太极图形课

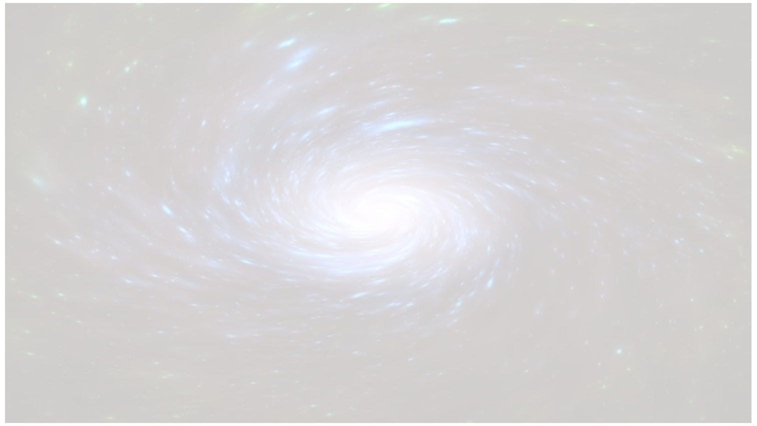
第11讲 Fluid Simulation 02: The Grid-based Methods

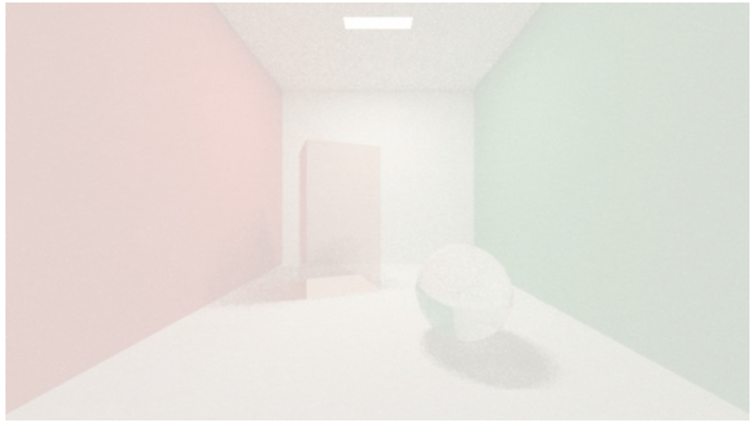
taichi

Season Finale Alert

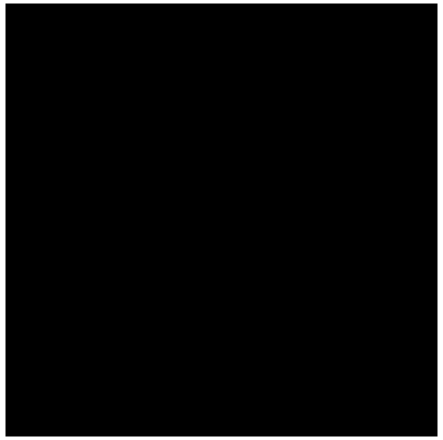
- Ailing Zhang 张爱玲
  - Compiler Architect @ Taichi Graphics
  - THU → UIUC → Facebook (PyTorch) → Taichi
- Dec. 14th:
  - 手把手教你如何向Taichi仓库贡献代码，成为Taichi开发者

Where are we?

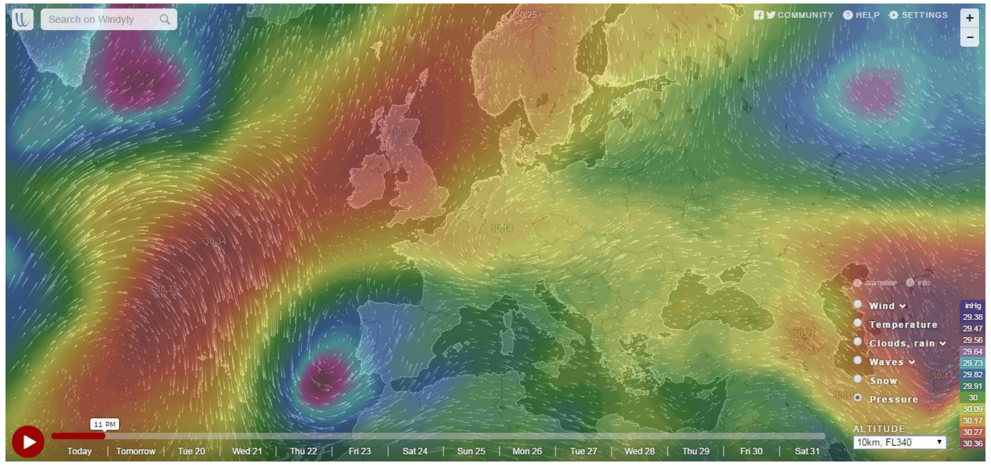

Procedural Animation

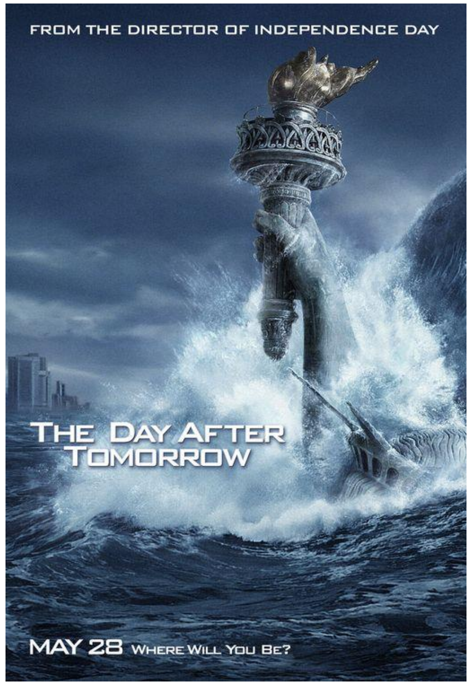

Rendering

Deformable Simulation

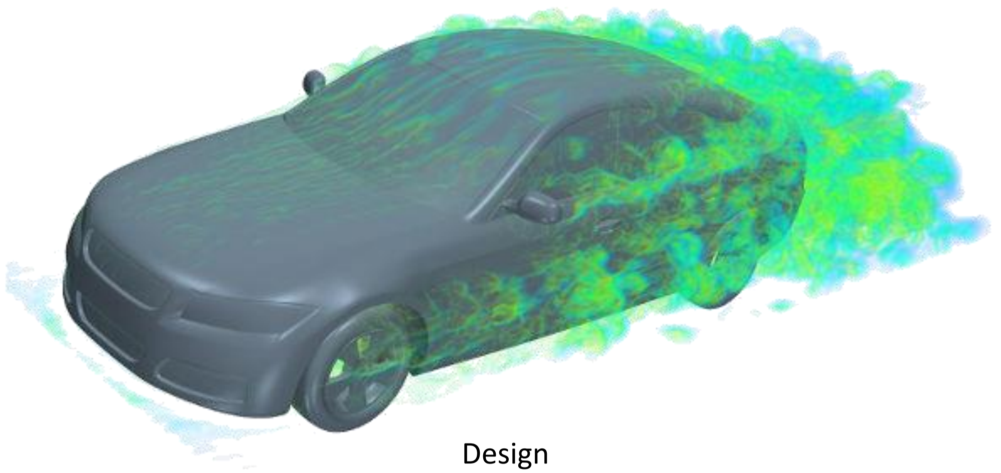

Fluid Simulation

Fluid simulation


Forecast


FROM THE DIRECTOR OF INDEPENDENCE DAY
THE DAY AFTER TOMORROW
MAY 28 WHERE WILL YOU BE?
VFX

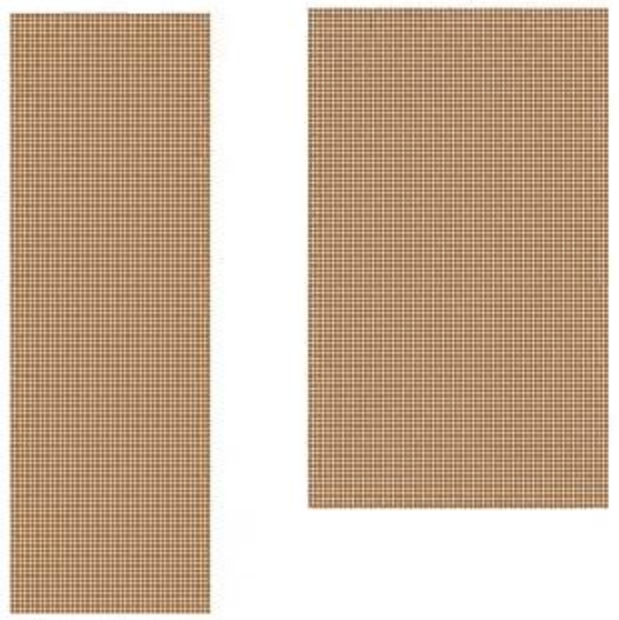
Game


Design

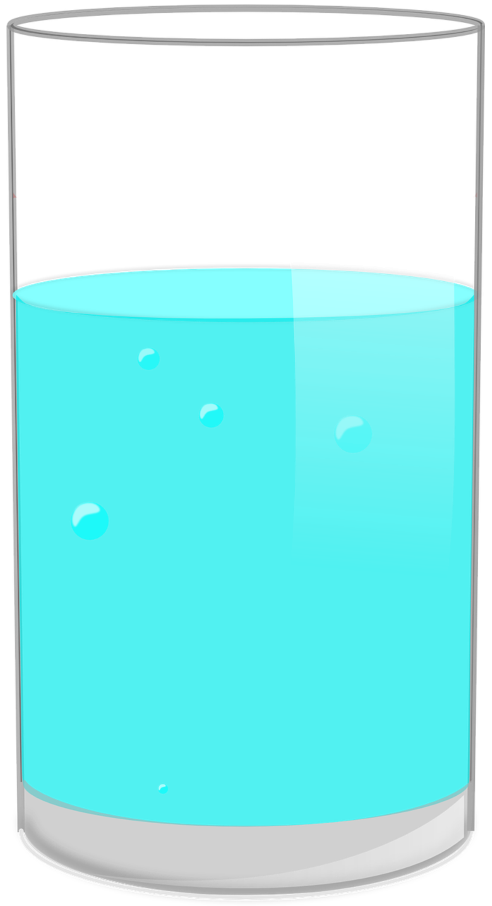
Animation

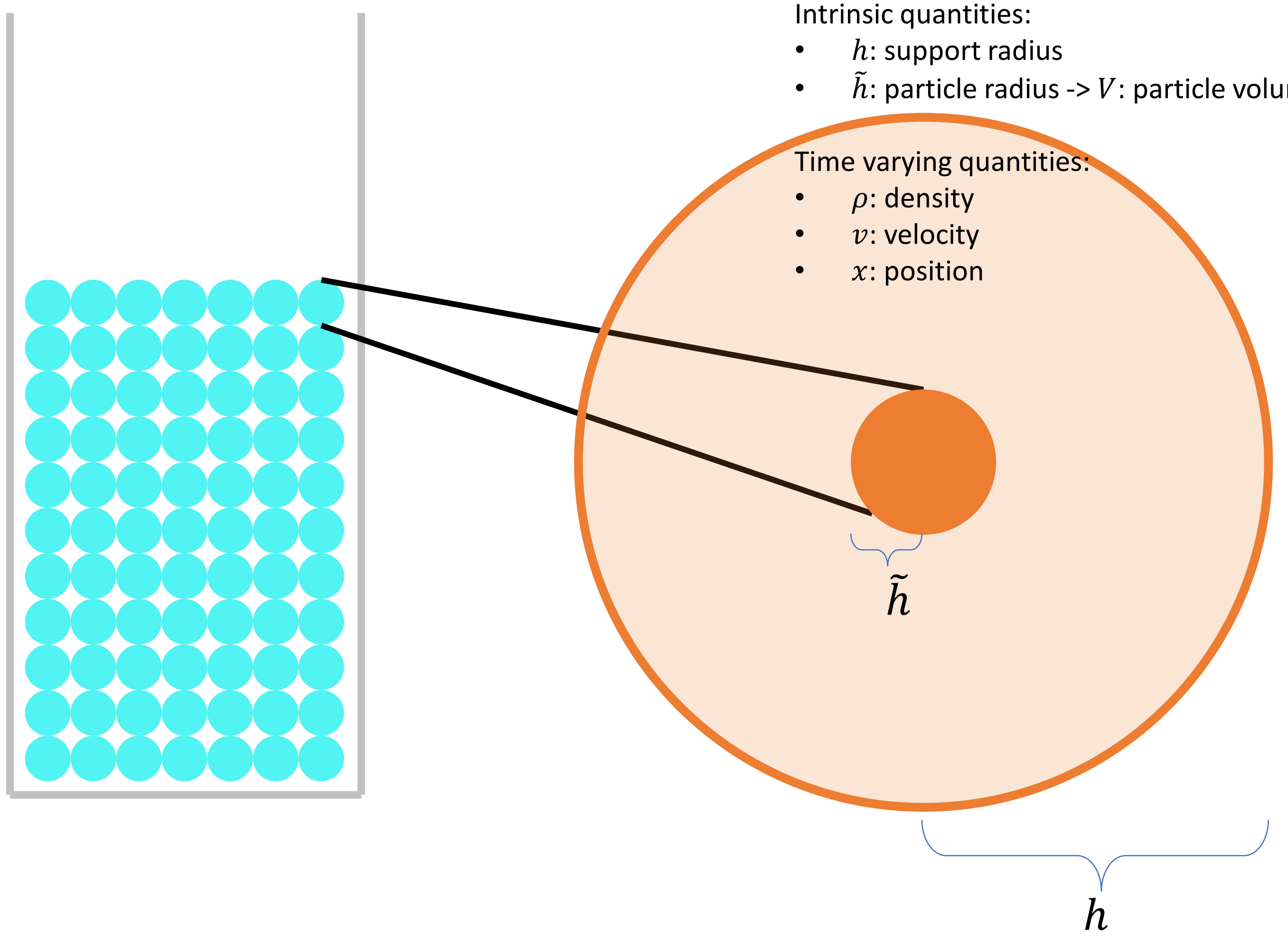
Recap

- Incompressible fluid dynamics
  - Incompressible Navier–Stokes equations
- Time discretization
  - Operator splitting
  - Integration with the weakly compressible assumption
- Spatial discretization
  - Smoothed particle hydrodynamics (SPH)
- Implementation details (WCSPH)
  - Simulation Pipeline
  - Boundary conditions
  - Neighbor search



Lagrangian view

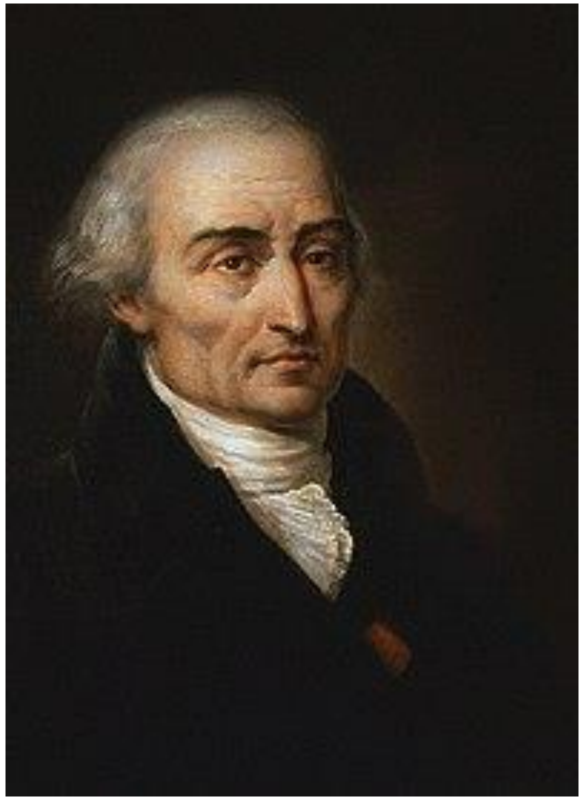




Intrinsic quantities:
- h: support radius
- $\tilde{h}$: particle radius -> V: particle volume

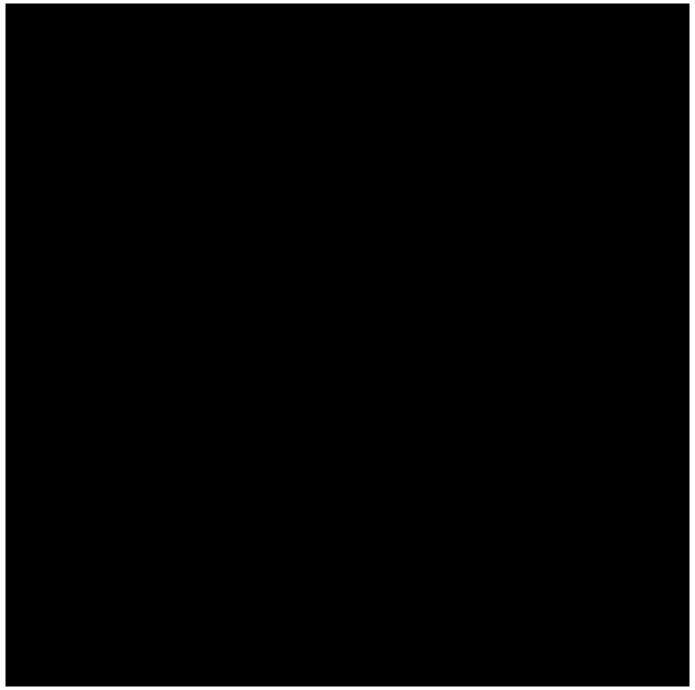
Time varying quantities:
- ρ: density
- v: velocity
- x: position

What will be covered today…



Code of the day

- Code:
  - https://github.com/taichi-dev/taichi/blob/master/python/taichi/examples/simulation/stable_fluid.py
- Code courtesy of 刘嘉枫 [@Hanke98]

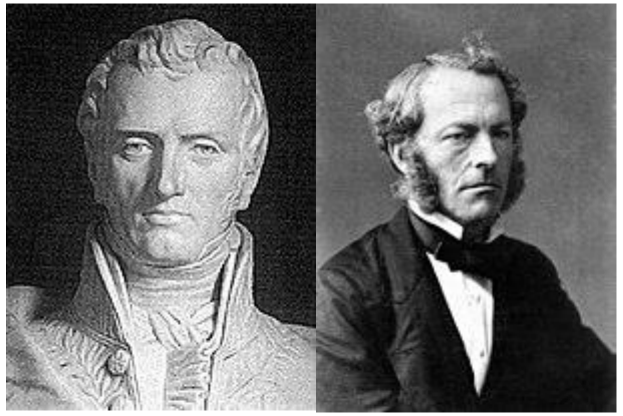


Outline today

- N-S equations and their time integration
  - Operator splitting
- From the Lagrangian view to the Eulerian view
  - Spatial derivatives using finite difference
  - MAC grid
- Advection
  - Material derivative
  - Quantity advection
- Projection
  - Poisson's equation
  - Boundary conditions

N-S equations and the time integration

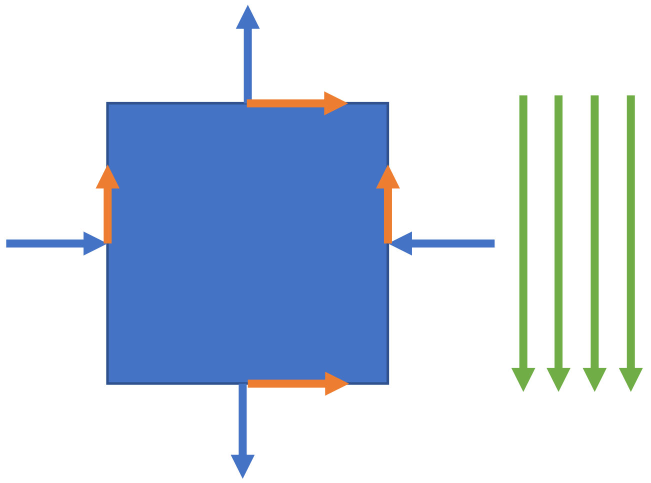
Incompressible Navier-Stokes equation



$$ma = f_{ext} + f_{pres} + f_{visc}$$

$$\rho \frac{Dv}{Dt} = \rho g - \nabla p + \mu \nabla^2 v$$

$$\nabla \cdot v = 0$$



Operator splitting: a toy example

- Let’s integrate $\frac{dq}{dt} = 1 + 2$
- (We know that the answer is:$q^{n+1} = q^n + 3\Delta t$)

- Operator splitting:
  - $\tilde{q} = q^n + 1\Delta t$
  - $q^{n+1} = \tilde{q} + 2\Delta t$

Operator splitting: a general example

- Let's integrate $\frac{dq}{dt} = f(q) + g(q)$

- Operator splitting:
  - $\tilde{q} = q_n + \Delta t f(q^n)$
  - $q^{n+1} = \tilde{q} + \Delta t g(\tilde{q})$

Operator splitting: N-S equations

- Let's integrate $\frac{Dv}{Dt} = g - \frac{1}{\rho}\nabla p + \nu\nabla^2 v$
  $\nabla \cdot v = 0$

- Operator splitting:
  - Advection: $\frac{Dq}{Dt} = 0$, where q can be velocity, density, temperature etc.
  - Applying forces: $\frac{\partial v}{\partial t} = g + \nu\nabla^2 v$
  - Projection: $\frac{\partial v}{\partial t} = -\frac{1}{\rho}\nabla p \; s.t. \nabla \cdot v = 0$

One numerical time-stepping for N-S equations

- Given q^n, where q can be velocity, density, temperature etc.
  - Step 1 Advection:
    - $q^{n+1} = advect(v^n, \Delta t, q^n)$
    - $\tilde{v} = advect(v^n, \Delta t, v^n)$
  - Step 2 Applying forces:
    - $\tilde{\tilde{v}} = \tilde{v} + \Delta t(g + \nu \nabla^2 \tilde{v})$
  - Step 3 Projection:
    - $v^{n+1} = project(\Delta t, \tilde{\tilde{v}})$
  - Return v^{n+1}, q^{n+1}

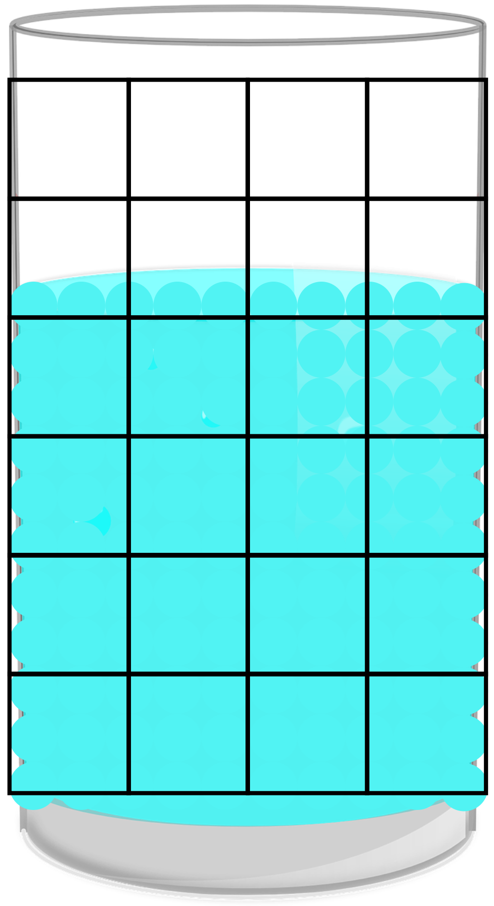
$$\frac{Dv}{Dt} = g - \frac{1}{\rho}\nabla p + \nu \nabla^2 v$$

$$\nabla \cdot v = 0$$

From the Lagrangian view to the Eulerian view

Lagrangian view v.s. Eulerian view

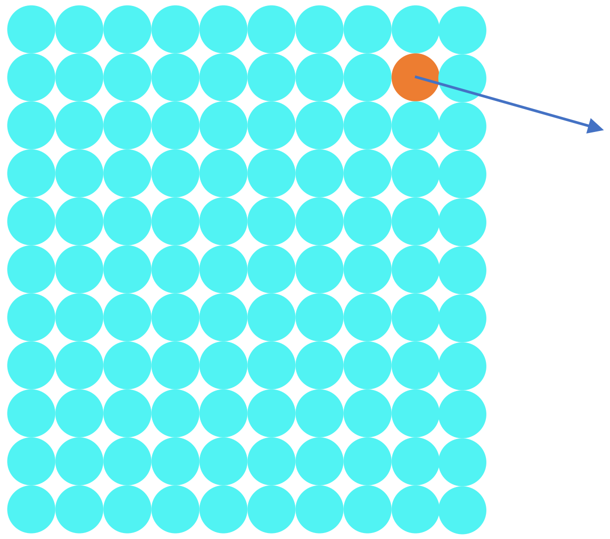
- Lagrangian view：



- Eulerian view：

Lagrangian view v.s. Eulerian view

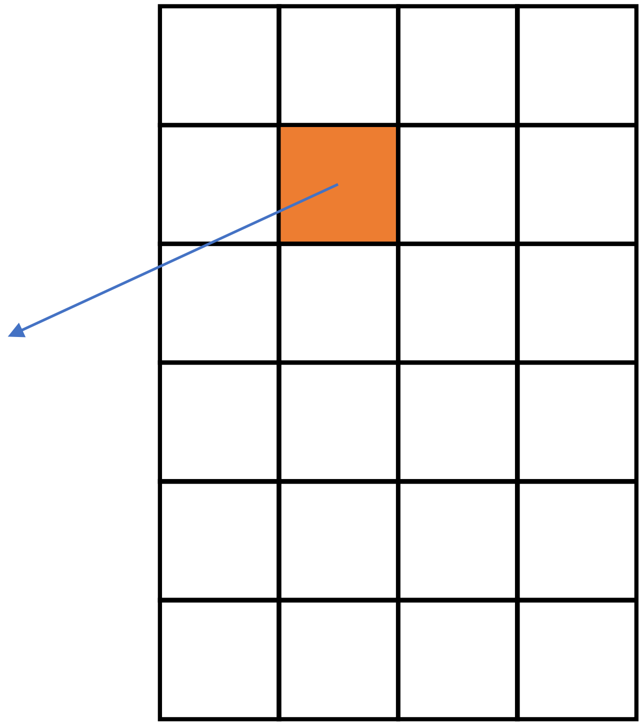
- Lagrangian view：



Position
Velocity
Density
Temperature
Volume
Support radius
…
etc.

Dynamic Markers

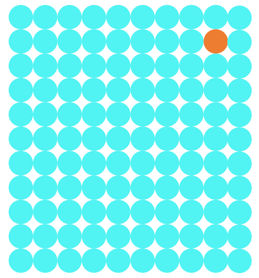
- Eulerian view：

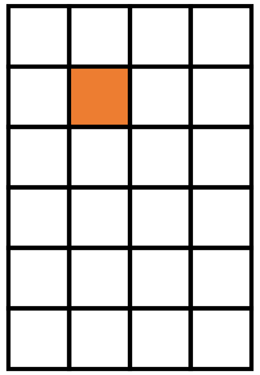


Grid index
Velocity
Density
Temperature
Grid size
…
etc.

Static Markers

Pros and cons



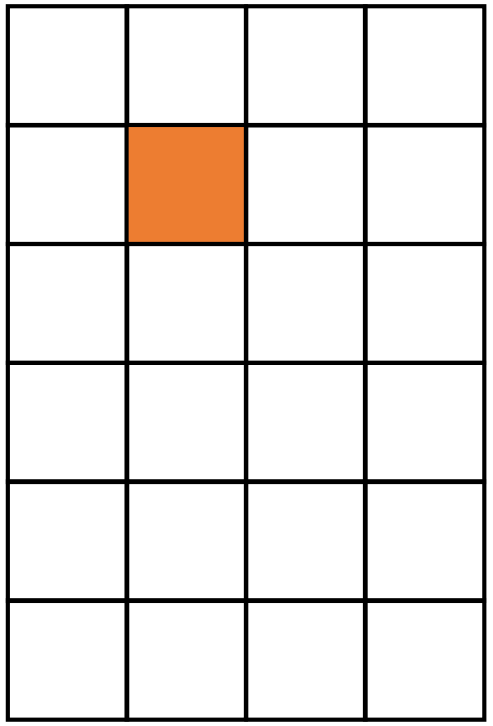


- Lagrangian view:
  - Pros:
    - Advection (Quantity preservation)
    - Boundary condition (Conformal discretization)
    - Coupling with solids
  - Cons:
    - Spatial derivative
    - High spatial discretization error
    - Neighbor search
    - Unbounded distortion
    - Explicit collision handling

- Eulerian view:
  - Pros:
    - Spatial derivative for free (finite difference)
    - Low spatial discretization error
    - Fixed topology (good for neighbor search)
    - Bounded distortion
    - Collision free
  - Cons:
    - Advection
    - Boundaries
    - Coupling with solids

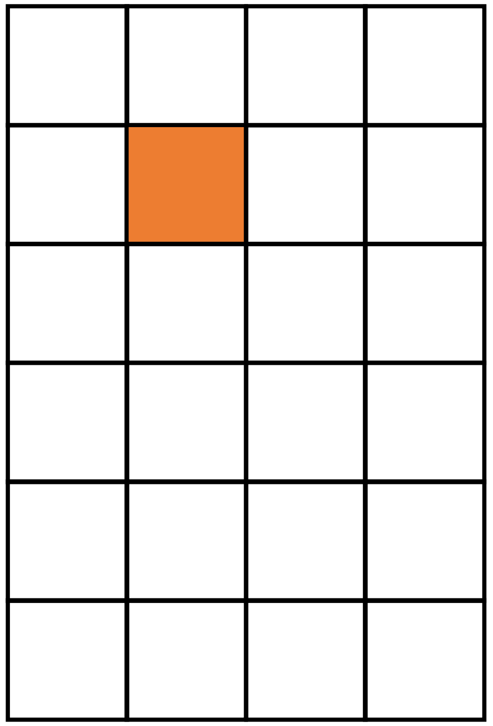
Spatial derivatives under the Eulerian viewpoint

- Spatial derivative in a grid:
  - $\nabla q_{i,j,k} = \begin{bmatrix} \partial q_{i,j,k}/\partial x \\ \partial q_{i,j,k}/\partial y \\ \partial q_{i,j,k}/\partial z \end{bmatrix}$
  - The dimensions can be ***decoupled*** when computing the spatial derivatives due to the ***structural grid***.



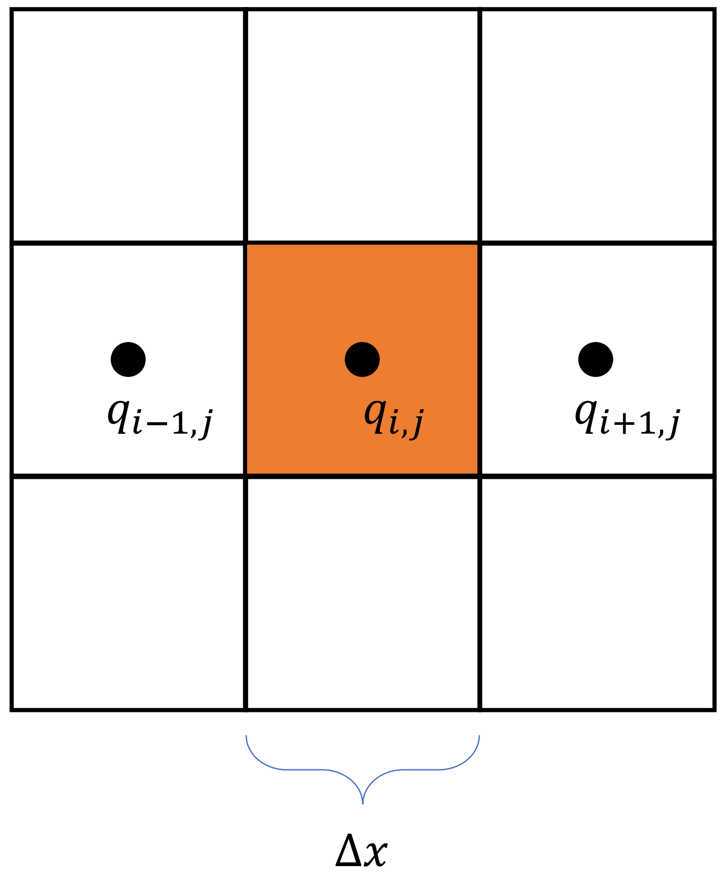
Spatial derivatives under the Eulerian viewpoint

- Spatial derivative in a grid:
  - $\nabla q_{i,j,k} = \begin{bmatrix} \partial q_{i,j,k}/\partial x \\ \partial q_{i,j,k}/\partial y \\ \partial q_{i,j,k}/\partial z \end{bmatrix}$
  - The dimensions can be ***decoupled*** when computing the spatial derivatives due to the ***structural grid***.



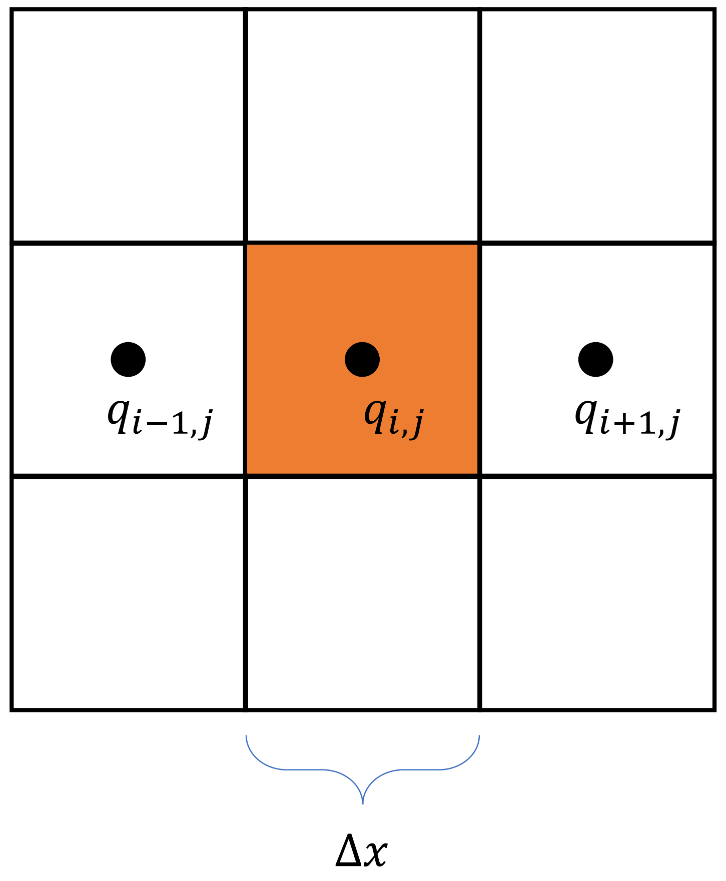
How to compute $\partial q/\partial x$?

- Finite difference, two options:
  - Forward difference:
    - $\left(\frac{\partial q}{\partial x}\right)_i \approx \frac{q_{i+1}-q_i}{\Delta x}$
    - Accurate to $\mathcal{O}(\Delta x)$
    - Biased
  - Central difference:
    - $\left(\frac{\partial q}{\partial x}\right)_i \approx \frac{q_{i+1}-q_{i-1}}{2\Delta x}$
    - Accurate to $\mathcal{O}(\Delta x^2)$
    - Unbiased



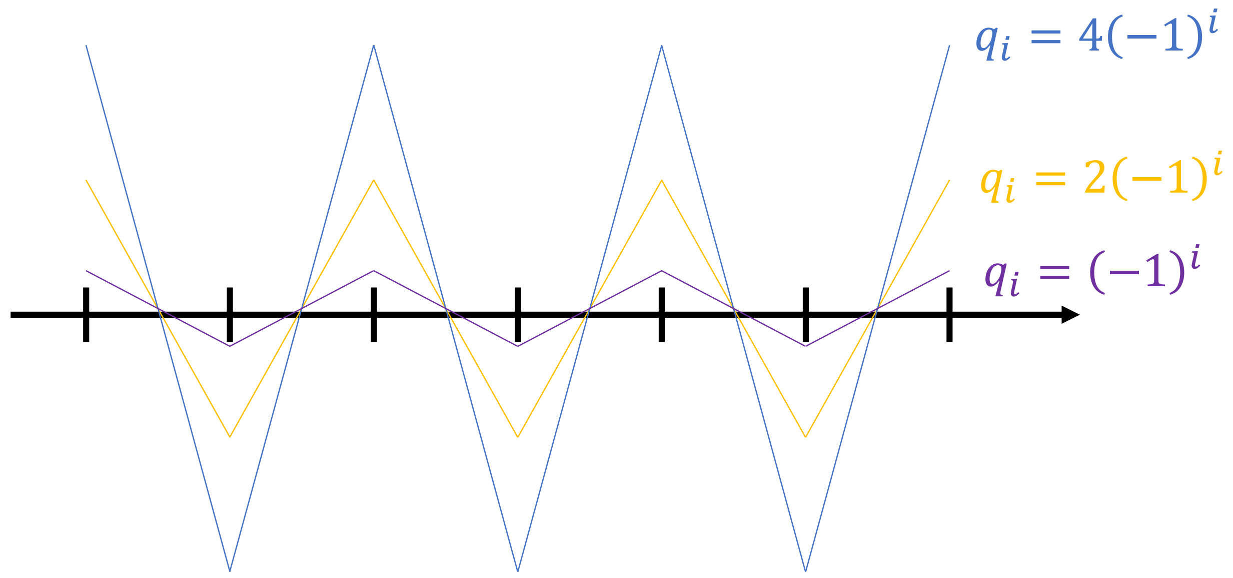
How to compute $\partial q / \partial x$?

- Finite difference, two options:
  - Forward difference:
    - $\left(\frac{\partial q}{\partial x}\right)_i \approx \frac{q_{i+1} - q_i}{\Delta x}$
    - Accurate to $\mathcal{O}(\Delta x)$
    - Biased
  - Central difference:
    - $\left(\frac{\partial q}{\partial x}\right)_i \approx \frac{q_{i+1} - q_{i-1}}{2\Delta x}$
    - Accurate to $\mathcal{O}(\Delta x^2)$
    - Unbiased



The problem of central difference $\left(\frac{\partial q}{\partial x}\right)_i \approx \frac{q_{i+1}-q_{i-1}}{2\Delta x}$:

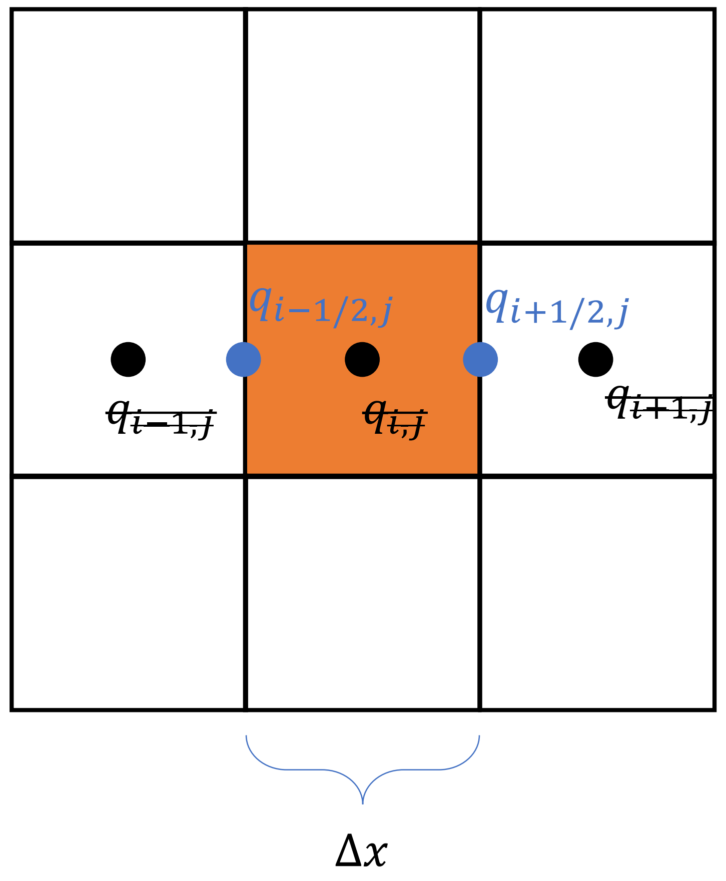
- Non-constant functions are able to register a zero spatial derivative:



Solution: central difference with a “staggered” grid

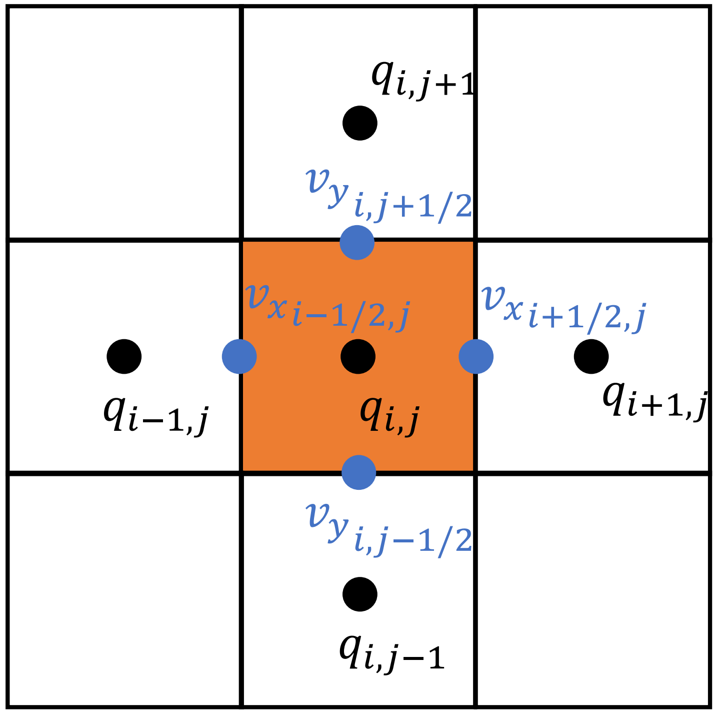
- $\left(\frac{\partial q}{\partial x}\right)_i \approx \frac{q_{i+1/2} - q_{i-1/2}}{\Delta x}$
- Also accurate to $\mathcal{O}(\Delta x^2)$
- Unbiased

- Usually we store the ***velocity*** using the staggered fashion.
- … and store the other (scalar) quantities in the grid centers:
  - e.g. temperature / density / pressure



Staggered grid for fluid simulation

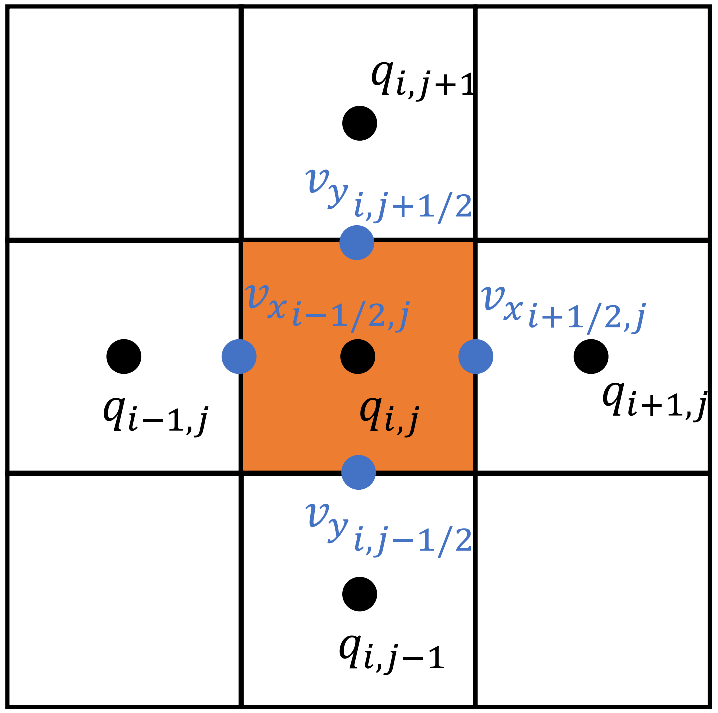
- Compositing a velocity vector in a staggered grid:
  - $v_{i,j} = \left[\frac{v_{x_{i-1/2,j}} + v_{x_{i+1/2,j}}}{2}, \quad \frac{v_{y_{i,j-1/2}} + v_{y_{i,j+1/2}}}{2}\right]$
  - $v_{i+1/2,j} = \left[v_{x_{i+1/2,j}}, \quad \frac{v_{y_{i,j-1/2}} + v_{y_{i,j+1/2}} + v_{y_{i+1,j-1/2}} + v_{y_{i+1,j+1/2}}}{4}\right]$
  - $v_{i,j+1/2} = \left[\frac{v_{y_{i-1/2,j}} + v_{y_{i+1/2,j}} + v_{y_{i-1/2,j+1}} + v_{y_{i+1/2,j+1}}}{4}, \quad v_{y_{i,j+1/2}}\right]$



Note: The staggered grid is first introduced to computational fluid dynamics by Harlow and Welch [1965]. It was called the Marker-and-Cell (MAC) method. Sometimes the staggered grid is also called the MAC grid.

Quiz:

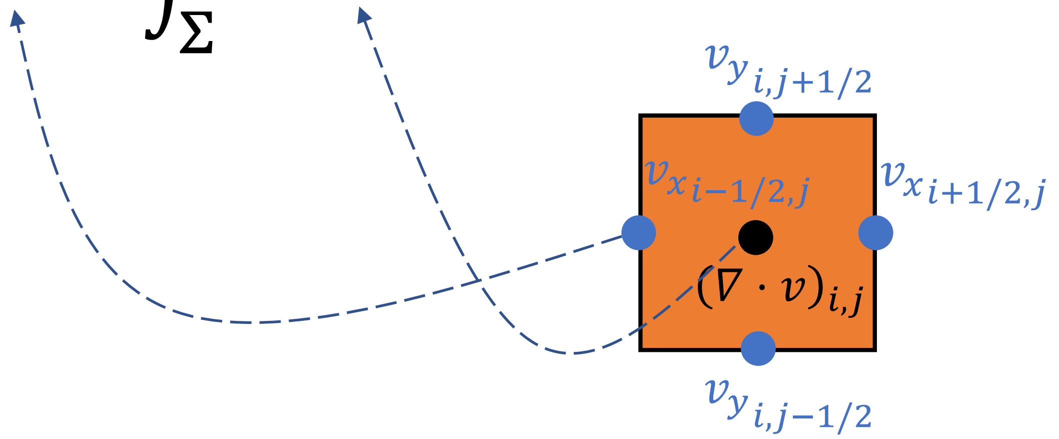
- For a $row \times col$ grid ($row = 3, col = 3$):
  - How many temperature T values do we need to store? $row \times col = 9$
  - How many horizontal velocity v_x values do we need to store? $row \times (col + 1) = 12$
  - How many vertical velocity v_y values do we need to store? $(row + 1) \times col = 12$



Staggered grid: another viewpoint (optional)

- Stokes Theorem (exterior calculus):

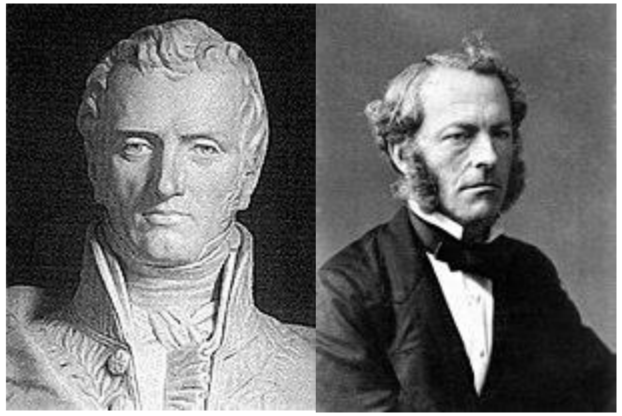
$$\int_{\partial\Sigma} \omega = \int_{\Sigma} d\omega$$



Further Readings:
Discrete Differential Geometry: An Applied Introduction [Crane 2019][Course][Video]
《简明微积分》-- 龚昇

Advection

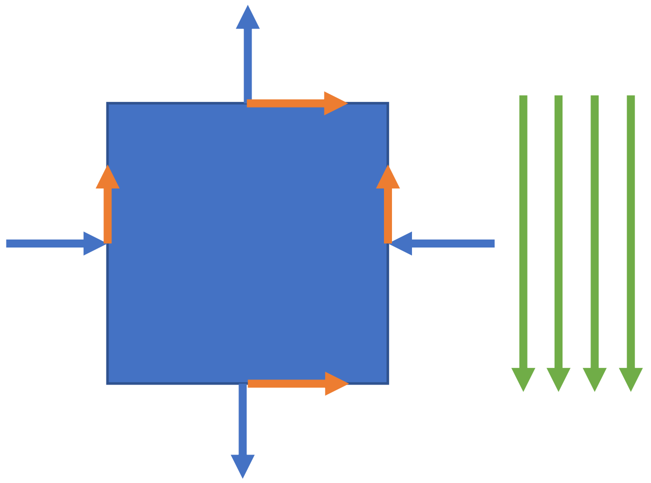
Revisit incompressible Navier-Stokes equation



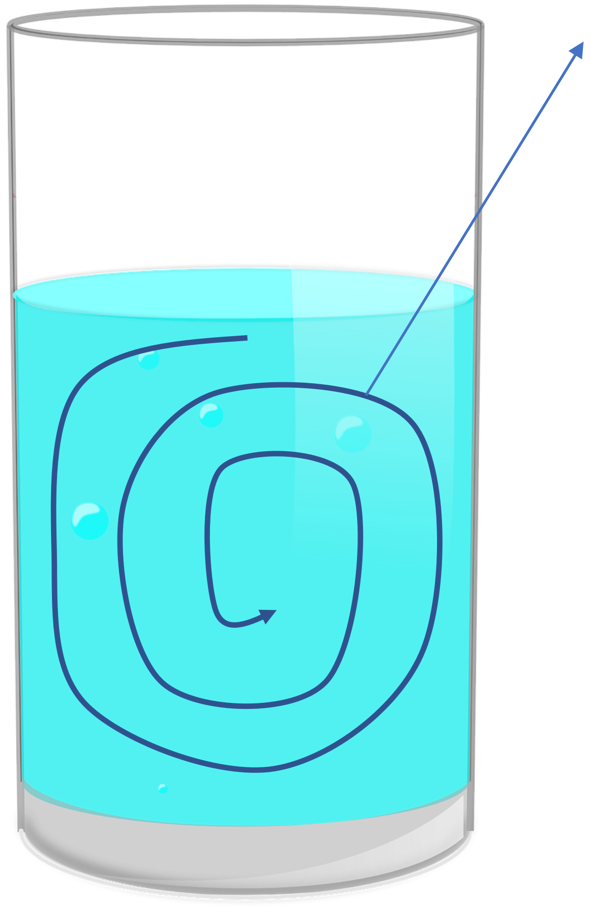
$$ma = f_{ext} + f_{pres} + f_{visc}$$

$$\rho \frac{Dv}{Dt} = \rho g - \nabla p + \mu \nabla^2 v$$

$$\nabla \cdot v = 0$$



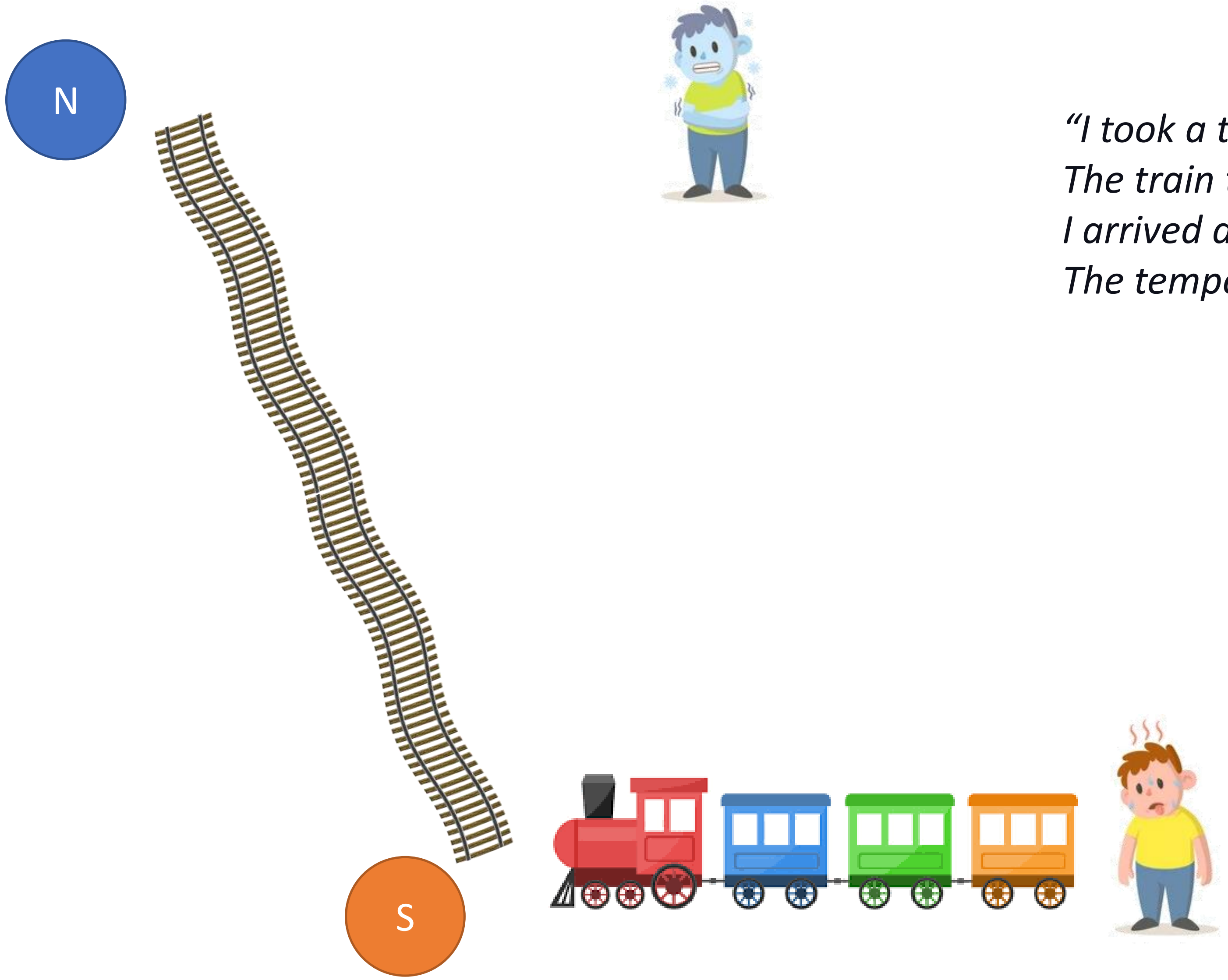
Material derivative $\frac{Df}{Dt} = \frac{\partial f}{\partial t} + v \cdot \nabla f$



- Some functions we want to evaluate depends on both ***space*** and ***time*** coordinates:
  - $f = f(x, t)$
- We have the **total derivative w.r.t. *time*** of f expanded using chain rule:
  - $\frac{d}{\mathrm{dt}} f(x,t) = \frac{\partial f}{\partial t} + \frac{dx}{dt} : \frac{\partial f}{\partial x} = \frac{\partial f}{\partial t} + v \cdot \nabla f$

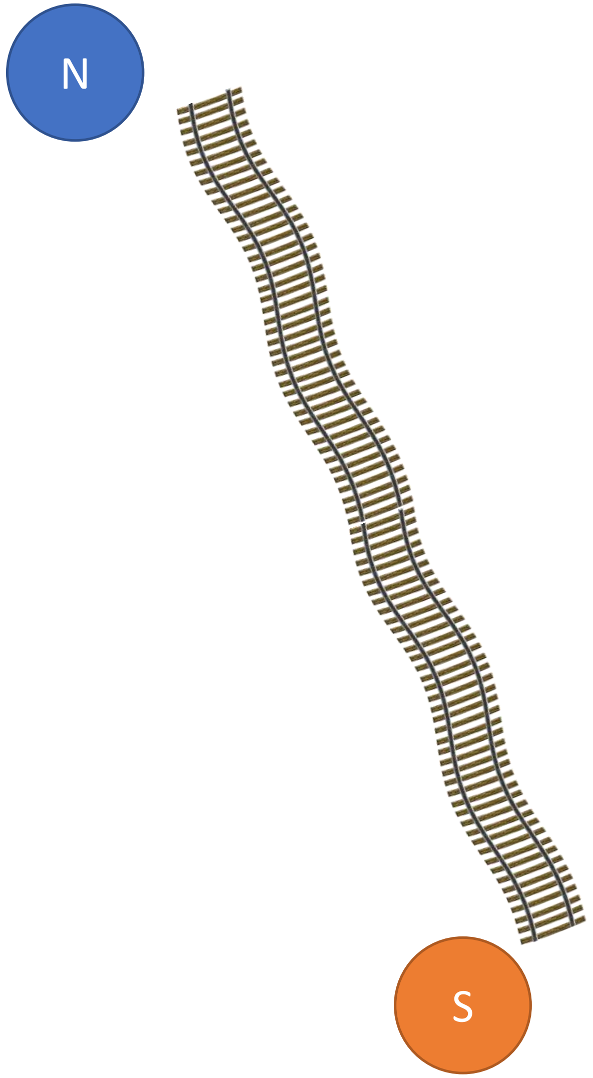
- Other names for $\frac{Df}{Dt}$:
  advective derivative
  convective derivative
  derivative following the motion
  hydrodynamic derivative
  Lagrangian derivative
  particle derivative
  substantial derivative
  substantive derivative
  Stokes derivative

Material derivative explained using a train



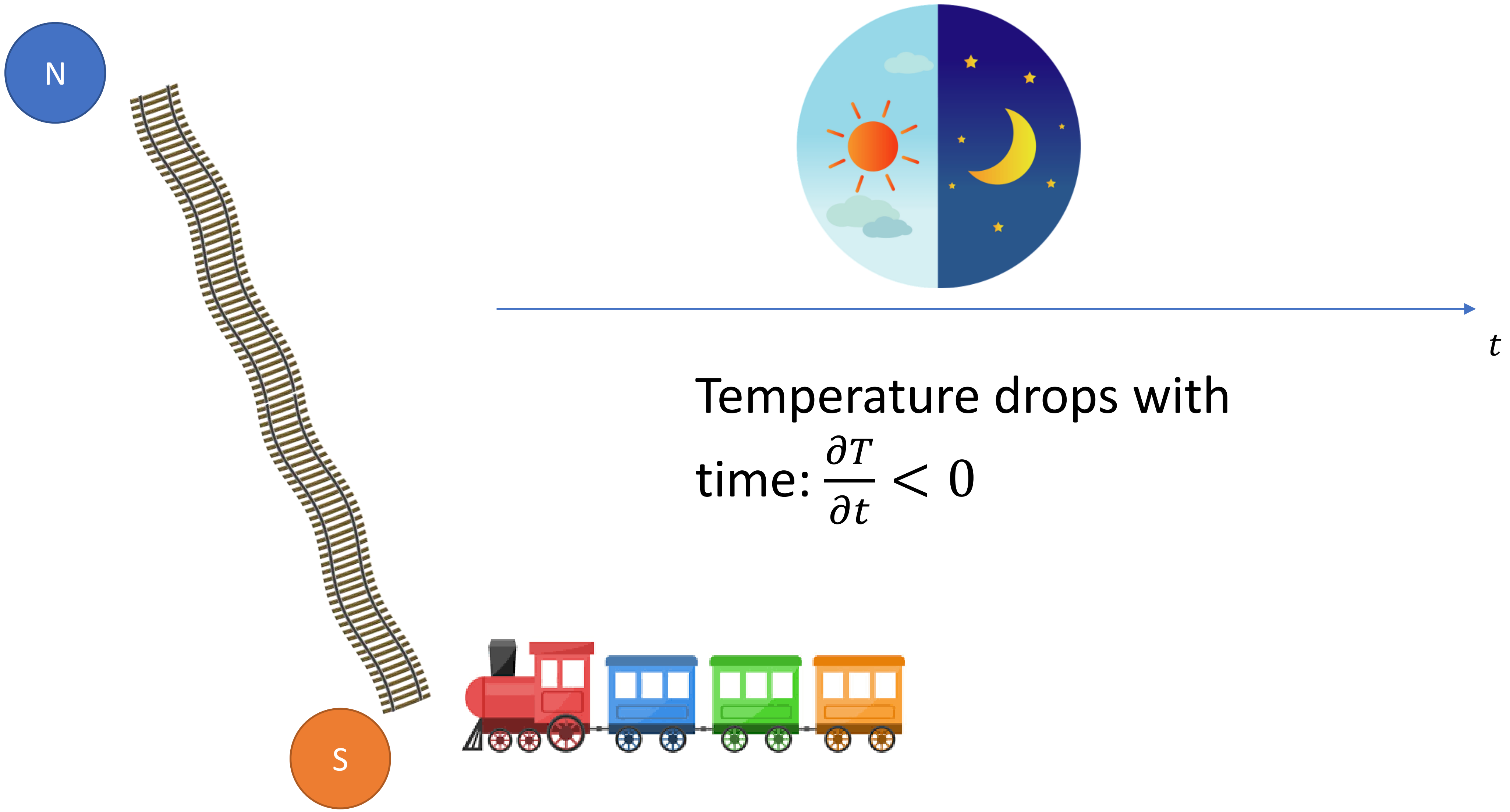
*"I took a train from the **south** to the **north**.*
*The train took off at **1:00 PM**. It was **35 °C** outside.*
*I arrived at **7:00 PM**, and it was so freaking cold there.*
*The temperature dropped to only **5 °C**!"*

Material derivative explained using a train



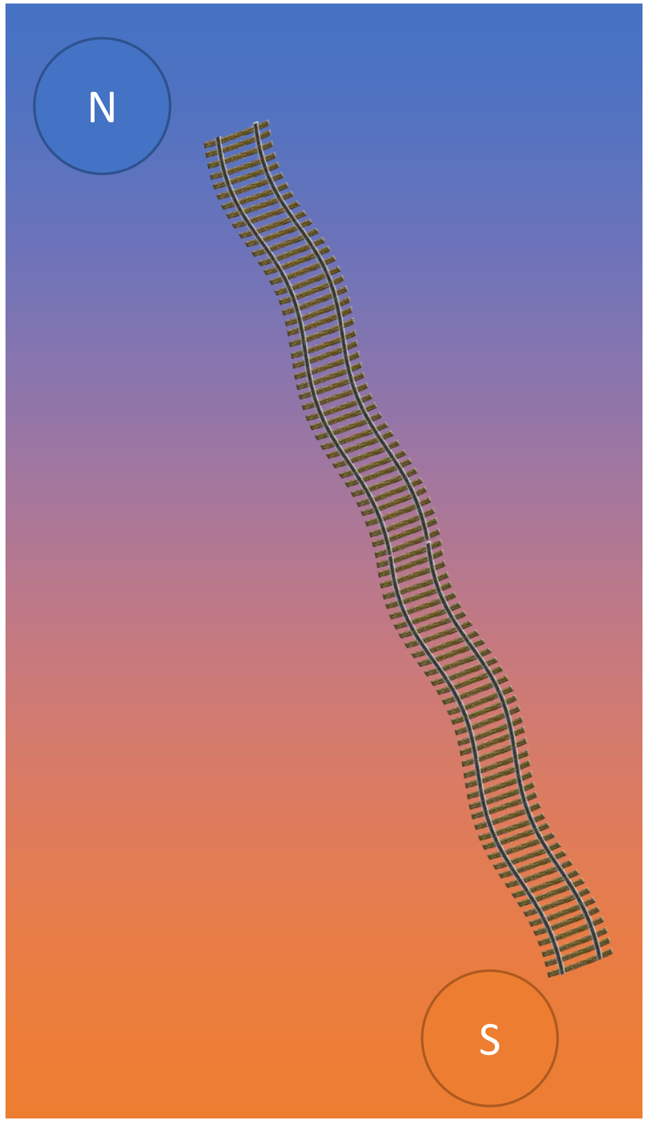
"I took a train from the ***south*** *to the* ***north****.*
The train took off at ***1:00 PM****. It was* ***35 °C*** *outside.*
I arrived at ***7:00 PM****, and it was so freaking cold there.*
The temperature dropped to only ***5 °C****!"*

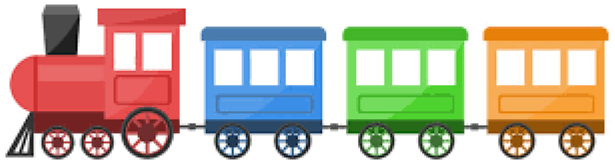
When we freeze the ***space*** coordinate



Temperature drops with time: $\frac{\partial T}{\partial t} < 0$

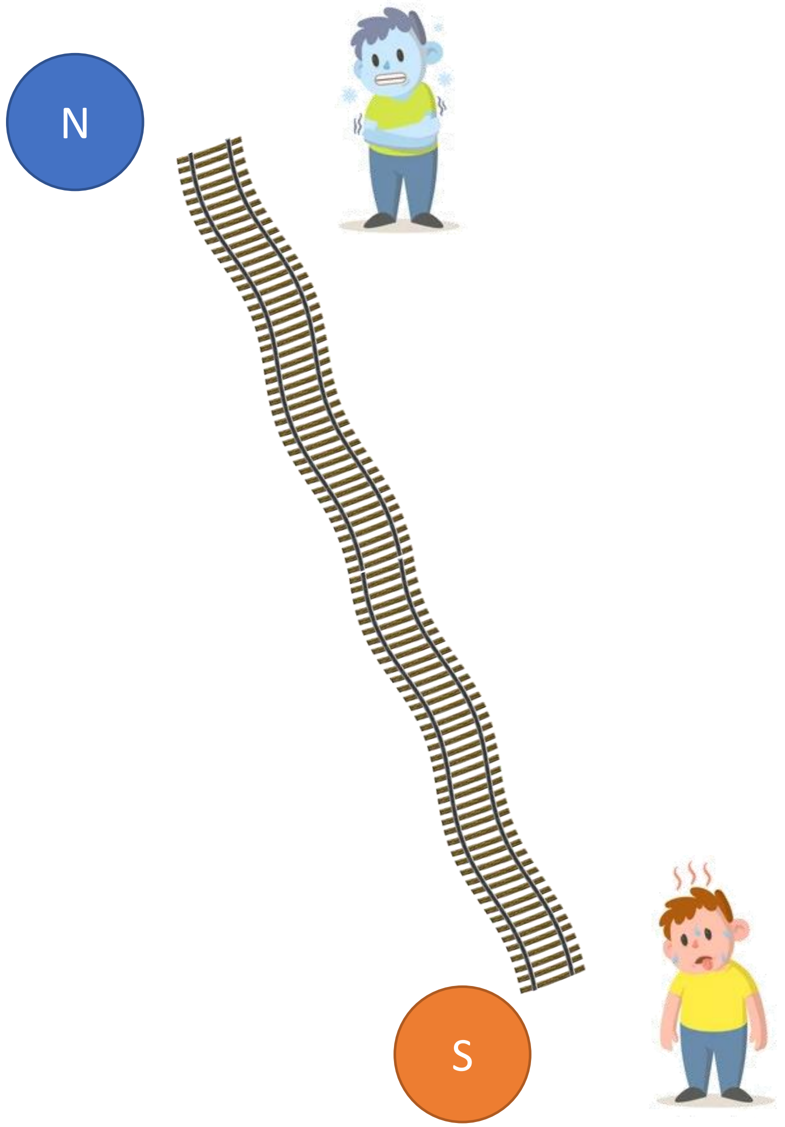
When we freeze the *time* coordinate





Temperature drops along my moving trajectory: $v \cdot \nabla T < 0$

Material derivative explained using a train



*“I took a train from the **south** to the **north**.*
*The train took off at **1:00 PM**. It was **35 °C** outside.*
*I arrived at **7:00 PM**, and it was so freaking cold there.*
*The temperature dropped to only **5 °C**!”*

It was getting colder because:

$$\frac{DT}{Dt} = \frac{\partial T}{\partial t} + v \cdot \nabla T < 0$$

Material derivative of vectors

- $\frac{Dq}{Dt} = \frac{\partial q}{\partial t} + v \cdot \nabla q$
- If $\boldsymbol{q}$ is a vector: $\boldsymbol{q} = \left[q_x, q_y, q_z\right]^T$
    - $\frac{D\boldsymbol{q}}{Dt} = \frac{\partial \boldsymbol{q}}{\partial t} + v \cdot \nabla \boldsymbol{q} = \frac{\partial \boldsymbol{q}}{\partial t} + \boldsymbol{v}: \begin{bmatrix} \begin{bmatrix} \frac{\partial q_x}{\partial x} \\ \frac{\partial q_y}{\partial x} \\ \frac{\partial q_z}{\partial x} \end{bmatrix} \\ \begin{bmatrix} \frac{\partial q_x}{\partial y} \\ \frac{\partial q_y}{\partial y} \\ \frac{\partial q_z}{\partial y} \end{bmatrix} \\ \begin{bmatrix} \frac{\partial q_x}{\partial z} \\ \frac{\partial q_y}{\partial z} \\ \frac{\partial q_z}{\partial z} \end{bmatrix} \end{bmatrix} = \frac{\partial \boldsymbol{q}}{\partial t} + v_x \begin{bmatrix} \frac{\partial q_x}{\partial x} \\ \frac{\partial q_y}{\partial x} \\ \frac{\partial q_z}{\partial x} \end{bmatrix} + v_y \begin{bmatrix} \frac{\partial q_x}{\partial y} \\ \frac{\partial q_y}{\partial y} \\ \frac{\partial q_z}{\partial y} \end{bmatrix} + v_z \begin{bmatrix} \frac{\partial q_x}{\partial z} \\ \frac{\partial q_y}{\partial z} \\ \frac{\partial q_z}{\partial z} \end{bmatrix} = \frac{\partial \boldsymbol{q}}{\partial t} + \begin{bmatrix} v_x \frac{\partial q_x}{\partial x} + v_y \frac{\partial q_x}{\partial y} + v_z \frac{\partial q_x}{\partial z} \\ v_x \frac{\partial q_y}{\partial x} + v_y \frac{\partial q_y}{\partial y} + v_z \frac{\partial q_y}{\partial z} \\ v_x \frac{\partial q_z}{\partial x} + v_y \frac{\partial q_z}{\partial y} + v_z \frac{\partial q_z}{\partial z} \end{bmatrix}$

Material derivative of vectors (cont'd)

- If $\boldsymbol{q}$ is a vector: $\boldsymbol{q} = \left[q_x, q_y, q_z\right]^T$
  - $\frac{D\boldsymbol{q}}{Dt} = \frac{\partial \boldsymbol{q}}{\partial t} + v \cdot \nabla \boldsymbol{q} = \begin{bmatrix} \frac{\partial q_x}{\partial t} + v \cdot \nabla q_x \\ \frac{\partial q_y}{\partial t} + v \cdot \nabla q_y \\ \frac{\partial q_z}{\partial t} + v \cdot \nabla q_z \end{bmatrix}$

Material derivative of velocity

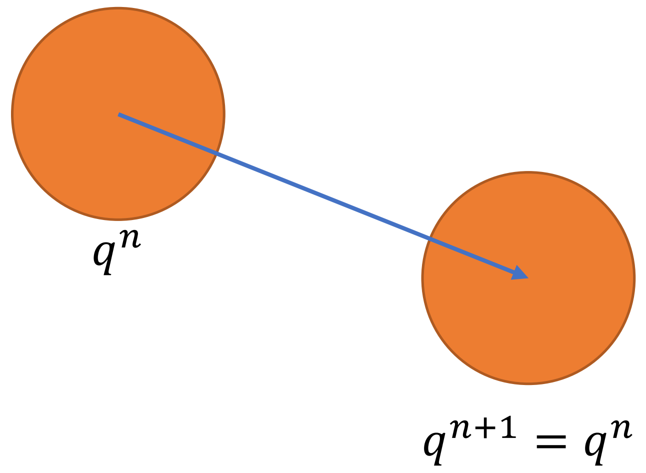
- The *“self-advection”*
  - $\frac{D\boldsymbol{v}}{Dt} = \frac{\partial \boldsymbol{v}}{\partial t} + v \cdot \nabla \boldsymbol{v} = \begin{bmatrix} \frac{\partial v_x}{\partial t} + v \cdot \nabla v_x \\ \frac{\partial v_y}{\partial t} + v \cdot \nabla v_y \\ \frac{\partial v_z}{\partial t} + v \cdot \nabla v_z \end{bmatrix}$
  - … is nothing but the material derivative of the velocity itself.

Advection: $\frac{Dq}{Dt} = \frac{\partial q}{\partial t} + v \cdot \nabla q = 0$

- "Quantities flow with the velocity field"

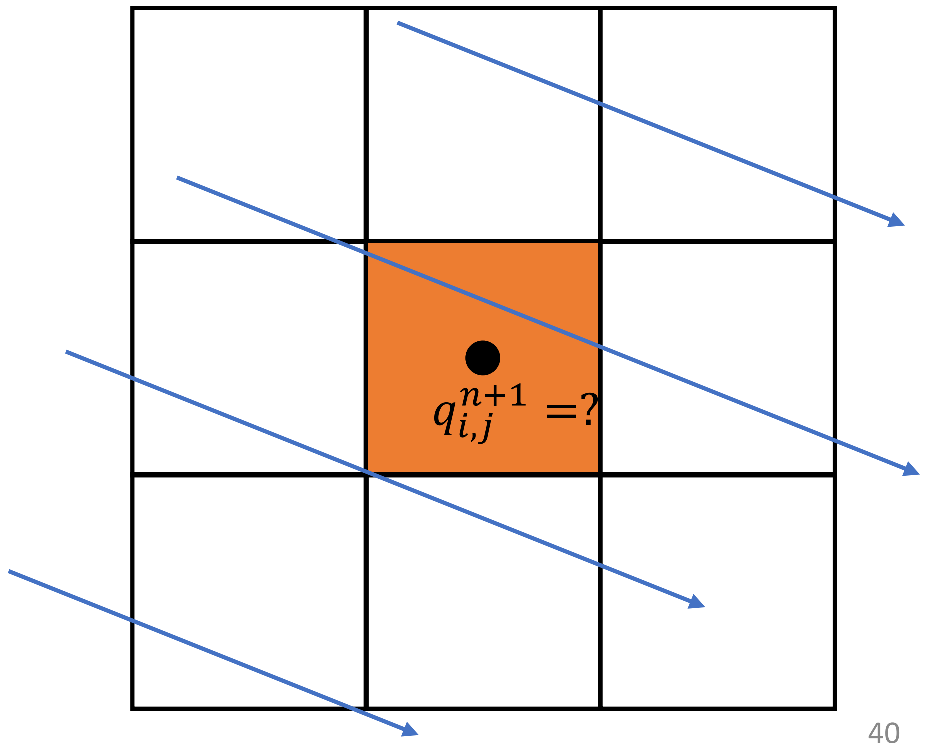
Lagrangian view

$q^{n+1} = q^n$ if $\frac{Dq}{Dt} = 0$



Eulerian view

$q_{i,j}^{n+1} = ?$

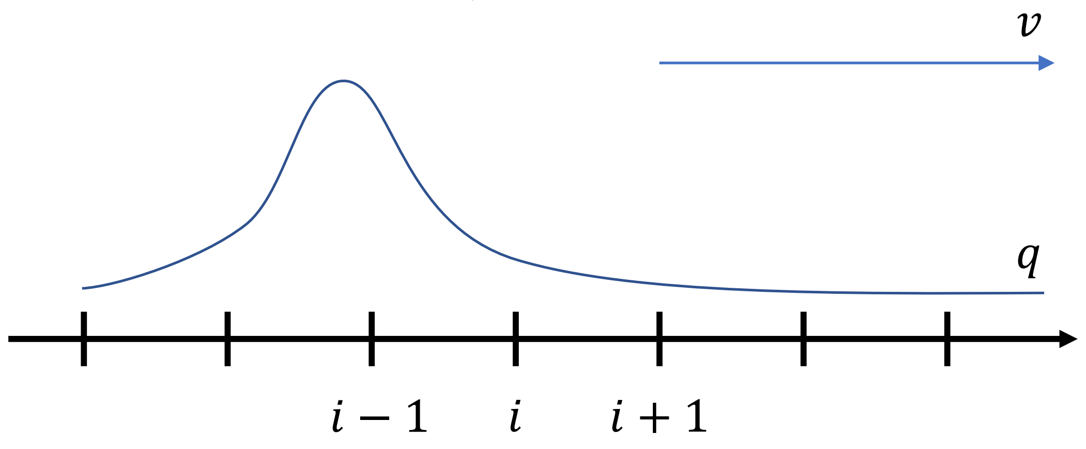


Attempt 1: Finite difference

- $\frac{\partial q}{\partial t} + v \cdot \nabla q = 0$
- $\Rightarrow \frac{q_i^{n+1} - q_i^n}{\Delta t} + v^n \cdot \frac{q_{i+1}^n - q_{i-1}^n}{2\Delta x} = 0$
- $\Rightarrow \; q_i^{n+1} = q_i^n - \Delta t v^n \cdot \frac{q_{i+1}^n - q_{i-1}^n}{2\Delta x}$

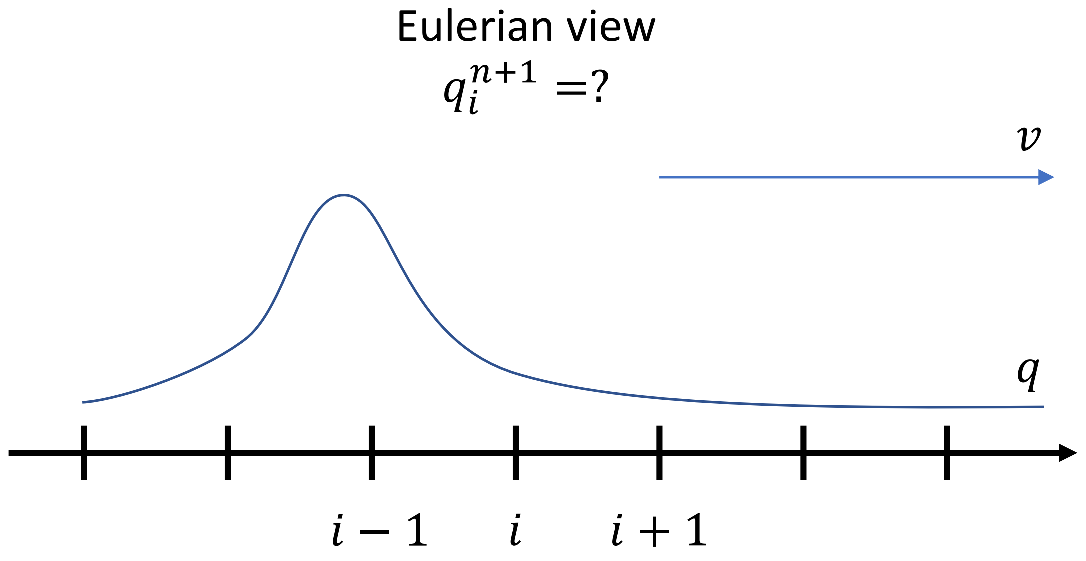
Eulerian view

$q_i^{n+1} = ?$



Attempt 1: Finite difference

- $\frac{\partial q}{\partial t} + v \cdot \nabla q = 0$
- $\Rightarrow \frac{q_i^{n+1} - q_i^n}{\Delta t} + v^n \cdot \frac{q_{i+1}^n - q_{i-1}^n}{2\Delta x} = 0$
- $\Rightarrow \; q_i^{n+1} = q_i^n - \Delta t v^n \cdot \frac{q_{i+1}^n - q_{i-1}^n}{2\Delta x}$



This advection scheme is ***unconditionally unstable***!

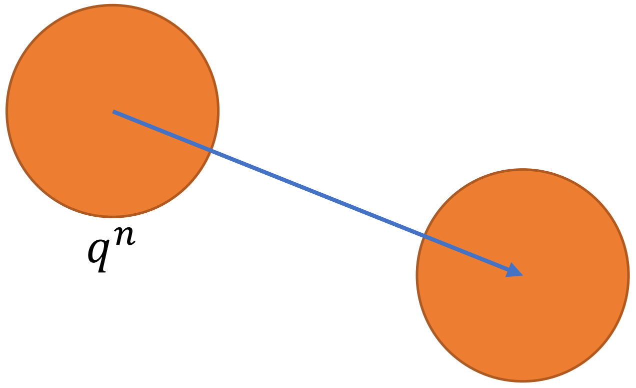
Attempt 2: “semi-Lagrangian”

- It is extremely simple to handle advection in the Lagrangian view…
  - shall we reuse this idea in our Eulerian grid too?
  - The answer is “yes”

  - $q^{n+1} \qquad\qquad = q^n$

Lagrangian view

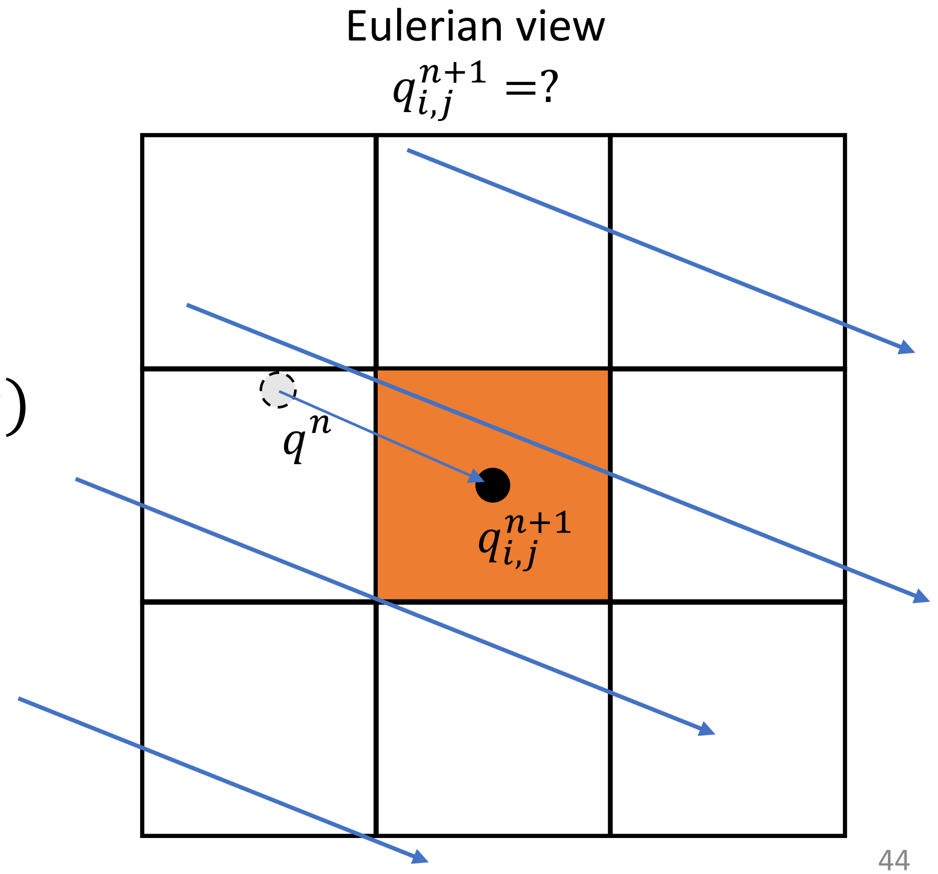
$q^{n+1} = q^n$ if $\frac{Dq}{Dt} = 0$



Attempt 2: “semi-Lagrangian”

- It is extremely simple to handle advection in the Lagrangian view…
  - shall we reuse this idea in our Eulerian grid too?
  - The answer is “yes”

  - $q^{n+1}(x^{n+1}) = q^n(x^n) = q^n(x^{n+1} - \Delta t v^n)$

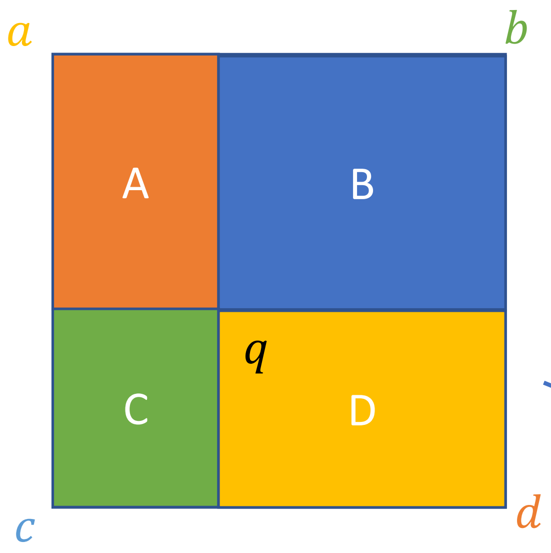


Attempt 2: "semi-Lagrangian"

- How do we get the value for $q^n(x^{n+1} - \Delta t v^n)$?
  - Interpolation!
  - $q^{n+1}(x^{n+1}) = \text{interpolate}(q^n, x^{n+1} - \Delta t v^n)$

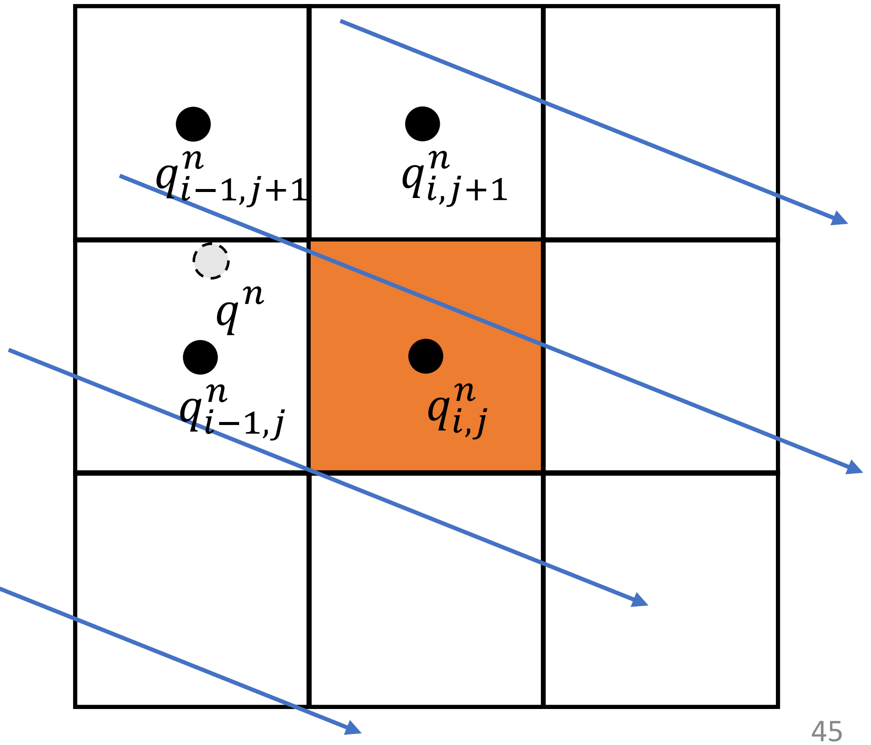
Bi-linear interpolation in 2D:

$$\begin{aligned} q &= lerp(a,b,c,d) \\ &= lerp\big(lerp(a,b), lerp(c,d)\big) \\ &= \frac{D*a + C*b + B*c + A*d}{A+B+C+D} \end{aligned}$$



Eulerian view

$q_{i,j}^{n+1} = q^n$

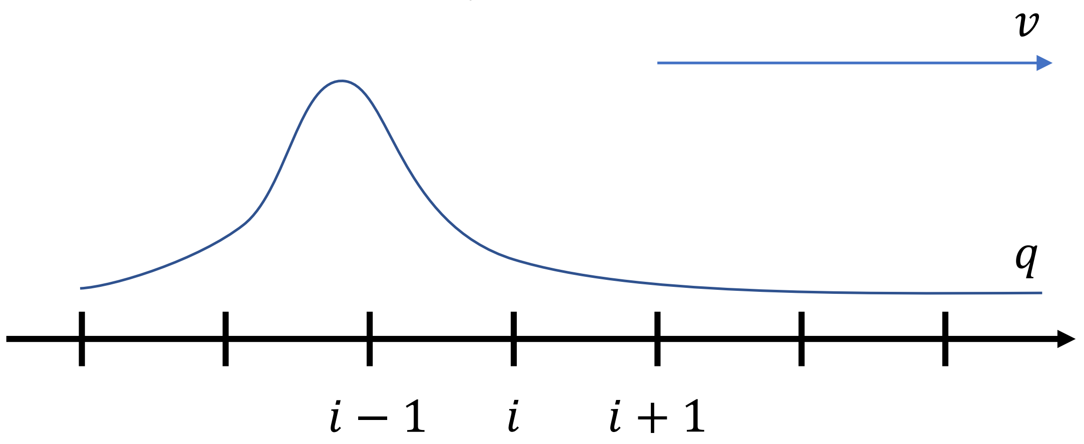


Attempt 2: “semi-Lagrangian”

- $\frac{\partial q}{\partial t} + v \cdot \nabla q = 0$
- $\Rightarrow \ q_i^{n+1} = \text{interpolate}(q^n, x^{n+1} - \Delta t v^n)$

Eulerian view

$q_i^{n+1} = ?$

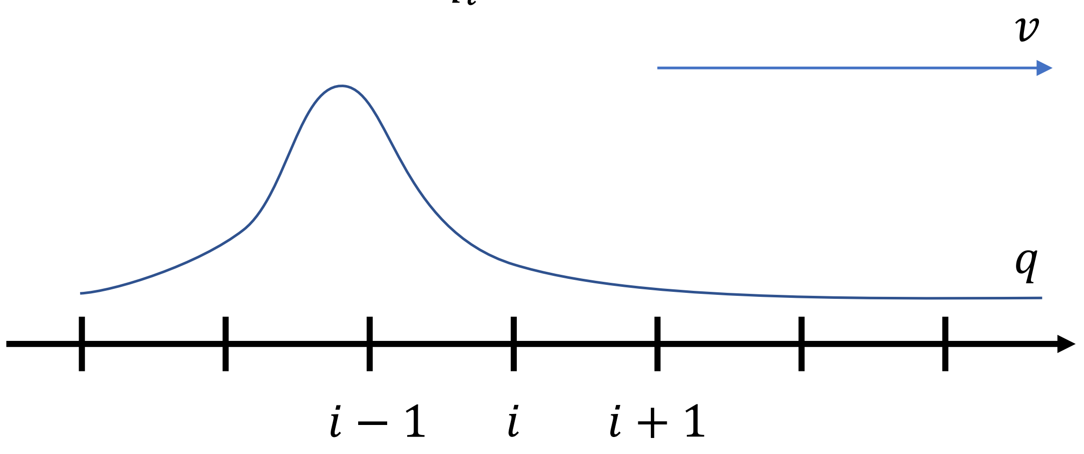


Attempt 2: "semi-Lagrangian"

- $\frac{\partial q}{\partial t} + v \cdot \nabla q = 0$
- $\Rightarrow \; q_i^{n+1} = \text{interpolate}(q^n, x^{n+1} - \Delta t v^n)$

Eulerian view

$q_i^{n+1} = ?$

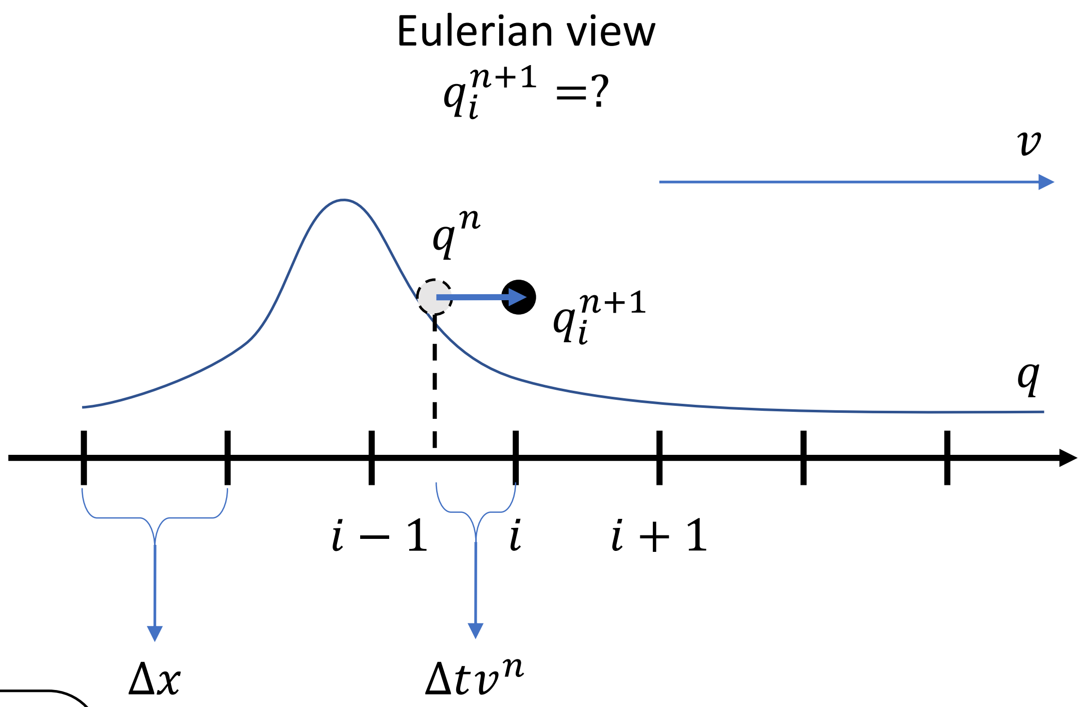


This advection scheme is ***unconditionally stable***!

Semi-Lagrangian advection: what is that?

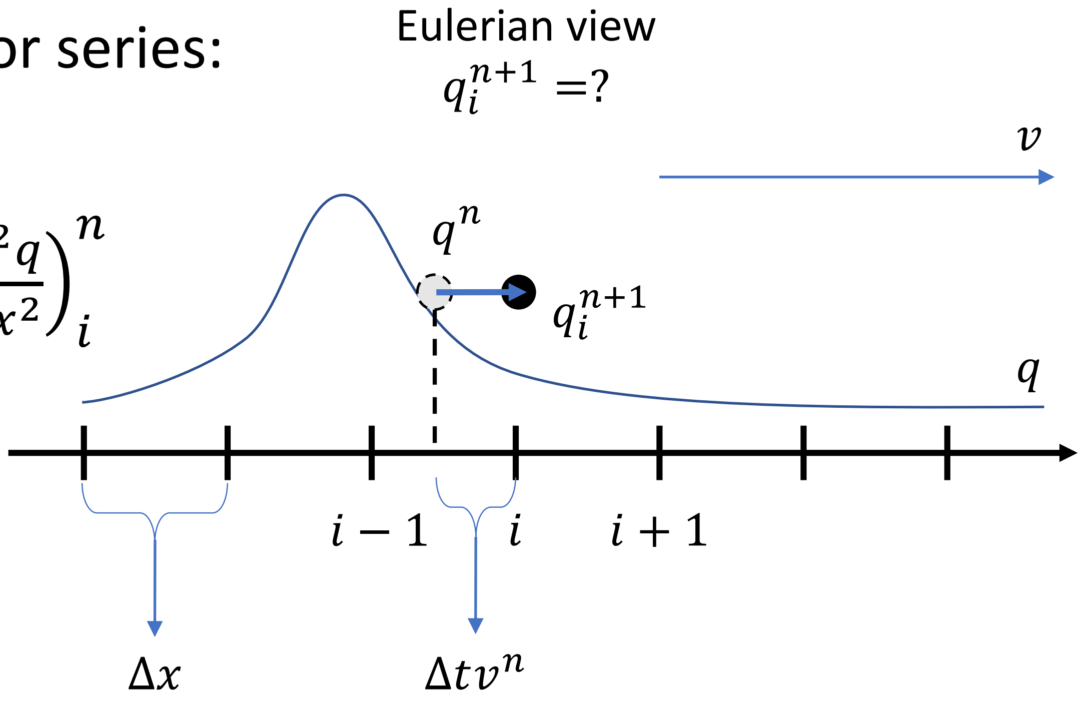
- What we want (in 1D):
  - $\frac{Dq}{Dt} = \frac{\partial q}{\partial t} + v\frac{\partial q}{\partial x} = 0$
- Assuming $v^n\Delta t < \Delta x$:
  - $q_i^{n+1} = \frac{\Delta t v^n}{\Delta x} q_{i-1}^n + \left(1 - \frac{\Delta t v^n}{\Delta x}\right) q_i^n$
  - $\Rightarrow q_i^{n+1} = q_i^n - \Delta t v^n \frac{q_i^n - q_{i-1}^n}{\Delta x}$
  - $\Rightarrow \frac{q_i^{n+1} - q_i^n}{\Delta t} + v^n \frac{q_i^n - q_{i-1}^n}{\Delta x} = 0$

The semi-Lagrangian scheme is essentially a forward Euler scheme with a "velocity-aware" one-sided finite difference



Semi-Lagrangian advection: what do we lose?

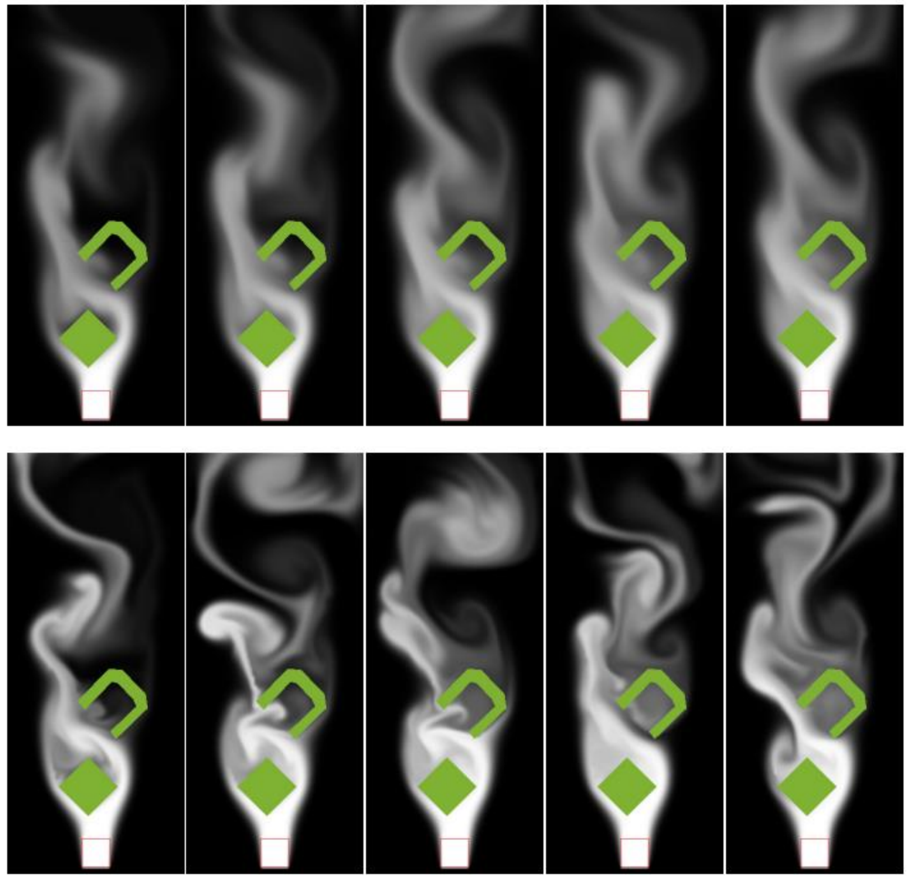
- $q_i^{n+1} = q_i^n - \Delta t v^n \frac{q_i^n - q_{i-1}^n}{\Delta x}$
- We can also expand q_{i-1}^n at q_i^n using the Taylor series:
  - $q_{i-1}^n \approx q_i^n - \left(\frac{\partial q}{\partial x}\right)_i^n \Delta x + \left(\frac{\partial^2 q}{\partial x^2}\right)_i^n \frac{\Delta x^2}{2}$
  - $\Rightarrow q_i^{n+1} \approx q_i^n - \Delta t v^n \left(\frac{\partial q}{\partial x}\right)_i^n + \frac{\Delta t v^n \Delta x}{2} \left(\frac{\partial^2 q}{\partial x^2}\right)_i^n$
  - $\Rightarrow \frac{q_i^{n+1} - q_i^n}{\Delta t} + v^n \left(\frac{\partial q}{\partial x}\right)_i^n \approx \frac{v^n \Delta x}{2} \left(\frac{\partial^2 q}{\partial x^2}\right)_i^n$
  - $\Rightarrow \frac{Dq}{Dt} \approx \frac{v^n \Delta x}{2} \frac{\partial^2 q}{\partial x^2} \neq 0$



The semi-Lagrangian advection scheme introduces "numerical dissipation/viscosity"

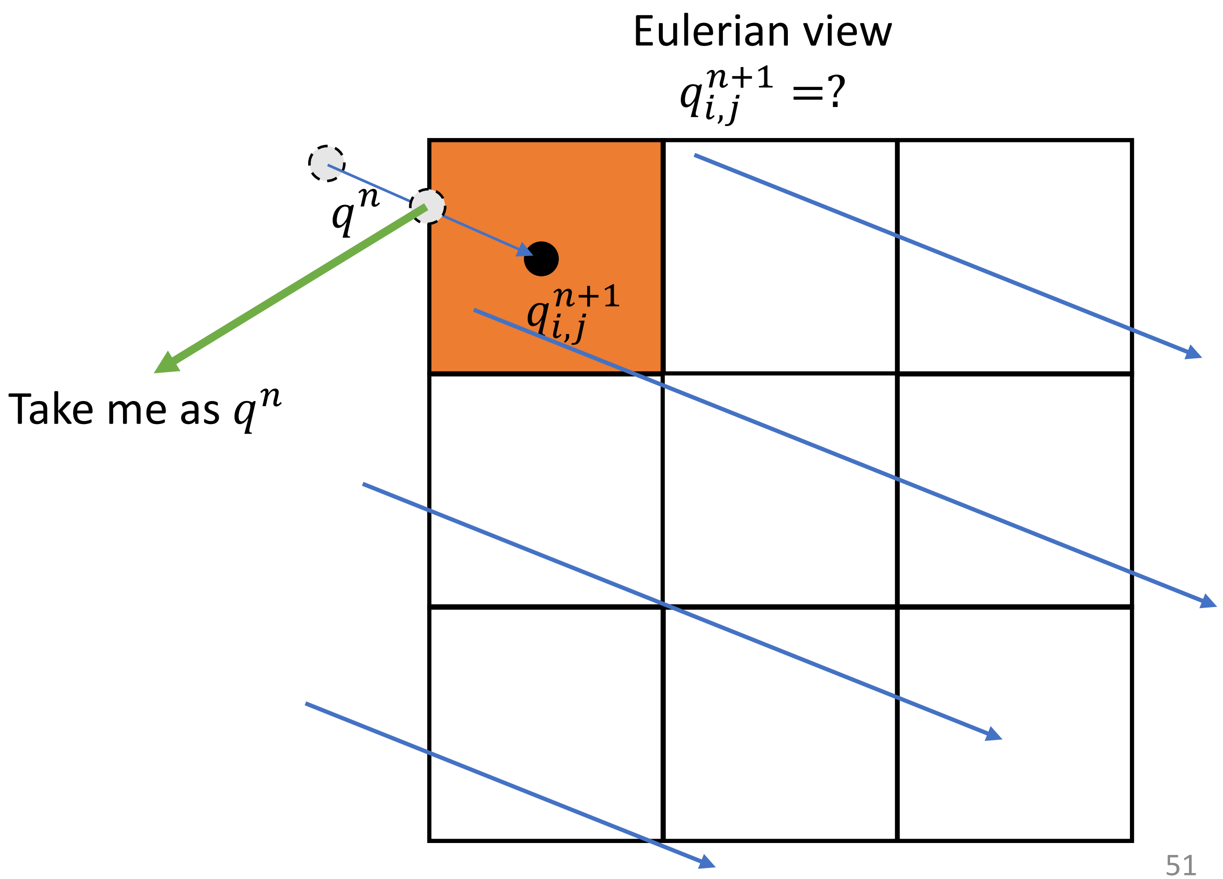
A better advection scheme with less dissipation?

- Sharper interpolation
  - Cubic Hermite spline interpolation [Link]
- Better error correction schemes
  - MacCormack method [Link]
  - Back and Forth Error Compensation and Correction (BFECC) [Kim et al. 2005][Paper]


BFECC [Kim et al. 2005]

What if the backtracked “particle” is out-of-boundary?

- Simplest solution:
  - Take the value on the boundary



One numerical time-stepping for N-S equations

- Given q^n, where q can be velocity, density, temperature etc.
  - Step 1 Advection:
    - $q^{n+1} = advect(v^n, \Delta t, q^n)$
    - $\tilde{v} = advect(v^n, \Delta t, v^n)$
  - Step 2 Applying forces:
    - $\tilde{\tilde{v}} = \tilde{v} + \Delta t(g + \nu \nabla^2 \tilde{v})$
  - Step 3 Projection:
    - $v^{n+1} = project(\Delta t, \tilde{\tilde{v}})$
  - Return v^{n+1}, q^{n+1}

One numerical time-stepping for N-S equations

- Given q^n, where q can be velocity, density, temperature etc.
  - Step 1 Advection:
    - $q^{n+1} = advect(v^n, \Delta t, q^n)$
    - $\tilde{v} = advect(v^n, \Delta t, v^n)$
  - Step 2 Applying forces:
    - $\tilde{\tilde{v}} = \tilde{v} + \Delta t(g + \nu \nabla^2 \tilde{v})$
  - Step 3 Projection:
    - $v^{n+1} = project(\Delta t, \tilde{\tilde{v}})$
  - Return v^{n+1}, q^{n+1}

Sometimes we can drop the viscous force term $\nu \nabla^2 \tilde{v}$ for water/smoke/"inviscid fluid" simulations. Sometimes we can even drop the gravitational force term g for smoke simulations.

One numerical time-stepping for N-S equations

- Given q^n, where q can be velocity, density, temperature etc.
  - Step 1 Advection:
    - $q^{n+1} = advect(v^n, \Delta t, q^n)$
    - $\tilde{v} = advect(v^n, \Delta t, v^n)$
  - Step 2 Applying forces:
    - $\tilde{\tilde{v}} = \tilde{v} + \Delta t(g + \nu \nabla^2 \tilde{v})$
  - Step 3 Projection:
    - $v^{n+1} = project(\Delta t, \tilde{\tilde{v}})$
  - Return v^{n+1}, q^{n+1}

$$\frac{Dv}{Dt} = g - \frac{1}{\rho}\nabla p + \nu \nabla^2 v$$

$$\nabla \cdot v = 0$$

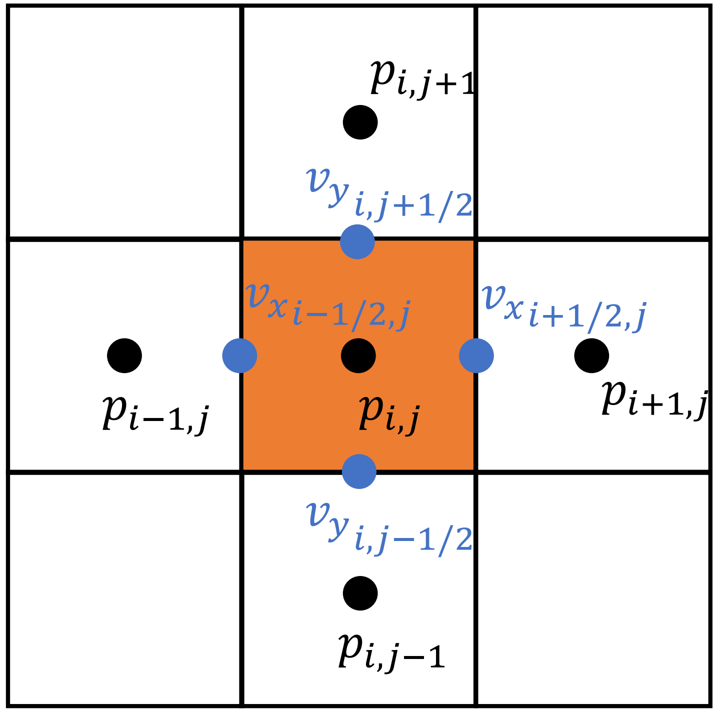
Projection

The projection step

- We want to solve: $\frac{\partial v}{\partial t} = -\frac{1}{\rho}\nabla p$

  $\nabla \cdot v = 0$

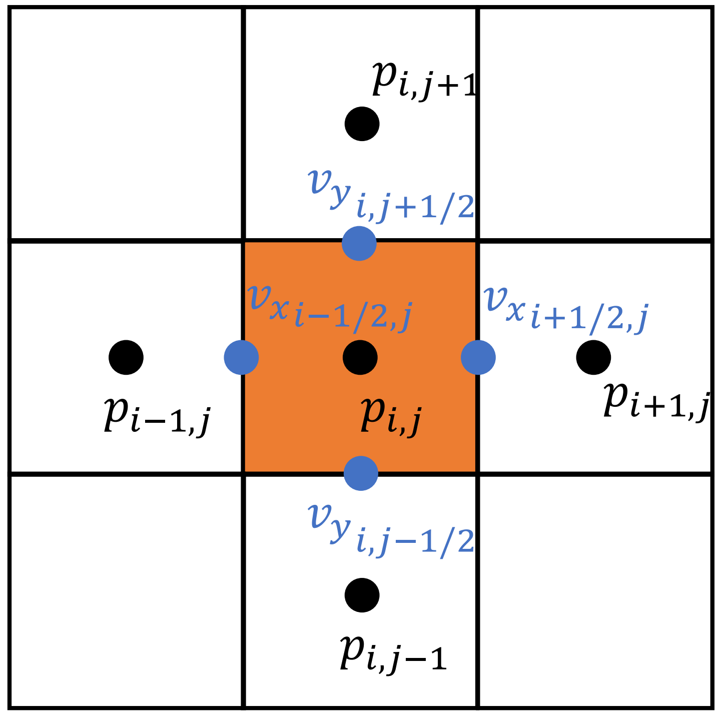
- Well, the finite difference comes to the rescue again:
  - $\frac{{v_x}_{i-1/2,j}^{n+1} - {v_x}_{i-1/2,j}^{n}}{\Delta t} = -\frac{1}{\rho}\frac{p_{i,j} - p_{i-1,j}}{\Delta x}$
  - s.t. $\underbrace{\frac{{v_x}_{i+1/2,j}^{n+1} - {v_x}_{i-1/2,j}^{n+1}}{\Delta x}}_{\frac{\partial v_x}{\partial x}} + \underbrace{\frac{{v_y}_{i,j+1/2}^{n+1} - {v_y}_{i,j-1/2}^{n+1}}{\Delta x}}_{\frac{\partial v_y}{\partial y}} = 0$



Finite difference

- $\frac{{v_x}_{i-1/2,j}^{n+1} - {v_x}_{i-1/2,j}^{n}}{\Delta t} = -\frac{1}{\rho}\frac{p_{i,j} - p_{i-1,j}}{\Delta x}$ ①
- $\frac{{v_x}_{i+1/2,j}^{n+1} - {v_x}_{i+1/2,j}^{n}}{\Delta t} = -\frac{1}{\rho}\frac{p_{i+1,j} - p_{i,j}}{\Delta x}$ ②
- $\frac{{v_y}_{i,j-1/2}^{n+1} - {v_y}_{i,j-1/2}^{n}}{\Delta t} = -\frac{1}{\rho}\frac{p_{i,j} - p_{i,j-1}}{\Delta x}$ ③
- $\frac{{v_y}_{i,j+1/2}^{n+1} - {v_y}_{i,j+1/2}^{n}}{\Delta t} = -\frac{1}{\rho}\frac{p_{i,j+1} - p_{i,j}}{\Delta x}$ ④

- $\frac{{v_x}_{i+1/2,j}^{n+1} - {v_x}_{i-1/2,j}^{n+1}}{\Delta x} + \frac{{v_y}_{i,j+1/2}^{n+1} - {v_y}_{i,j-1/2}^{n+1}}{\Delta x} = 0$



Finite difference

1 $\dfrac{{v_x}_{i-1/2,j}^{n+1} - {v_x}_{i-1/2,j}^{n}}{\Delta t} = -\dfrac{1}{\rho}\dfrac{p_{i,j} - p_{i-1,j}}{\Delta x}$

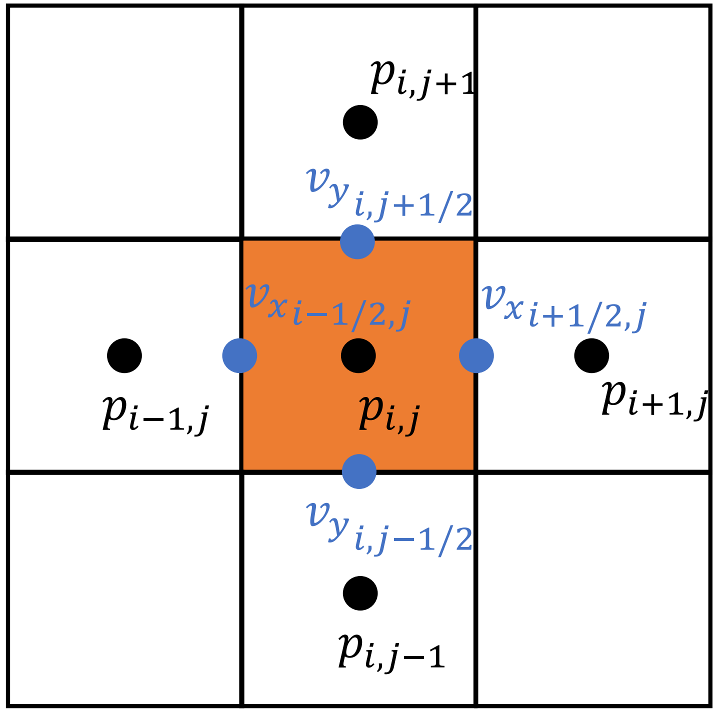
2 $\dfrac{{v_x}_{i+1/2,j}^{n+1} - {v_x}_{i+1/2,j}^{n}}{\Delta t} = -\dfrac{1}{\rho}\dfrac{p_{i+1,j} - p_{i,j}}{\Delta x}$

3 $\dfrac{{v_y}_{i,j-1/2}^{n+1} - {v_y}_{i,j-1/2}^{n}}{\Delta t} = -\dfrac{1}{\rho}\dfrac{p_{i,j} - p_{i,j-1}}{\Delta x}$

4 $\dfrac{{v_y}_{i,j+1/2}^{n+1} - {v_y}_{i,j+1/2}^{n}}{\Delta t} = -\dfrac{1}{\rho}\dfrac{p_{i,j+1} - p_{i,j}}{\Delta x}$

2 − 1 + 4 − 3 :

$$\frac{{v_x}_{i+1/2,j}^{n+1} - {v_x}_{i-1/2,j}^{n+1} + {v_y}_{i,j+1/2}^{n+1} - {v_y}_{i,j-1/2}^{n+1}}{\Delta t}$$
$$-\frac{{v_x}_{i+1/2,j}^{n} - {v_x}_{i-1/2,j}^{n} + {v_y}_{i,j-1/2}^{n} - {v_y}_{i,j-1/2}^{n}}{\Delta t}$$
$$= \frac{1}{\rho}\frac{4p_{i,j} - p_{i+1,j} - p_{i-1,j} - p_{i,j+1} - p_{i,j-1}}{\Delta x}$$



- $\dfrac{{v_x}_{i+1/2,j}^{n+1} - {v_x}_{i-1/2,j}^{n+1}}{\Delta x} + \dfrac{{v_y}_{i,j+1/2}^{n+1} - {v_y}_{i,j-1/2}^{n+1}}{\Delta x} = 0$

Finite difference

① $\dfrac{{v_x}_{i-1/2,j}^{n+1} - {v_x}_{i-1/2,j}^{n}}{\Delta t} = -\dfrac{1}{\rho}\dfrac{p_{i,j} - p_{i-1,j}}{\Delta x}$

② $\dfrac{{v_x}_{i+1/2,j}^{n+1} - {v_x}_{i+1/2,j}^{n}}{\Delta t} = -\dfrac{1}{\rho}\dfrac{p_{i+1,j} - p_{i,j}}{\Delta x}$

③ $\dfrac{{v_y}_{i,j-1/2}^{n+1} - {v_y}_{i,j-1/2}^{n}}{\Delta t} = -\dfrac{1}{\rho}\dfrac{p_{i,j} - p_{i,j-1}}{\Delta x}$

④ $\dfrac{{v_y}_{i,j+1/2}^{n+1} - {v_y}_{i,j+1/2}^{n}}{\Delta t} = -\dfrac{1}{\rho}\dfrac{p_{i,j+1} - p_{i,j}}{\Delta x}$

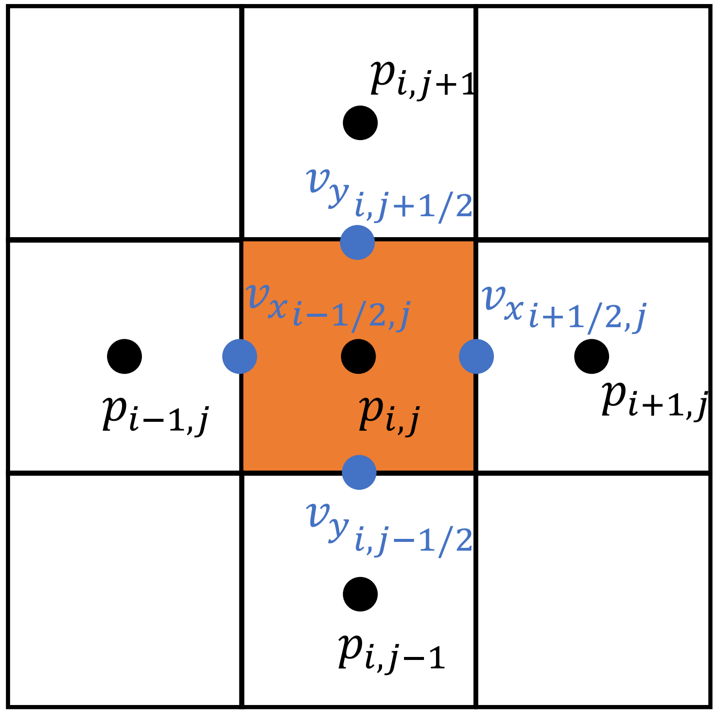
② − ① + ④ − ③ $\dfrac{\Delta t}{\Delta x}$:

$$\frac{{v_x}_{i+1/2,j}^{n+1} - {v_x}_{i-1/2,j}^{n+1} + {v_y}_{i,j+1/2}^{n+1} - {v_y}_{i,j-1/2}^{n+1}}{\Delta x} \quad \boxed{\nabla \cdot v^{n+1}}$$

$$-\frac{{v_x}_{i+1/2,j}^{n} - {v_x}_{i-1/2,j}^{n} + {v_y}_{i,j-1/2}^{n} - {v_y}_{i,j-1/2}^{n}}{\Delta x} \quad \boxed{-\nabla \cdot v^{n}}$$

$$= \frac{\Delta t}{\rho}\frac{4p_{i,j} - p_{i+1,j} - p_{i-1,j} - p_{i,j+1} - p_{i,j-1}}{\Delta x^2} \quad \boxed{-\frac{\Delta t}{\rho}\nabla \cdot \nabla p}$$

$$\boxed{-\frac{\Delta t}{\rho}\nabla \cdot \nabla p = -\nabla \cdot v^{n}}$$



- $\dfrac{{v_x}_{i+1/2,j}^{n+1} - {v_x}_{i-1/2,j}^{n+1}}{\Delta x} + \dfrac{{v_y}_{i,j+1/2}^{n+1} - {v_y}_{i,j-1/2}^{n+1}}{\Delta x} = 0$

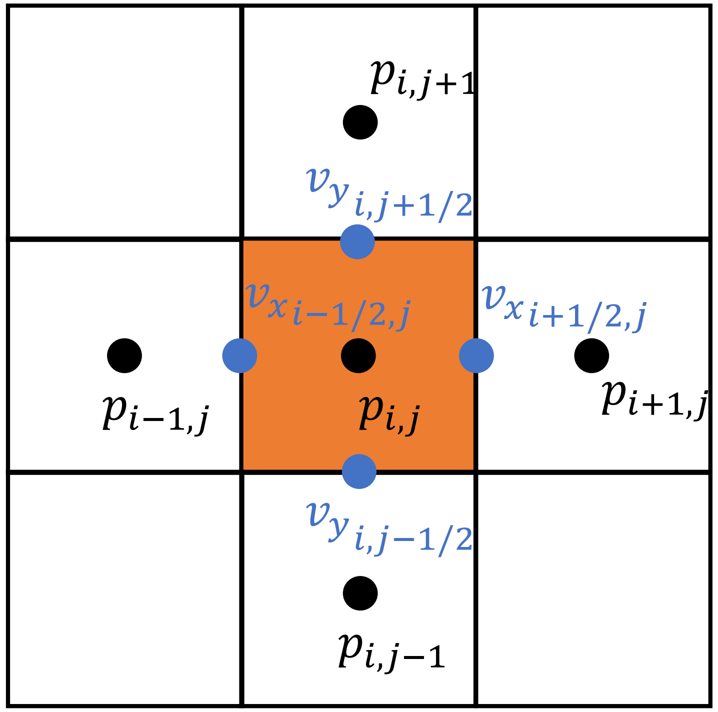
The projection step (A Poisson problem)

- $-\frac{\Delta t}{\rho}\nabla\cdot\nabla p = -\nabla\cdot v^n$

- Or:
  - $\frac{\Delta t}{\rho}\frac{4p_{i,j}-p_{i+1,j}-p_{i-1,j}-p_{i,j+1}-p_{i,j-1}}{\Delta x^2} = -\frac{{v_x}^n_{i+1/2,j}-{v_x}^n_{i-1/2,j}+{v_y}^n_{i,j-1/2}-{v_y}^n_{i,j-1/2}}{\Delta x}$

Another way to achieve the Poisson problem

- What we want (the pressure equation):
  - $\frac{\partial v}{\partial t} = -\frac{1}{\rho}\nabla p$ s.t. $\nabla \cdot v = 0$
- Discretize the pressure equation in time:
  - $v^{n+1} - v^n = -\frac{\Delta t}{\rho}\nabla p$ s.t. $\nabla \cdot v^{n+1} = 0$
- Apply divergence operator ($\nabla \cdot$) on both sides:
  - $-\nabla \cdot v^n = -\frac{\Delta t}{\rho}\nabla \cdot \nabla p$



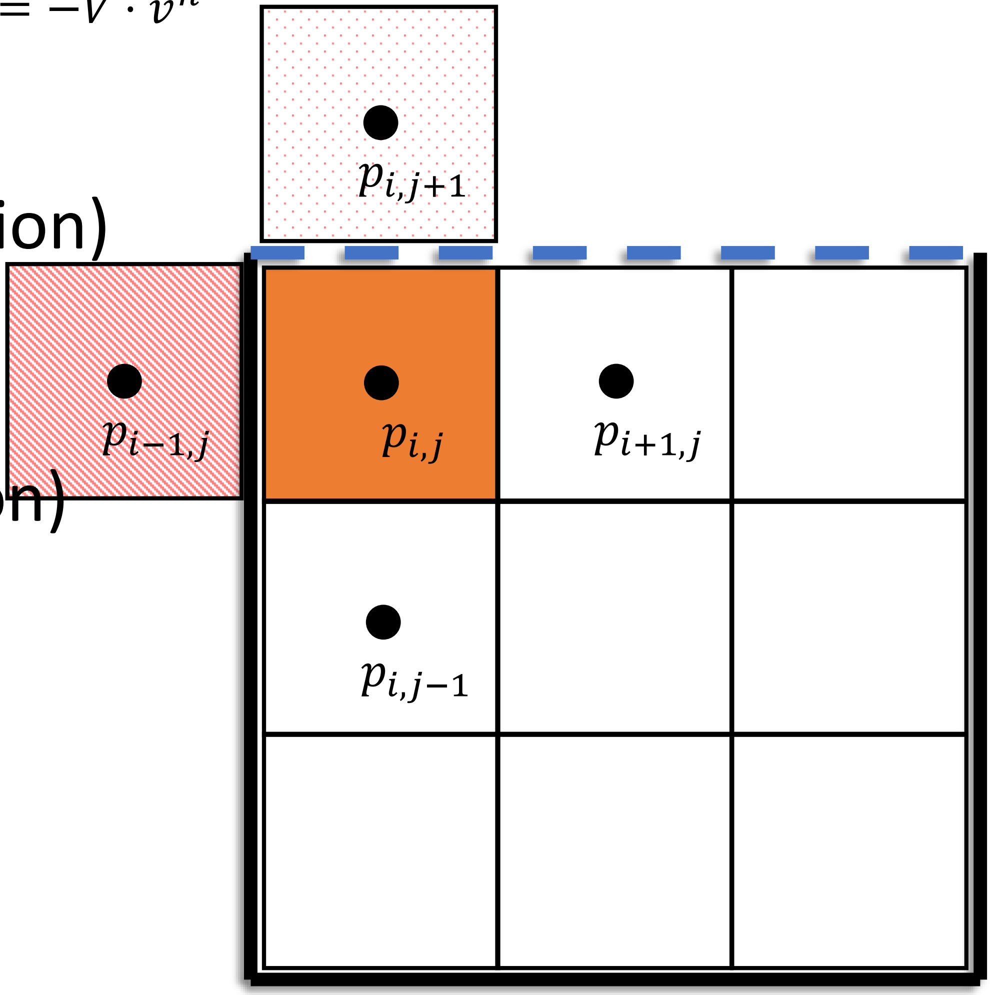
Pressure solve $-\frac{\Delta t}{\rho}\nabla \cdot \nabla p = -\nabla \cdot v^n$

- For every grid, there is one unknown $p_{i,j}$
- For every grid, there will be one equation:
  - $\frac{\Delta t}{\rho}\frac{4p_{i,j}-p_{i+1,j}-p_{i-1,j}-p_{i,j+1}-p_{i,j-1}}{\Delta x^2} = -\frac{{v_x}^n_{i+1/2,j}-{v_x}^n_{i-1/2,j}+{v_y}^n_{i,j-1/2}-{v_y}^n_{i,j-1/2}}{\Delta x}$

- This requires us only a linear solve $Ap = -d$
- Once we have all the pressure values…
  - We can solve for the velocity update:
    - $v^{n+1}_{x_{i-1/2,j}} = v^n_{x_{i-1/2,j}} - \frac{\Delta t}{\rho}\frac{p_{i,j}-p_{i-1,j}}{\Delta x}$

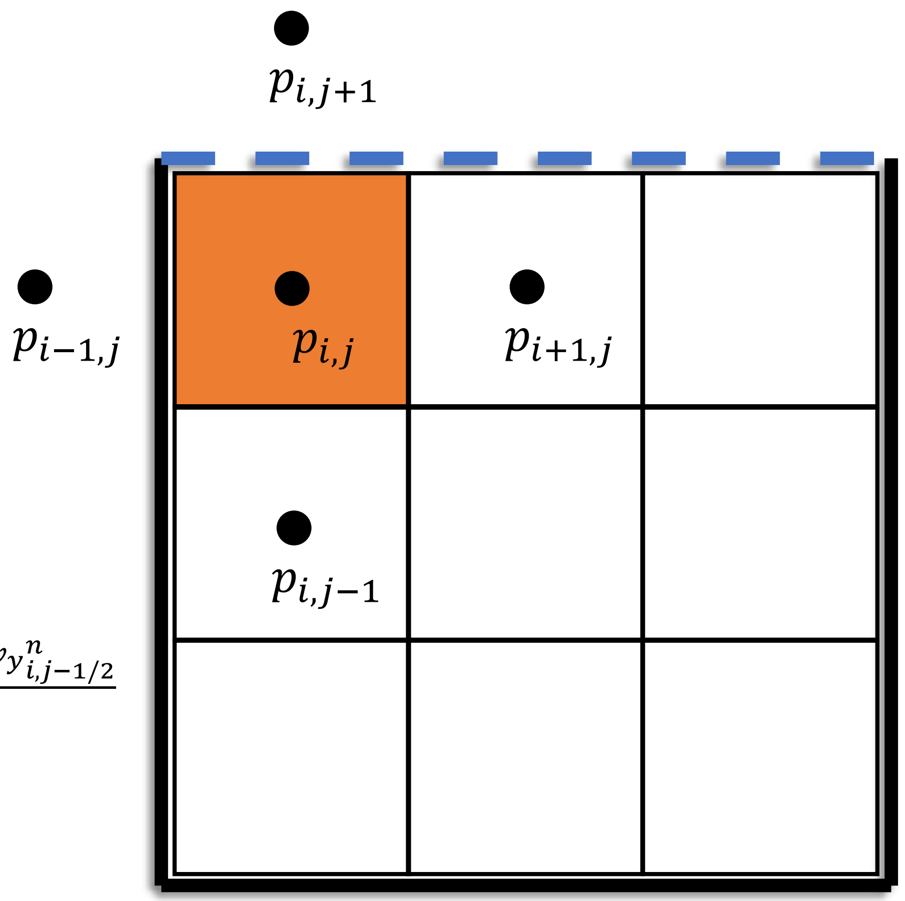
Boundary conditions $-\frac{\Delta t}{\rho}\nabla \cdot \nabla p = -\nabla \cdot v^n$

- Free surface (Dirichlet boundary condition)
  - $p = 0$ for void grids
- Solid wall (Neumann boundary condition)
  - $v^{n+1} \cdot n = v^{solid} \cdot n$
    - or $v_x^{n+1} = v_x^{solid}$, $v_y^{n+1} = v_y^{solid}$
  - For solid grids:
    - $v_x^{solid} = v_{x_{i-1/2,j}}^{n+1} = v_{x_{i-1/2,j}}^{n} - \frac{\Delta t}{\rho}\frac{p_{i,j}-p_{i-1,j}}{\Delta x}$
    - $\Rightarrow p_{i-1,j} = p_{i,j} - \frac{\rho \Delta x}{\Delta t}\left(v_{x_{i-1/2,j}}^{n} - v_x^{solid}\right)$



The Poisson's equation with boundaries

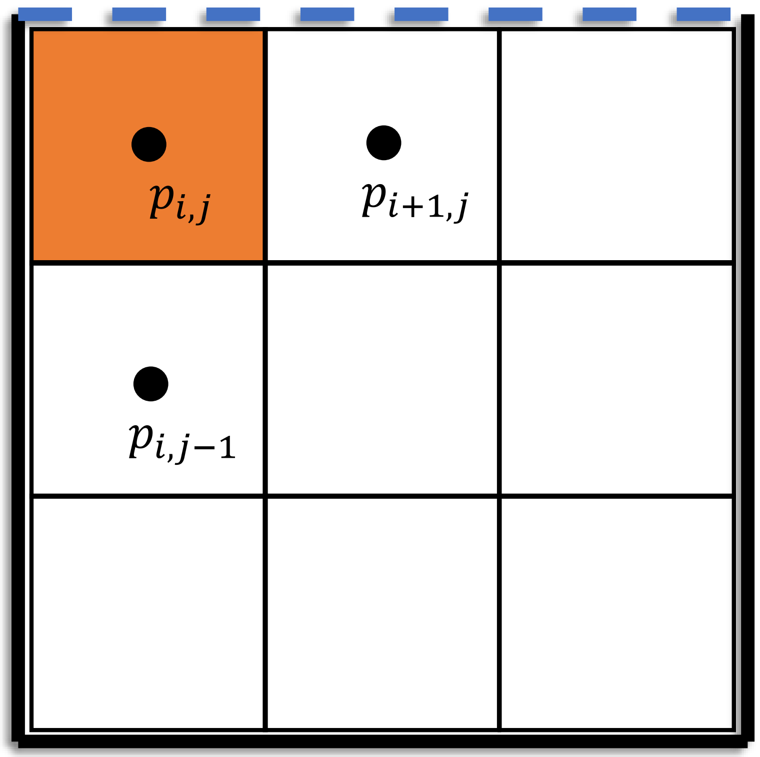
- Dirichlet boundary:
  - $p_{i,j+1} = 0$
- Neumann boundary:
  - $p_{i-1,j} = p_{i,j} - \frac{\rho\Delta x}{\Delta t}\left(v^n_{x_{i-1/2,j}} - v_x^{solid}\right)$
- The original Poisson's equation:
  - $\frac{\Delta t}{\rho}\frac{4p_{i,j}-p_{i+1,j}-p_{i-1,j}-p_{i,j+1}-p_{i,j-1}}{\Delta x^2} = -\frac{v{x}^n_{i+1/2,j}-v{x}^n_{i-1/2,j}+v{y}^n_{i,j-1/2}-v{y}^n_{i,j-1/2}}{\Delta x}$
- The Poisson's equation with boundaries:
  - $\frac{\Delta t}{\rho}\frac{4p_{i,j}-p_{i+1,j}-\left(p_{i,j}-\frac{\rho\Delta x}{\Delta t}\left(v^n_{x_{i-1/2,j}}-v_x^{solid}\right)\right)-0-p_{i,j-1}}{\Delta x^2} = -\frac{v{x}^n_{i+1/2,j}-v{x}^n_{i-1/2,j}+v{y}^n_{i,j-1/2}-v{y}^n_{i,j-1/2}}{\Delta x}$
  - Or equivalently:
  - $\frac{\Delta t}{\rho}\frac{3p_{i,j}-p_{i+1,j}-p_{i,j-1}}{\Delta x^2} = -\frac{v{x}^n_{i+1/2,j}-v_x^{solid}+v{y}^n_{i,j-1/2}-v{y}^n_{i,j-1/2}}{\Delta x}$



The Poisson’s equation with boundaries

- $Ap = -d$

$$\mathbf{A} = \frac{\Delta t}{\rho \Delta x^2} \left[\begin{array}{ccccccccc}
+3 & -1 & & -1 & & & & & \\
-1 & +4 & -1 & & -1 & & & & \\
& -1 & +3 & & & -1 & & & \\
-1 & & & +3 & -1 & & -1 & & \\
& -1 & & -1 & +4 & -1 & & -1 & \\
& & -1 & & -1 & +3 & & & -1 \\
& & & -1 & & & +2 & -1 & \\
& & & & -1 & & -1 & +3 & -1 \\
& & & & & -1 & & -1 & +2
\end{array}\right]
\begin{array}{l}
\\
\text{Dirichlet} \\
\\
\\
\text{Interior} \\
\\
\text{Neumann} \\
\\
\\
\end{array}$$



Linear solvers $Ax = b$ [Lecture 09]

- Direct solvers:
    - Inversion: $x = \mathrm{A}^{-1}b$
    - Factorization: $\mathrm{A} = \begin{cases} \mathrm{LU} & , if\ \mathrm{A}\ is\ a\ square\ matrix \\ \mathrm{LDL^T} & , if\ \mathrm{A} = \mathrm{A^T} \\ \mathrm{LL^T} & , if\ \mathrm{A} = \mathrm{A^T}\ and\ \mathrm{A} \succ 0 \end{cases}$
- Iterative solvers:
    - Stationary iterative linear solvers: Jacobi / Gauss-Seidel / SOR / Multigrid
    - Krylov subspace methods: Conjugate Gradient (CG) / biCG / CR / MinRes / GMRes

Put things together

- Given q^n, where q can be velocity, density, temperature etc.
  - Step 1 Advection:
    - $q^{n+1} = advect(v^n, \Delta t, q^n)$
    - $\tilde{v} = advect(v^n, \Delta t, v^n)$
  - Step 2 Applying forces:
    - $\tilde{\tilde{v}} = \tilde{v} + \Delta t(g + \nu \nabla^2 \tilde{v})$
  - Step 3 Projection:
    - $v^{n+1} = project(\Delta t, \tilde{\tilde{v}})$
  - Return v^{n+1}, q^{n+1}

$$\frac{Dv}{Dt} = g - \frac{1}{\rho}\nabla p + \nu \nabla^2 v$$

$$\nabla \cdot v = 0$$

Remark

Remark

- N-S equations and their time integration
  - Operator splitting
- From the Lagrangian view to the Eulerian view
  - Spatial derivatives using finite difference
  - MAC grid
- Advection
  - Material derivative
  - Quantity advection
- Projection
  - Poisson's equation
  - Boundary conditions

$$\frac{Dv}{Dt} = g - \frac{1}{\rho}\nabla p + \nu\nabla^2 v \qquad \nabla \cdot v = 0$$

Operator splitting:

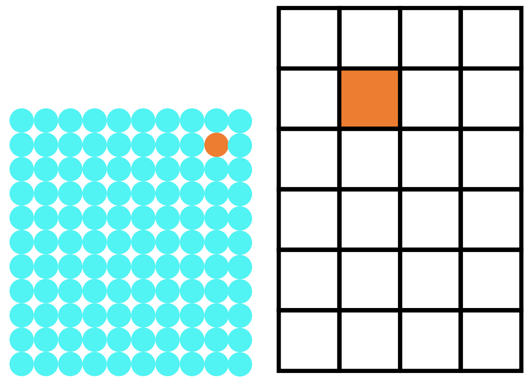
Advection: $\frac{Dq}{Dt} = 0$

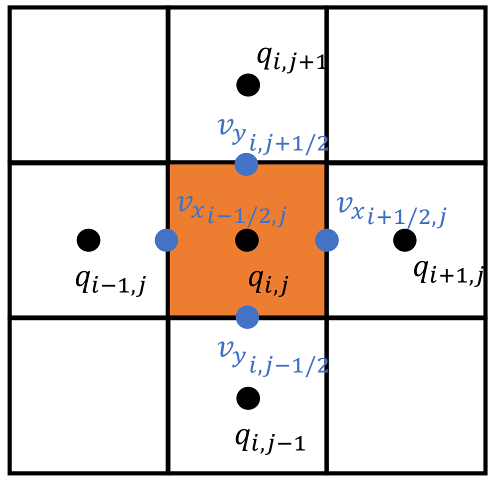
forcing: $\frac{\partial v}{\partial t} = g + \nu\nabla^2 v$

Projection: $\frac{\partial v}{\partial t} = -\frac{1}{\rho}\nabla p \; s.t. \nabla \cdot v = 0$

Remark

- N-S equations and their time integration
  - Operator splitting
- From the Lagrangian view to the Eulerian view
  - Spatial derivatives using finite difference
  - MAC grid
- Advection
  - Material derivative
  - Quantity advection
- Projection
  - Poisson’s equation
  - Boundary conditions

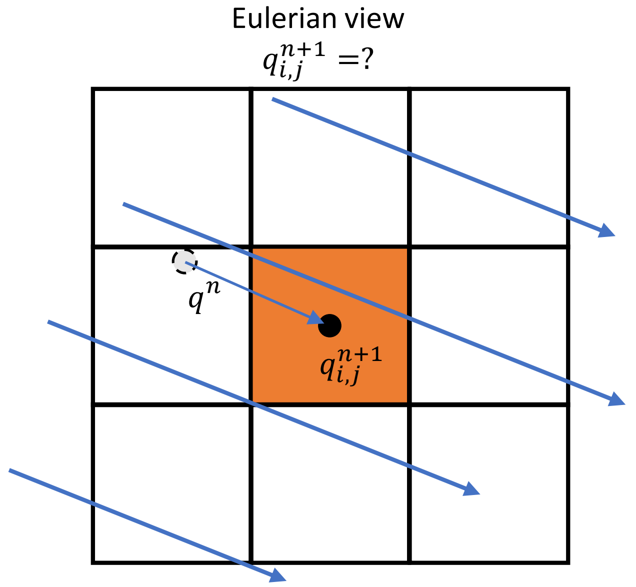


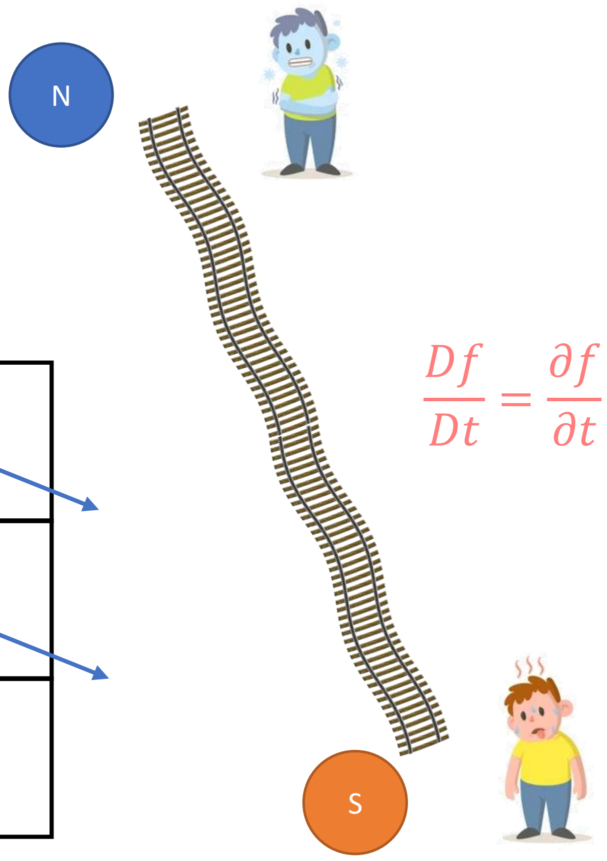


Remark

- N-S equations and their time integration
  - Operator splitting
- From the Lagrangian view to the Eulerian view
  - Spatial derivatives using finite difference
  - MAC grid
- **Advection**
  - Material derivative
  - Quantity advection

$$\frac{Dq}{Dt} = 0$$





$$\frac{Df}{Dt} = \frac{\partial f}{\partial t} + v \cdot \nabla f$$

- Projection
  - Poisson's equation
  - Boundary conditions

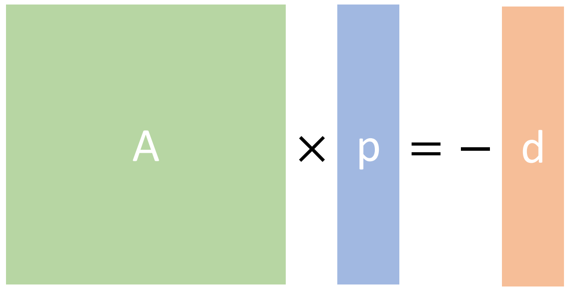
Remark

- N-S equations and their time integration
  - Operator splitting
- From the Lagrangian view to the Eulerian view
  - Spatial derivatives using finite difference
  - MAC grid
- Advection
  - Material derivative
  - Quantity advection
- Projection
  - Poisson's equation
  - Boundary conditions

$$\frac{\partial v}{\partial t} = -\frac{1}{\rho}\nabla p$$

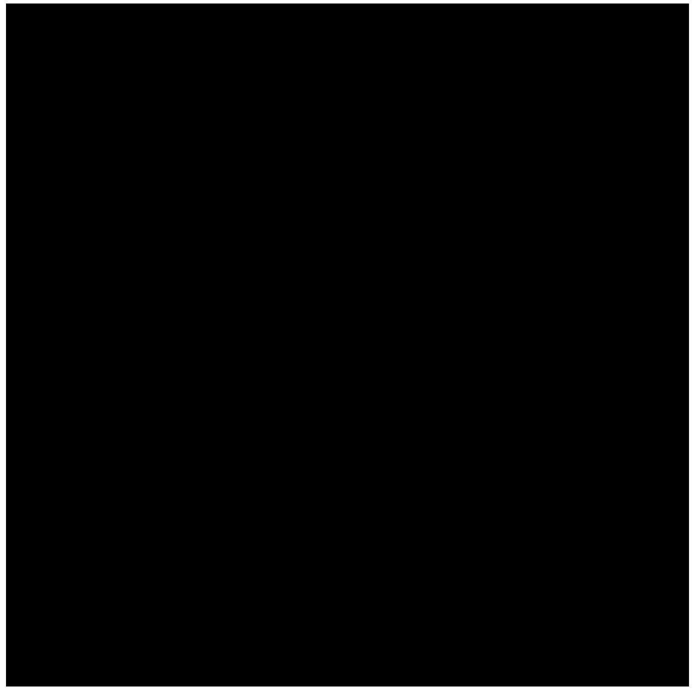
$$\nabla \cdot v = 0$$

$$-\nabla \cdot v^n = -\frac{\Delta t}{\rho}\nabla \cdot \nabla p$$



Remark

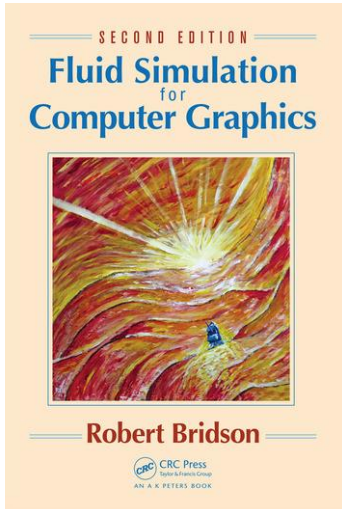
- N-S equations and their time integration
  - Operator splitting
- From the Lagrangian view to the Eulerian view
  - Spatial derivatives using finite difference
  - MAC grid
- Advection
  - Material derivative
  - Quantity advection
- Projection
  - Poisson’s equation
  - Boundary conditions

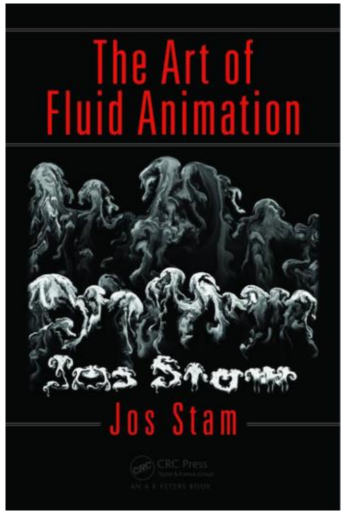


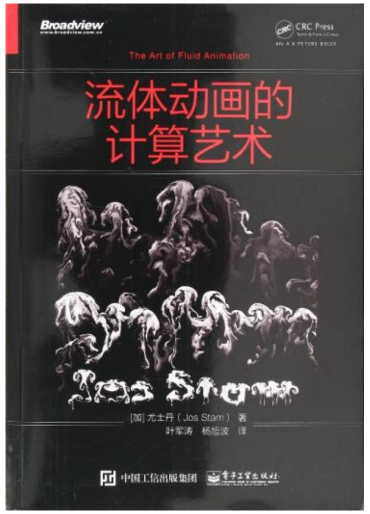
Further readings

- *Stable Fluids* [Stam 1999][Paper][Slides]
- *Fluid Simulation,* Chapter 1 ~ 6 [Bridson and Müller 2007][Course Notes]

- Books
  - *Fluid Simulation for Computer Graphics* [Bridson 2015]
  - 科普向：
    - *The Art of Fluid Animation* [Stam 2015]
    - 《流体动画的计算艺术》 -- 叶军涛、杨旭波译







Homework

Homework Today

- Check Taichi examples:
  - https://github.com/taichi-dev/taichi/blob/master/python/taichi/examples/simulation/stable_fluid.py
- Try:
  - free surface (Dirichlet boundary condition)
  - buoyancy/vorticity confinement force for smoke [Chapter 5]
  - sharper interpolation schemes [Link]
  - MacCormack method [Link] / BFECC [Paper]
  - Conjugate gradient linear solvers

Candidate projects for your final

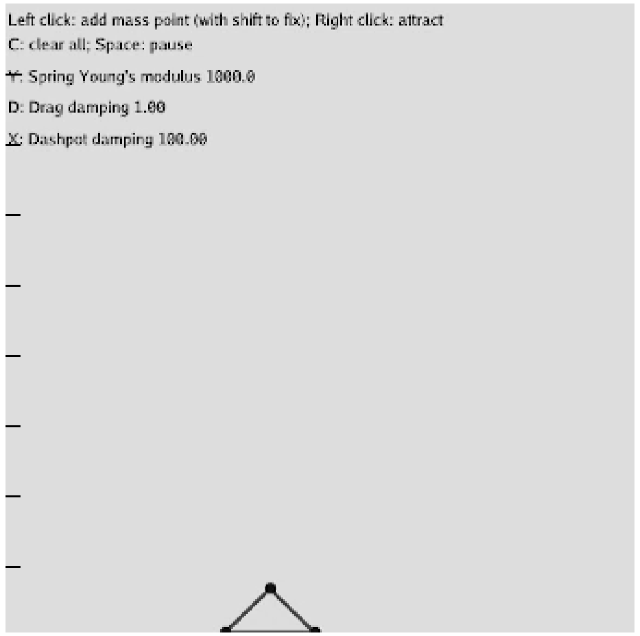
- Candidate topics:
  - Good performance! [Section 4.3]
    - Multigrid-preconditioned CG / Modified-incomplete-Cholesky-preconditioned CG
  - Complex boundaries [Section 4.5]
  - Eulerian water [Chapter 6]
  - Advection-reflection method [Paper]
  - Render your Eulerian fluid with your own renderer

- Both 2D and 3D projects are great!
  - As long as your pictures look great ☺

Final project

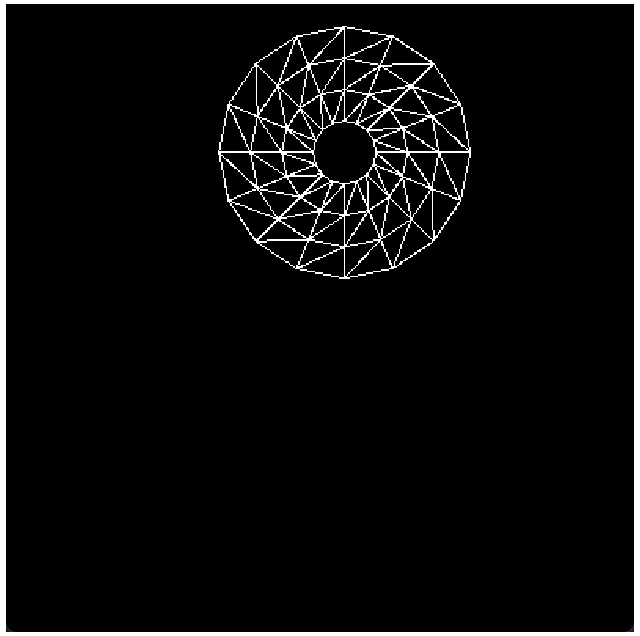
- 死线：2022年1月3日
- 要求：
  - 使用作业模板
  - 需要有设计文档，如果有参照代码也需要标明
- 题材：
  - 任何使用Taichi完成的内容（图形学更佳）
  - 可以参考每节图形课后给出的大作业选题灵感 [参考第07,09,10,11讲]
  - 鼓励实现任意图形学论文/图形学课程内容
  - 可以在小作业的基础上完成大作业 (Homework Promotion!)
- 形式：
  - 使用 GitHub/Gitee提交并邀请tgc01@taichi.graphics加入你的代码仓
  - 允许三人以下合作，记得管理多人合作的git commits
- 奖励：
  - 太极图形课第一季结业证书一份+神秘Taichi小礼物一份

Excellent homework assignments

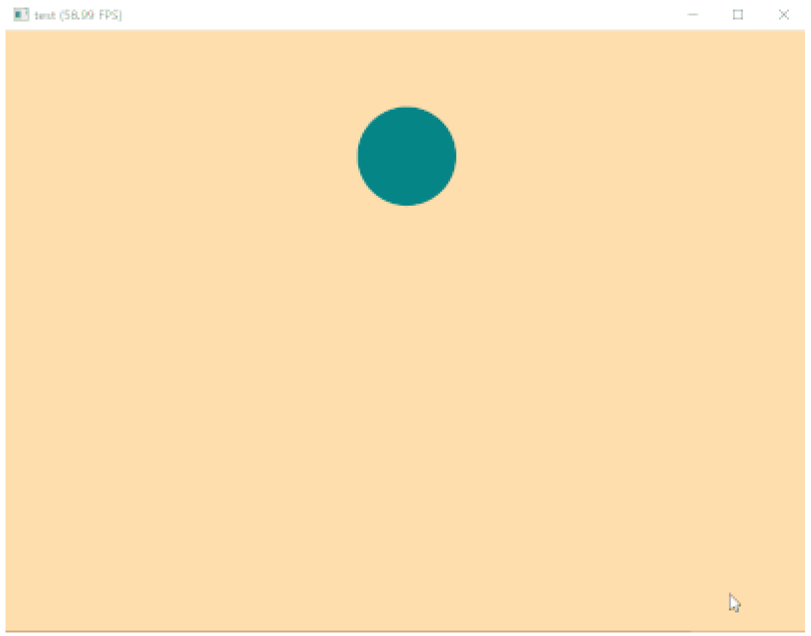


Left click: add mass point (with shift to fix); Right click: attract
C: clear all; Space: pause
Y: Spring Young's modulus 1000.0
D: Drag damping 1.00
X: Dashpot damping 100.00

[@Vineyo]

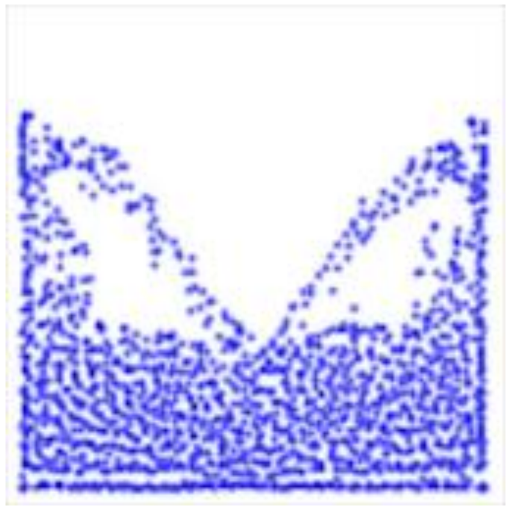


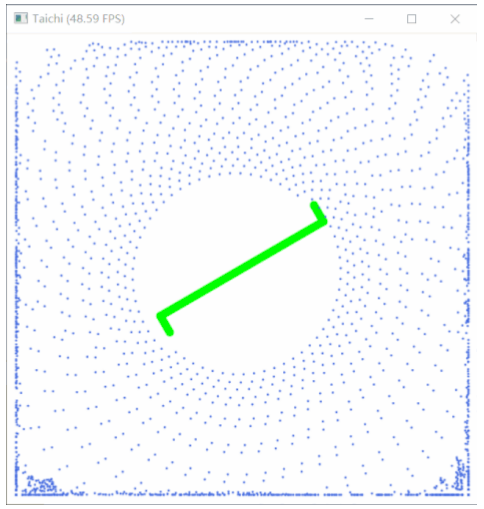
[@cflw]



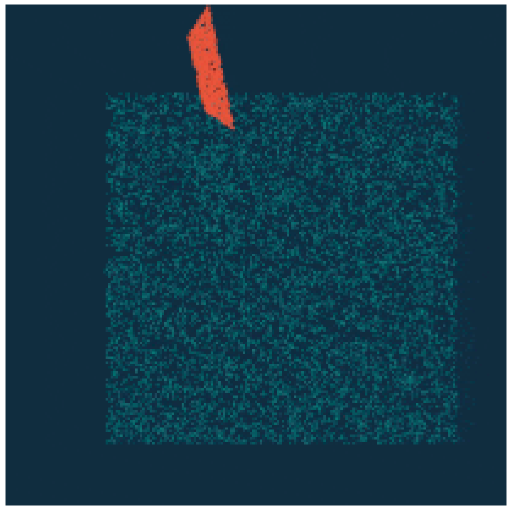
[@niushuqing123]

Excellent homework assignments


[@kphmd]



[@Pierce-qiang]

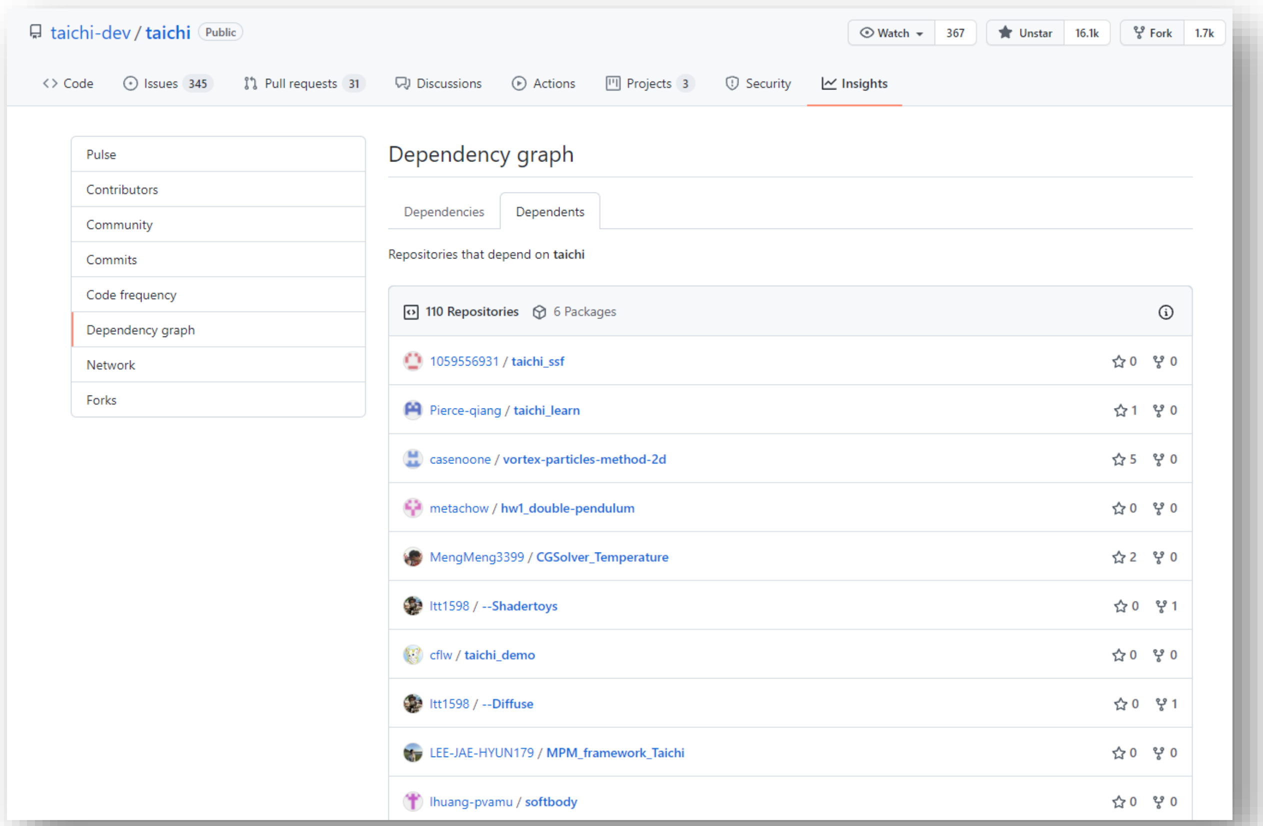

[@runck]

Excellent homework assignments

[@moxunbai]

Gifts for the gifted

- Use Template for your homework
- Next check in the next week! Dec. 14, 2021


taichi-dev / taichi Public

Watch 367 | Unstar 16.1k | Fork 1.7k

Code | Issues 345 | Pull requests 31 | Discussions | Actions | Projects 3 | Security | Insights

Pulse
Contributors
Community
Commits
Code frequency
Dependency graph
Network
Forks

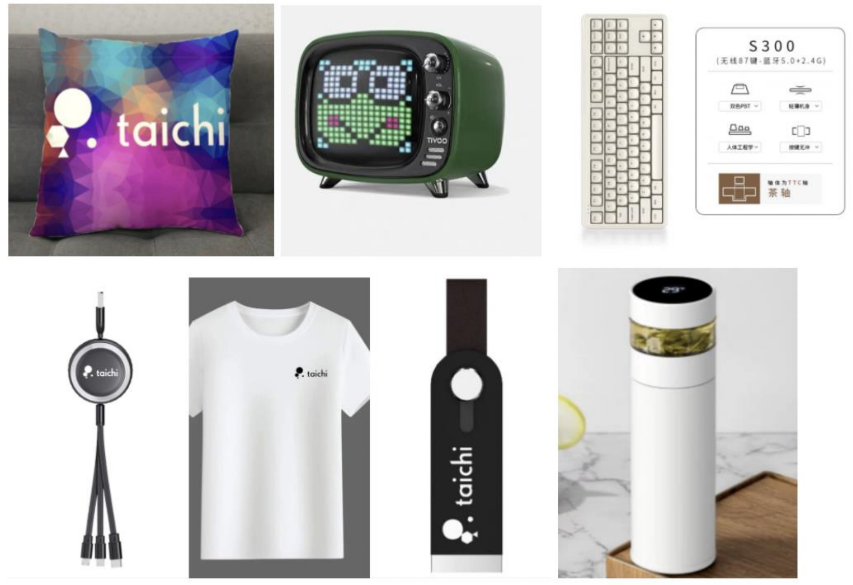
Dependency graph

Dependencies | Dependents

Repositories that depend on taichi

110 Repositories | 6 Packages

Repository	Stars	Forks
1059556931 / taichi_ssf	0	0
Pierce-qiang / taichi_learn	1	0
casenoone / vortex-particles-method-2d	5	0
metachow / hw1_double-pendulum	0	0
MengMeng3399 / CGSolver_Temperature	2	0
ltt1598 / --Shadertoys	0	1
cflw / taichi_demo	0	0
ltt1598 / --Diffuse	0	1
LEE-JAE-HYUN179 / MPM_framework_Taichi	0	0
lhuang-pvamu / softbody	0	0



Questions?

本次答疑：12/09 🡐作业分享也在这里

下次直播：12/14 🡐小作业抽奖以及第一季大结局

直播回放：Bilibili 搜索「太极图形」

主页&课件：https://github.com/taichiCourse01

主页&课件(backup)：https://docs.taichi.graphics/tgc01