CS**5643**

13 Brief survey of fluid animation

Steve Marschner
Cornell University
Spring 2023

Fluid simulation landscape

Eulerian methods

- fluid properties are written down at points that stay fixed while the fluid moves by
- more classical approach, easier to make accurate in the limit, hard to make fast

Lagrangian methods

- fluid properties are written down at points that move with the fluid
- can be simpler, easier to make fast, harder to achieve high accuracy

Hybrid methods

- convert back and forth between Eulerian and Lagrangian representations in hopes of getting the advantages of both

Leonhard Euler
(1707–1783)

Joseph-Louis Lagrange
(1736–1813)

Outline

Elements of fluid mechanics

- reference: Bridson & Müller-Fischer SIGGRAPH course notes

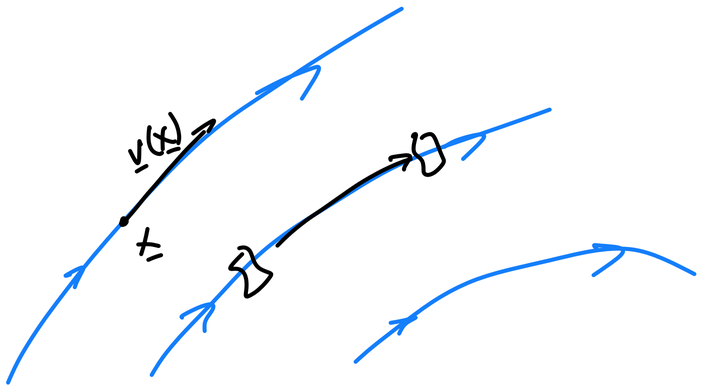
Smoothed Particle Hydrodynamics

- the basic scheme for particle-based (Lagrangian) fluids
- particles represent bits of fluid mass
- interact via inter-particle forces

Grid based fluids

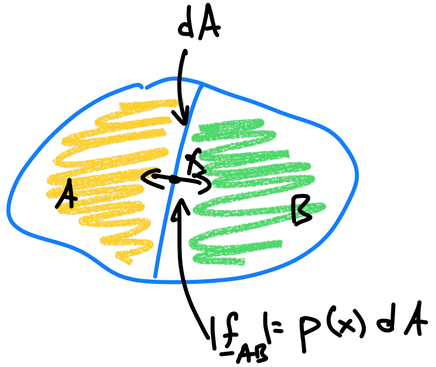
- represent fluid velocity and pressure on grids
- each timestep updates velocity and pressure fields
- operate in alternating advection (move along velocity) and projection (solve for pressure) steps

Key ideas of fluid mechanics



Velocity $\mathbf{u}(\mathbf{x})$

- blob at position $\mathbf{x}$ will be at position $\mathbf{x} + \mathbf{u}(\mathbf{x})dt$ at time dt



Pressure $p(\mathbf{x})$

- blob at position $\mathbf{x}$ exerts force $p(\mathbf{x})\,dA$ on a neighboring blob

Incompressibility

- water under most conditions and air under mild conditions do not appreciably compress
- thus pressure instantly adjusts to keep the volume density constant

Viscosity

- water an air have very little viscosity; honey or motor oil have higher viscosity

Material derivative

A quantity q in a fluid can change for two reasons

- because different bits of fluid show up where we are looking
- because the actual properties of the bits of fluid are changing

Measuring what is changing about the fluid itself

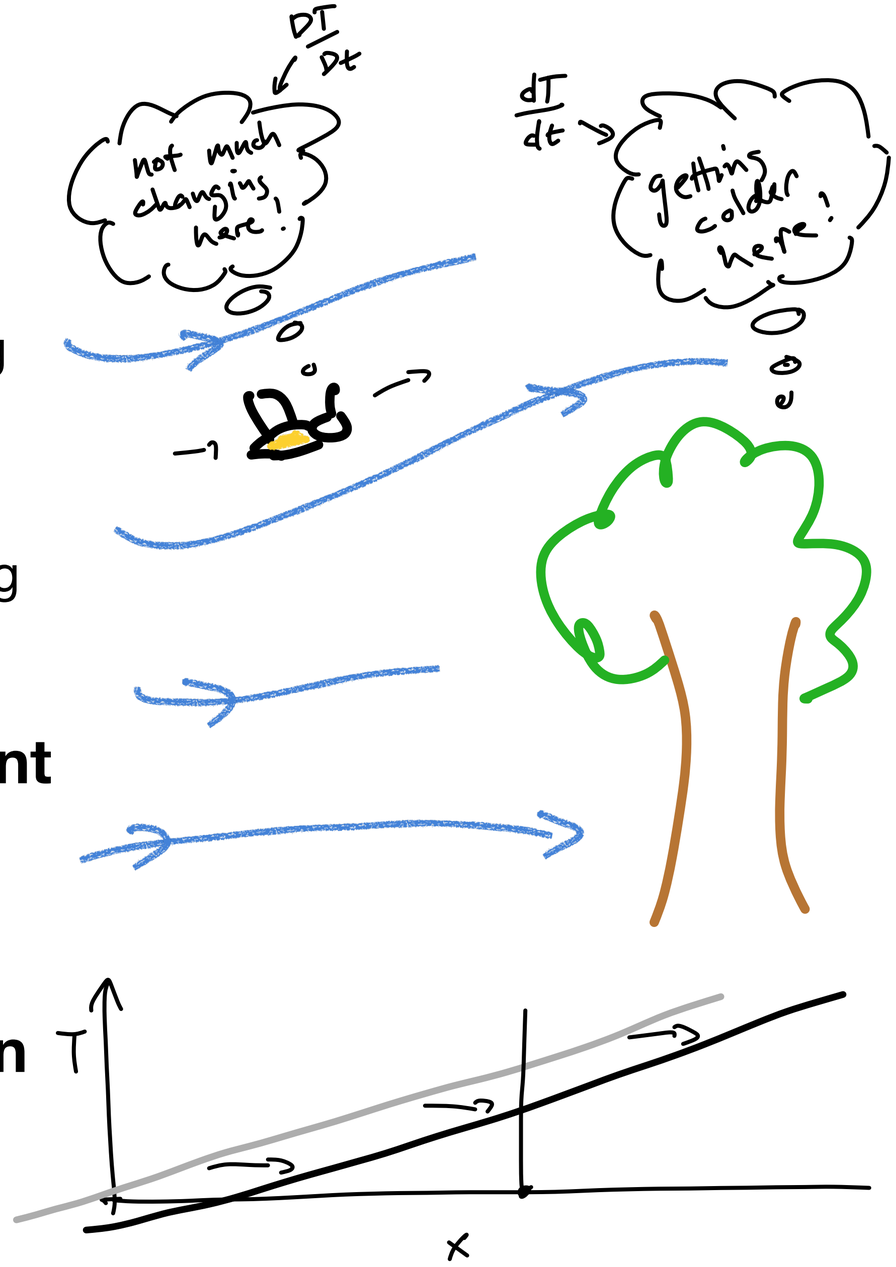
- *material derivative* Dq/Dt is the time derivative seen by a bug riding with the fluid

Measuring what changes at a fixed observation point

- ordinary time derivative dq/dt is the derivative seen by an observer at a fixed position

E.g.: temperature gradient in 1D, steady fluid motion

- $DT/Dt = 0$ but $dT/dt = -(dT/dx)u(x)$



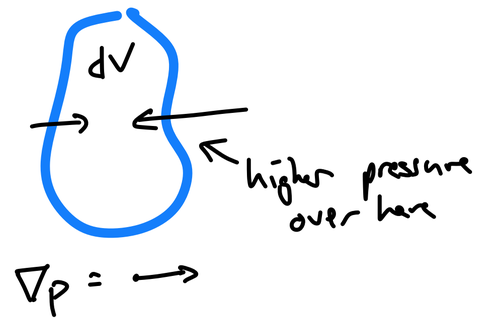
Rules for fluid motion

…for incompressible fluids with negligible viscosity

First rule: $\mathbf{f} = m\mathbf{a}$

const.

- consider a little blob of fluid with volume dV. It has mass $m = \rho dV$.
- it experiences a net force from its neighbors of $-\nabla p(\mathbf{x})\, dV$.
- it experiences a body force, e.g. due to gravity, of $m\mathbf{g}$.
- the acceleration of this blob of fluid is governed by
  - $m\dfrac{D\mathbf{u}(\mathbf{x})}{Dt} + \nabla p(\mathbf{x}) dV = m\mathbf{g}$ or $\rho\dfrac{D\mathbf{u}(\mathbf{x})}{Dt} dV + \nabla p(\mathbf{x})\, dV = \rho\mathbf{g}\, dV$
- in the limit as $dV \to 0$ we have
  - $\dfrac{D\mathbf{u}(\mathbf{x})}{Dt} + \dfrac{1}{\rho}\nabla p(\mathbf{x}) = \mathbf{g}$ which is the momentum balance equation

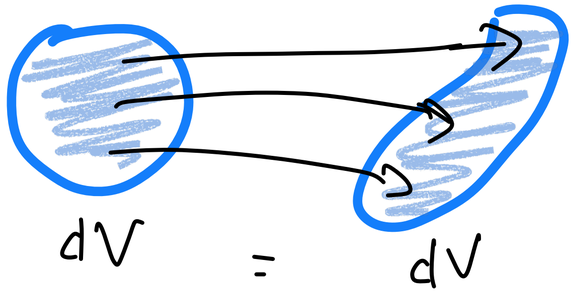


Rules for fluid motion

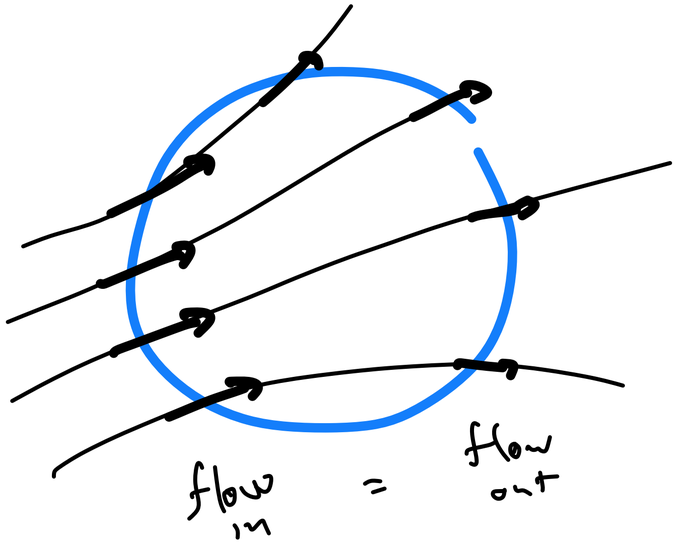
Second rule: conservation of mass

- since the density is constant, the flows into and out of a region must balance
  - $\int_{\partial\Omega} \mathbf{u}(\mathbf{x}) \cdot d\mathbf{n} = 0 = \int_{\Omega} \nabla \cdot \mathbf{u}(\mathbf{x})\, d\mathbf{x}$ —by the divergence theorem
- this has to be true for all volumes so the velocity field is divergence free
  - $\nabla \cdot \mathbf{u}(\mathbf{x}) = 0$

Lagrange:



Euler:



Rules for fluid motion

So the equations that define the problem are

$$\frac{D\mathbf{u}(\mathbf{x})}{Dt} + \frac{1}{\rho} p(\mathbf{x}) = \mathbf{g}$$

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

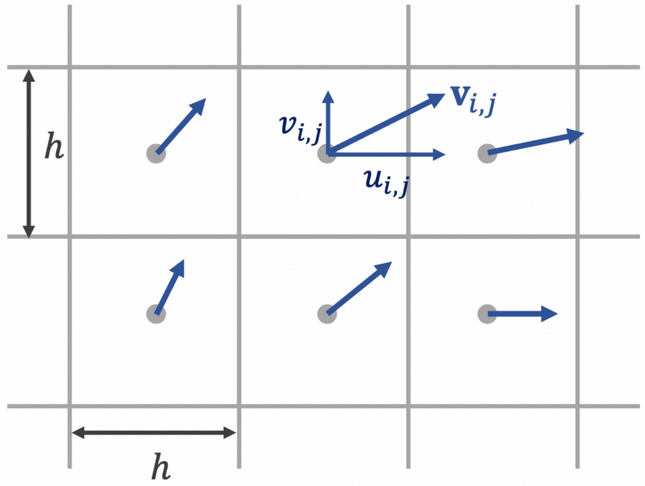
- these are the Euler equations for an incompressible and inviscid fluid
  (add viscosity and they are called Navier-Stokes equations)

Discretizing the Euler equations

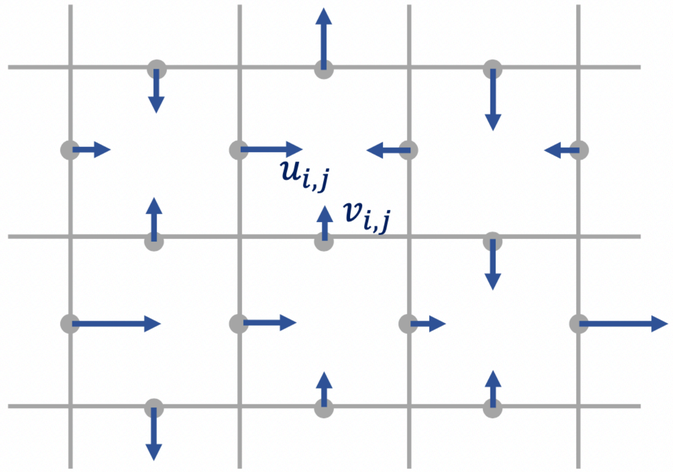
In some ways the grid-based Eulerian approach is the easiest to write down

We need to store velocity $\mathbf{u}(\mathbf{x})$ and store (or at least think about) pressure $p(\mathbf{x})$

- we will need to relate derivatives of pressure to velocity and vice versa
- it is convenient then to use a staggered rectangular grid
  - associate p with the cells of a regular grid
  - associate components of $\mathbf{u}$ with the edges of the grid



collocated grid



staggered grid

Matthias Müller

Solving the Euler equations

Successful algorithms treat terms of Euler equations separately

- first solve $\frac{D\mathbf{u}}{Dt} = 0$ to get $\mathbf{u}_A$ from $\mathbf{u}_k$

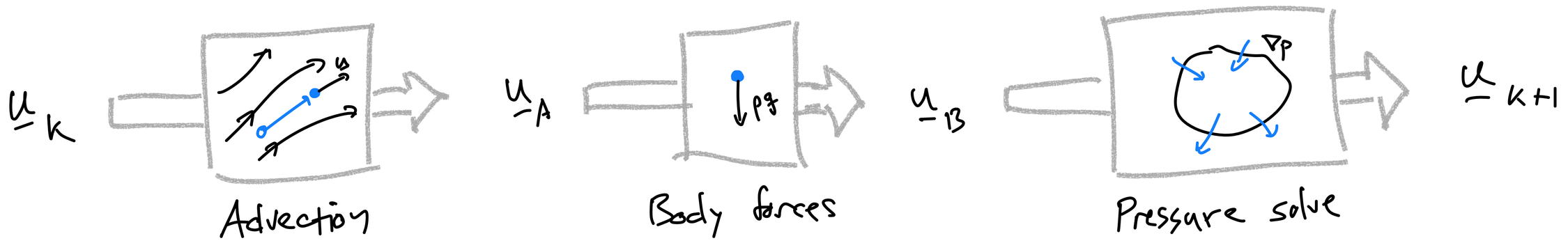
  (advection step: move velocity along the velocity)

- next add $\mathbf{g}$ to get $\mathbf{u}_B$ from $\mathbf{u}_A$

  (velocity update: can also add other body forces if desired)

- finally solve for pressure and update $\frac{d\mathbf{u}}{dt} = \frac{1}{\rho}\nabla p$ to get $\mathbf{u}_{k+1}$

  (projection: ensure $\mathbf{u}$ is divergence free)



Grid fluid solver: advection step

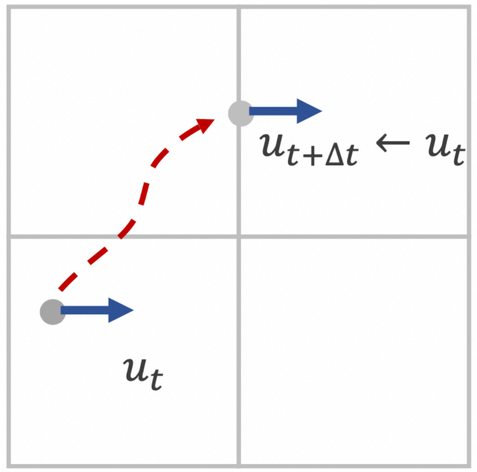
For the advection step we want $\mathbf{u}(\mathbf{x})$ to be the velocity that the bit of fluid at $\mathbf{x}$ had a time h ago

- one approach: work out in terms of space and time derivatives of $\mathbf{u}$ on the grid
  - sadly this requires very small time steps for stability
- better approach: simply look where this bit of fluid was a time h ago
  - step backward by h along the velocity vector
  - sample (interpolate) the fluid velocity at that point to get $\mathbf{u}_A$

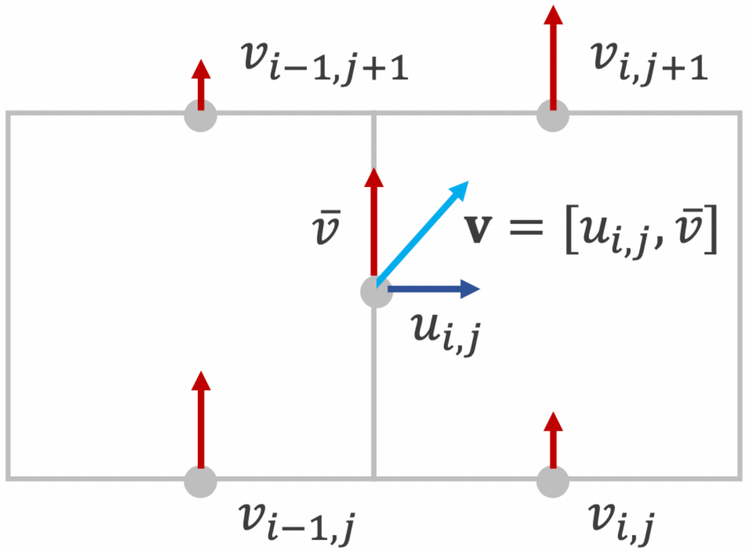
This approach is known as “semi-Lagrangian advection”

- it is unconditionally stable since it’s strictly interpolating; range of velocities gets smaller with each step, never larger
- the same approach can be used for other quantities as well as the velocity itself

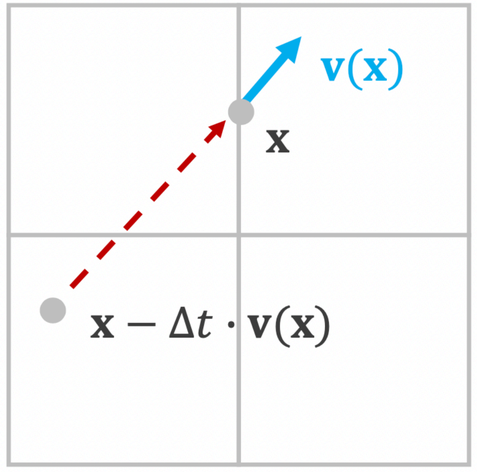
Semi-Lagrangian advection



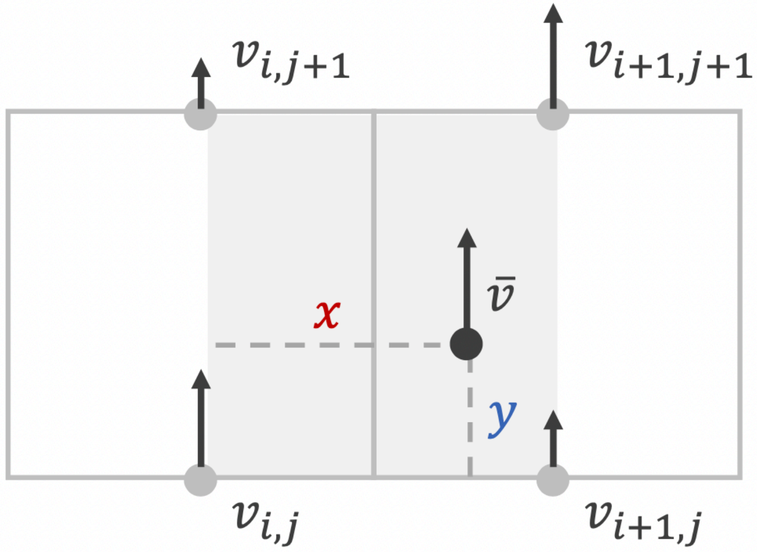
Trace back in time to a (non-grid) sample location at the previous time step



To get the velocity vector you need to interpolate the components from the staggered grid



Compute sample location using a simple Forward Euler step



To sample the previous velocity you need to linearly interpolate from the corresponding velocity grid

Matthias Müller

Velocity update

After the advection step we have new velocities everywhere

- they correspond to each bit of fluid flying straight
- no interaction between bits of fluid
- no concern for forces

First and simplest fix: add acceleration due to body forces

- $\mathbf{u}_B(\mathbf{x}) = \mathbf{u}_A(\mathbf{x}) + h\mathbf{g}$ — in practice for gravity only have to update the y component

Velocity projection (pressure update)

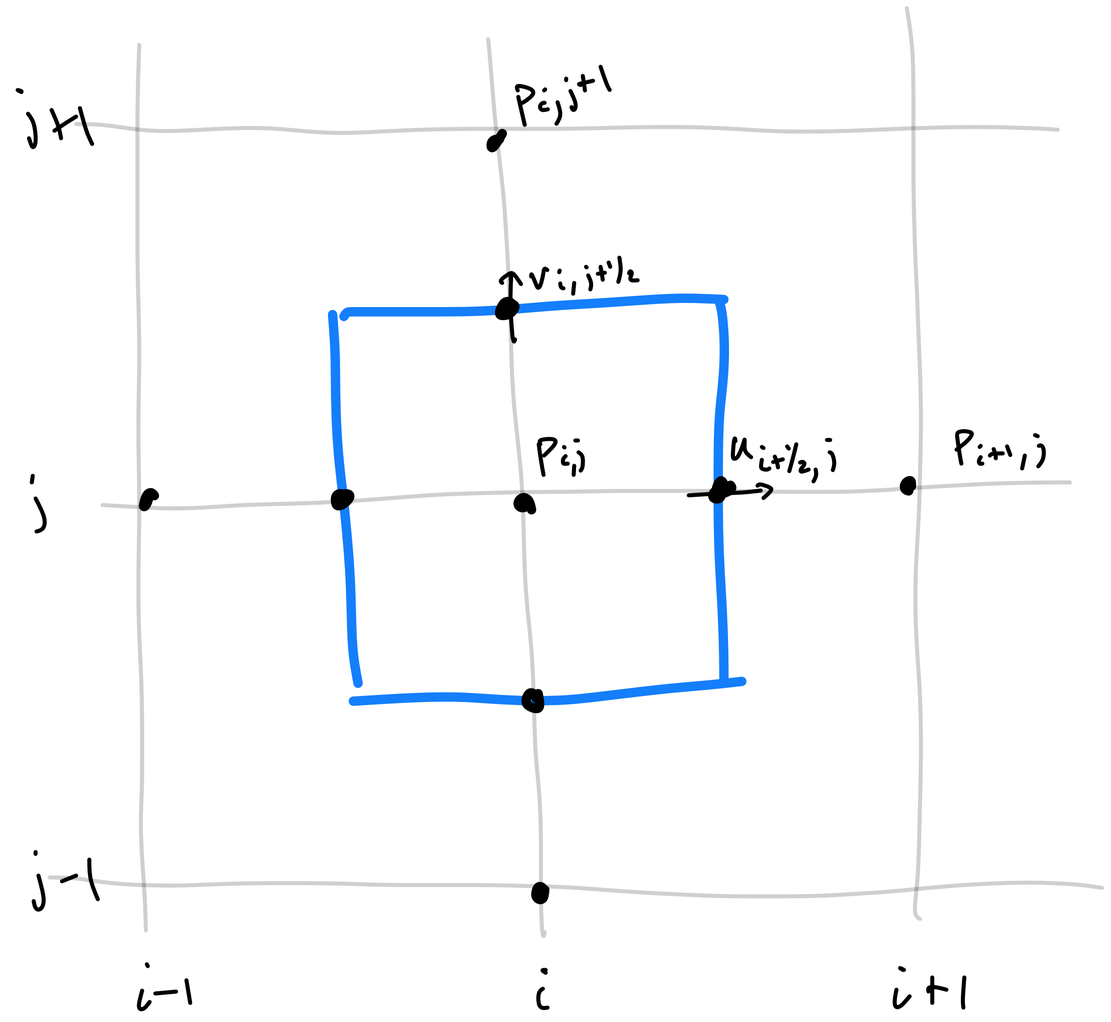
After advection and velocity update, velocity field is broken

- it is supposed to be divergence free (since the fluid is incompressible)
- in an incompressible fluid the pressure acts to enforce this constraint

Last substep is to solve for the pressure that will cause $\nabla \cdot \mathbf{u} = 0$

- we want a pressure field so that $\nabla \cdot \mathbf{u}_{k+1} = \nabla \cdot \left(\mathbf{u}_B(\mathbf{x}) - \frac{h}{\rho} \nabla p(\mathbf{x}) \right) = 0$
- this expands to $h\Delta p = -\rho \nabla \cdot \mathbf{u}_B$ where $\Delta p = \nabla \cdot \nabla p$ is the Laplacian of p
- this is a Poisson problem and there are good methods to solve it
- result is the final $\mathbf{u}_{k+1}$ for the next timestep
  (pressure does not need to be saved)

Velocity projection (pressure update)



update to velocity will be:

$$u^{+}_{i+1/2,j} = u^{-}_{i+1/2,j} - \frac{\Delta t}{\rho}(p_{i+1,j} - p_{i,j})/\Delta x$$

divergence apx is:

$$\nabla \cdot u_{i,j} \simeq \left(\frac{u_{i+1/2,j} - u_{i-1/2,j}}{\Delta x}\right) + \left(\frac{v_{i,j+1/2} - v_{i,j-1/2}}{\Delta x}\right) = 0$$

pressure must satisfy:

$$\frac{\Delta t}{\rho}\left[\frac{4p_{i,j} - p_{i+1,j} - p_{i,j-1} - p_{i+1,j} - p_{i,j+1}}{\Delta x^2}\right] =$$

$$-\nabla \cdot u^{-}_{i,j}$$

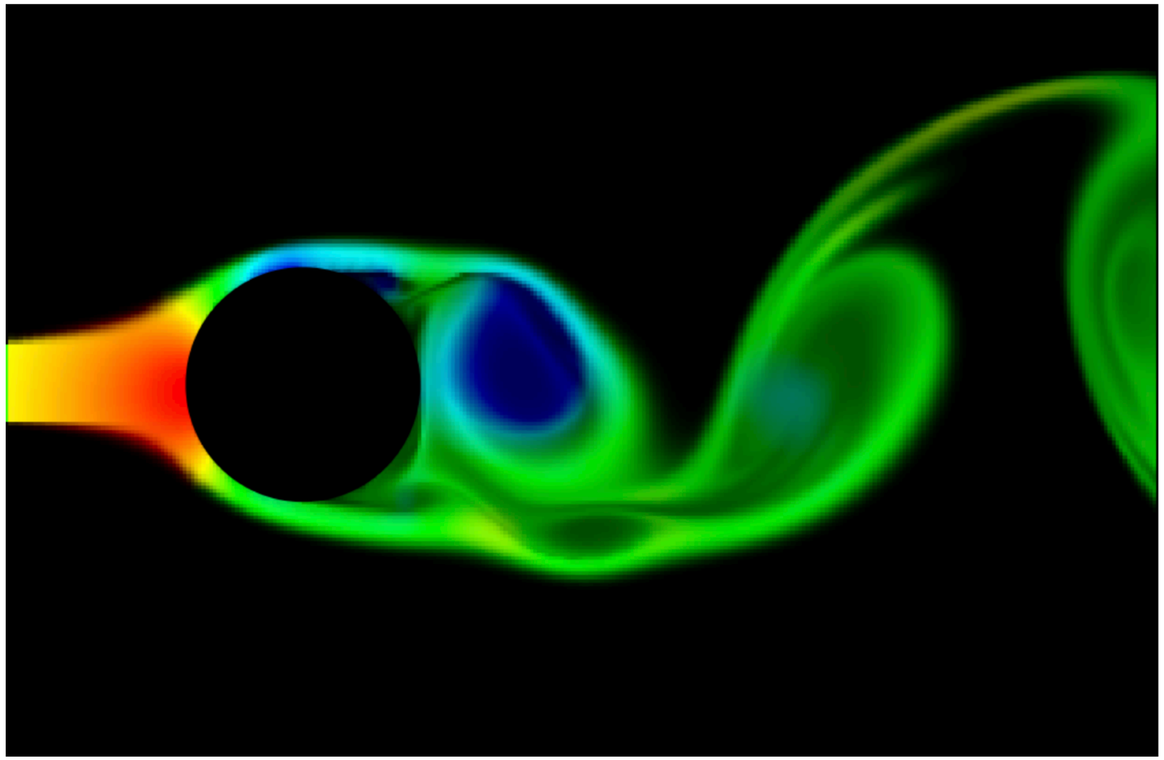
Grid simulator

Will need boundary conditions

- pressure = 0 (Neumann) conditions for free boundaries (open space)
- normal velocity = 0 (Dirichlet) conditions for solid boundaries (obstacles)
- require simple local changes in the pressure step

Those are all the pieces required

- often you'll want to advect smoke density
  or temperature passively along with the fluid
- simple approximation: upward buoyancy force
  proportional to temperature
- demo by Matthias Müller



Particle based fluids

A whole different approach to fluid simulation

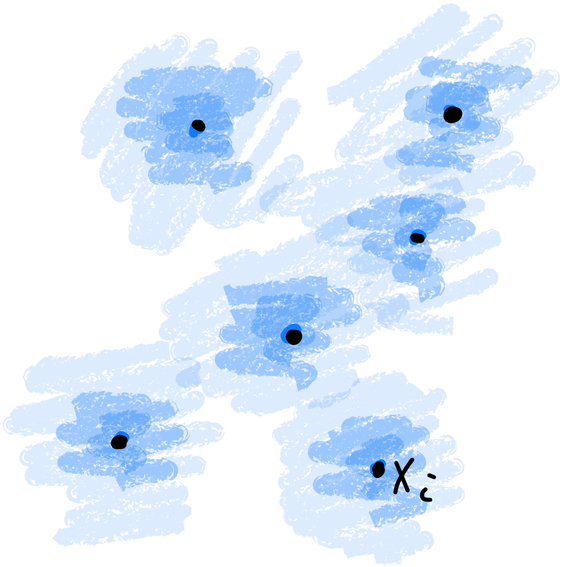
- represent fluid variables on movable particles
- advect the particles with the velocity (ballistic motion)
- use inter-particle forces to manage pressure and viscosity

Key mathematical formulation: Smoothed Particle Hydrodynamics (SPH)

- methods originated in astrophysics in the 90s
- idea: define continuously varying quantities by blurring the particles
- sample these quantities at particle locations to compute forces

Defining a continuous field through smoothing

Use a smooth kernel



- popular choice: $W_{\text{poly6}}(r) = \frac{315}{64\pi d^9} \begin{cases} (d^2 - r^2)^3 & 0 \le r \le d \\ 0 & \text{otherwise,} \end{cases}$
- define kernel by radial falloff: $W(\mathbf{r}) = W(\|\mathbf{r}\|)$
- example: fluid density
  - $\rho(\mathbf{x}) = \sum_i m_i W(\mathbf{x} - \mathbf{x}_i)$ — and let $\rho_i = \rho(\mathbf{x}_i)$ be the density at particle i
- once we have density, define interpolants of any quantity by generalizing this as
  - $q(\mathbf{x}) = \sum_i \frac{m_i}{\rho_i} q_i W(\mathbf{x} - \mathbf{x}_i)$
- this provides machinery to define the terms in the Navier-Stokes equations

Computing forces in SPH

Divergence-free velocity is not exactly maintained

- this means we can't assume no density variations

…but mass is exactly conserved

- they are particles, they can't help it
  (so no need to worry about losing mass even with density fluctuations)

Particles behave like regular ballistic particles subject to forces

- $$\rho \frac{D\mathbf{u}(\mathbf{x})}{Dt} = -\nabla p(\mathbf{x}) + \rho \mathbf{g}$$
- the left hand side defines the particle acceleration
- SPH lets us compute the forces on the right hand side
- gravity (or other external forces) is easy

Computing the pressure force

Pressure exerts a force on each particle

- the pressure is a function of the nearby particles
- given pressure p_i for each particle the interpolant is
  - $$p(\mathbf{x}) = \sum_i \frac{m_i}{\rho_i} p_i W(\mathbf{x} - \mathbf{x}_i)$$
- and its gradient is
  - $$\nabla p(\mathbf{x}) = \sum_i \frac{m_i}{\rho_i} p_i \nabla W(\mathbf{x} - \mathbf{x}_i)$$
- resulting in the force due to pressure on particle i
  - $$\mathbf{f}_i^{\mathrm{p}} = -\nabla p(\mathbf{x}_i) = -\sum_j \frac{m_j}{\rho_j} p_j \nabla W(\mathbf{x}_i - \mathbf{x}_j)$$

Computing the pressure force

Symmetrized force is typically used

- $$\mathbf{f}_i^{\mathrm{p}} = -\nabla p(\mathbf{x}_i) = -\sum_j \frac{m_j}{\rho_j}\frac{p_i + p_j}{2}\nabla W(\mathbf{x}_i - \mathbf{x}_j)$$

How to define pressures for the particles?

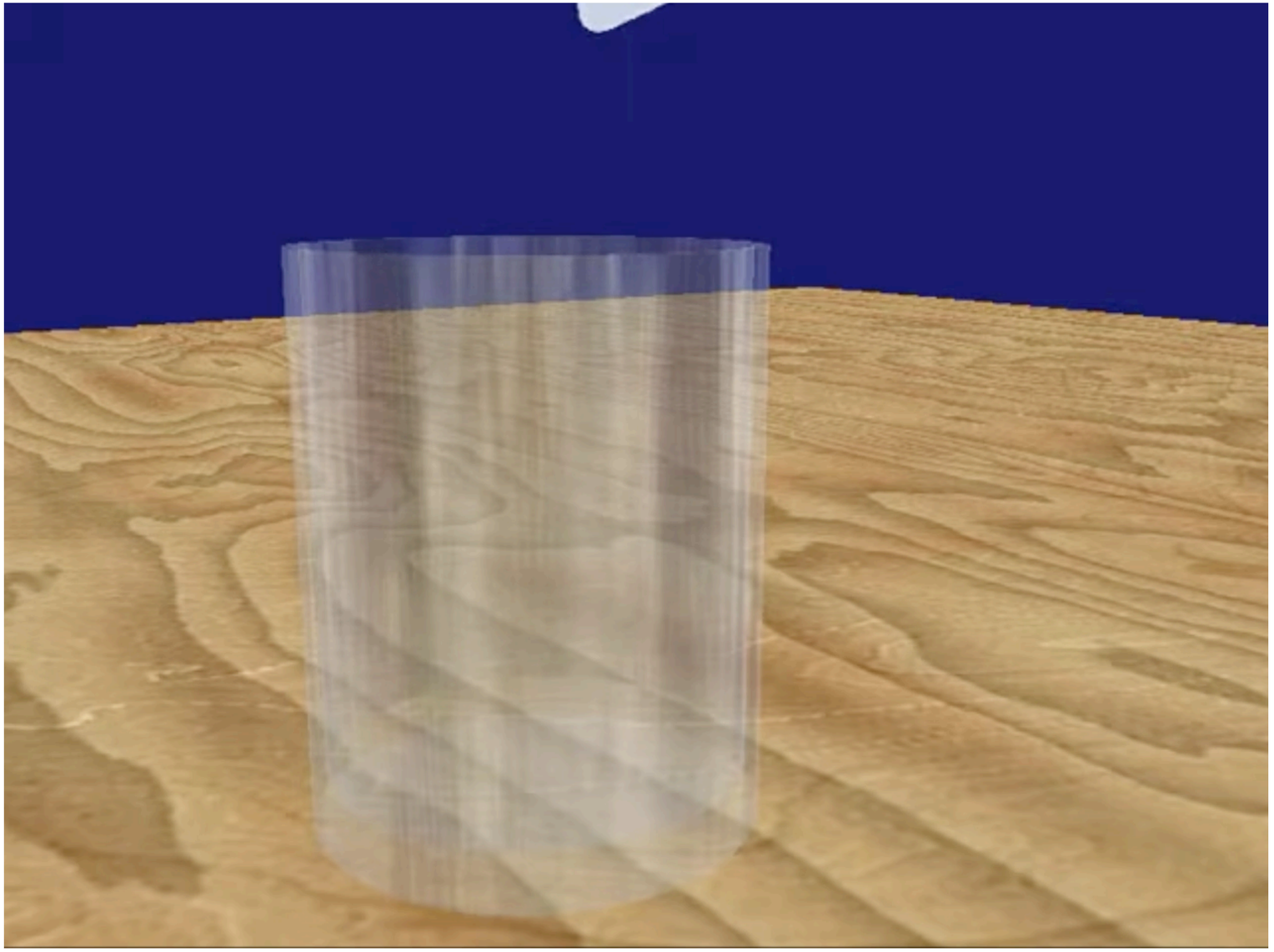
- simple: think of fluid as compressible and define pressure from density
- popular model $p(\mathbf{x}) = k(\rho(\mathbf{x}) - \rho_0)$ —kind of like ideal gas law
- advantage: simplicity
- disadvantage: springiness (if k is low) or numerical stiffness (if k is high)
- more sophisticated method: use iterative solve to find incompressible velocities

Algorithm 1 SPH / WCSPH

while animating **do**
  for all i **do**
    find neighborhoods $N_i(t)$
  for all i **do**
    compute density $\rho_i(t)$
    compute pressure $p_i(t)$
  for all i **do**
    compute forces $\mathbf{F}^{p,v,g,ext}(t)$
  for all i **do**
    compute new velocity $\mathbf{v}_i(t+1)$
    compute new position $\mathbf{x}_i(t+1)$

[Solenthaler and Pajarola 2009]

Early graphics SPH: real time c. 2003



Matthias Müller

Iteratively solving for pressure: predictive-corrective incompressible SPH

- iterate within each timestep to find pressures that lead to velocities that lead to uniform density
- allows larger timesteps
- ultimately faster than the alternative of using tiny simtesteps to achieve sufficient pressure uniformity

Algorithm 2 PCISPH

while animating **do**
  for all i **do**
    find neighborhoods $N_i(t)$
  for all i **do**
    compute forces $\mathbf{F}^{v,g,ext}(t)$
    initialize pressure $p(t) = 0.0$
    initialize pressure force $\mathbf{F}^p(t) = 0.0$
  while $(\rho^*_{err}(t+1) > \eta)$ || $(iter < minIterations)$ **do**
    for all i **do**
      predict velocity $\mathbf{v}^*_i(t+1)$
      predict position $\mathbf{x}^*_i(t+1)$
    for all i **do**
      predict density $\rho^*_i(t+1)$
      predict density variation $\rho^*_{err}(t+1)$
      update pressure $p_i(t) \mathrel{+}= f(\rho^*_{err}(t+1))$
    for all i **do**
      compute pressure force $\mathbf{F}^p(t)$
  for all i **do**
    compute new velocity $\mathbf{v}_i(t+1)$
    compute new position $\mathbf{x}_i(t+1)$

[Solenthaler and Pajarola 2009]

SPH with better pressure solve

Barbara Solenthaler