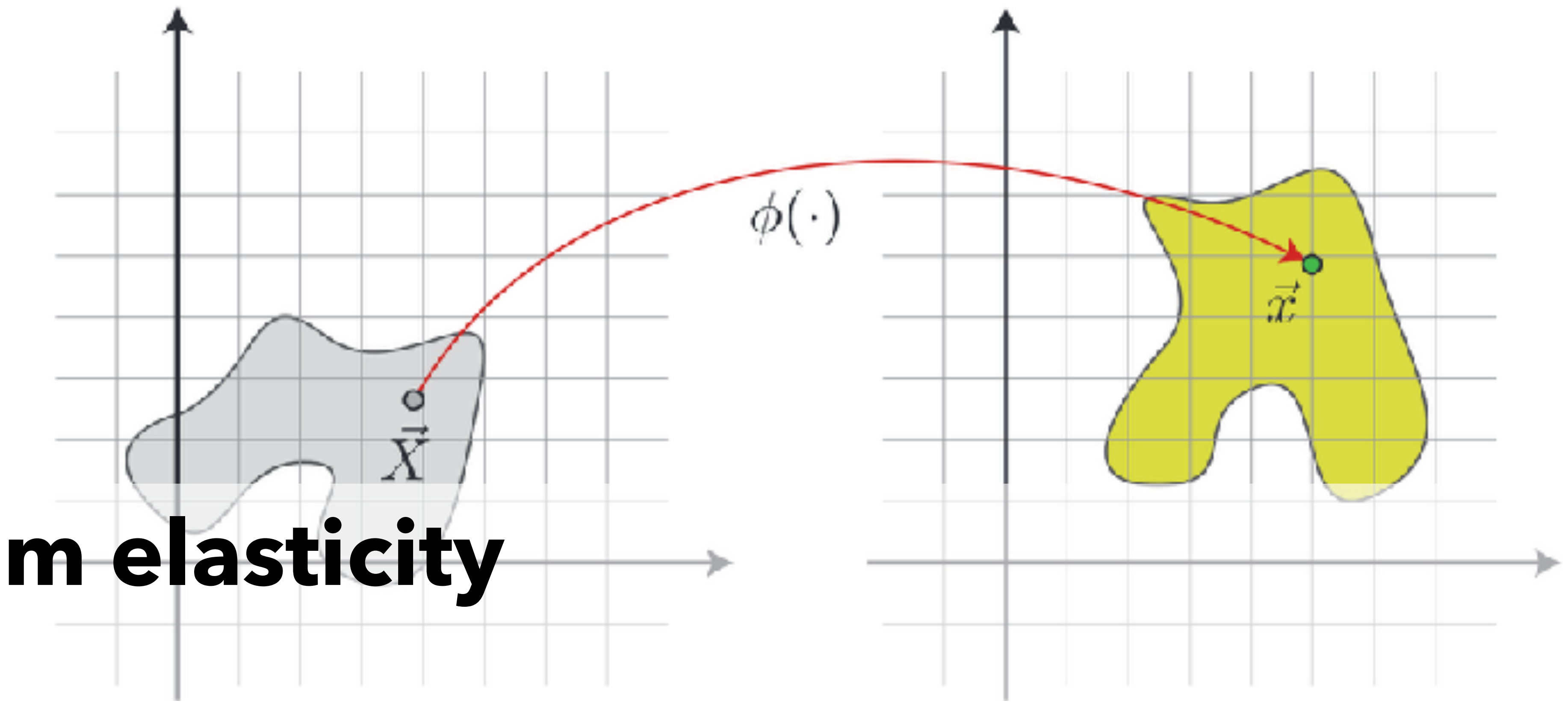

COL781: Computer Graphics
39. Fluid Simulation
Chern et al. 2016

Recap: Elastic solids



Mass-spring systems

Degrees of freedom: $\mathbf{q} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$

Deformed shape of one spring: $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$

Strain: $\varepsilon = \dfrac{\|\mathbf{x}_{ij}\|}{\|\mathbf{X}_{ij}\|} - 1$

Spring energy: $U_{ij} = ½\, k_s\, \varepsilon\, \|\mathbf{X}_{ij}\|$

Total internal energy: $U = \sum U_{ij}$

Force on ith particle: $\mathbf{f}_i = -\dfrac{\partial U}{\partial \mathbf{x}_i} = -\sum \dfrac{\partial U_{ij}}{\partial \mathbf{x}_i}$

Continuum elasticity

Degrees of freedom: $\varphi : \mathbf{X} \to \mathbf{x}$

Deformation of infinitesimal patch: $\mathbf{F} = \dfrac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{X}}$

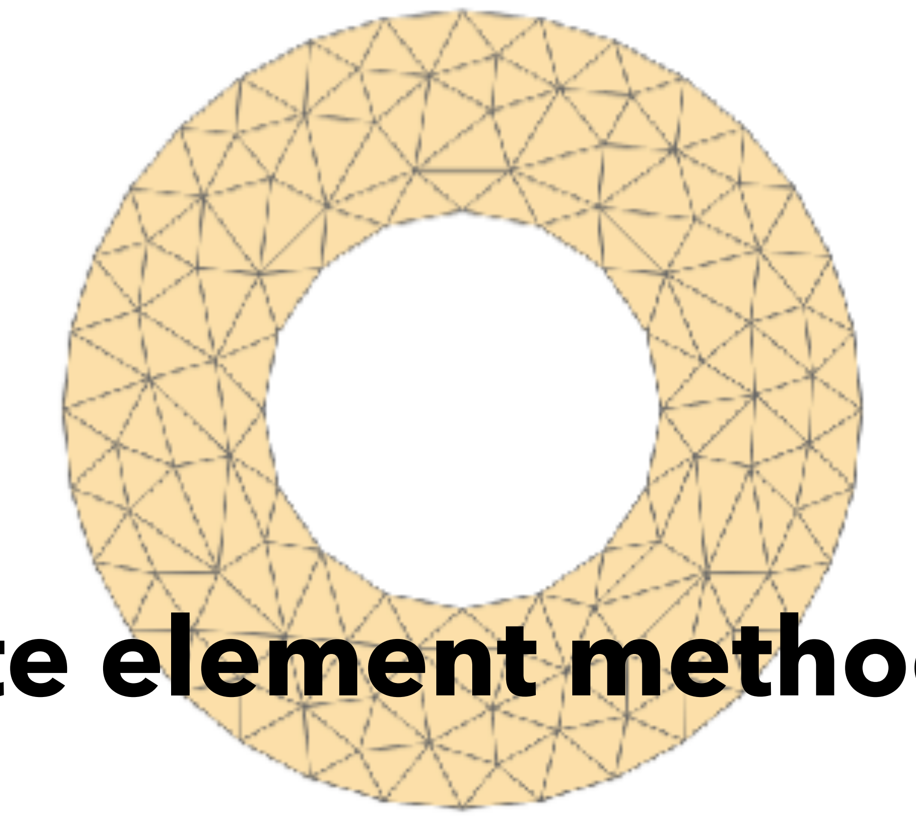
Strain: $\mathbf{E} = ½\,(\mathbf{F}^\mathsf{T}\mathbf{F} - \mathbf{I})$

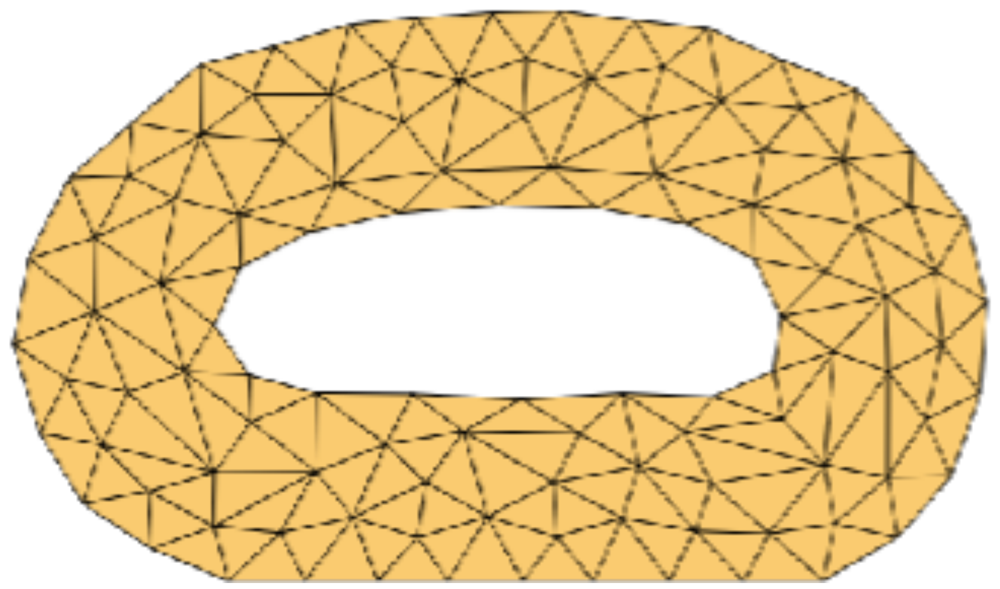
Strain energy density: $\Psi(\mathbf{E})$

Total internal energy: $U = \int \Psi(\mathbf{E})\, \mathrm{d}V$

Generalized force: $-\dfrac{\partial U}{\partial \phi}$?

Recap: Elastic solids





Mass-spring systems

Degrees of freedom: $\mathbf{q} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$

Deformed shape of one spring: $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$

Strain: $\varepsilon = \dfrac{\|\mathbf{x}_{ij}\|}{\|\mathbf{X}_{ij}\|} - 1$

Spring energy: $U_{ij} = \frac{1}{2}\, k_s\, \varepsilon\, \|\mathbf{X}_{ij}\|$

Total internal energy: $U = \sum U_{ij}$

Force on ith particle: $\mathbf{f}_i = -\dfrac{\partial U}{\partial \mathbf{x}_i} = -\sum \dfrac{\partial U_{ij}}{\partial \mathbf{x}_i}$

Finite element method

Degrees of freedom: $\mathbf{q} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$

Deformation of jth element: $\mathbf{F}_j = \dfrac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{X}}$

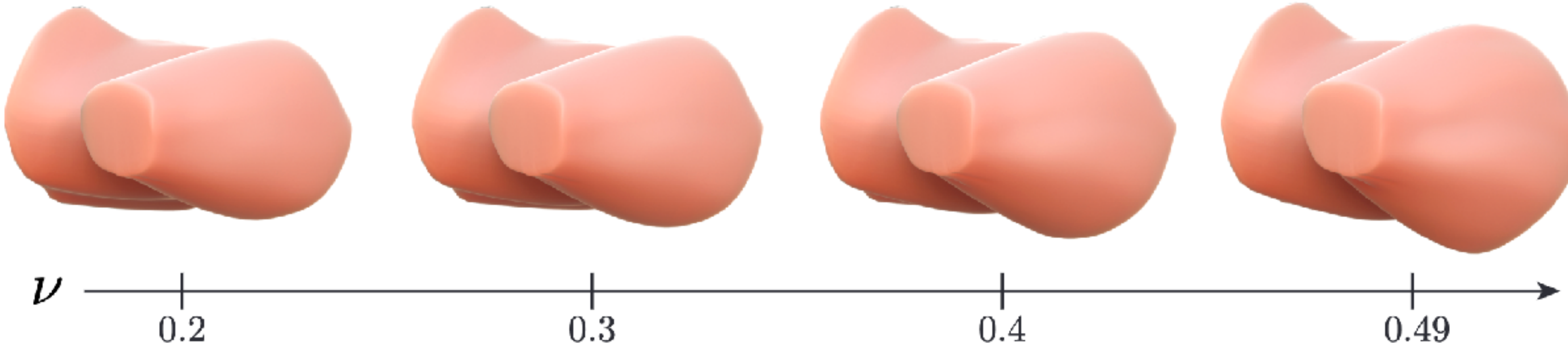
Strain: $\mathbf{E} = \frac{1}{2}\,(\mathbf{F}^\top \mathbf{F} - \mathbf{I})$

Strain energy density: $\Psi(\mathbf{E})$

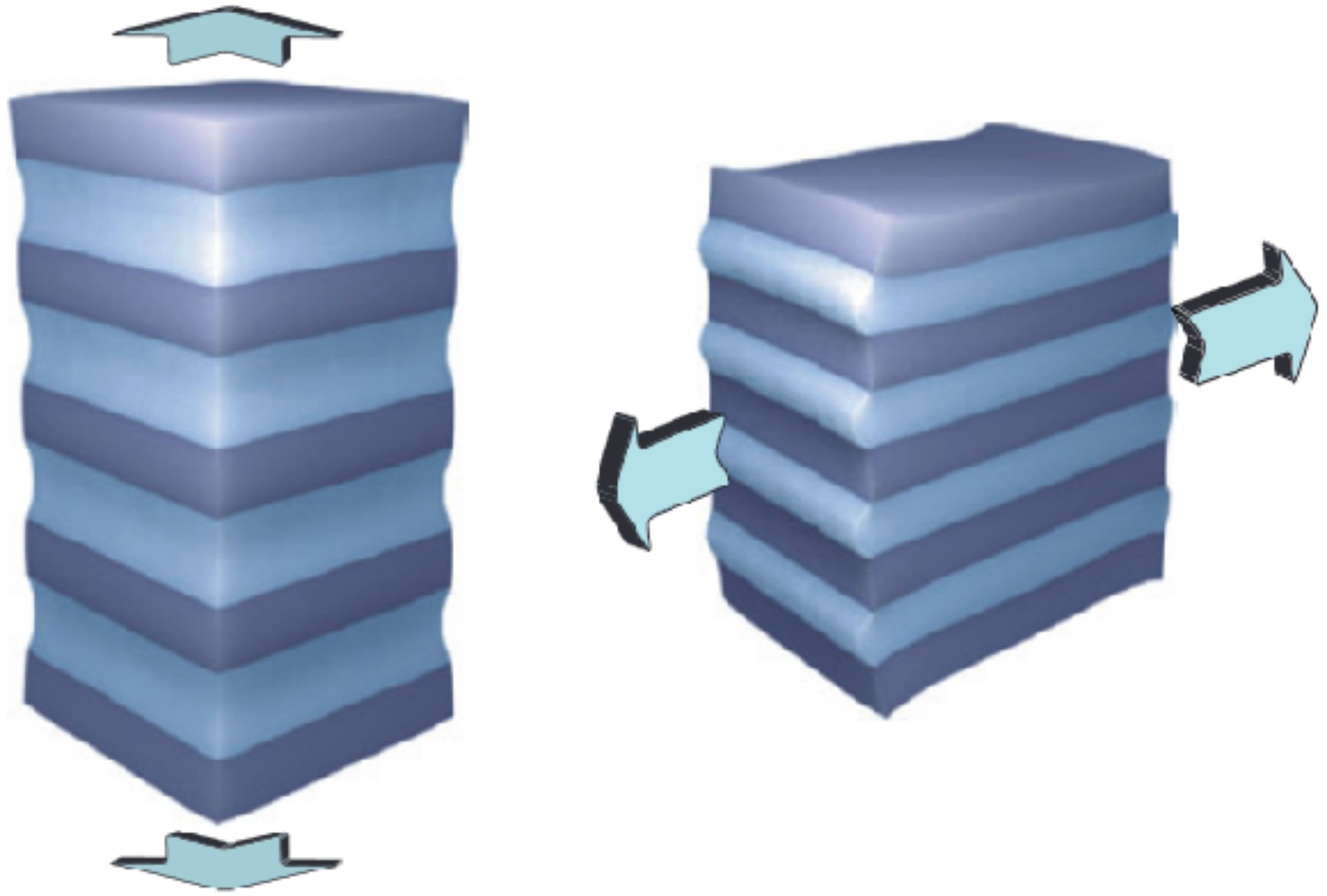
Total internal energy: $U = \sum \Psi(\mathbf{E}_j)\, \mathrm{d}V_j$

Force on ith particle: $\mathbf{f}_i = -\dfrac{\partial U}{\partial \mathbf{x}_i} = -\sum \dfrac{\partial U_j}{\partial \mathbf{x}_i}$

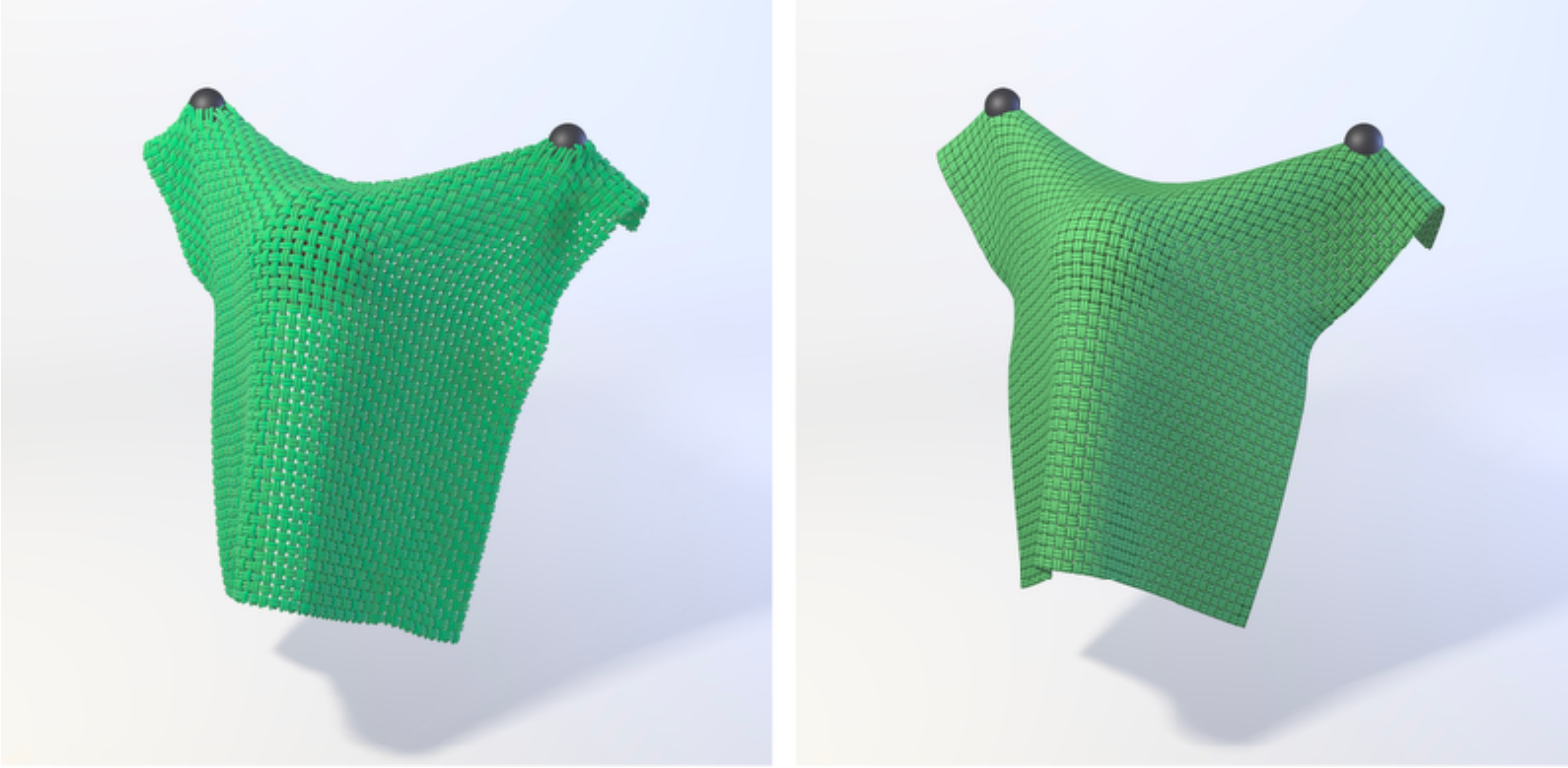
Choice of strain energy density $\Psi(\mathbf{E})$ determines material behaviour, including volume preservation (Poisson's ratio), anisotropy, and all other effects



Smith et al. 2018



Kharevych et al. 2009



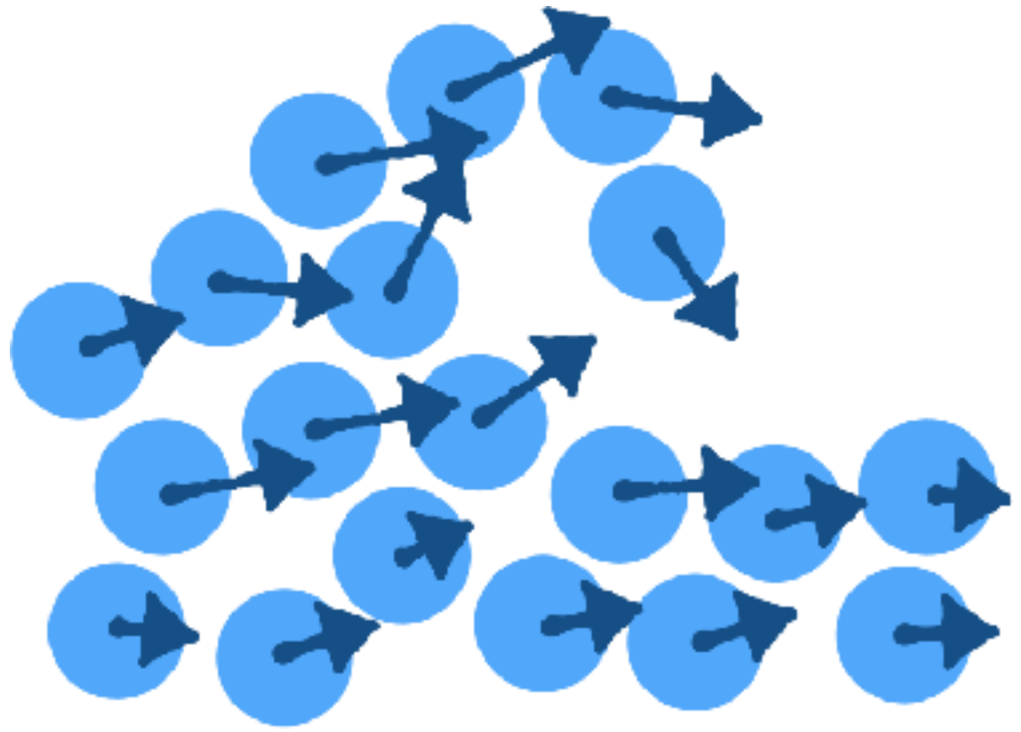
Sperl et al. 2020

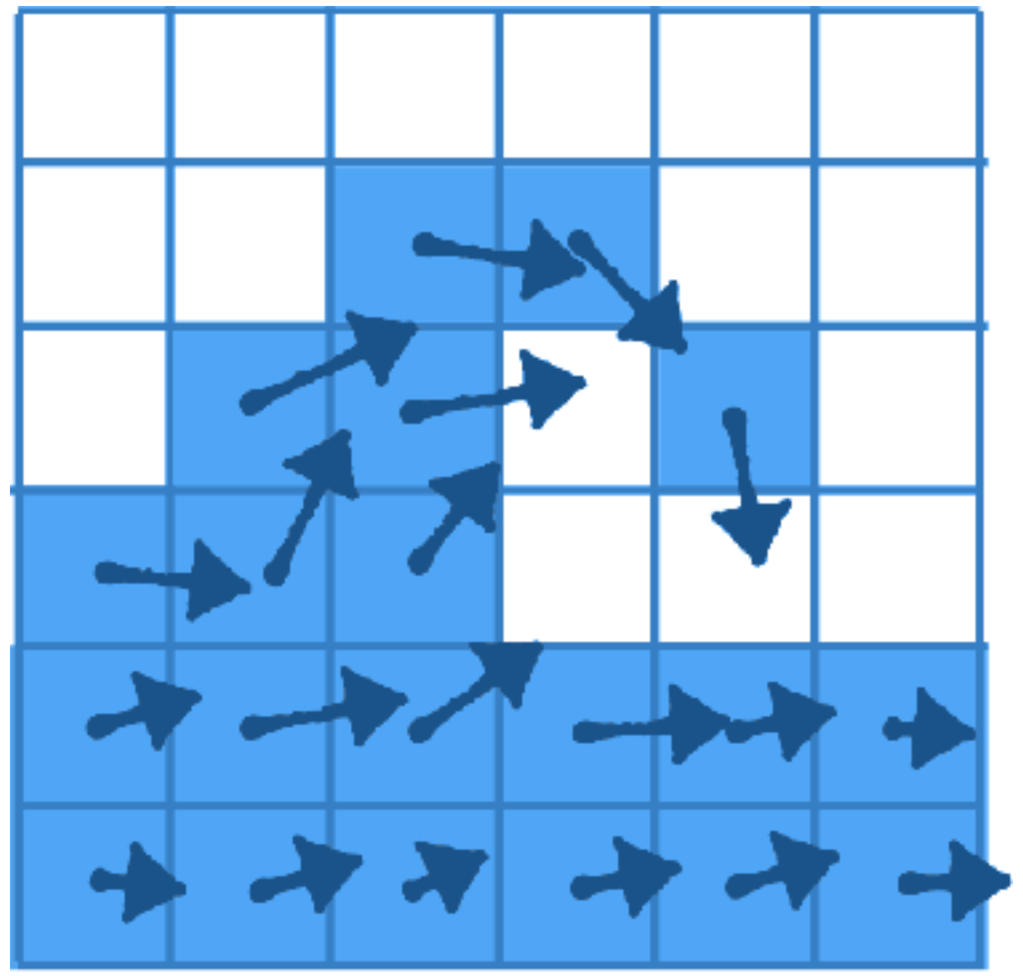
Fluids

No rest shape, so no reference space **X** needed.
No deformation map **x**(**X**), no time derivative **v**(**X**) = $\dot{\mathbf{x}}$(**X**)

Still need **v** as a function of **x** though: the velocity field

Can discretize using particles or a grid:



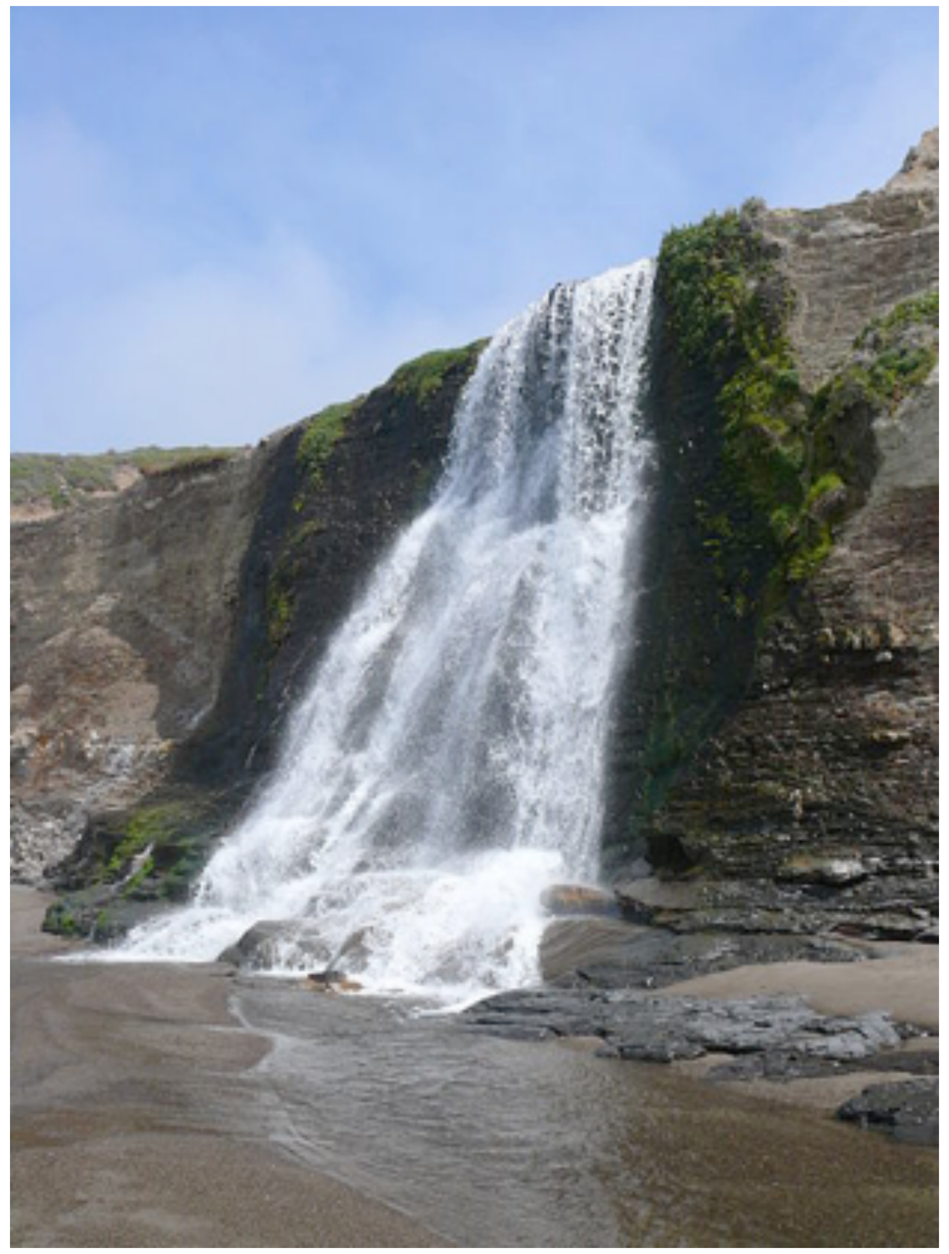


Forces acting on the fluid:

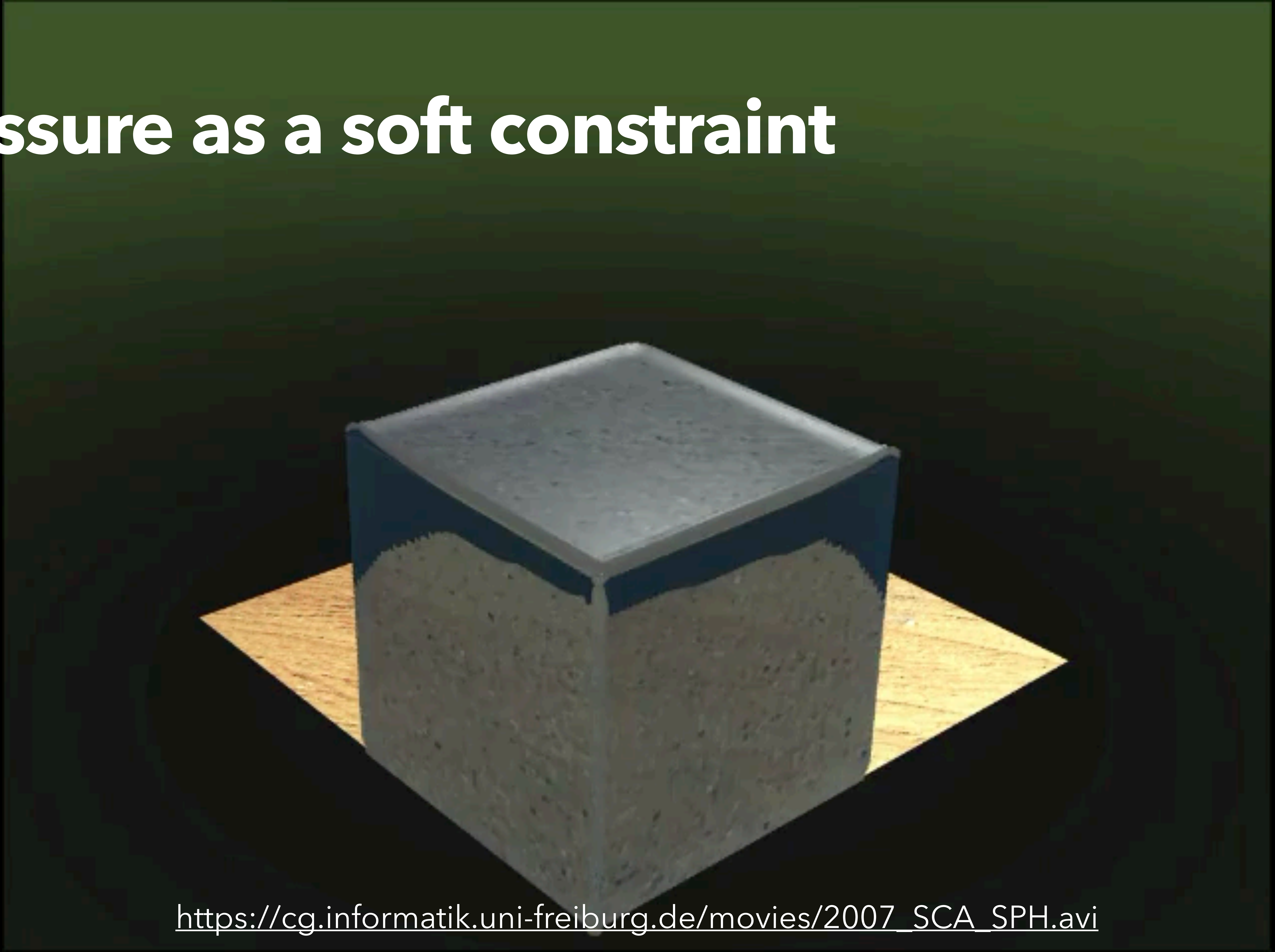
- External forces e.g. gravity
- Pressure $\mathbf{f}_{\text{pres}} = -\nabla p(\mathbf{x})$

  Fluid is pushed away from high pressure towards low pressure
- Viscosity $\mathbf{f}_{\text{visc}} = \mu \, \nabla^2 \, \mathbf{u}(\mathbf{x})$

  Resists relative motion within the fluid
- ~~Surface tension (out of scope)~~
- ~~Interaction with solids (out of scope)~~



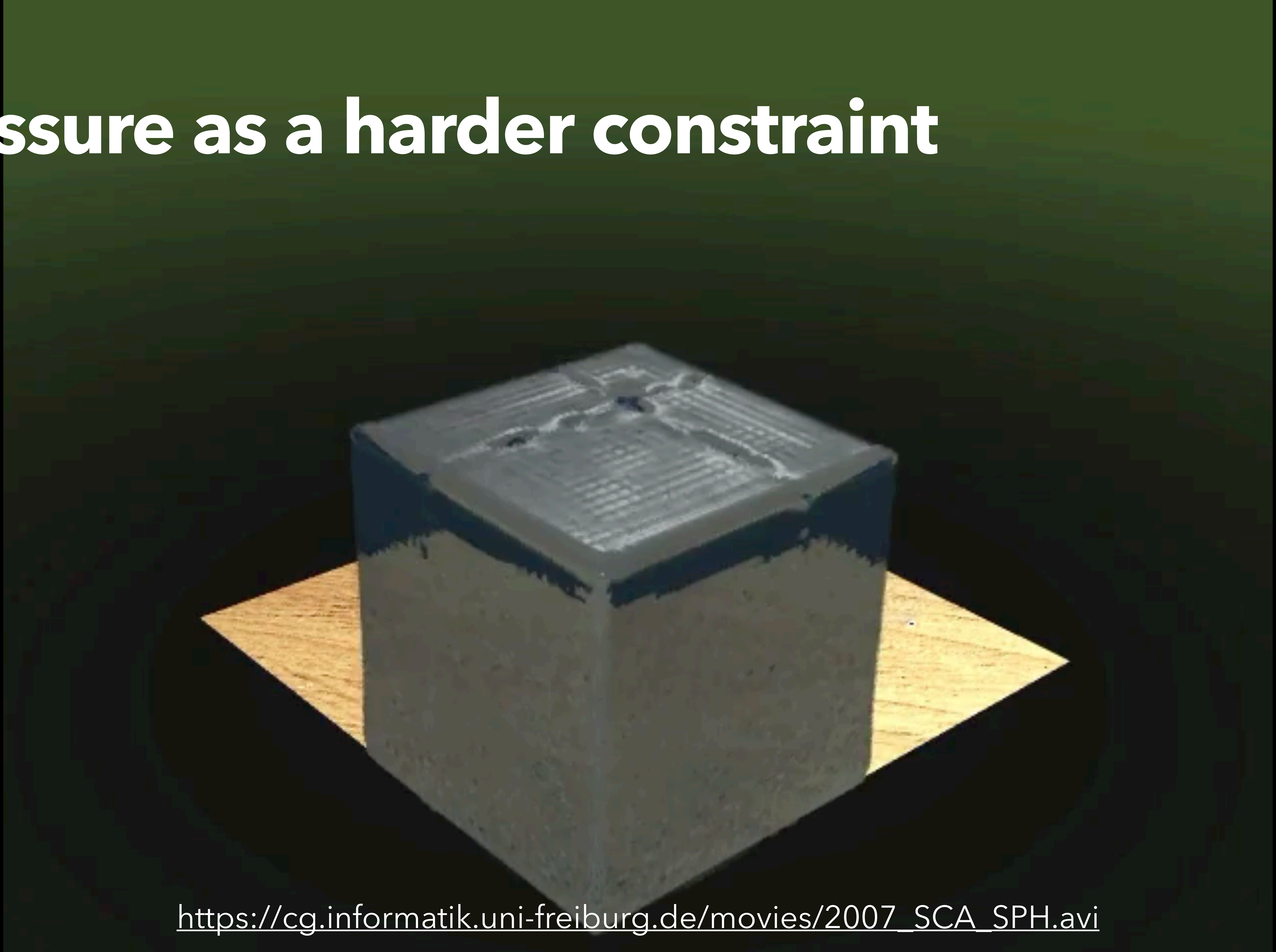
Pressure as a soft constraint



https://cg.informatik.uni-freiburg.de/movies/2007_SCA_SPH.avi

Becker & Teschner 2007

Pressure as a harder constraint



https://cg.informatik.uni-freiburg.de/movies/2007_SCA_SPH.avi

Becker & Teschner 2007

Pressure

In real life, pressure is a restoring force that opposes changes in density.

Restoring force → oscillations (sound, shock waves)!

In simulation, we'll treat fluid as perfectly incompressible → no oscillations, stable!

Change in volume of any "blob" of fluid should be zero everywhere:

$$(\nabla \cdot \mathbf{u})(\mathbf{x}) = 0$$

Solve for the pressure field p so that, after applying the pressure force $-\nabla p(\mathbf{x})$, the above remains true.

The Navier-Stokes equations

$$\frac{D\mathbf{u}}{Dt} = \mathbf{f}_{ext} - \nabla p + \mu \nabla^2 \mathbf{u}$$

Wait, what's D/Dt?

- Partial derivative $\frac{\partial \mathbf{u}(t, \mathbf{x})}{\partial t}$ = time derivative at a fixed point $\mathbf{x}$
- Material derivative $\frac{D\mathbf{u}(t, \mathbf{x})}{Dt}$ = time derivative seen by point moving with the fluid

Actually $\frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u}$

S. Blackaller-Johnson

Burazin

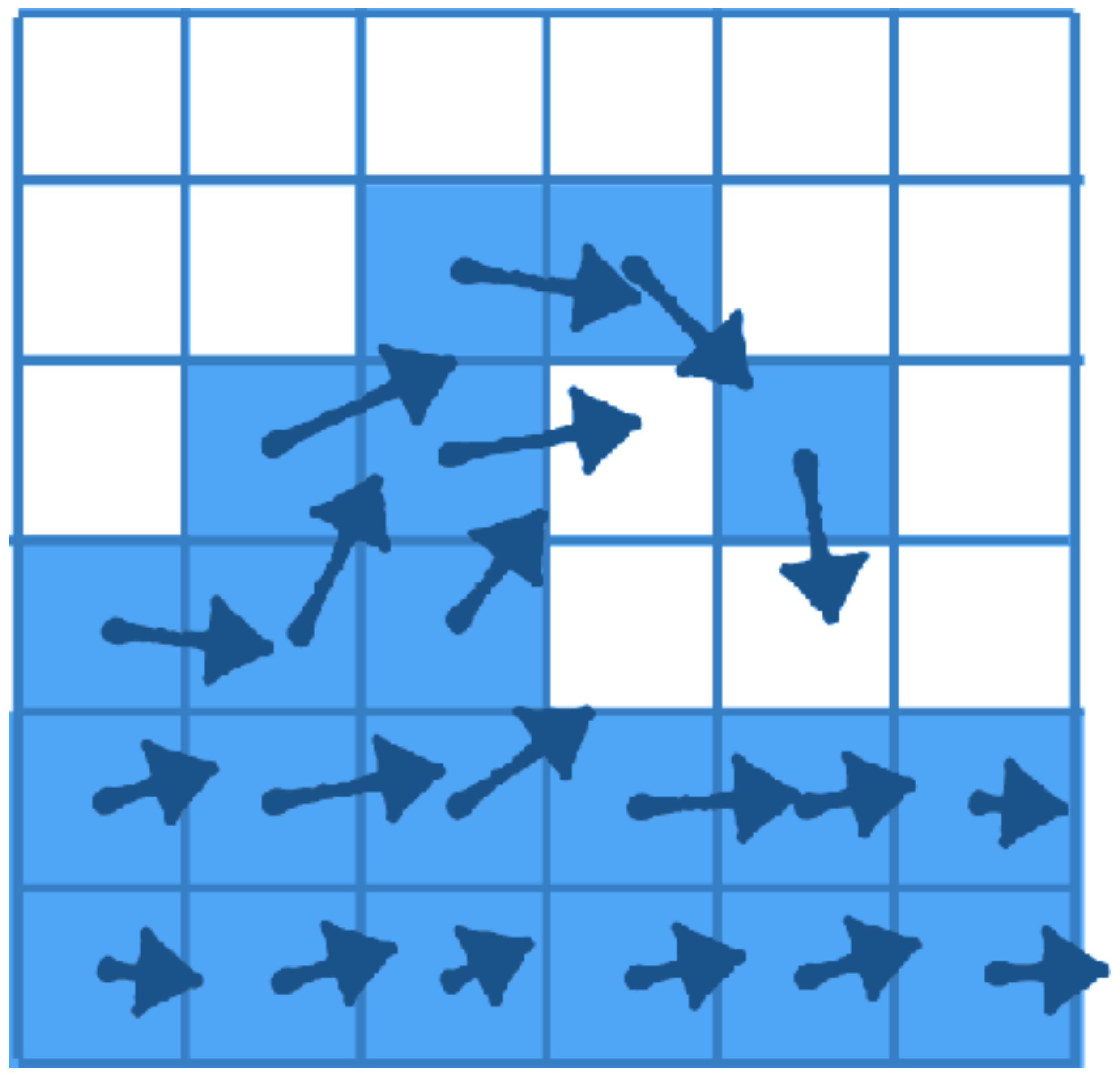
$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = \mathbf{f}_{\text{ext}} - \nabla p + \mu\,\nabla^2\,\mathbf{u}$$

$$\nabla \cdot \mathbf{u} = 0$$

How to solve this complicated thing?

Hard to handle all terms at once! But one at a time is easy: **splitting method**

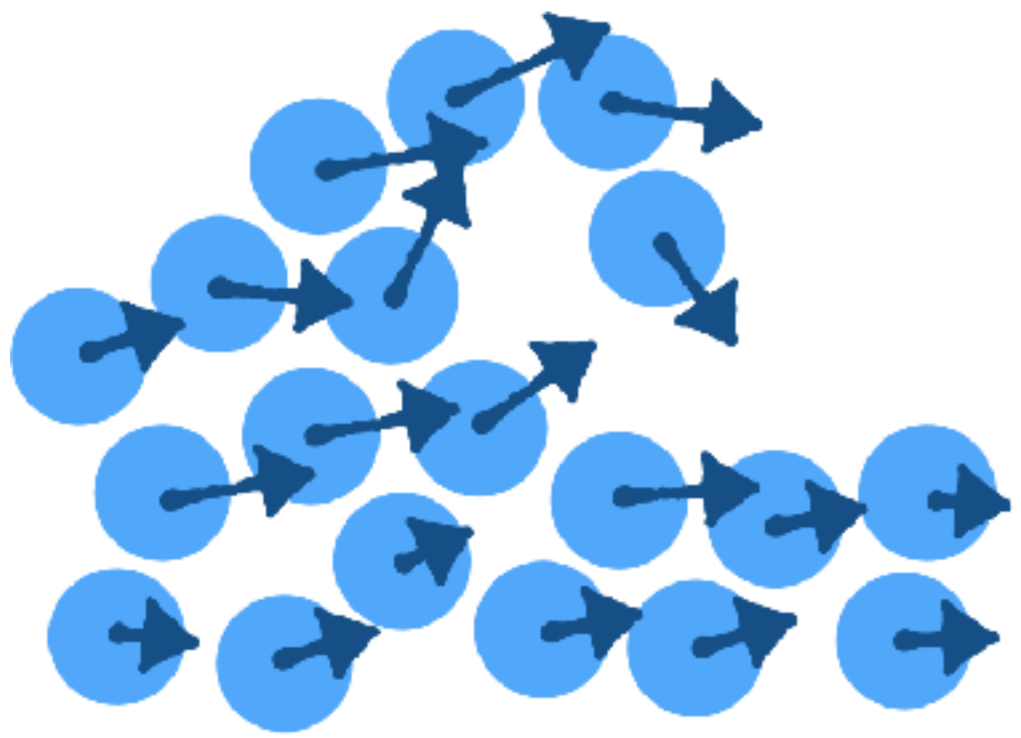
1. Advection: $\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = 0$
2. External forces: $\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}_{\text{ext}}$
3. Viscosity: $\frac{\partial \mathbf{u}}{\partial t} = \mu\,\nabla^2\,\mathbf{u}$
4. Pressure: $\frac{\partial \mathbf{u}}{\partial t} = -\nabla p,\ \nabla \cdot \mathbf{u} = 0$



Advection

$$\frac{Dq}{Dt} = \frac{\partial q}{\partial t} + (\mathbf{u} \cdot \nabla) q = 0$$

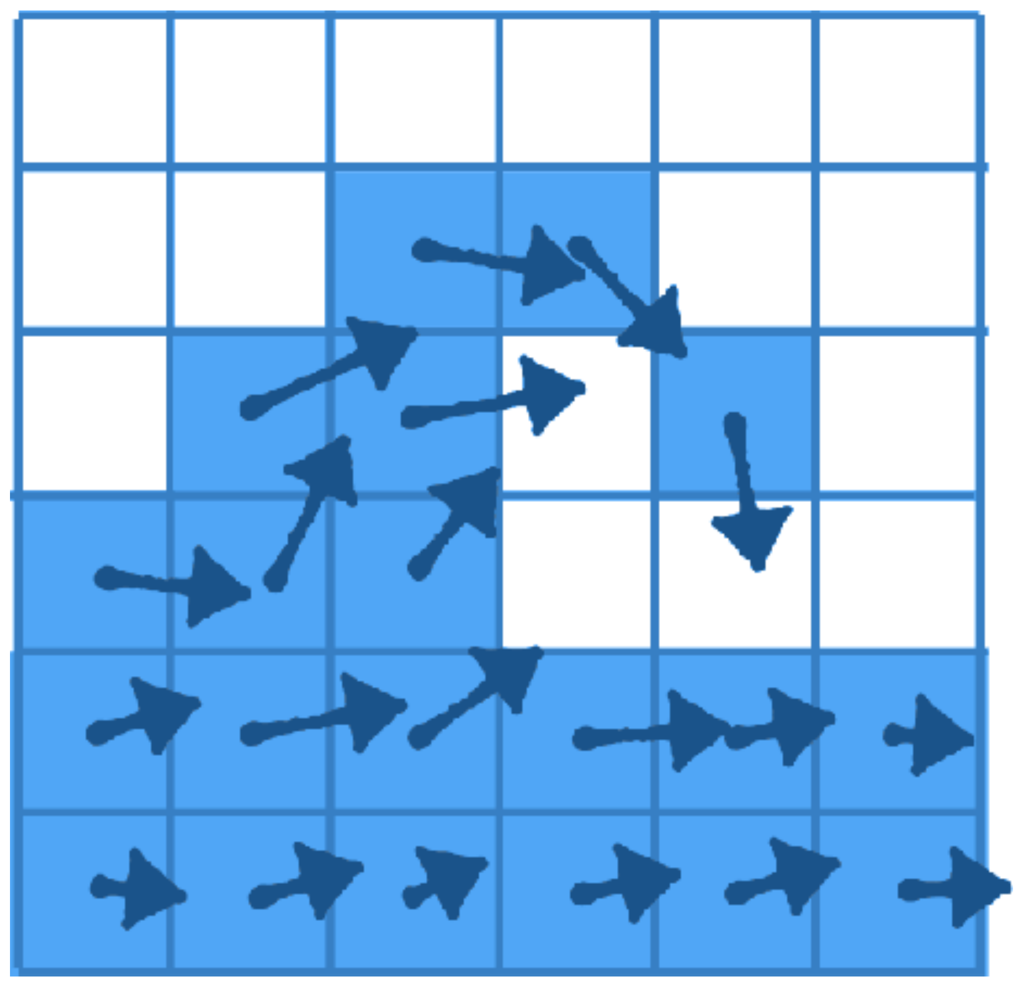
Basically says: just move everything using the velocity field **u**



Would be easy if we had particles:

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \mathbf{v}_i^n \Delta t$$

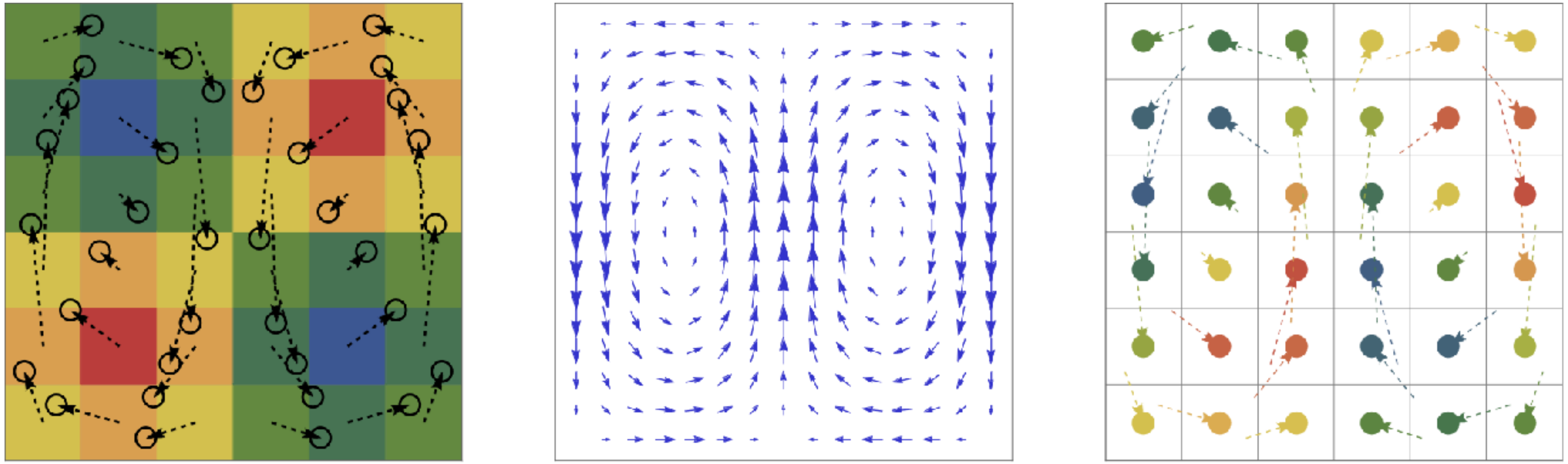
How to get the same effect on a grid?



- Create a temporary particle at each grid cell of q^n, move it forward for time Δt, write its data to q^{n+1} at the new position?

Semi-Lagrangian advection: figure out what location the particle should have started to land at $\mathbf{x}_i$, and pick up the value from there.

- Create a temporary particle at each grid cell of q^{n+1}
- Trace it backwards using current velocity field $\mathbf{u}$ for time $-\Delta t$
- Look up interpolated value $q^n(\mathbf{x})$ and write into original grid cell



q^n $\mathbf{u}$ q^{n+1}

Pressure

$$\frac{\partial \mathbf{u}}{\partial t} = -\nabla p$$
$$\nabla \cdot \mathbf{u} = 0$$

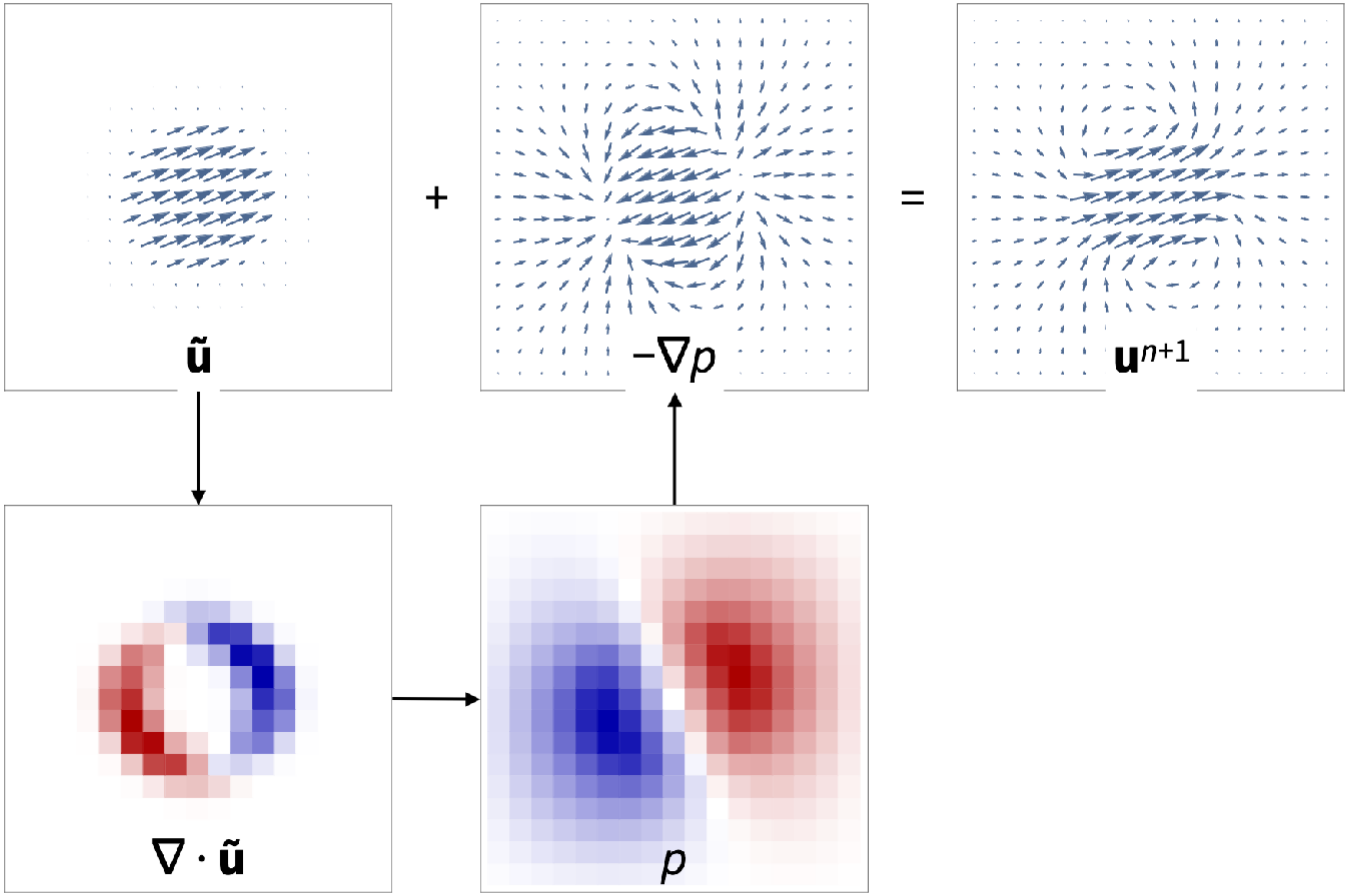
Standard constraint projection strategy:

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \nabla p \, \Delta t$$
$$\nabla \cdot \mathbf{u}^{n+1} = 0$$

Plug it in and solve for p:

$$\nabla^2 p \, \Delta t = \nabla \cdot \mathbf{u}^n$$

This is a Laplace problem with nonzero right-hand side, a.k.a. a Poisson problem



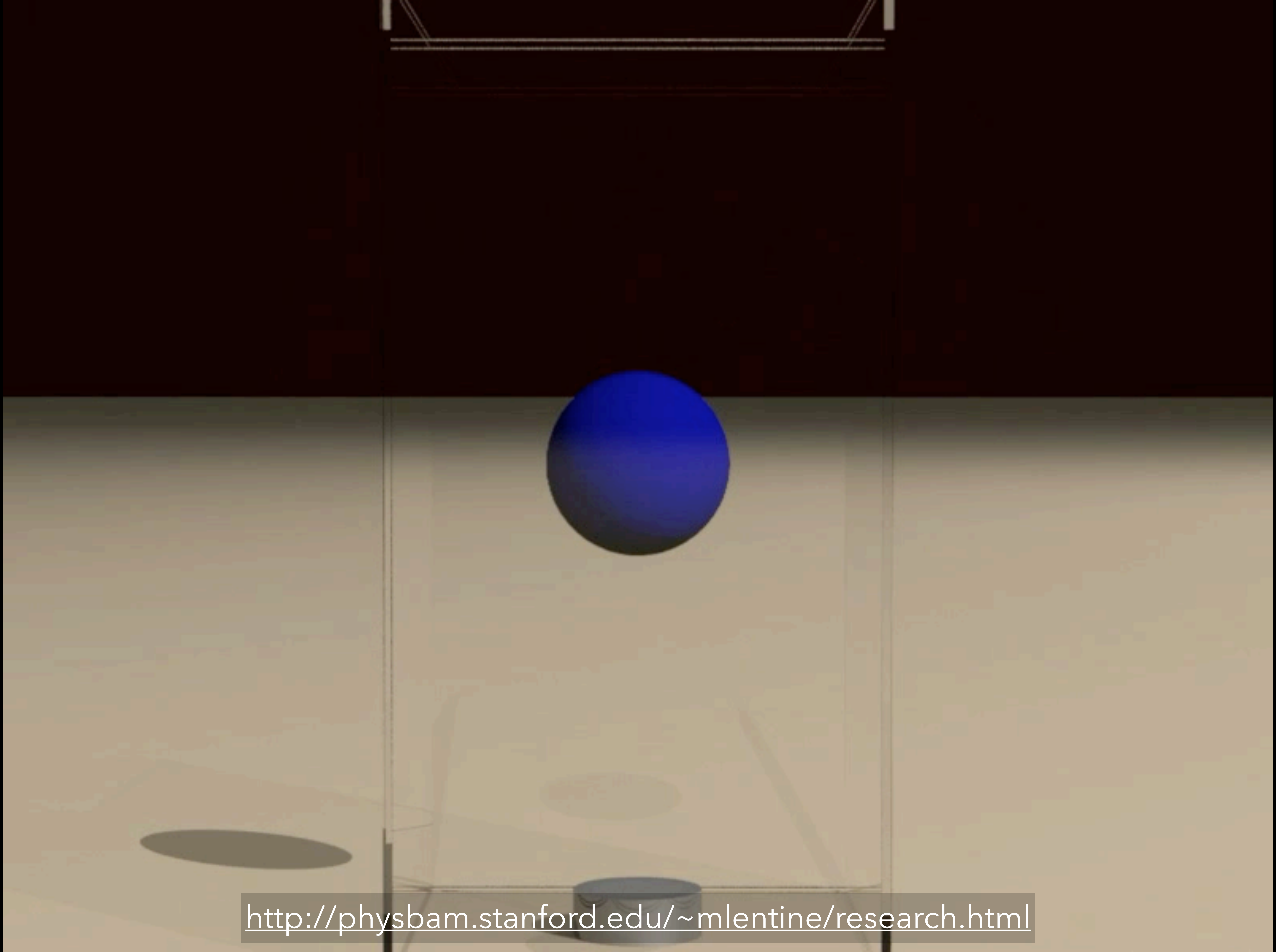
$\tilde{\mathbf{u}}$
+
$-\nabla p$
=
$\mathbf{u}^{n+1}$
$\nabla \cdot \tilde{\mathbf{u}}$
p

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = \mathbf{f}_{\text{ext}} - \nabla p + \mu\, \nabla^2 \mathbf{u}$$

$$\nabla \cdot \mathbf{u} = 0$$

1. Advection: $\mathbf{u}^{(1)} = \text{advect}(\mathbf{u}^n, \mathbf{u}^n, \Delta t)$
2. External forces: $\mathbf{u}^{(2)} = \mathbf{u}^{(1)} + \mathbf{f}_{\text{ext}}\,\Delta t$
3. Viscosity: $\mathbf{u}^{(3)} = \mathbf{u}^{(2)} + \mu\, \nabla^2 \mathbf{u}\,\Delta t$
4. Pressure: $\mathbf{u}^{n+1} = \mathbf{u}^{(3)} - \nabla p\,\Delta t$ so that $\nabla \cdot \mathbf{u}^{n+1} = 0$

This is the classic stable fluids algorithm [Stam 1999]

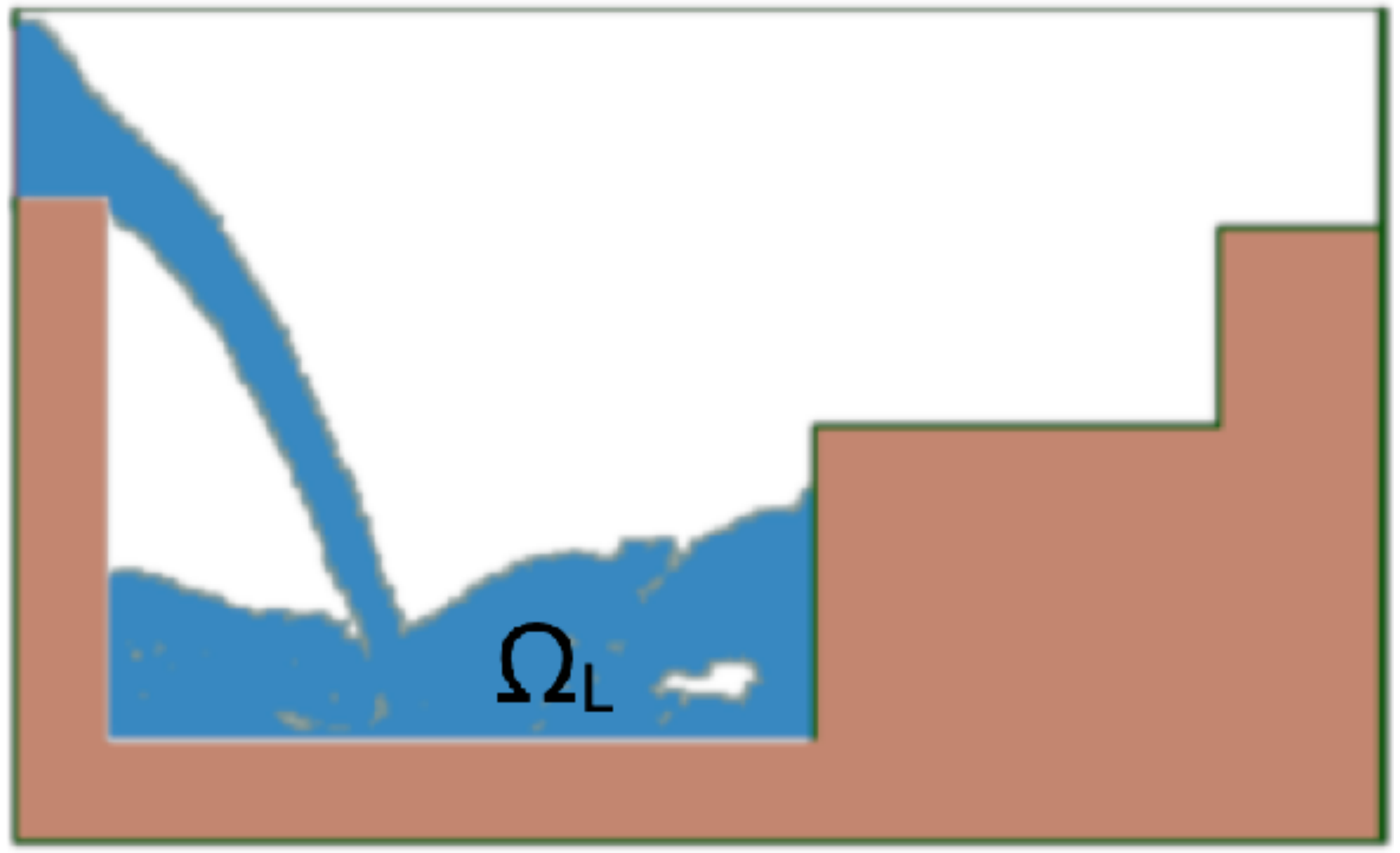


Lentine et al. 2010

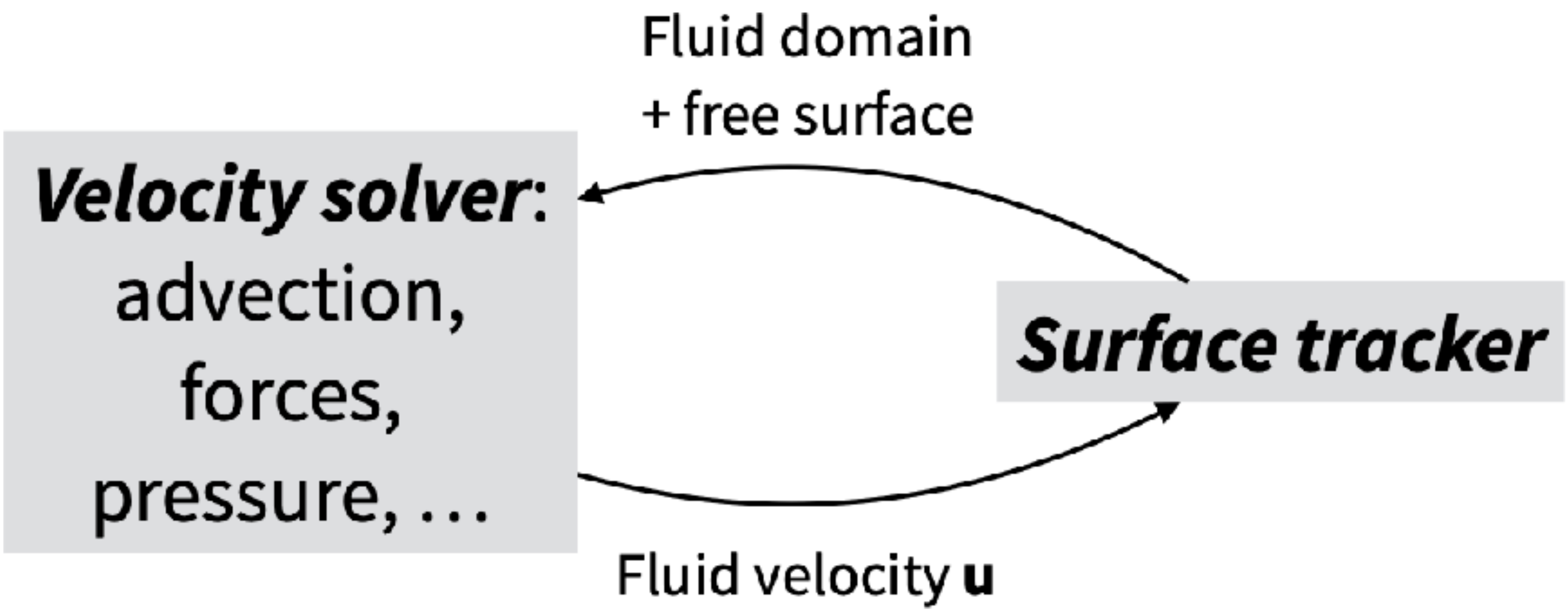
Liquids

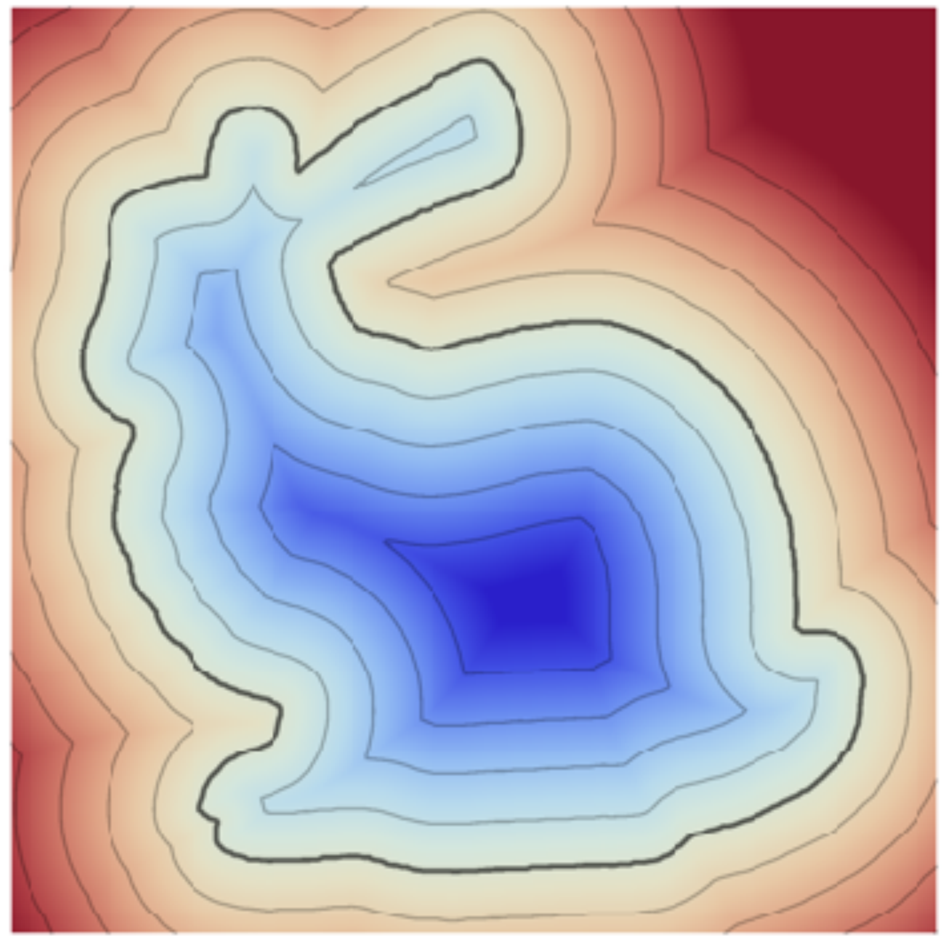
Fluid only occupies a finite region Ω ⊂ ℝ3

Need a **surface tracking** algorithm to represent Ω (usually with an implicit representation)

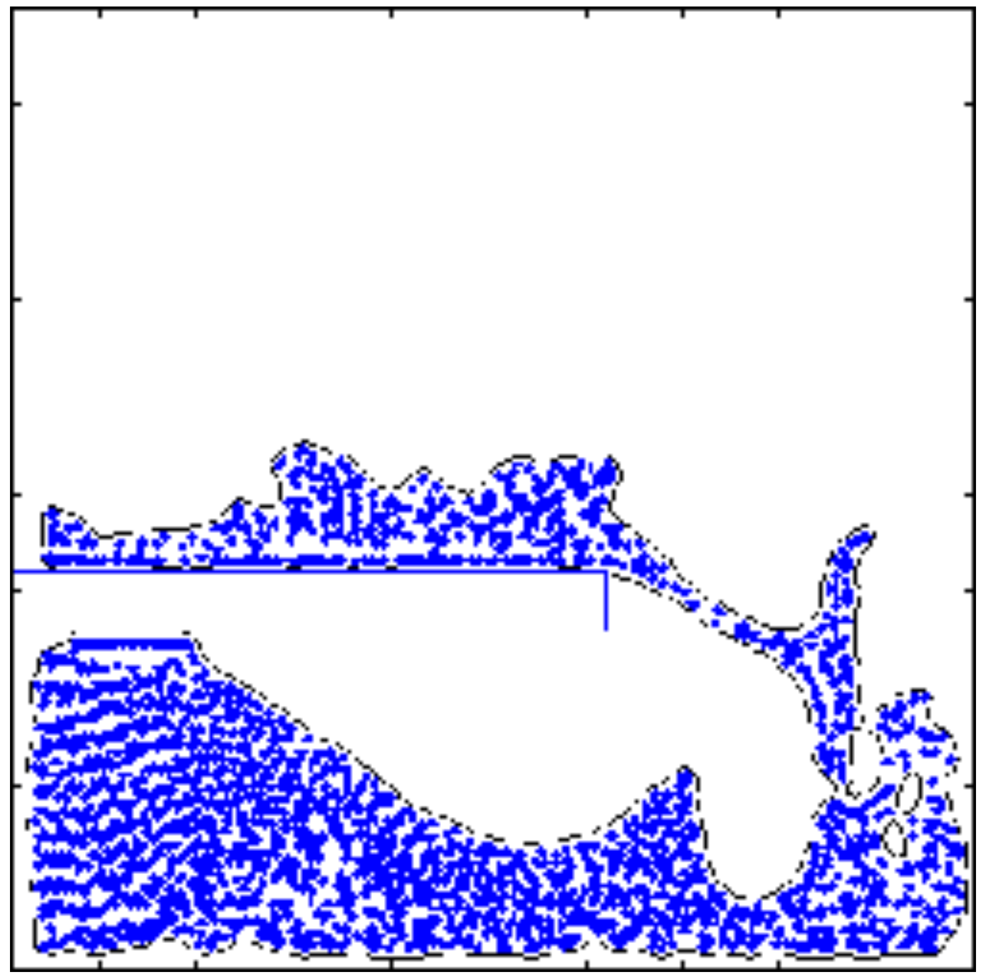


Foster & Metaxas 1996





Level sets



Particles

Particle fluids

There are also algorithms for simulating fluids with only particles:
smoothed particle hydrodynamics

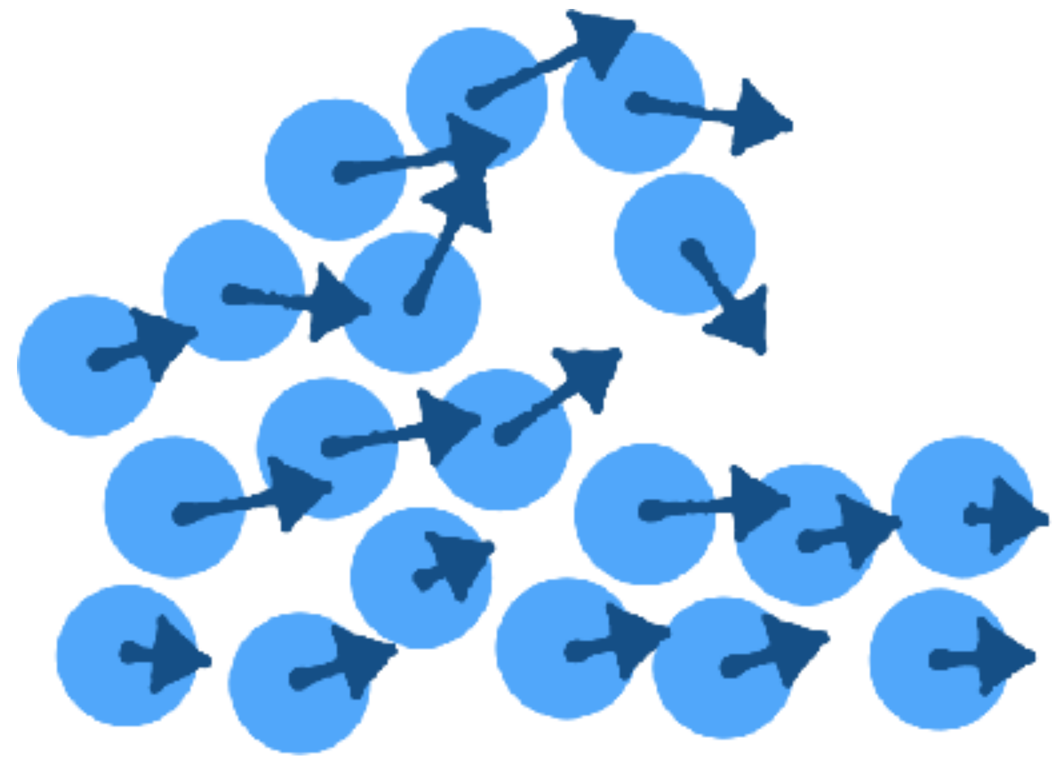
Some things become easy:

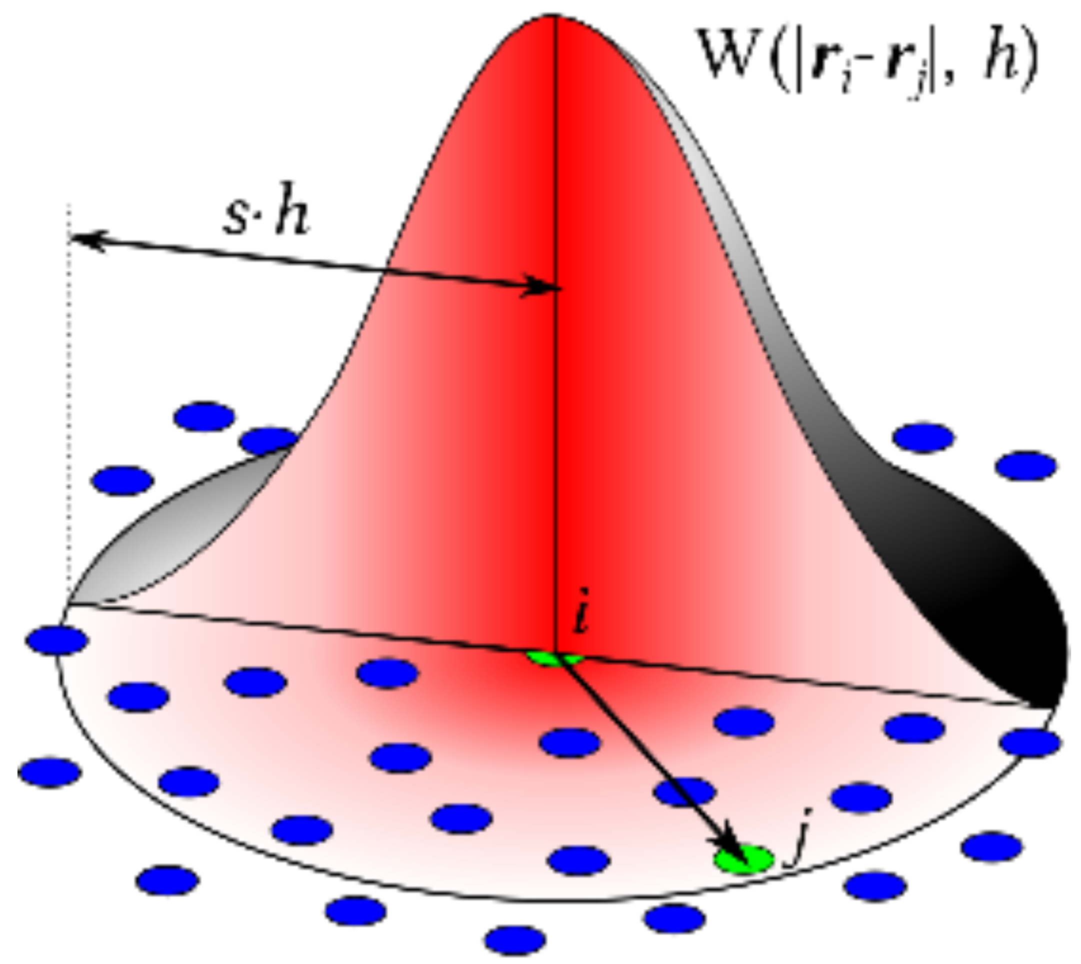
- Advection = just move the particles

Some things become hard:

- How to compute spatial derivatives ($\nabla \cdot \mathbf{u}$, ∇p, etc.)?
  Need to do weighted averaging over all nearby particles

Hybrid **particle-in-cell** methods (FLIP, APIC, MPM):
Use particles for advection, use grids for everything else!





Where to learn more

Simulation in general:

- Witkin & Baraff, *Physically Based Modeling* (2001)
- Bargteil & Shinar, *An Introduction to Physics-Based Animation* (2019)

Elastic bodies:

- Kim & Eberle, *Dynamic Deformables* (2022)

Contact handling:

- Andrews et al., *Contact and Friction Simulation for Computer Graphics* (2022)

Fluids:

- Bridson & Müller-Fischer, *Fluid Simulation for Computer Animation* (2007)