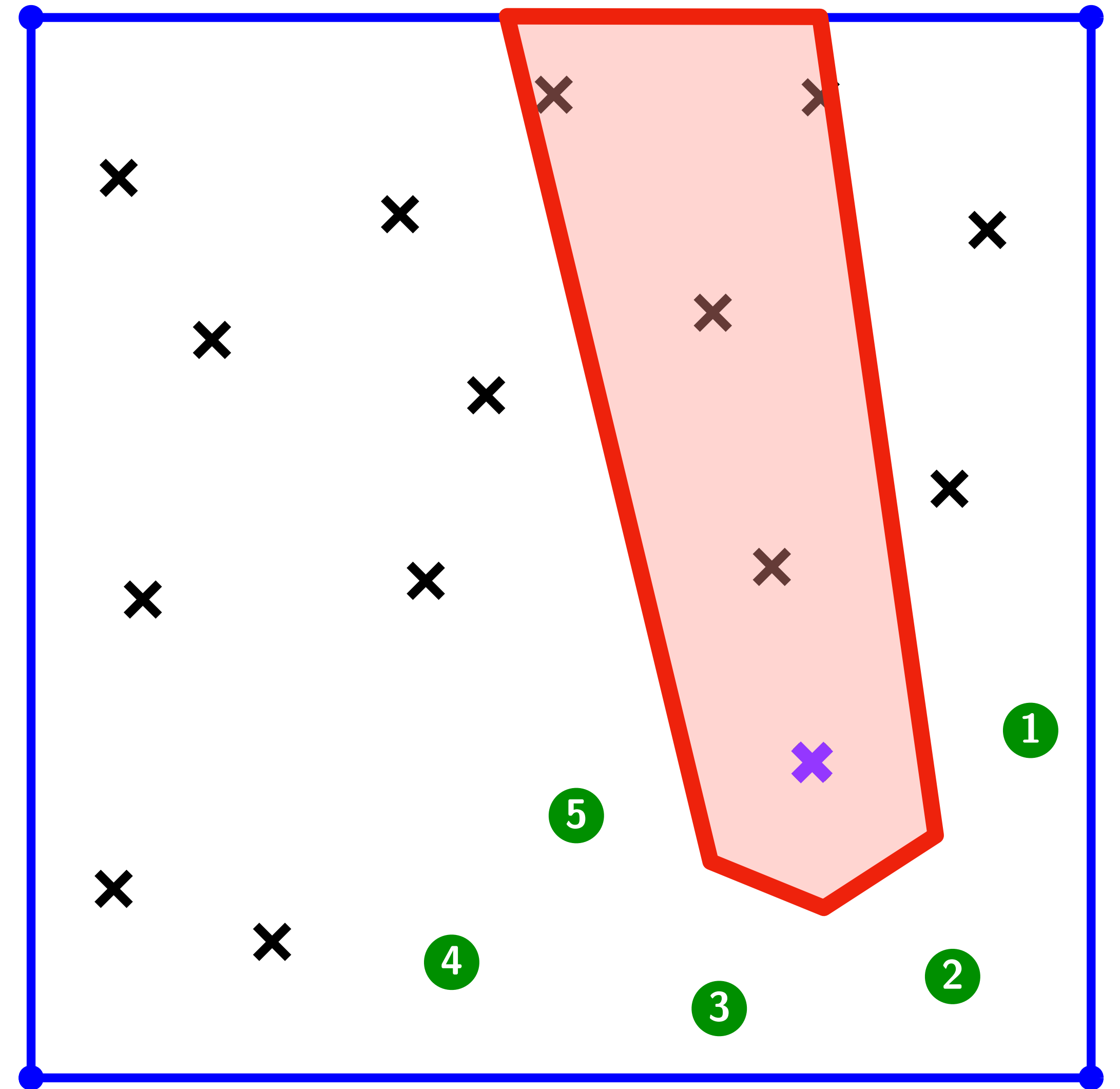
- An alternative method is to compute each Voronoi cell separately
- Initialize Voronoi cell to fill  $\Omega$
- Each neighboring particle cuts the Voronoi cell by a plane that is a perpendicular bisector
- After enough nearby particles are considered, the Voronoi cell is complete





# Calculating an individual Voronoi cell

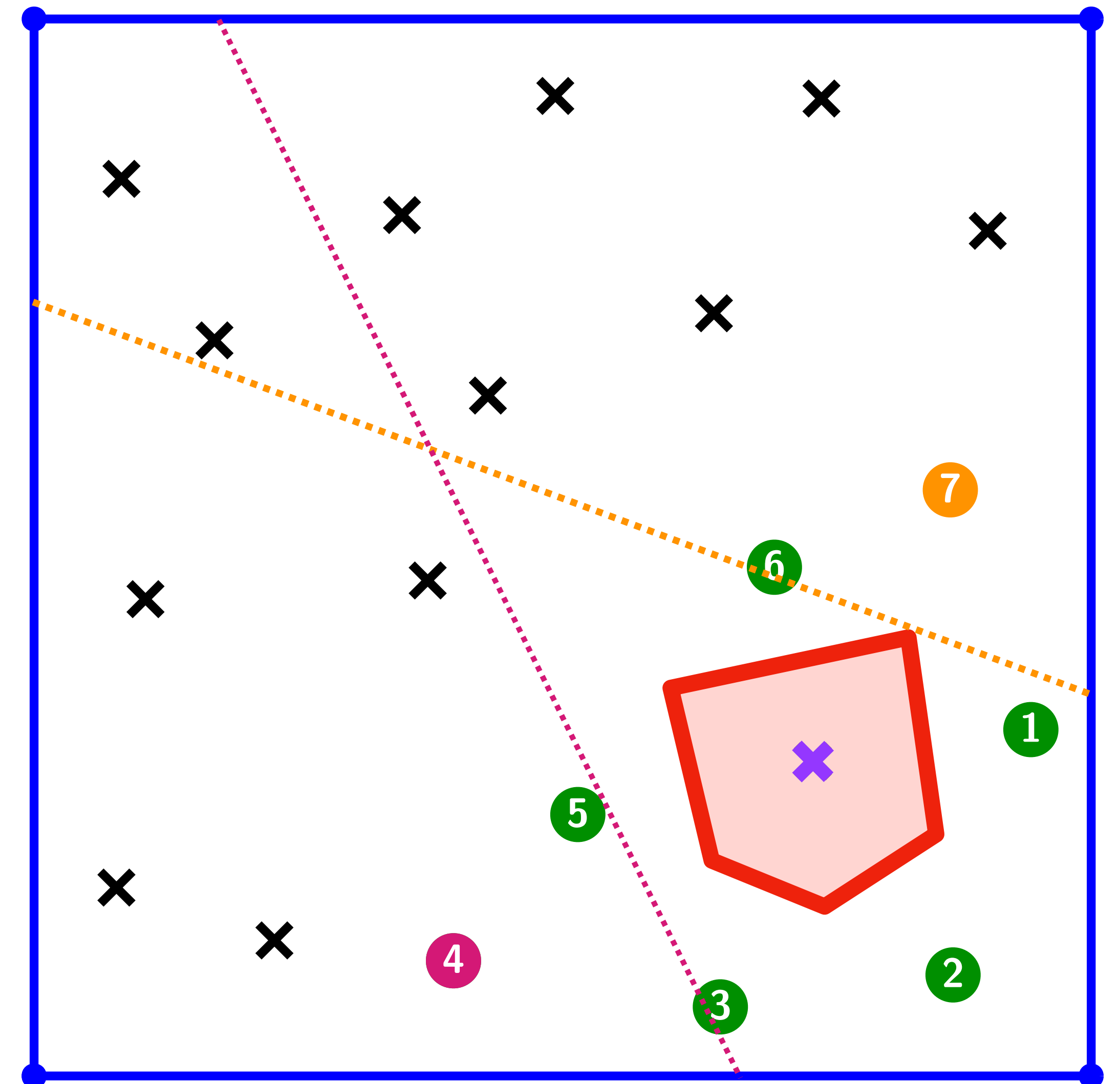
- An alternative method is to compute each Voronoi cell separately
- Initialize Voronoi cell to fill  $\Omega$
- Each neighboring particle cuts the Voronoi cell by a plane that is a perpendicular bisector
- After enough nearby particles are considered, the Voronoi cell is complete





# Calculating an individual Voronoi cell

- An alternative method is to compute each Voronoi cell separately
- Initialize Voronoi cell to fill  $\Omega$
- Each neighboring particle cuts the Voronoi cell by a plane that is a perpendicular bisector
- After enough nearby particles are considered, the Voronoi cell is complete



7 : considered, but didn't cut cell

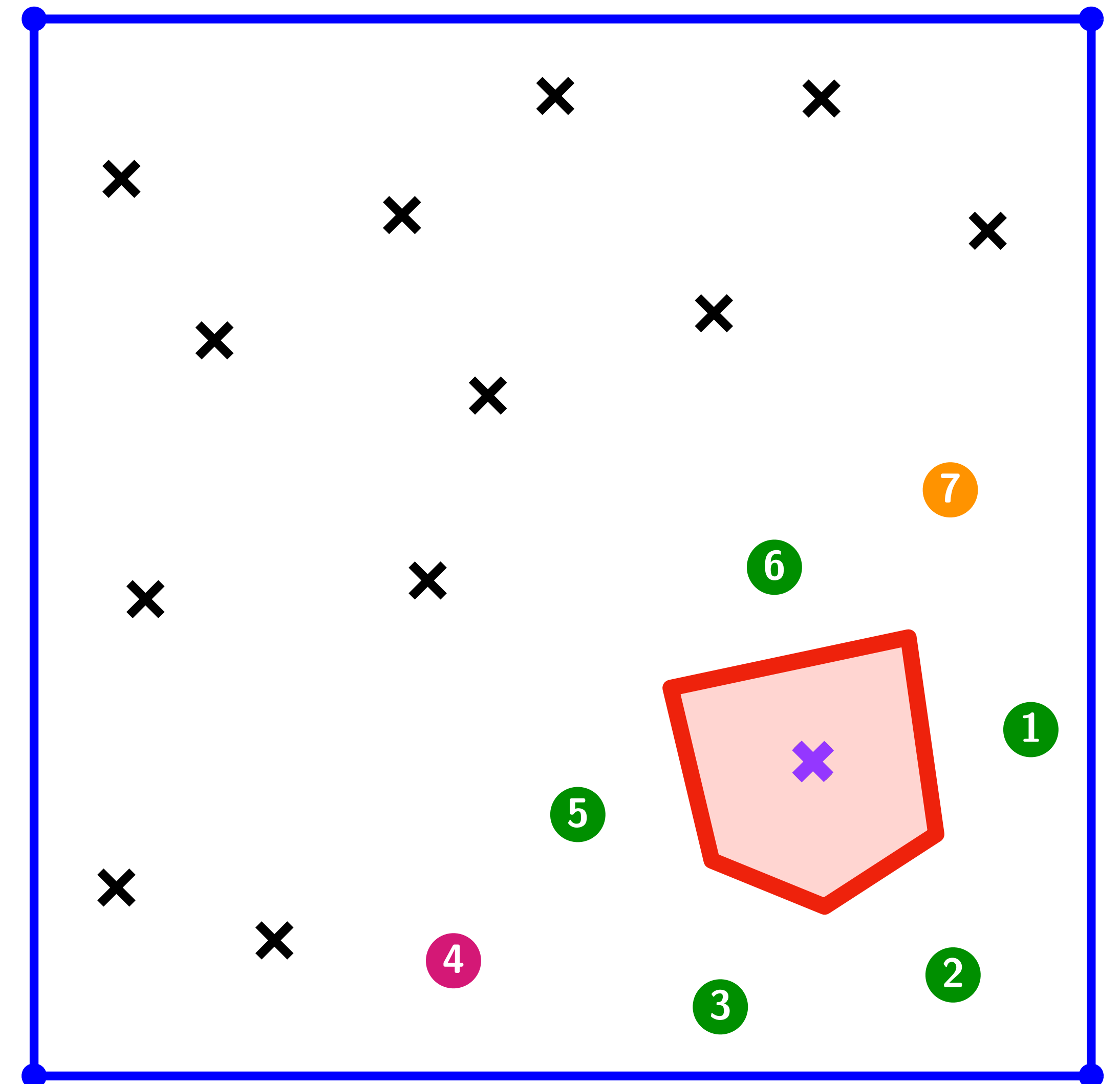
4 : cut cell, but ultimately not needed

1,2,3,5,6 : required



# Computational considerations

- Each cell can be computed independently; has advantages for parallel computation
- The algorithm needs to consider nearby particles efficiently—we want to find the **green** particles, and minimize choosing **orange/magenta** particles
- The algorithm has some drawbacks—we will return to this later



7 : considered, but didn't cut cell

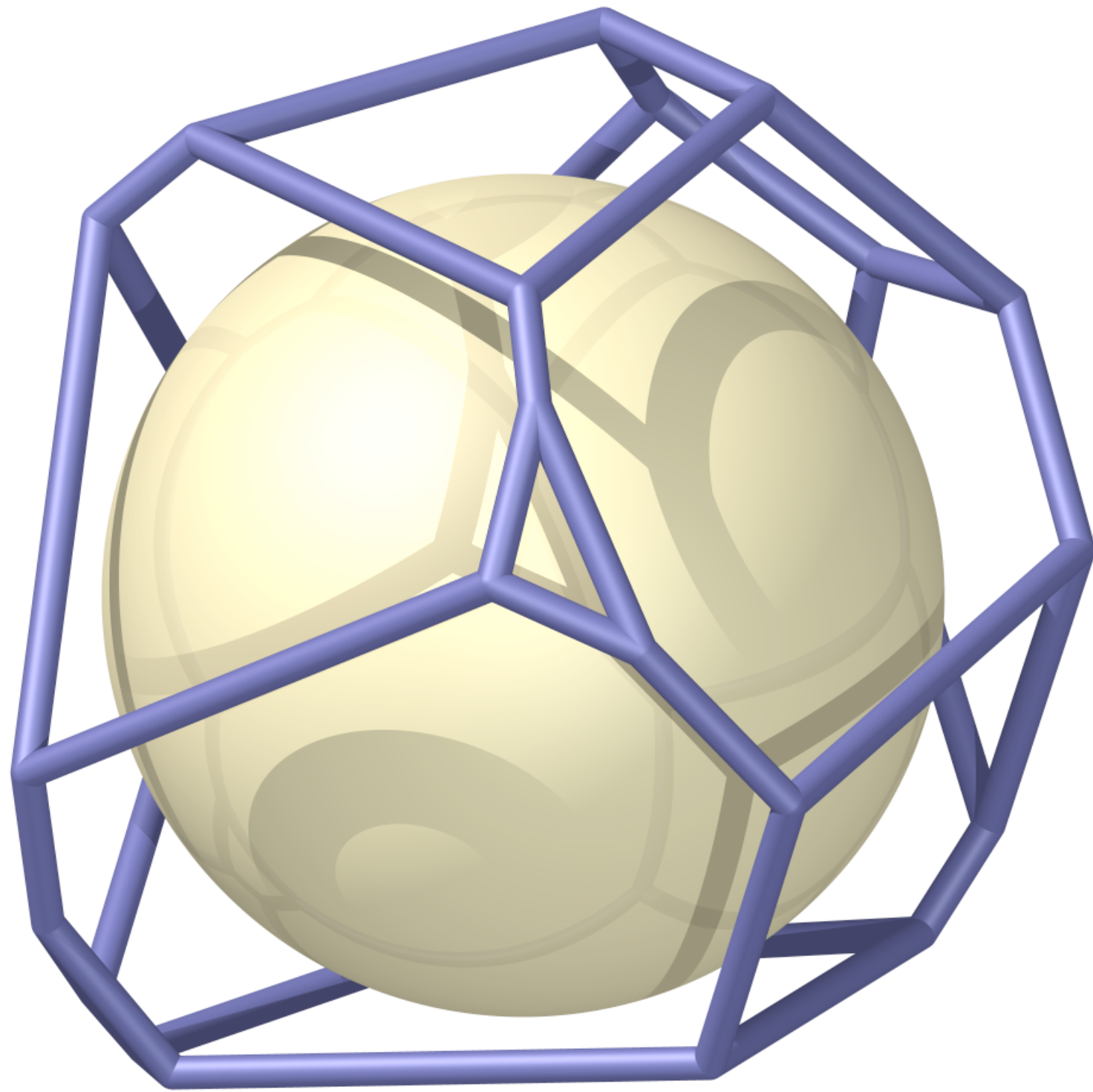
4 : cut cell, but ultimately not needed

1,2,3,5,6 : required



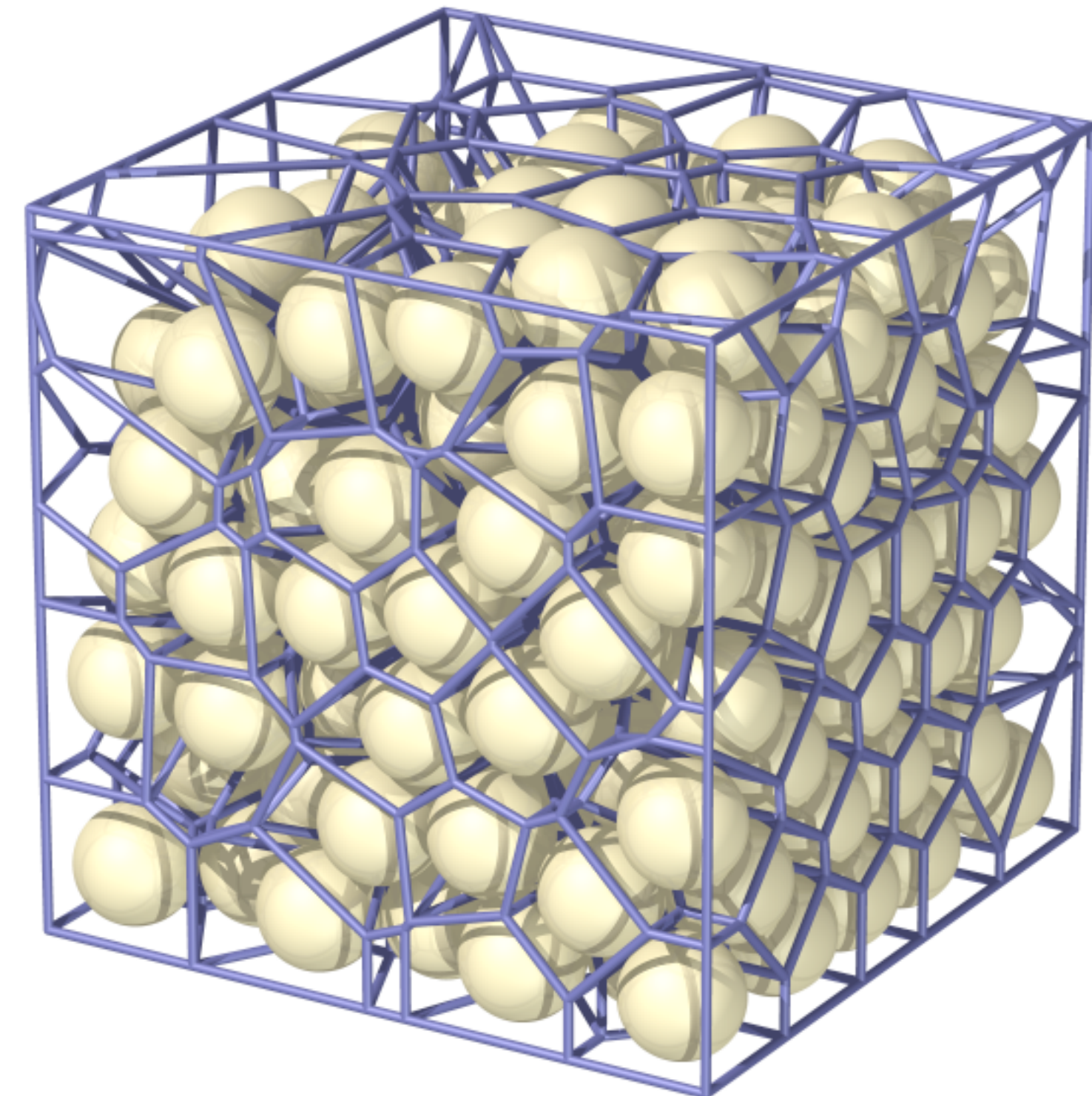
# Voro++, a software library for cell-based Voronoi calculations

- Primary audience: diagnostics for particle-based simulation
- Good performance (90,000 cells per second on a typical desktop, 3x faster than Qhull)
- Written in C++ around several *classes*



`voronocell_3d` class

(Code module for doing single-cell computations)



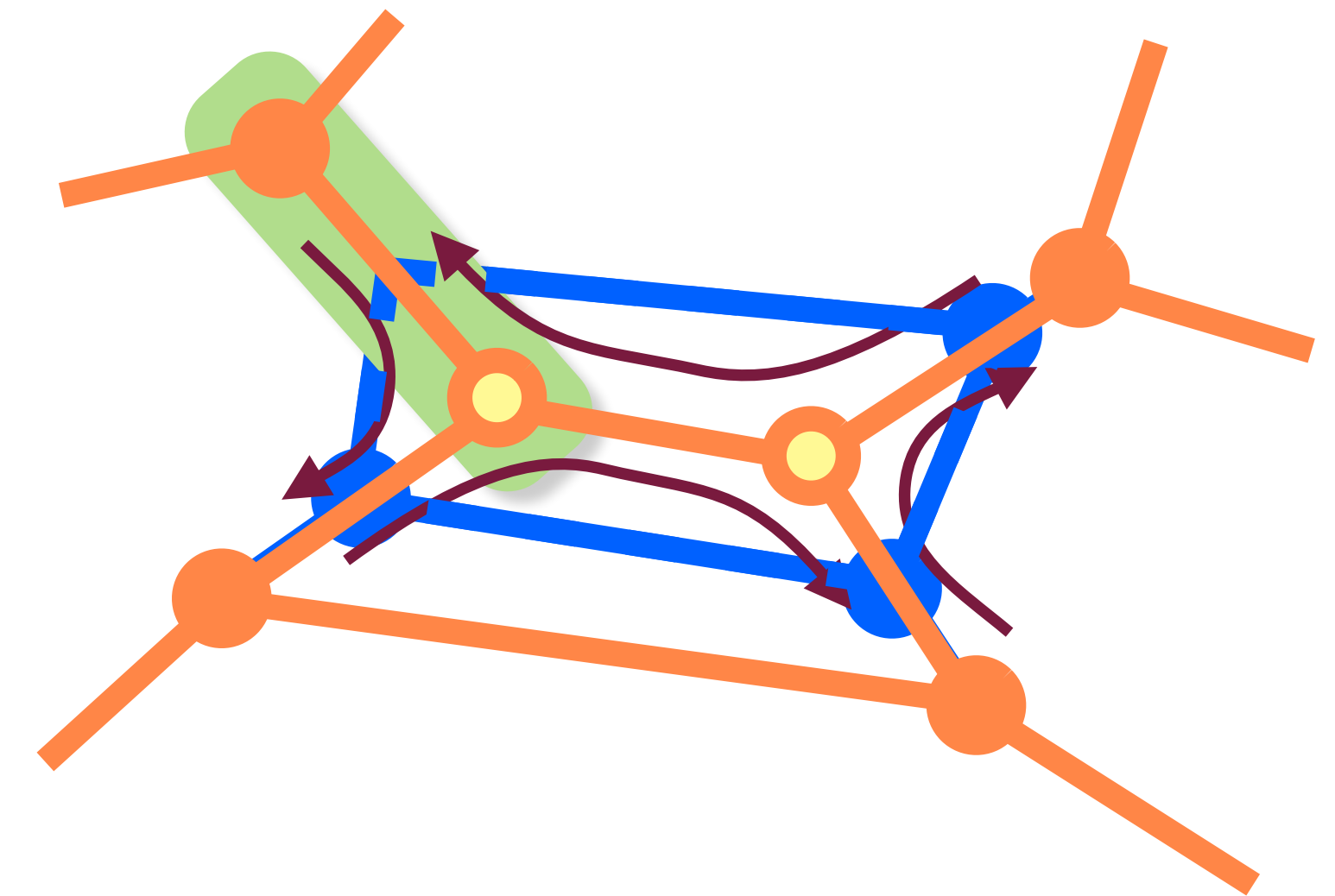
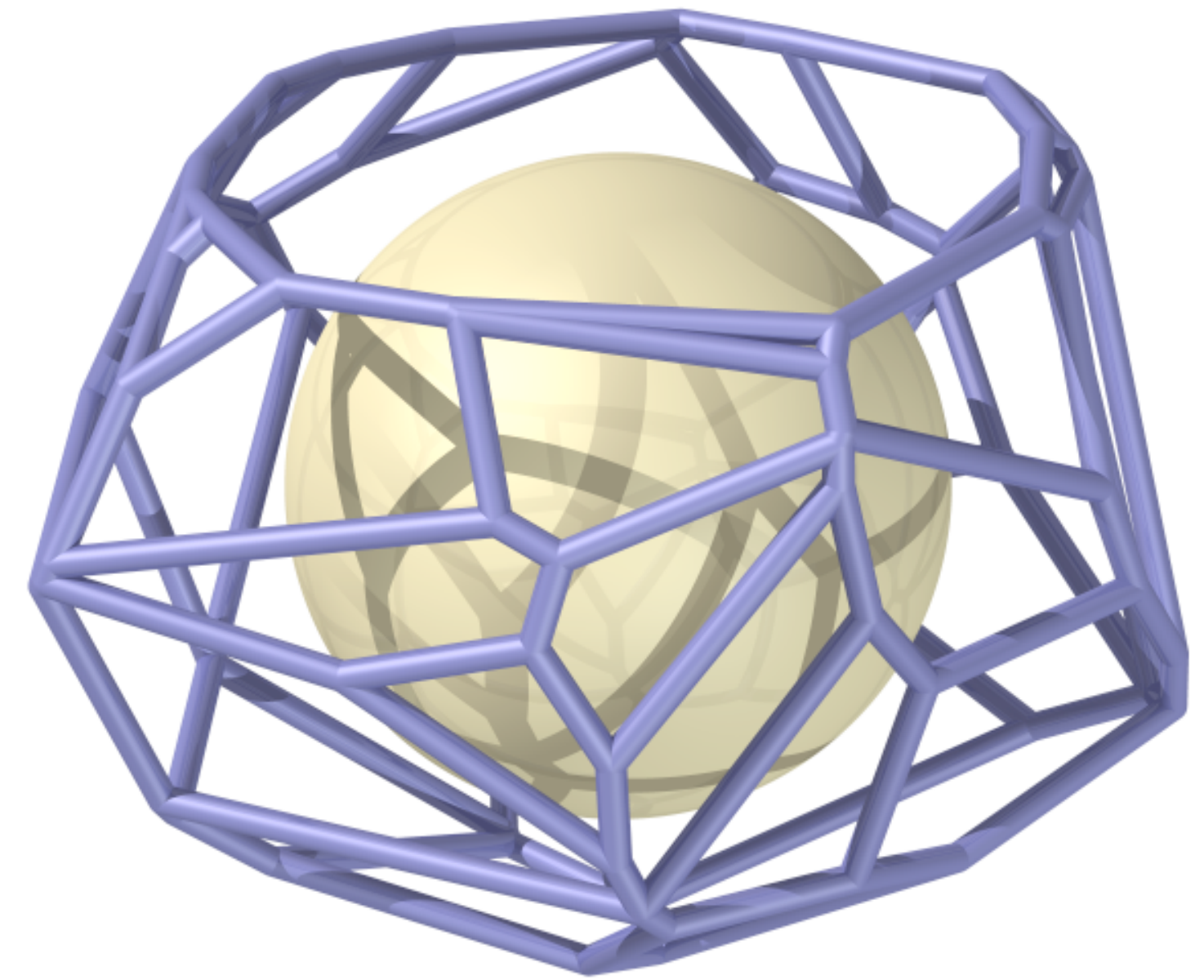
`container_3d` class

(Code module for Voronoi analysis of particle arrangements in a domain)



# The voronocell\_3d class

- An irregular convex polyhedron described by:
  - Vertex positions
  - Table of edges
- Primary operation: recompute vertices when cut by a plane
- Diagnostic routines:
  - Number of planes, edges
  - Volume, centroid, *etc.*



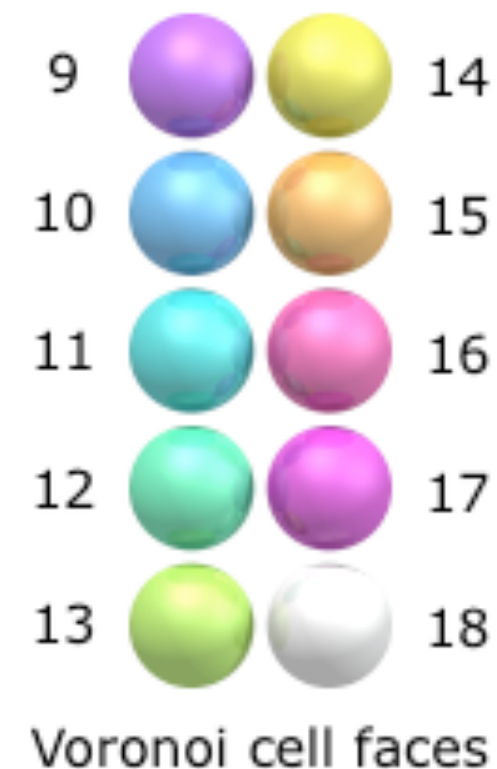
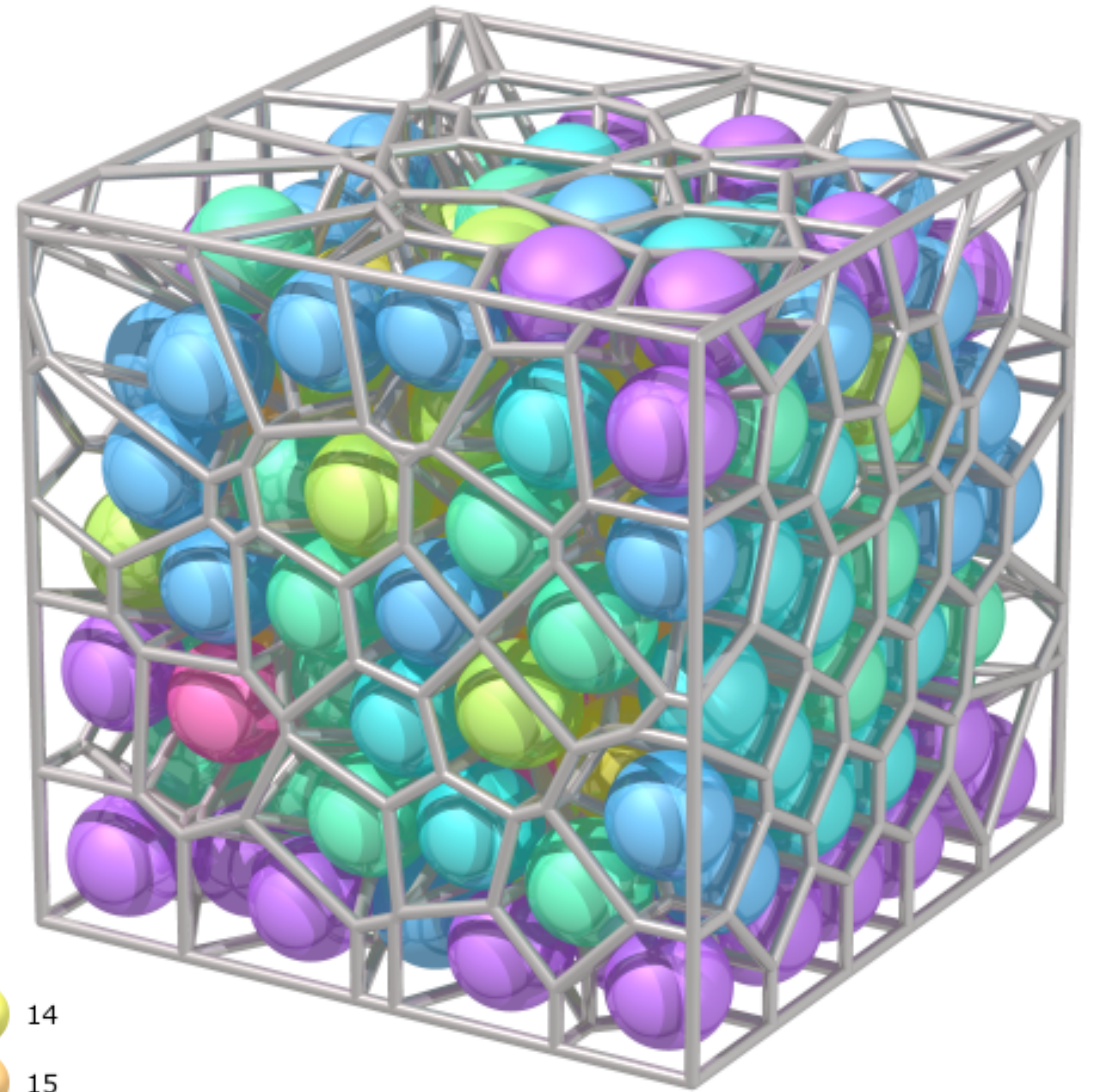
● : inside

● : outside



# The `container_3d` class

- Sorts particles into rectangular grid of blocks
- Uses `voronoicell` class to construct individual cells by testing over the blocks
- Can carry out a variety of calculations on the computed cells
- Library and example programs available for download from <http://math.lbl.gov/voro++/>



Voronoi tessellation with  
216 particles in a cube of side length 6,  
generated by DEVSIM simulation  
the Voronoi cell faces