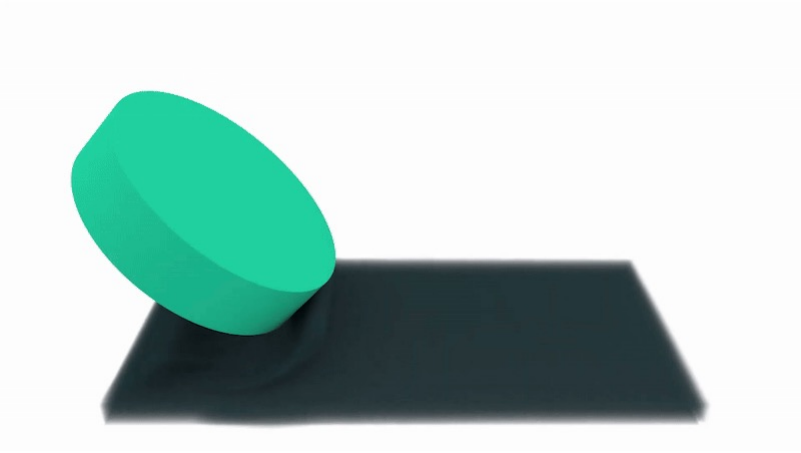
CSC417 Physics-Based Animation

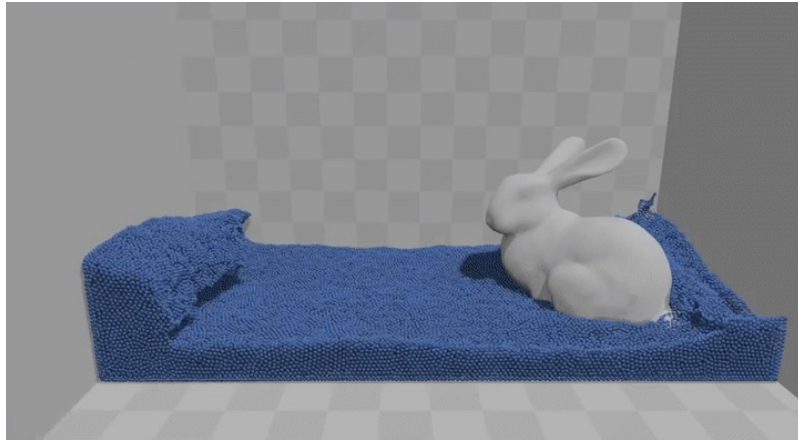
Fluids

Avatar: The Way of Water (2022)

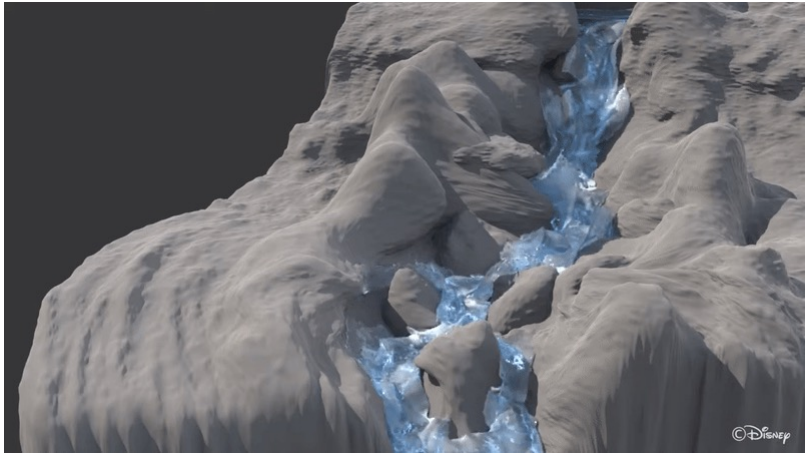
Larionov et al. 2017



Cui et al. 2018



Macklin & Müller 2013



Jiang et al. 2015

Resources

- Chorin and Marsden “A Mathematical Introduction to Fluid Mechanics”
- Bridson Siggraph 2007 Coursenotes
  - https://www.cs.ubc.ca/~rbridson/fluidsimulation/fluids_notes.pdf
- Ten Minute Physics
  - https://matthias-research.github.io/pages/tenMinutePhysics/17-fluidSim.pdf
  - https://matthias-research.github.io/pages/tenMinutePhysics/18-flip.pdf
- Sebastian Lague “I Tried Creating Smoke with a Fluid Simulation”
  - https://www.youtube.com/watch?v=Q78wvrQ9xsU

- Stam 1999 – “Stable fluids”
- Zhu and Bridson 2005 – “Animating sand as a fluid”
- Stomakhin et al. 2013 – “A material point method for snow simulation”

Deriving Fluids Equations

Pirates of the Caribbean: At World's End (2007)

Deriving Fluids

$$F = ma$$

Deriving Fluids

$$F = ma$$

$$a = \frac{D\boldsymbol{u}}{Dt}$$

$$\frac{du}{dt}$$

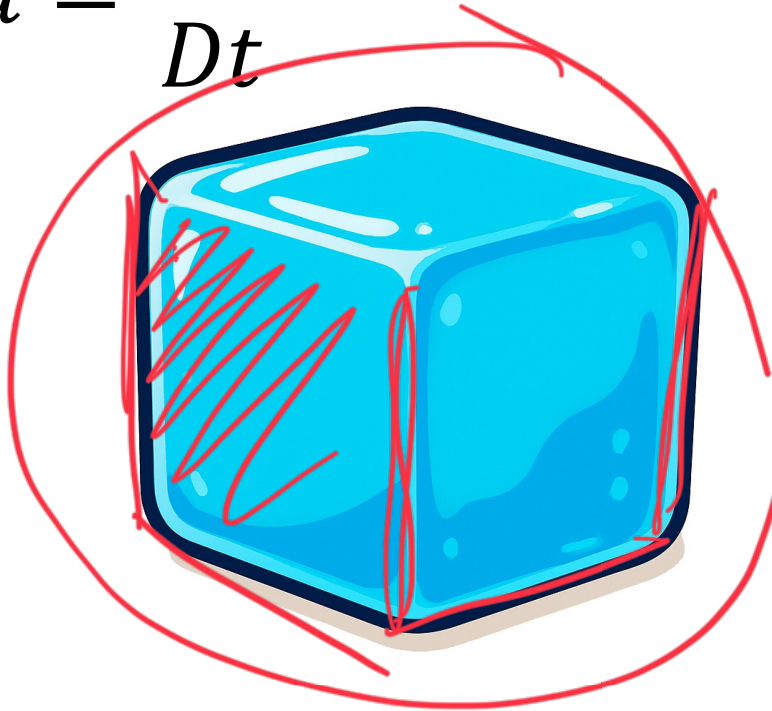
$$F = m\frac{D\boldsymbol{u}}{Dt}$$

Deriving Fluids

$$F = ma \qquad a = \frac{D\boldsymbol{u}}{Dt}$$

$$F = m\frac{D\boldsymbol{u}}{Dt}$$

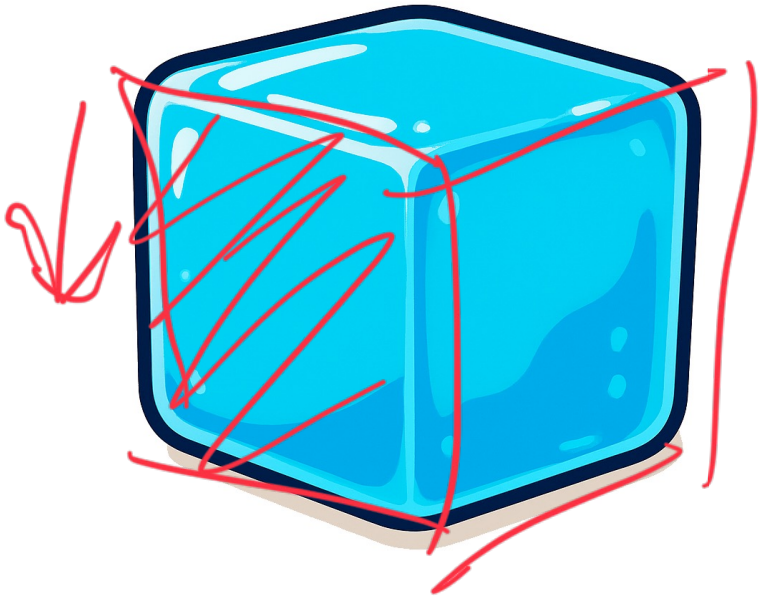
?



Forces

gravity $F_g = mg$

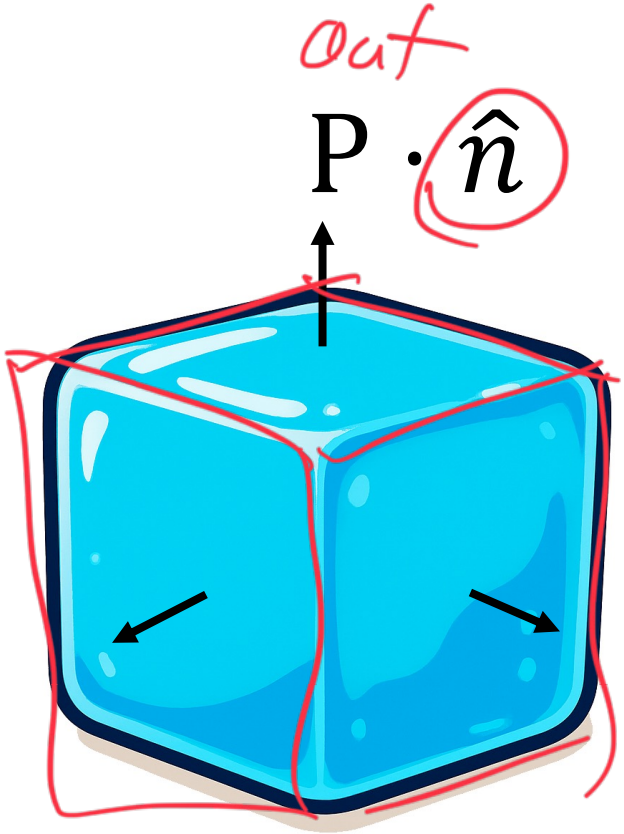
9.8



Forces

gravity $F_g = mg$

pressure $F_P = \sum(P \cdot \hat{\boldsymbol{n}})\mathrm{A}$



Forces

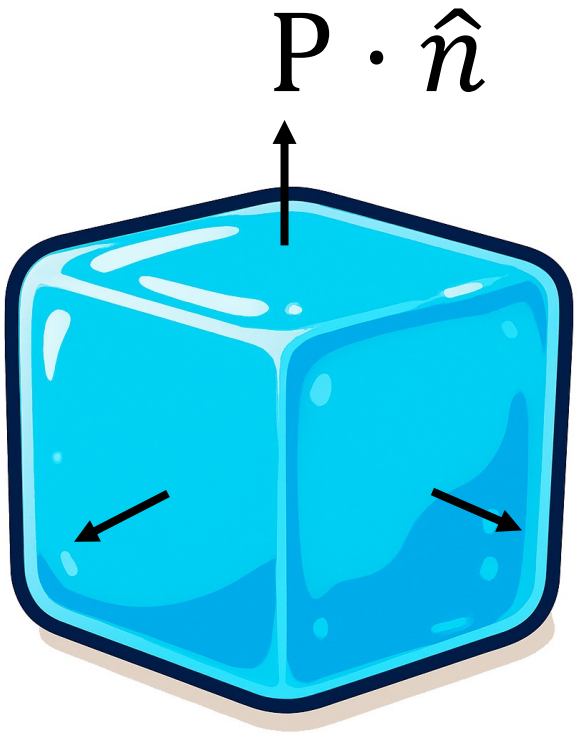
gravity $F_g = mg$

pressure $F_P = -\sum(P \cdot \hat{\boldsymbol{n}})\mathrm{A}$

$= -\int P \cdot \hat{\boldsymbol{n}}\, \mathrm{dA}$

divergence theorem

$= -\int \nabla P\, \mathrm{dV}$



Forces

gravity $F_g = mg$

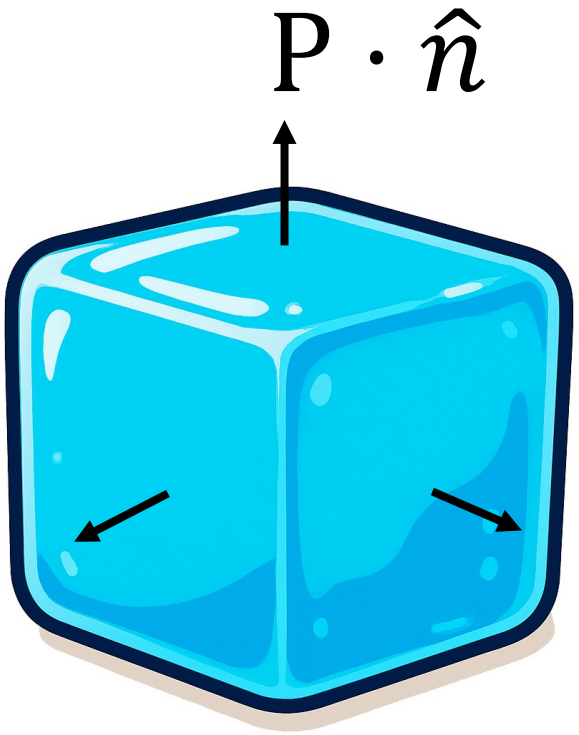
pressure $F_P = -\sum(P \cdot \hat{\boldsymbol{n}})\mathrm{A}$

$= -\int P \cdot \hat{\boldsymbol{n}}\, \mathrm{dA}$

divergence theorem

$= -\int \nabla P\, \mathrm{dV}$

$= -V\nabla P$



Deriving Fluids

F = ma

$$m\frac{D\boldsymbol{u}}{Dt} = F_g + F_P$$

$$m\frac{D\boldsymbol{u}}{Dt} = mg - V\nabla P$$

Deriving Fluids

mu

momentum

density

$$m\frac{D\boldsymbol{u}}{Dt} = F_g + F_P$$

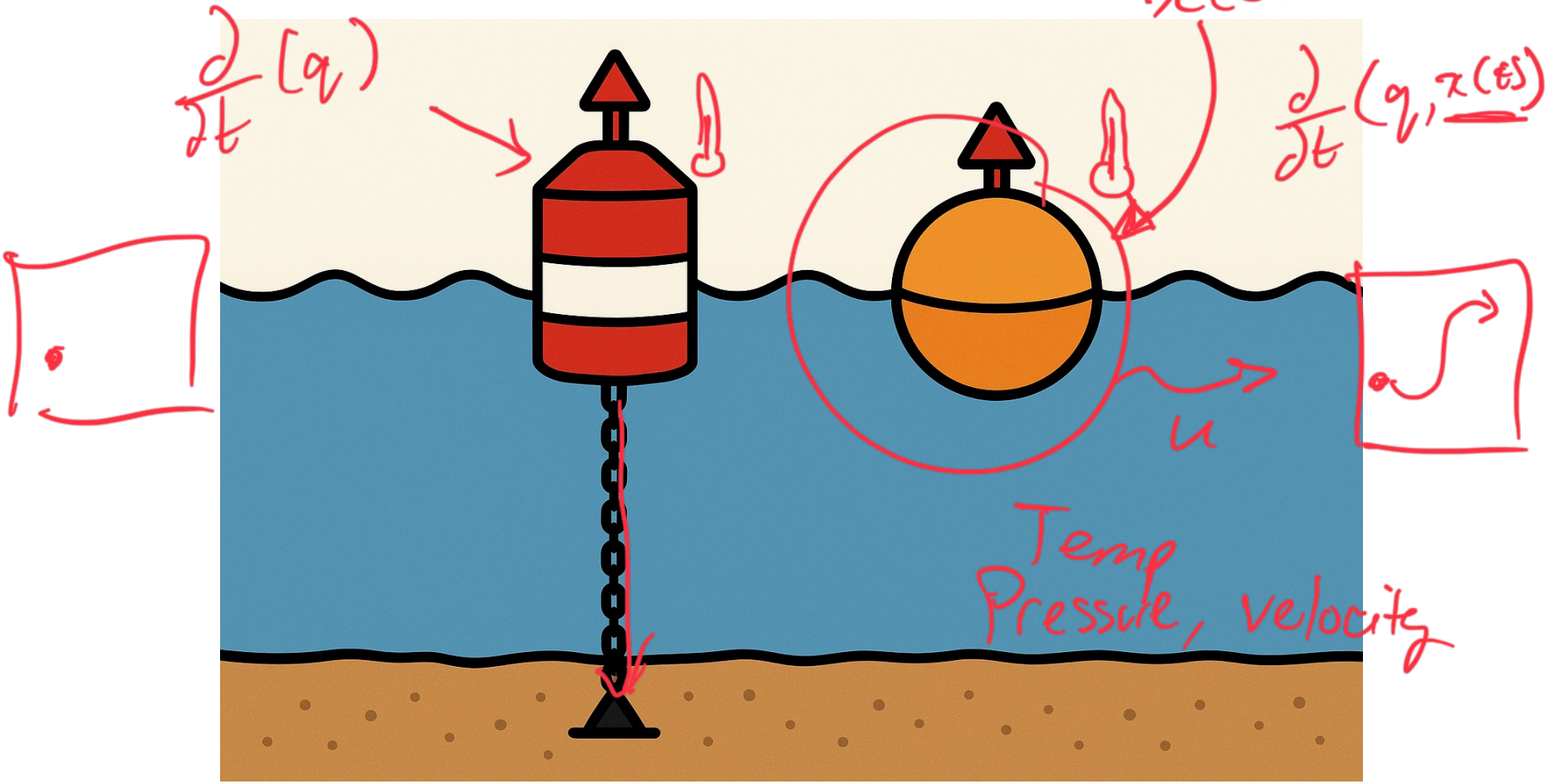
$$m\frac{D\boldsymbol{u}}{Dt} = mg - V\nabla P$$

$$\rho = \frac{m}{\cancel{\nabla P}\ V}$$

$$\boxed{\frac{D\boldsymbol{u}}{Dt} = g - \frac{1}{\rho}\nabla P}$$

"momentum equation"

Two Buoys


$\frac{\partial}{\partial t}(q)$
$x(t)$
$\frac{\partial}{\partial t}(q, x(t))$
u
Temp
Pressure, velocity

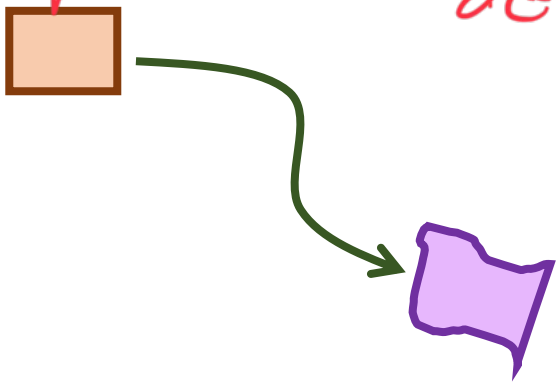
Funky Derivative

how temporary quantity q changes over time $\neq \frac{d}{dt} q(t)$

$$\frac{d}{dt} q(t, \boldsymbol{x}(t)) = \frac{dq}{dt} + \frac{dq}{d\boldsymbol{x}} \frac{d\boldsymbol{x}}{dt}$$

chain rule

q



Funky Derivative

spatial derivative

∇q "gradient"

$$\frac{d}{dt}q(t,\boldsymbol{x}(t)) = \frac{dq}{dt} + \frac{dq}{d\boldsymbol{x}}\frac{d\boldsymbol{x}}{dt}$$

chain rule

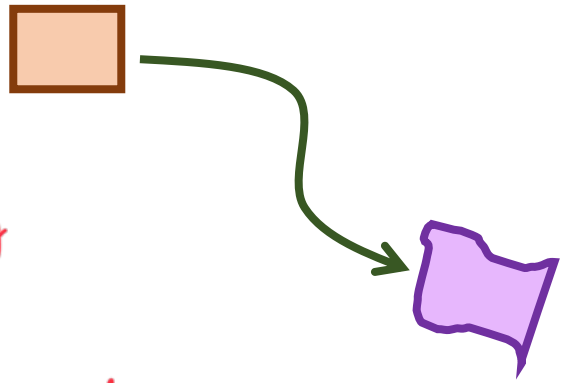
velocity

$$= \frac{dq}{dt} + \nabla q \cdot \boldsymbol{u}$$

"commutative"

$a \cdot b = b \cdot a$

$$= \frac{dq}{dt} + \underline{\boldsymbol{u}} \cdot \nabla \mathrm{q}$$

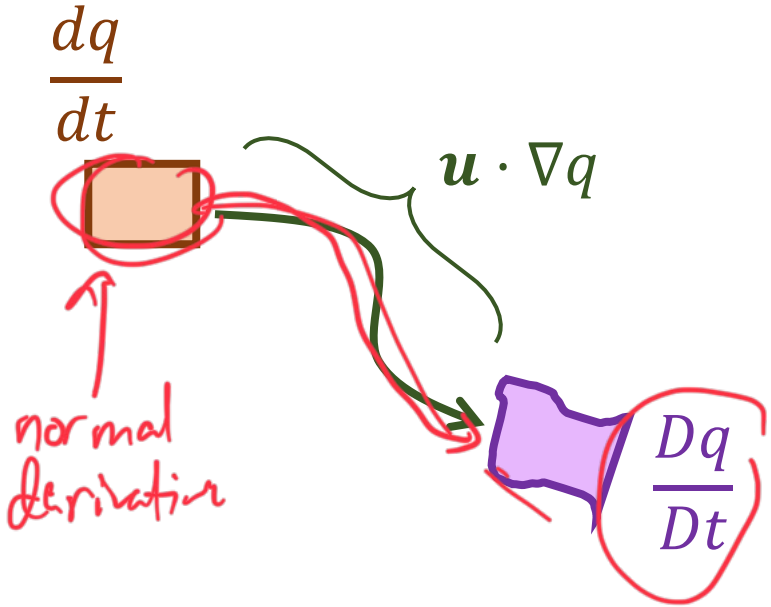


Funky Derivative

$$\frac{d}{dt}q(t,\boldsymbol{x}(t)) = \frac{dq}{dt} + \frac{dq}{d\boldsymbol{x}}\frac{d\boldsymbol{x}}{dt}$$

chain rule

$$= \frac{dq}{dt} + \nabla q \cdot \boldsymbol{u}$$



normal stationary derivative

$$= \frac{dq}{dt} + \boldsymbol{u} \cdot \nabla q$$

how it changed by moving

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

"material derivative"

Funky Derivative

$$\frac{d}{dt}q(t,\boldsymbol{x}(t)) = \frac{dq}{dt} + \frac{dq}{d\boldsymbol{x}}\frac{d\boldsymbol{x}}{dt}$$

chain rule

$$= \frac{dq}{dt} + \nabla q \cdot \boldsymbol{u}$$

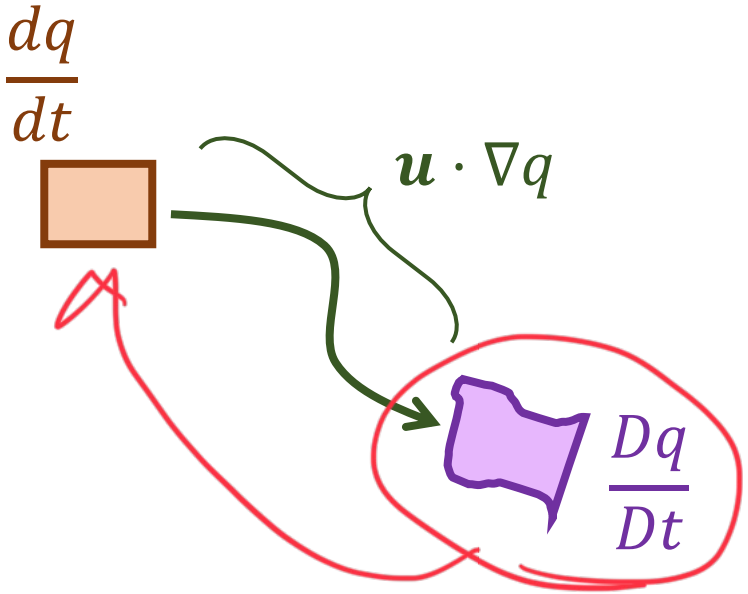
static

$$= \frac{dq}{dt} + \boldsymbol{u} \cdot \nabla \mathrm{q}$$

motion

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

"material derivative"



Funky Derivative

$$\frac{d}{dt}q(t,\boldsymbol{x}(t)) = \frac{dq}{dt} + \frac{dq}{d\boldsymbol{x}}\frac{d\boldsymbol{x}}{dt}$$

chain rule

$$= \frac{dq}{dt} + \nabla q \cdot \boldsymbol{u}$$

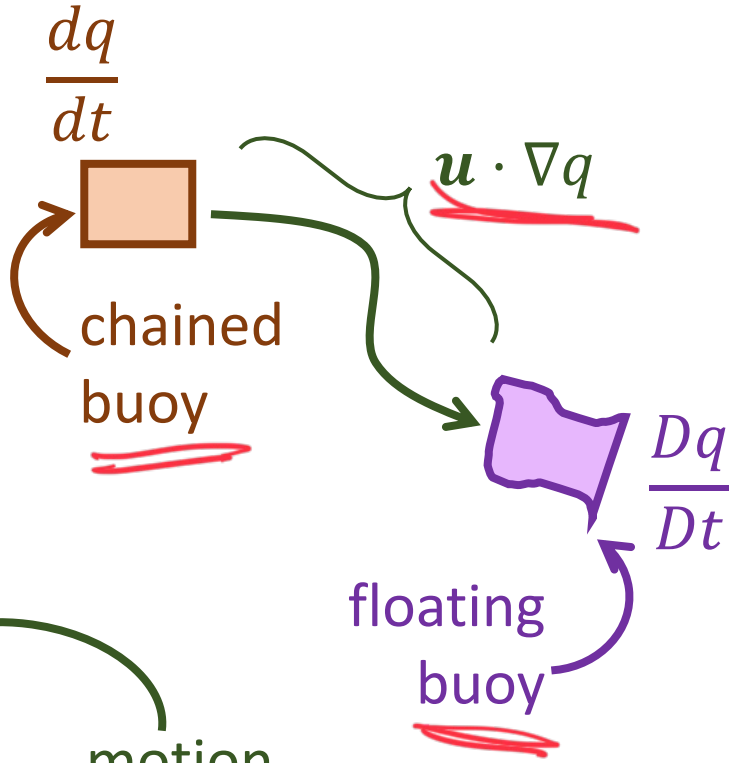
static

$$= \frac{dq}{dt} + \boldsymbol{u} \cdot \nabla \mathrm{q}$$

motion

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

"material derivative"



Material Derivative In 3D

$$\frac{Dq}{Dt} = \frac{dq}{dt} + u\frac{dq}{dx} + v\frac{dq}{dy} + w\frac{dq}{dz}$$

$$\frac{\partial q}{\partial t} + u \cdot \nabla q$$

scalar

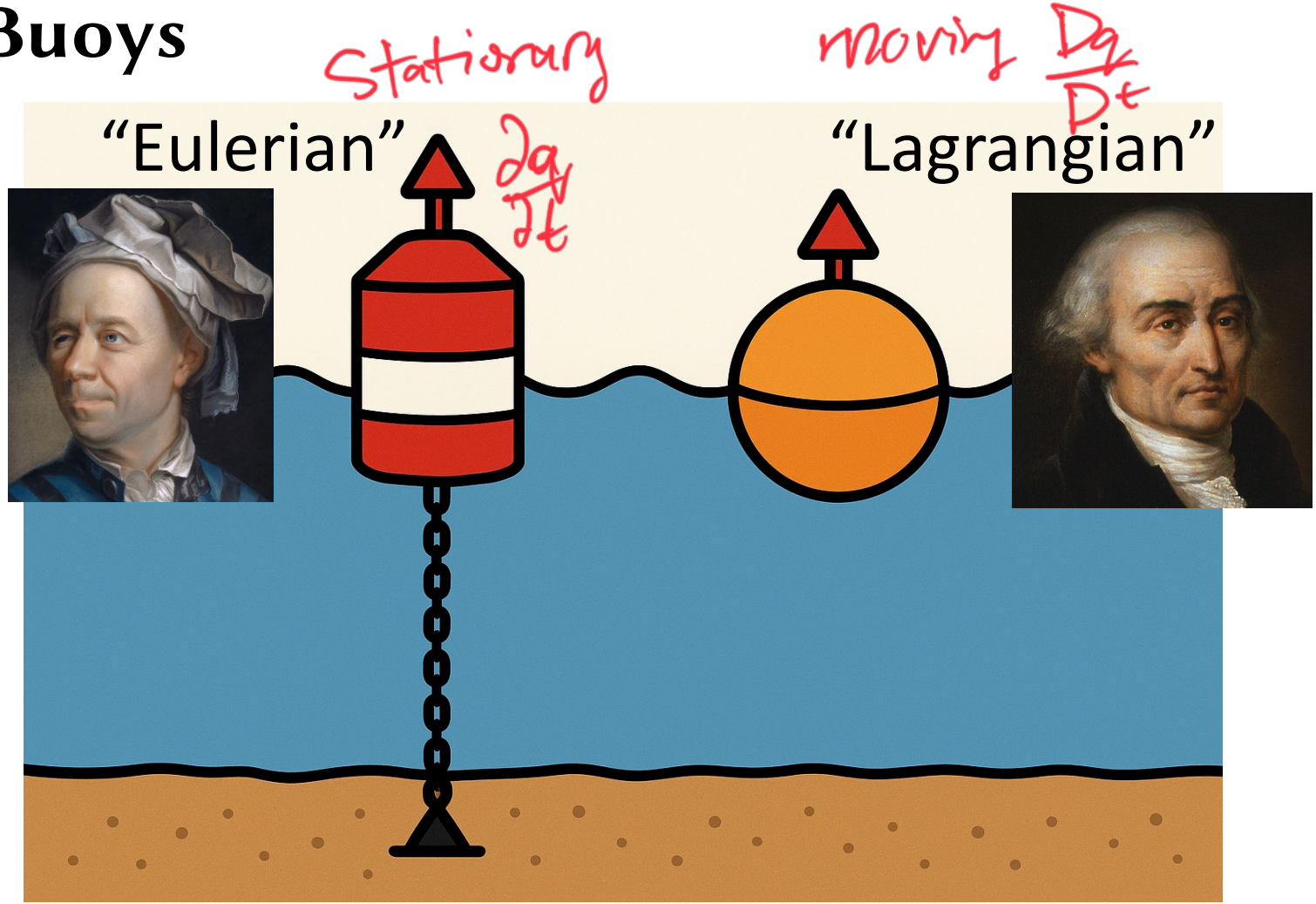
$$\frac{D\bar{q}}{Dt} = \begin{bmatrix} \frac{Dq_x}{Dt} \\ \frac{Dq_y}{Dt} \\ \frac{Dq_z}{Dt} \end{bmatrix}$$

Material Derivative In 3D

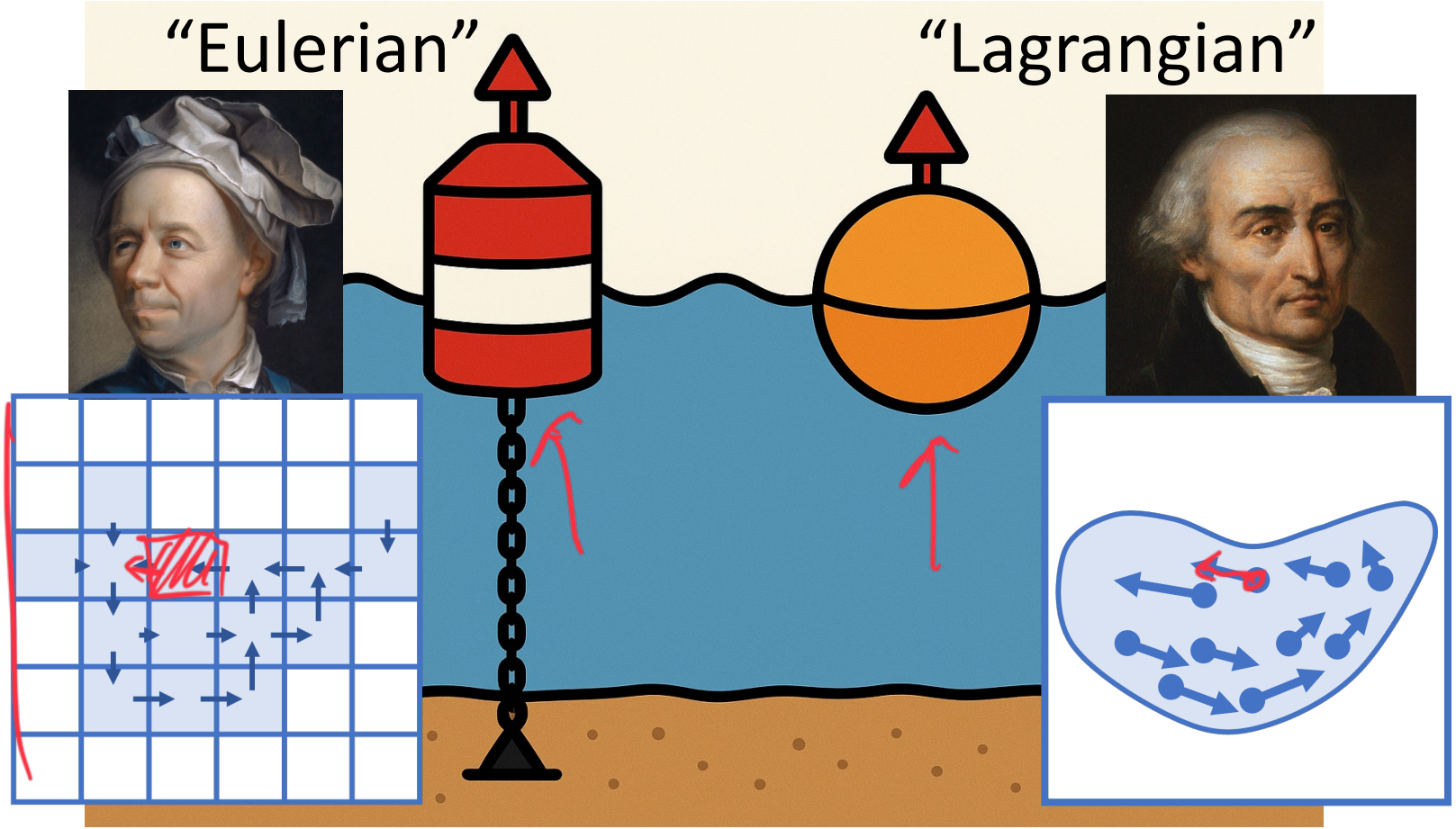
$$\frac{Dq}{Dt} = \frac{dq}{dt} + u\frac{dq}{dx} + v\frac{dq}{dy} + w\frac{dq}{dz}$$

$$\frac{D\boldsymbol{u}}{Dt} = \begin{bmatrix} \frac{Du}{Dt} \\ \frac{Dv}{Dt} \\ \frac{Dw}{Dt} \end{bmatrix} = \begin{bmatrix} \frac{du}{dt} + u\frac{du}{dx} + v\frac{du}{dy} + w\frac{du}{dz} \\ \frac{dv}{dt} + u\frac{dv}{dx} + v\frac{dv}{dy} + w\frac{dv}{dz} \\ \frac{dw}{dt} + u\frac{dw}{dt} + v\frac{dw}{dy} + w\frac{dw}{dz} \end{bmatrix}$$

Two Buoys


Stationary
moving $\frac{Dq}{Dt}$
"Eulerian"
$\frac{\partial q}{\partial t}$
"Lagrangian"

Two Buoys

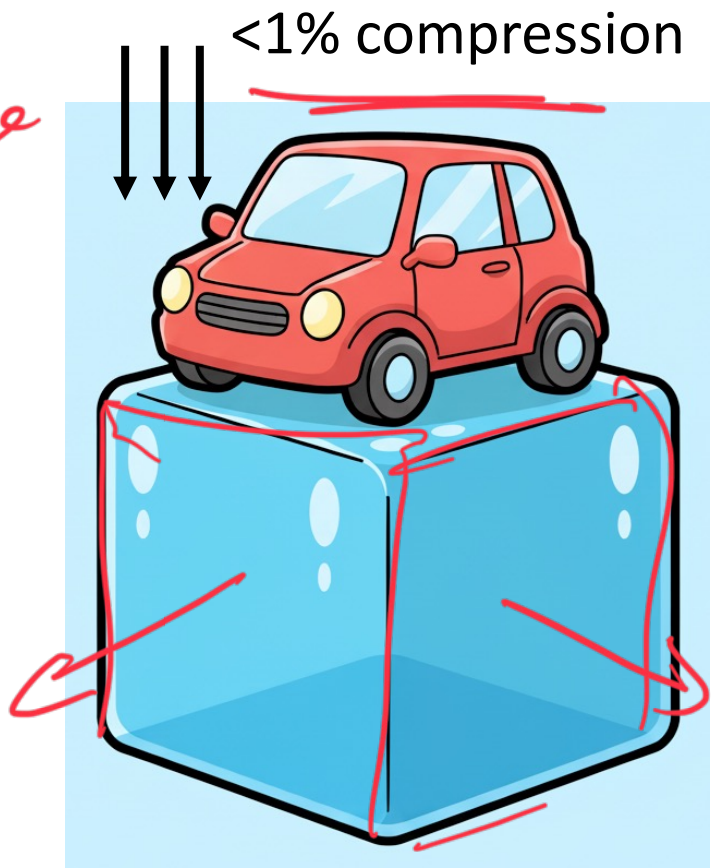


Incompressibility

Incompressibility

$$\frac{d}{dt}V(\Omega) = 0$$

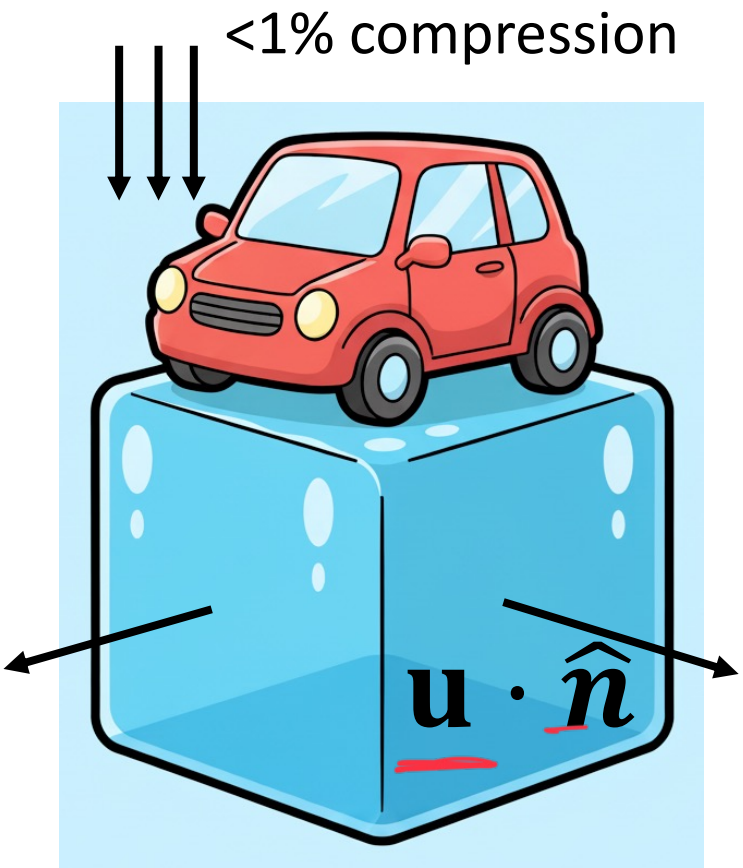
volume doesn't change



Incompressibility

$$\frac{d}{dt}V(\Omega) = 0$$

$$\iint_{\partial\Omega} \boldsymbol{u} \cdot \hat{\boldsymbol{n}}\, dA = 0$$



Incompressibility

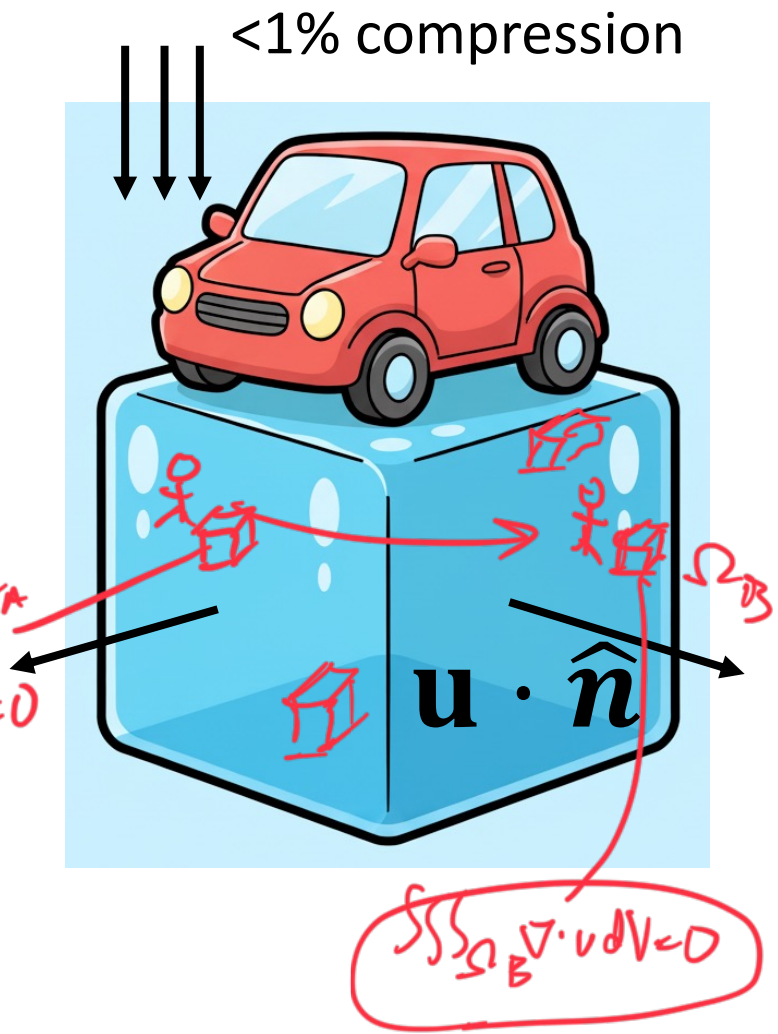
$$\frac{d}{dt}V(\Omega) = 0$$

$$\iint_{\partial\Omega} \boldsymbol{u} \cdot \hat{\boldsymbol{n}}\, dA = 0$$

divergence theorem

$$\iiint_{\Omega} \nabla \cdot \boldsymbol{u}\, dV = 0$$

$$\iiint_{\Omega_A} \nabla \cdot u\, dV = 0$$



Incompressibility

$$\frac{d}{dt}V(\Omega) = 0$$

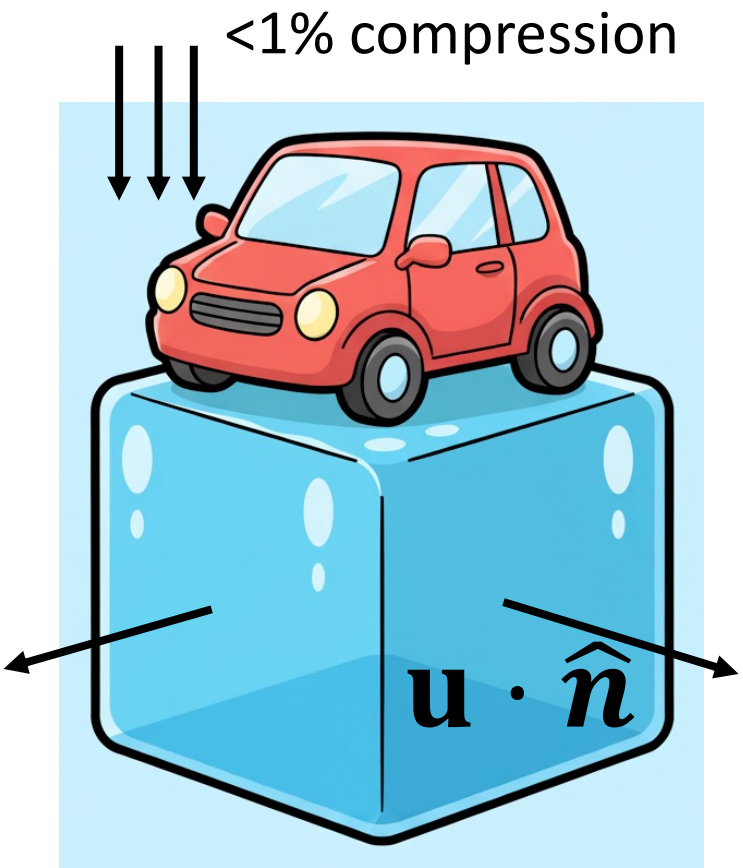
$$\iint_{\partial\Omega} \boldsymbol{u} \cdot \hat{\boldsymbol{n}}\, dA = 0$$

divergence theorem

$$\boxed{\iiint_{\Omega} \nabla \cdot \boldsymbol{u}\, dV = 0}$$

true for any Ω

$$\boxed{\nabla \cdot \boldsymbol{u} = 0}$$



"divergence-free condition"

The Fluid Equations

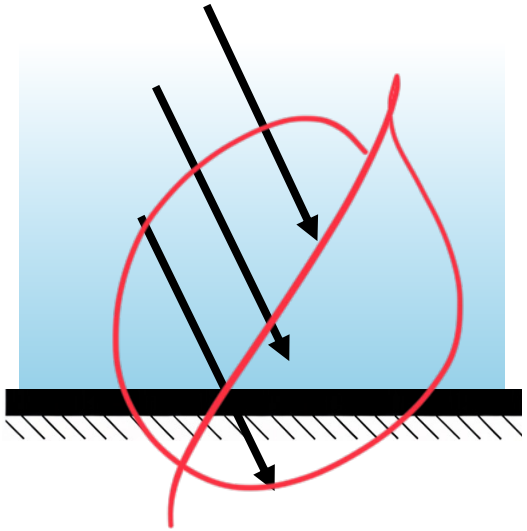
$$\frac{D\boldsymbol{u}}{Dt} = g - \frac{1}{\rho}\nabla P$$

1: momentum equation

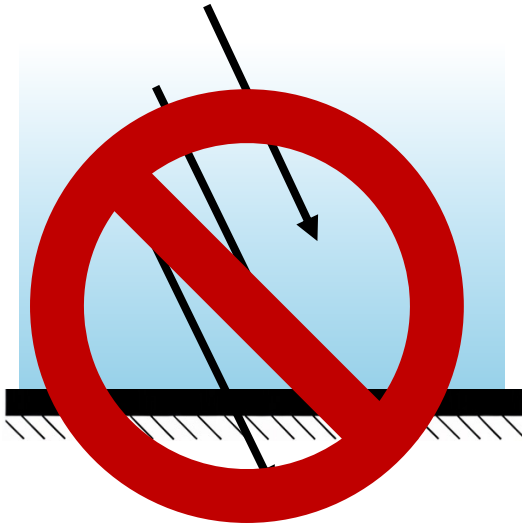
$$\nabla \cdot \boldsymbol{u} = 0$$

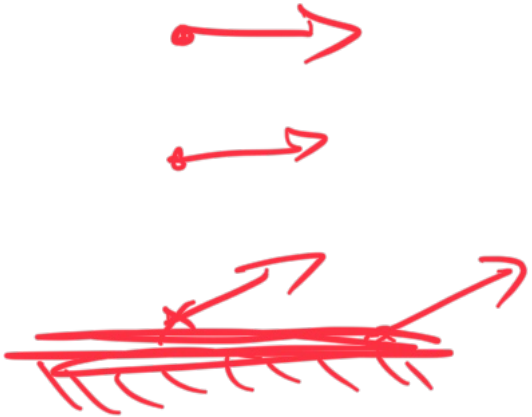
2: divergence-free condition

Boundary Conditions - Solids



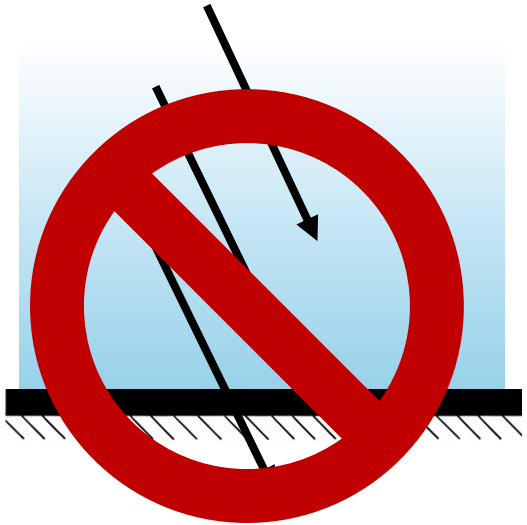
Boundary Conditions - Solids

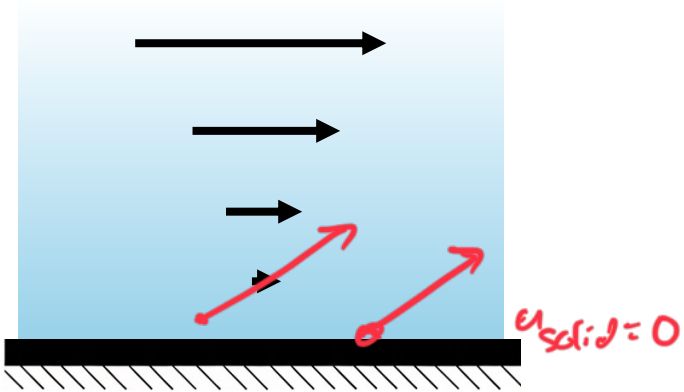




Boundary Conditions - Solids

"viscosity"

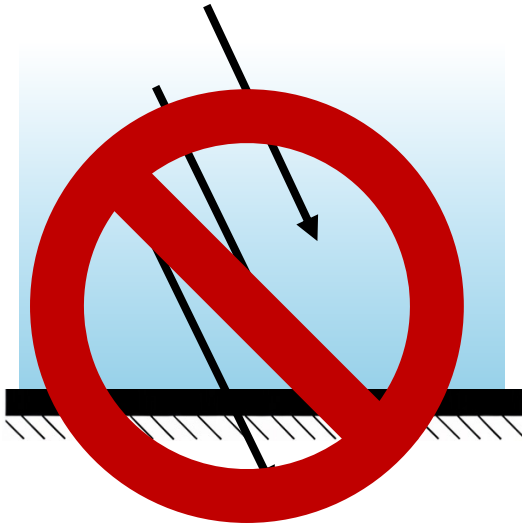


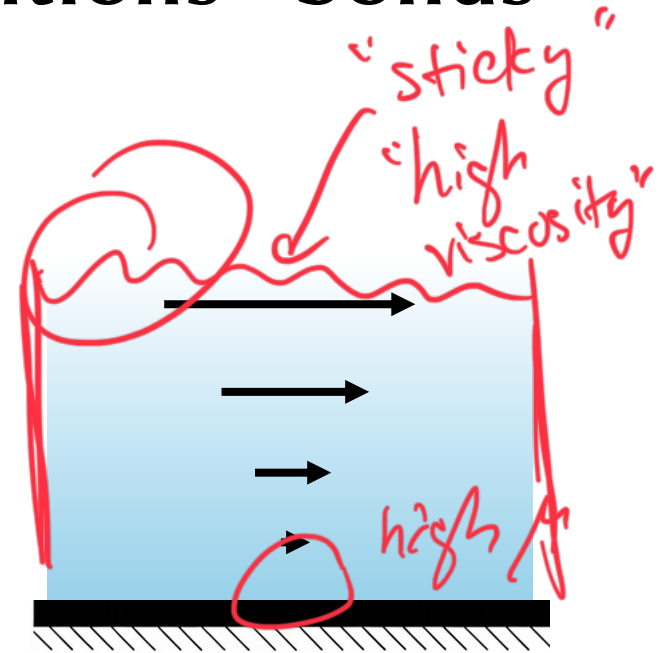


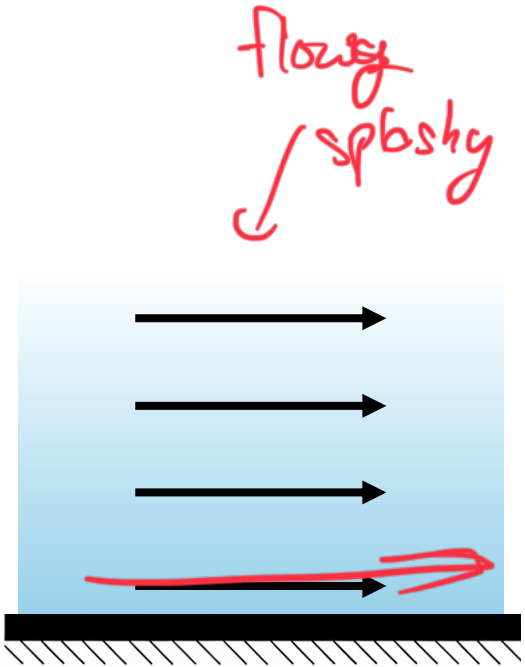
$$\boldsymbol{u} = \boldsymbol{u}_{solid}$$

"no slip"

Boundary Conditions - Solids







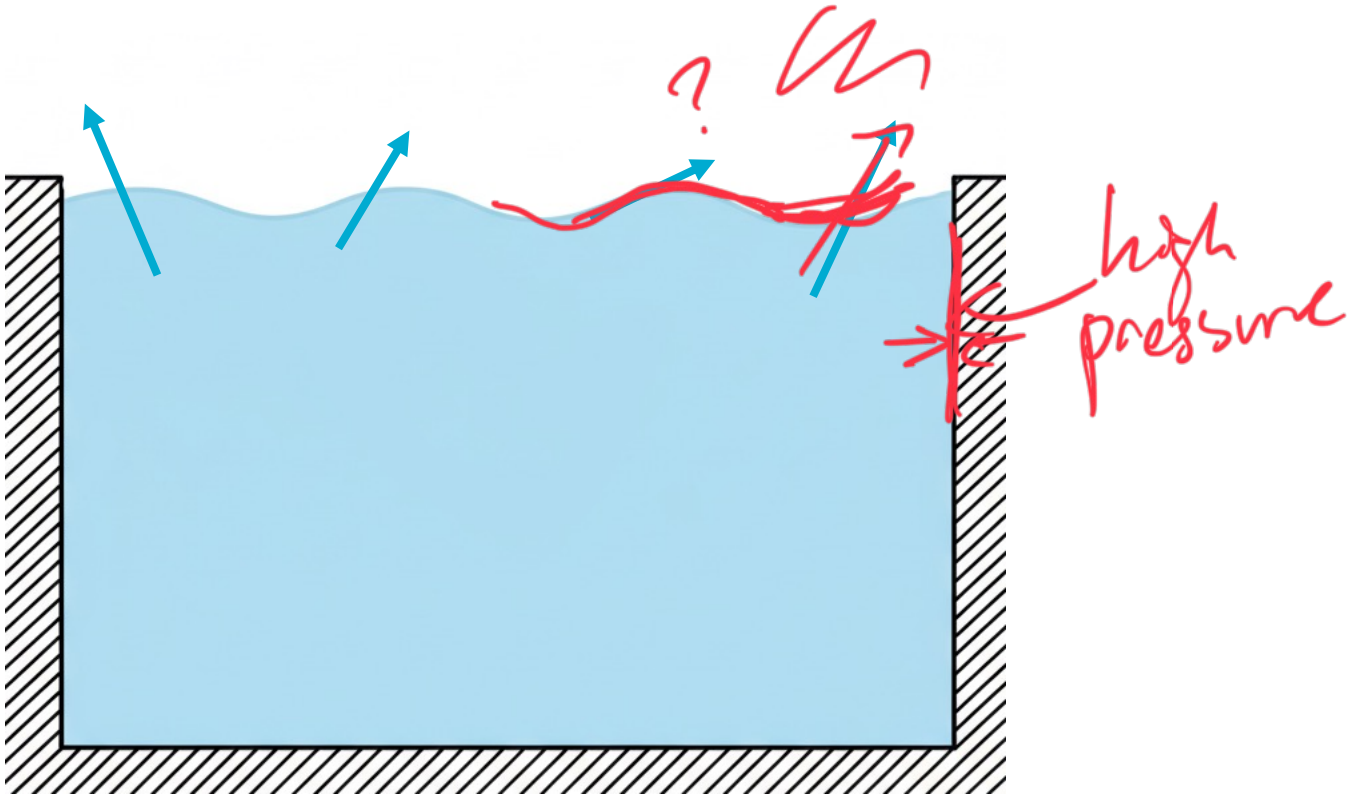
$$\boldsymbol{u} = \boldsymbol{u}_{solid}$$

"no slip"

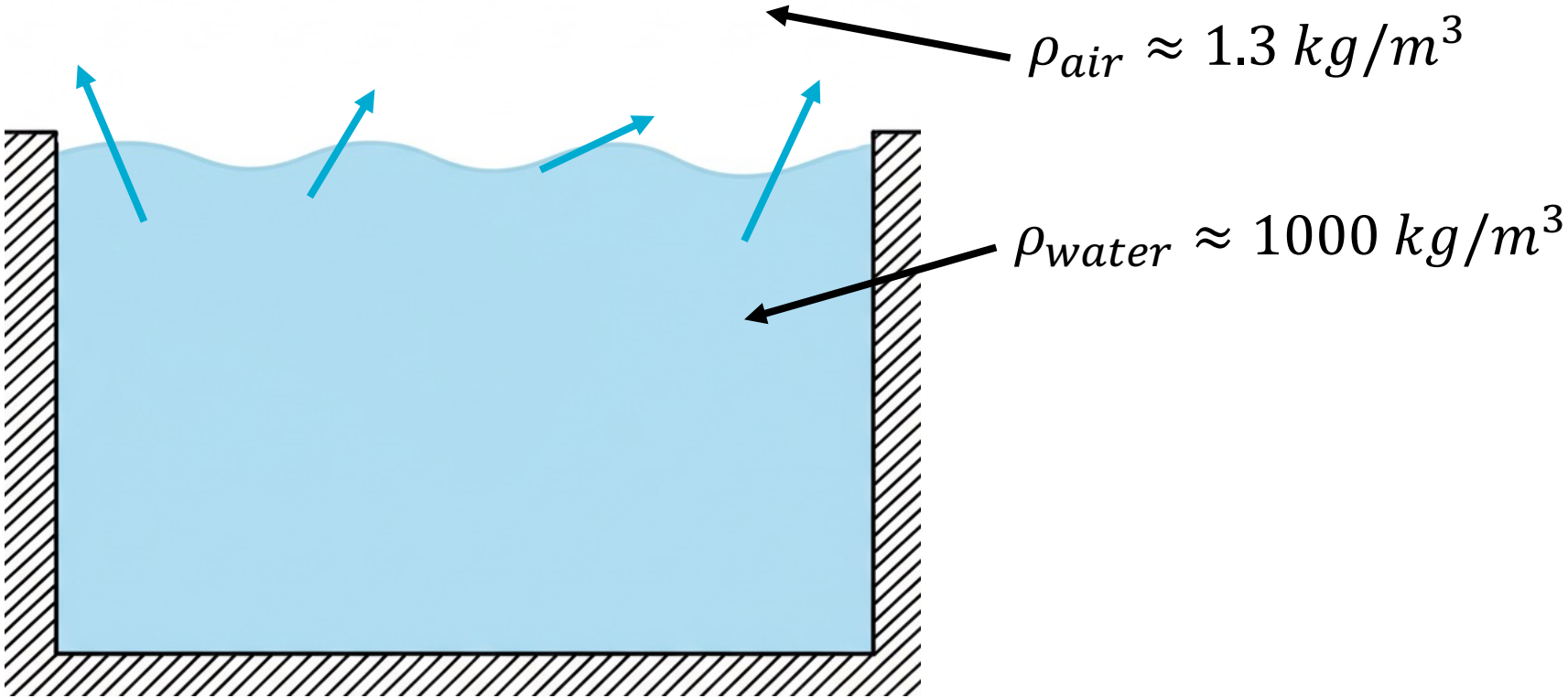
$$\boldsymbol{u} \cdot \hat{\boldsymbol{n}} = \boldsymbol{u}_{solid} \cdot \hat{\boldsymbol{n}}$$

"free slip"

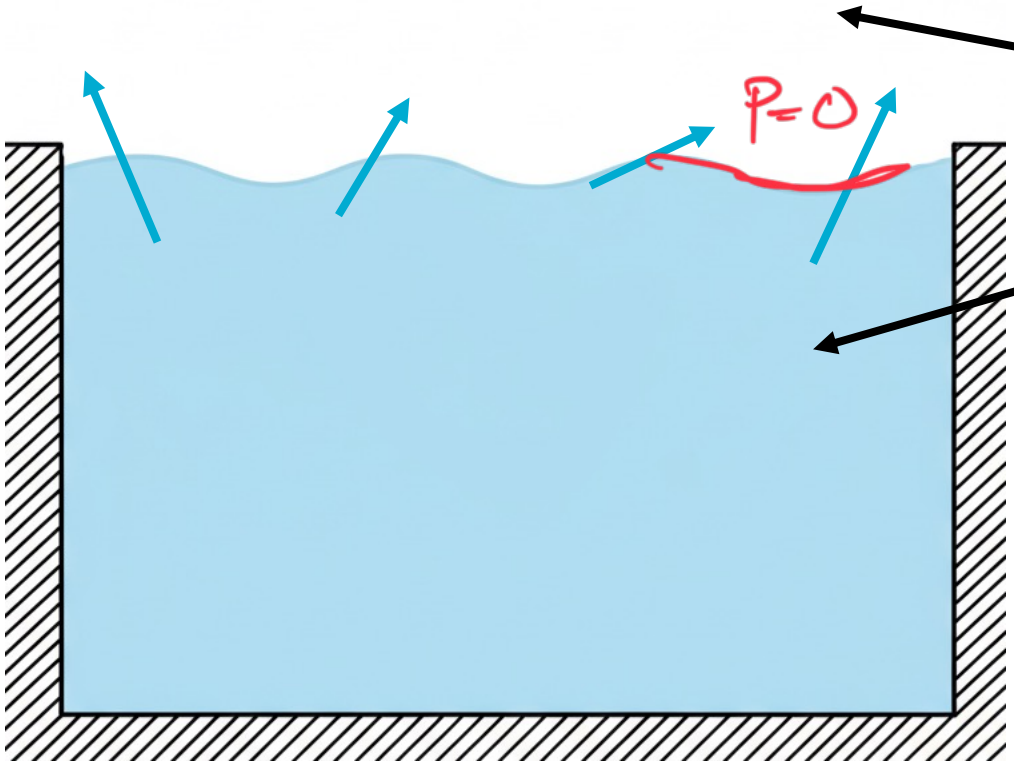
Boundary Conditions - Air



Boundary Conditions - Air



Boundary Conditions - Air



$\rho_{air} \approx 1.3\ kg/m^3$

$\rho_{water} \approx 1000\ kg/m^3$

$P = 0$

"free surface"

Algorithm
The Day After Tomorrow (2004)

Recap

momentum

$$\frac{D\boldsymbol{u}}{Dt} = g - \frac{1}{\rho}\nabla P$$

subject to

B.C

↳ solid

↳ air

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

div-free

incompressibility

Recap

$$\frac{D\boldsymbol{u}}{Dt} = g - \frac{1}{\rho}\nabla P \longrightarrow \boxed{\frac{d\boldsymbol{u}}{dt} = -\boldsymbol{u}\cdot\nabla\boldsymbol{u} + g - \frac{1}{\rho}\nabla P}$$

static motion

definition of material derivative

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

$\frac{\partial x}{\partial t} = 3 + 4$

$\frac{\partial x}{\partial t} = 7$

$x = 7t + c$

$\frac{\partial x^1}{\partial t} = 3 \leftarrow x^1 = 3t$

$\frac{\partial x^2}{\partial t} = 4 \leftarrow x^2 = 4t$

$x = 3t + 4t = 7t$

Operator Splitting

1: $\boxed{\dfrac{d\boldsymbol{u}}{dt} = -\boldsymbol{u}\cdot\nabla\boldsymbol{u}}$ $\xrightarrow{\text{material derivative}}$ $\dfrac{D\boldsymbol{u}}{Dt} = 0$

advection

2: $\dfrac{d\boldsymbol{u}}{dt} = g$ ← body force

3: $\dfrac{d\boldsymbol{u}}{dt} = -\dfrac{1}{\rho}\nabla P$ ← momentum term such that $\nabla\cdot\boldsymbol{u} = 0$

incompressibility

Algorithm

for each timestep:

$$\boldsymbol{u}^A = advect(\Delta t, \boldsymbol{u}^n)$$

$$\boldsymbol{u}^B = body(\Delta t, \boldsymbol{u}^A)$$

$$\boldsymbol{u}^{n+1} = project(\Delta t, \boldsymbol{u}^B)$$

- convention: superscripts = time discretization
  subscripts = space discretization

Algorithm

for each timestep:

$$\boldsymbol{u}^A = advect(\Delta t, \boldsymbol{u}^n)$$

$$\boldsymbol{u}^B = body(\Delta t, \boldsymbol{u}^A) = \boldsymbol{u}^A + \Delta t g$$

$$\boldsymbol{u}^{n+1} = project(\Delta t, \boldsymbol{u}^B)$$

- convention: superscripts = time discretization
  subscripts = space discretization

Pressure Projection

$$\frac{d\boldsymbol{u}}{dt} = -\frac{1}{\rho}\nabla P$$

Pressure Projection

$$\frac{d\boldsymbol{u}}{dt} = -\frac{1}{\rho}\nabla P$$

forward Euler

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^{B} - \frac{\cancel{1}\,\Delta t}{\rho}\nabla P$$

Pressure Projection

$$\frac{d\boldsymbol{u}}{dt} = -\frac{1}{\rho}\nabla P$$

forward Euler

$$\boldsymbol{u}^{n+1} = \boxed{\boldsymbol{u}^{B} - \frac{\Delta t}{\rho}\nabla P}$$

want to find P such that $\nabla \cdot \boldsymbol{u}^{n+1} = 0$

Pressure Projection

$$0 = \nabla \cdot \boldsymbol{u}^{n+1}$$

Pressure Projection

$$0 = \nabla \cdot \boldsymbol{u}^{n+1}$$

$$0 = \nabla \cdot \left(\boldsymbol{u}^{B} - \frac{\Delta t}{\rho} \nabla P \right)$$

Pressure Projection

$$0 = \nabla \cdot \boldsymbol{u}^{n+1}$$

$$0 = \nabla \cdot \left(\boldsymbol{u}^{B} - \frac{\Delta t}{\rho} \nabla P \right)$$

$$0 = \nabla \cdot \boldsymbol{u}^{B} - \frac{\Delta t}{\rho} \underline{\nabla \cdot \nabla} P$$

Δ "Laplacian"

Pressure Projection

$$0 = \nabla \cdot \boldsymbol{u}^{n+1}$$

$$0 = \nabla \cdot \left(\boldsymbol{u}^{B} - \frac{\Delta t}{\rho} \nabla P \right)$$

$$0 = \nabla \cdot \boldsymbol{u}^{B} - \frac{\Delta t}{\rho} \nabla \cdot \nabla P$$

$\Delta A = b$

"Laplacian"

$$\boxed{\Delta P = \frac{\rho}{\Delta t} \nabla \cdot \boldsymbol{u}^{B}}$$

"Poisson problem"

Poisson Problem – Matrix Form

$$\Delta P = \frac{\rho}{\Delta t} \nabla \cdot \boldsymbol{u}^B$$

$$\mathrm{L}\mathrm{p} = \frac{\rho}{\Delta t} \mathrm{D}\mathbf{u}^{\mathrm{B}}$$

matrix (n x n)

vector (n x 1)

vector (3n x 1)

matrix (n x 3n)

scalar

n points 3 dim

M

vector field

Scalar field

unknown

Scalar field → Scalar field

known

Lp = b

Pressure Projection

known

$$\boldsymbol{u}^{n+1} = \underline{\boldsymbol{u}^{B} - \frac{\Delta t}{\rho}\nabla P}$$

unknown

$$\mathbf{u}^{\mathrm{n+1}} = \mathbf{u}^{\mathrm{B}} - \frac{\Delta t}{\rho}\mathrm{Gp}$$

vector (3n x 1)

vector (n x 1)

matrix (3n x n)

scalar

vector (3n x 1)

Why does this work?

$$\boldsymbol{u} = \boldsymbol{u}_{div-free} + \boldsymbol{u}_{curl-free}$$

Why does this work?

$$\boldsymbol{u} = \boldsymbol{u}_{div-free} + \boldsymbol{u}_{curl-free}$$

$$\nabla \cdot \boldsymbol{u} = \nabla \cdot \left(\boldsymbol{u}_{div-free} + \boldsymbol{u}_{curl-free}\right)$$

Why does this work?

$$\boldsymbol{u} = \boldsymbol{u}_{div-free} + \boldsymbol{u}_{curl-free}$$

$$\nabla \cdot \boldsymbol{u} = \nabla \cdot \left(\boldsymbol{u}_{div-free} + \boldsymbol{u}_{curl-free}\right)$$

$$\nabla \cdot \boldsymbol{u} = \cancelto{0}{\nabla \cdot \boldsymbol{u}_{div-free}} + \nabla \cdot \boldsymbol{u}_{curl-free}$$

Why does this work?

$$\boldsymbol{u} = \boldsymbol{u}_{div-free} + \boldsymbol{u}_{curl-free}$$

$$\nabla \cdot \boldsymbol{u} = \nabla \cdot \left(\boldsymbol{u}_{div-free} + \boldsymbol{u}_{curl-free}\right)$$

$$\nabla \cdot \boldsymbol{u} = \cancelto{0}{\nabla \cdot \boldsymbol{u}_{div-free}} + \nabla \cdot \underline{\boldsymbol{u}_{curl-free}}$$

∇p

Scalar

Why does this work?

$$\boldsymbol{u} = \boldsymbol{u}_{div-free} + \boldsymbol{u}_{curl-free}$$

$$\nabla \cdot \boldsymbol{u} = \nabla \cdot \left(\boldsymbol{u}_{div-free} + \boldsymbol{u}_{curl-free}\right)$$

$$\nabla \cdot \boldsymbol{u} = \cancelto{0}{\nabla \cdot \boldsymbol{u}_{div-free}} + \nabla \cdot \boldsymbol{u}_{curl-free}$$

$$\nabla \cdot \boldsymbol{u} = \frac{\Delta \mathrm{t}}{\rho} \nabla \cdot \nabla P = \frac{\Delta t}{\rho} \Delta P$$

curl-free fields can be represented as the gradient of a scalar field

Projection as a Matrix System

$$\begin{bmatrix} \frac{\rho}{\Delta t}\mathrm{I} & \mathrm{G} \\ -\mathrm{D} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \frac{\rho}{\Delta t}\mathbf{u}^{\mathrm{B}} \\ 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \end{matrix}$$

Projection as a Matrix System

$$\begin{bmatrix} \frac{\rho}{\Delta t}\mathrm{I} & \mathrm{G} \\ -\mathrm{D} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \frac{\rho}{\Delta t}\mathbf{u}^{\mathrm{B}} \\ 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \end{matrix}$$

"KKT"

$$\frac{\rho}{\Delta t}\mathbf{u} + \mathrm{G}\mathbf{p} = \frac{\rho}{\Delta t}\mathbf{u}^{\mathrm{B}}$$

1 momentum equation

$$-\mathrm{D}\mathbf{p} = \mathbf{0}$$

2 divergence-free condition

Discrete Operators

Moana (2016)

Spatial Discretization

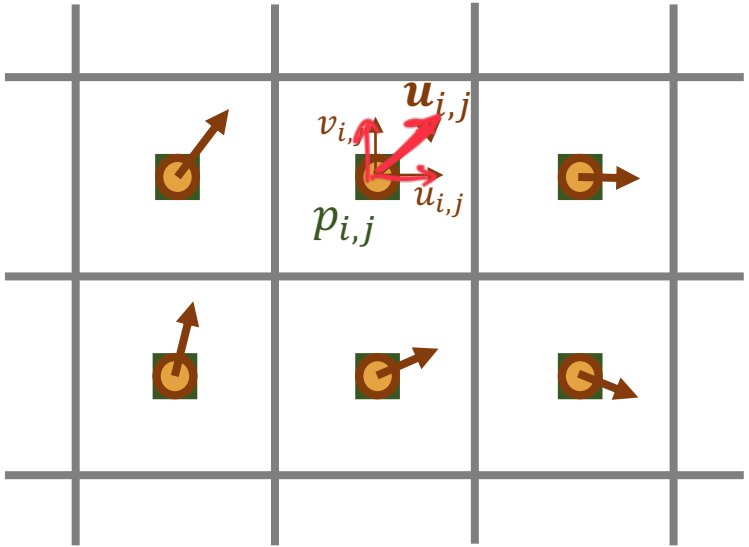
need to decide where the variables "live"

pressure scalar and velocity vector $\boldsymbol{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$

Spatial Discretization

need to decide where the variables "live"

pressure scalar and velocity vector $\boldsymbol{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$

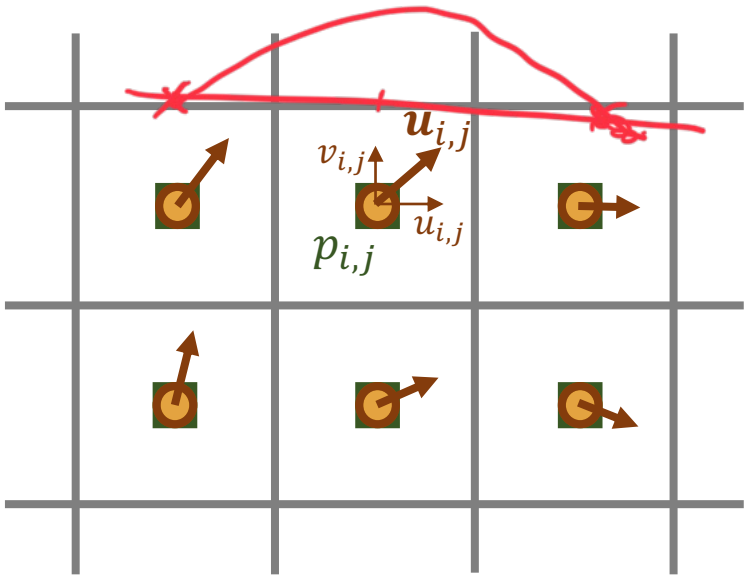


"collocated grid"

Spatial Discretization

need to decide where the variables "live"

pressure scalar and velocity vector $\boldsymbol{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$



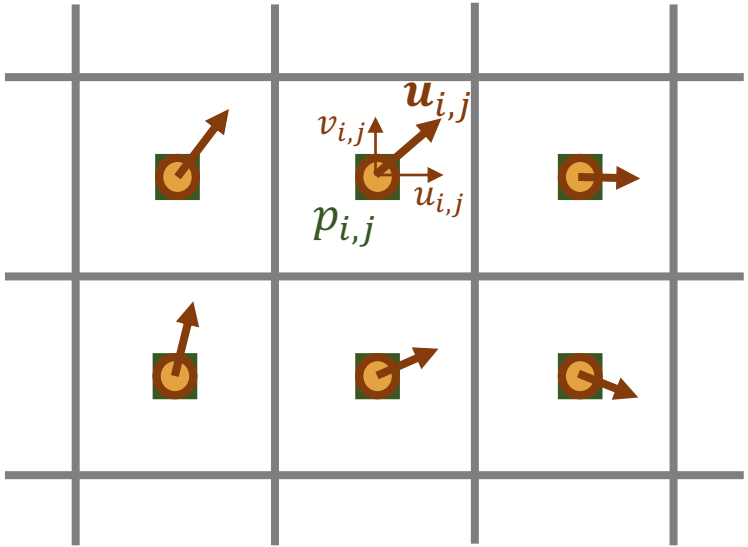
$$\frac{dq}{dx_i} = \frac{q_{i+1} - q_{i-1}}{2\Delta x}$$

"collocated grid"

Spatial Discretization

need to decide where the variables "live"

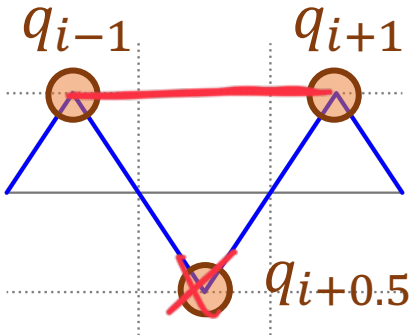
pressure scalar and velocity vector $\boldsymbol{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$



$$\frac{dq_i}{dx_i} = \frac{q_{i+1} - q_{i-1}}{2\Delta x}$$

q_i

consider:

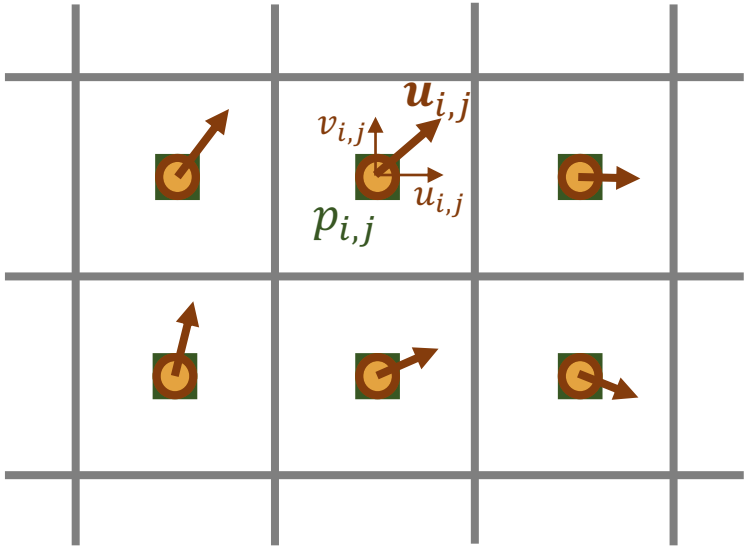


"collocated grid"

Spatial Discretization

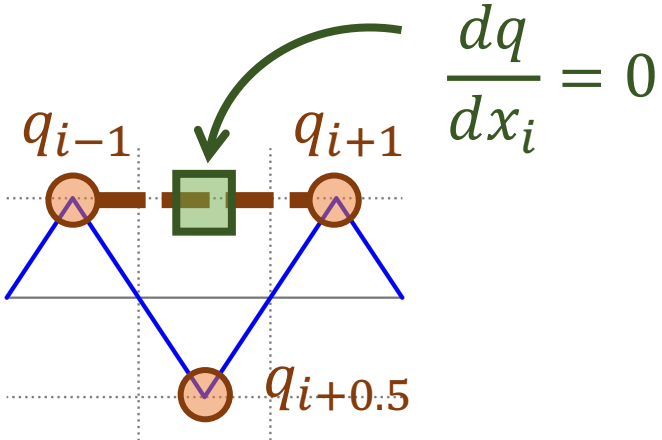
need to decide where the variables "live"

pressure scalar and velocity vector $\boldsymbol{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$



$$\frac{dq}{dx_i} = \frac{q_{i+1} - q_{i-1}}{2\Delta x}$$

consider:

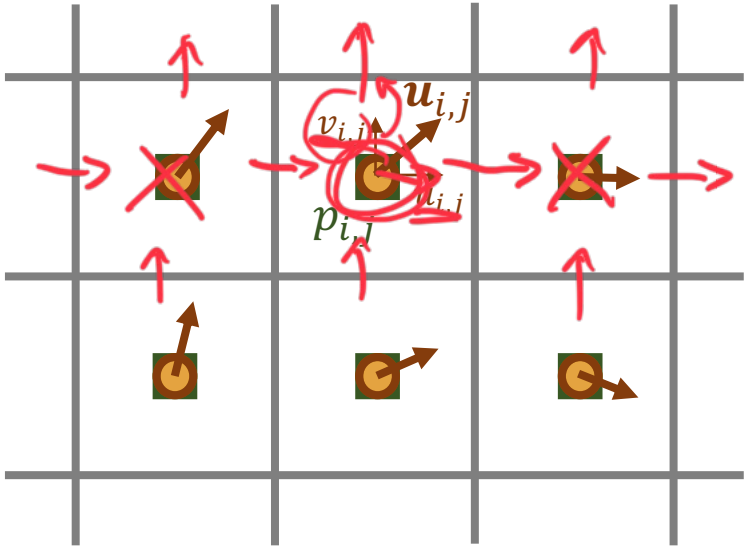


"collocated grid"

Spatial Discretization

need to decide where the variables "live"

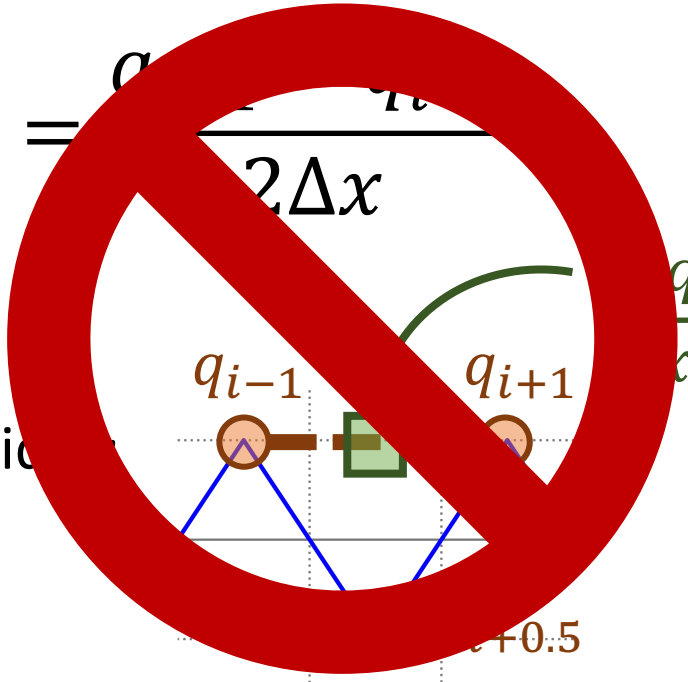
pressure scalar and velocity vector $\boldsymbol{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$



$$\frac{dq}{dx_i} = \frac{q}{2\Delta x}$$

$$\frac{q}{x_i} = 0$$

consi

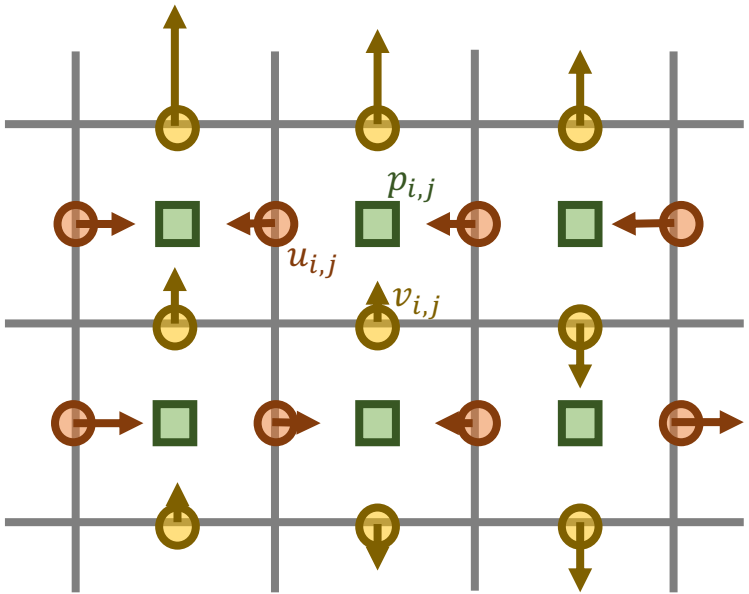


"collocated grid"

Spatial Discretization

need to decide where the variables "live"

pressure scalar and velocity vector $\boldsymbol{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$

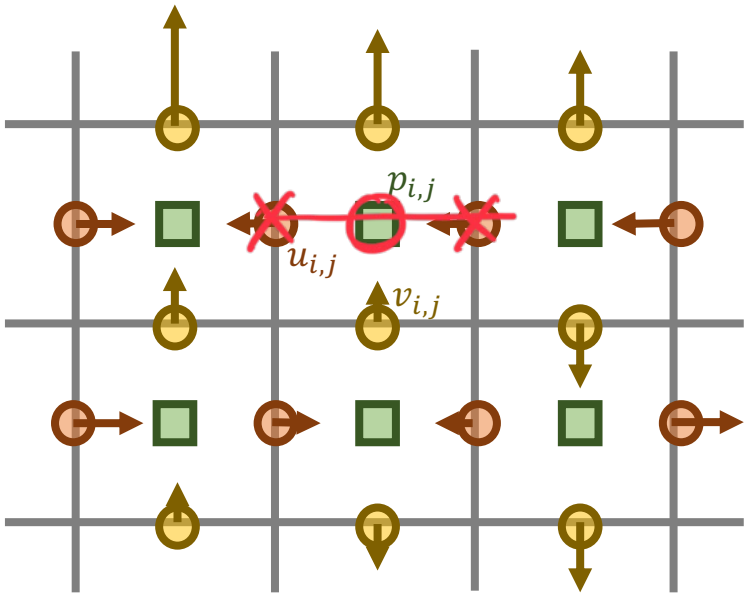


"staggered grid"

Spatial Discretization

need to decide where the variables "live"

pressure scalar and velocity vector $\boldsymbol{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$



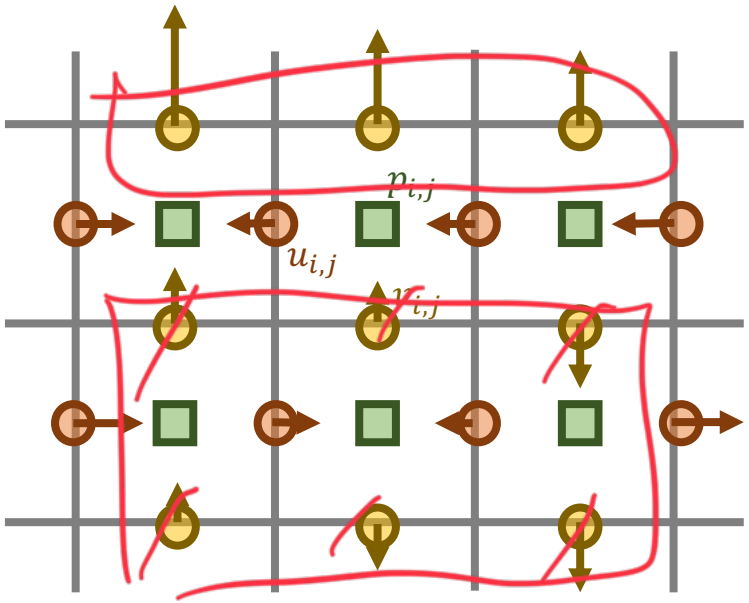
$$\frac{dq}{dx_i} = \frac{q_{i+0.5} - q_{i-0.5}}{\Delta x}$$

"staggered grid"

Spatial Discretization

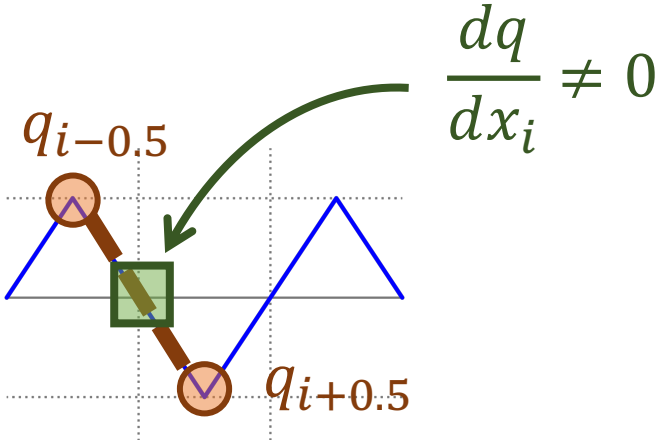
need to decide where the variables "live"

pressure scalar and velocity vector $\boldsymbol{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$



$$\frac{dq}{dx_i} = \frac{q_{i+0.5} - q_{i-0.5}}{\Delta x}$$

consider:

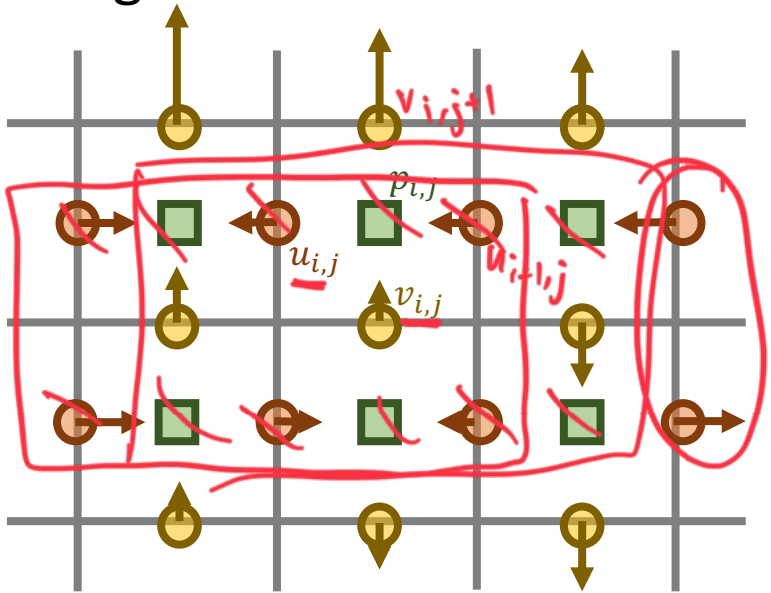


"staggered grid"

Grid Sizing

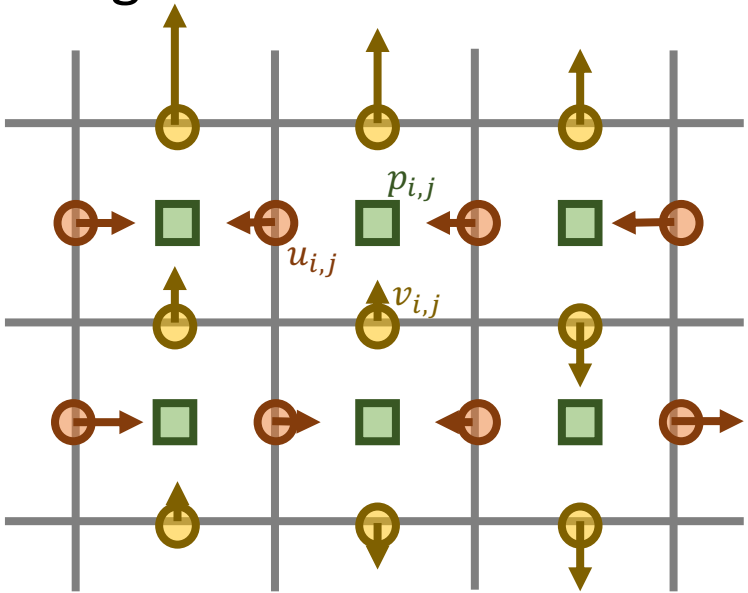
p[0.5][0.5]

- half-indexing doesn't exist
- grids for each variable become different sizes!



Grid Sizing

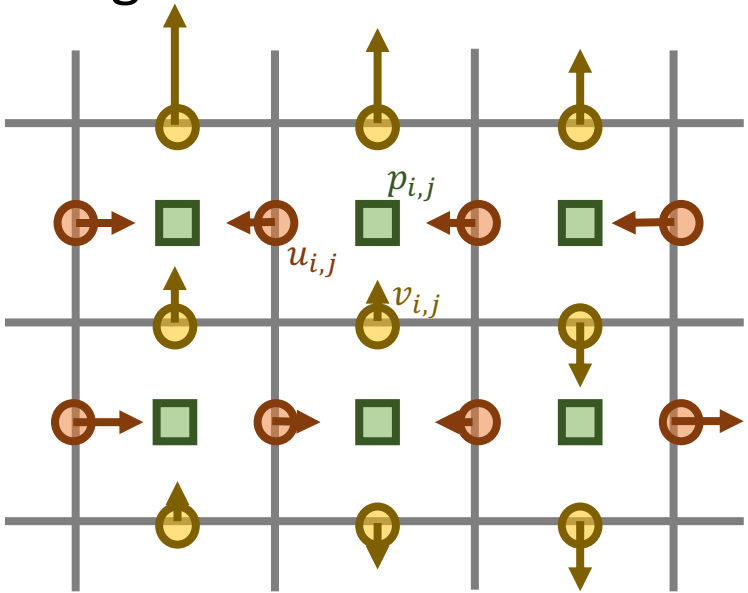
- half-indexing doesn't exist
- grids for each variable become different sizes!



$u \Rightarrow (4, 2) \Rightarrow (n_x + 1, n_y)$

Grid Sizing

- half-indexing doesn't exist
- grids for each variable become different sizes!

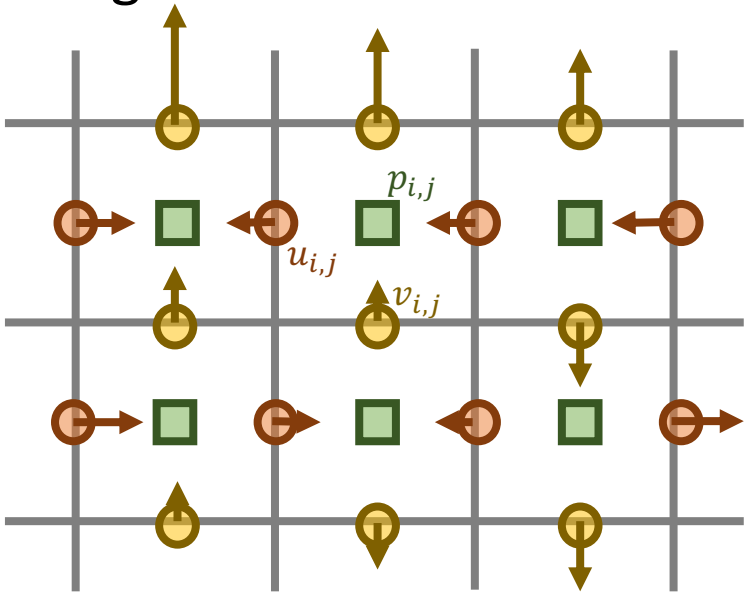


$u \Rightarrow (4, 2) \Rightarrow (n_x + 1, n_y)$

$v \Rightarrow (3, 3) \Rightarrow (n_x, n_y + 1)$

Grid Sizing

- half-indexing doesn't exist
- grids for each variable become different sizes!



$u \Rightarrow (4, 2) \Rightarrow (n_x + 1, n_y)$

$v \Rightarrow (3, 3) \Rightarrow (n_x, n_y + 1)$

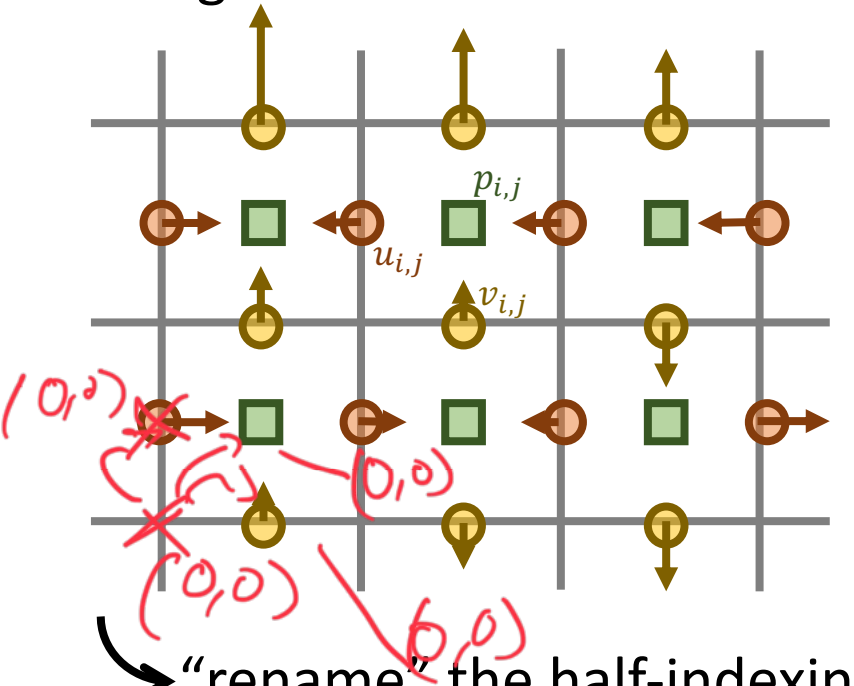
$w \Rightarrow (n_x, n_y, n_z + 1)$

$p \Rightarrow (3, 2) \Rightarrow (n_x, n_y)$

Grid Sizing

- half-indexing doesn't exist
- grids for each variable become different sizes!

$$\frac{q_i - q_{i-1}}{\Delta x}$$



- $u \Rightarrow (4, 2) \Rightarrow (n_x + 1, n_y)$
- $v \Rightarrow (3, 3) \Rightarrow (n_x, n_y + 1)$
- $p \Rightarrow (3, 2) \Rightarrow (n_x, n_y)$

- "rename" the half-indexing

$$\frac{q_{i+0.5} - q_{i-0.5}}{\Delta x} \Rightarrow \frac{q_{i+1} - q_i}{\Delta x}$$

Divergence Operator

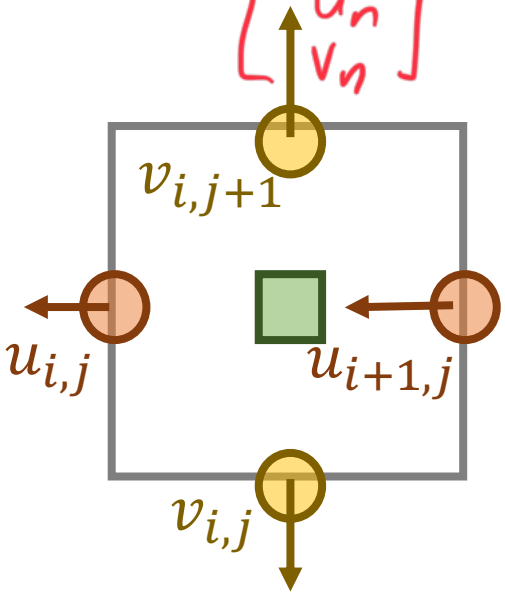
stacked vector! $\mathbf{u} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_n \\ v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix}$ $\longrightarrow$ $\mathrm{D} = \begin{bmatrix} D_x \\ D_y \end{bmatrix}$

$\vec{u} \begin{bmatrix} u_0 \\ v_0 \\ u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{bmatrix}$

$$\mathrm{D}\mathbf{u} = \nabla \cdot \boldsymbol{u}$$

$$= \frac{du}{dx} + \frac{dv}{dy}$$

finite diff



Divergence Operator

stacked vector! $\mathbf{u} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_n \\ v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix} \longrightarrow \mathrm{D} = \begin{bmatrix} D_x \\ D_y \end{bmatrix}$

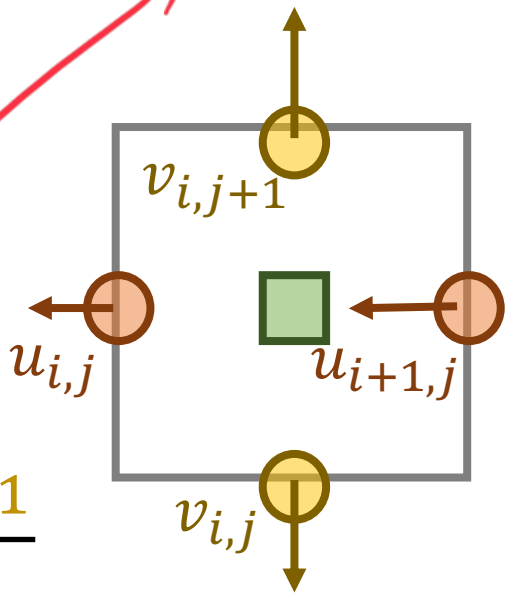
$$\mathrm{D}\mathbf{u} = \nabla \cdot \boldsymbol{u}$$

$$= \frac{du}{dx} + \frac{dv}{dy}$$

$$= \frac{u_{i,j} - u_{i+1,j}}{\Delta x} + \frac{v_{i,j} - v_{i,j+1}}{\Delta y}$$

+1

−1



Divergence Operator - Solid

stacked vector! $\mathbf{u} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_n \\ v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix} \longrightarrow \mathrm{D} = \begin{bmatrix} D_x \\ D_y \end{bmatrix}$

$$\mathrm{D}\mathbf{u} = \nabla \cdot \boldsymbol{u}$$

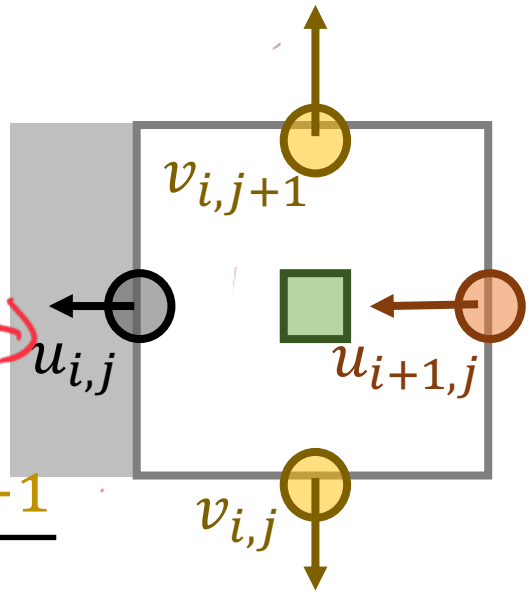
$$= \frac{du}{dx} + \frac{dv}{dy}$$

no longer an unknown, goes to RHS

$$= \frac{u_{i,j} - u_{i+1,j}}{\Delta x} + \frac{v_{i,j} - v_{i,j+1}}{\Delta y}$$

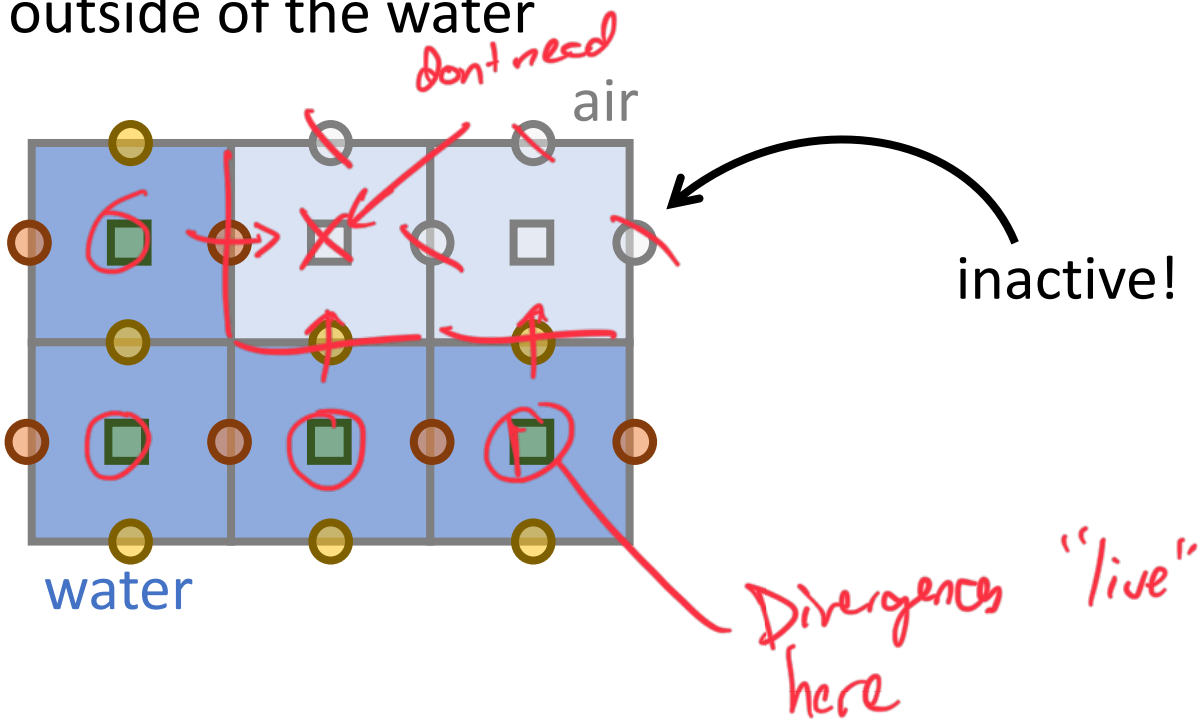
$u \cdot n = u_{solid} \cdot n$

known



Divergence Operator - Air

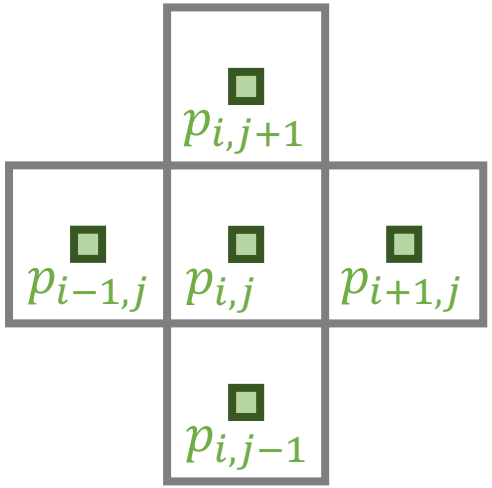
no change, never need to access unknowns outside of the water



Laplace Operator

$$\mathrm{Lp} = \Delta P = \nabla \cdot \nabla P$$

$$= \frac{d^2}{dx^2} p + \frac{d^2}{dy^2} p$$

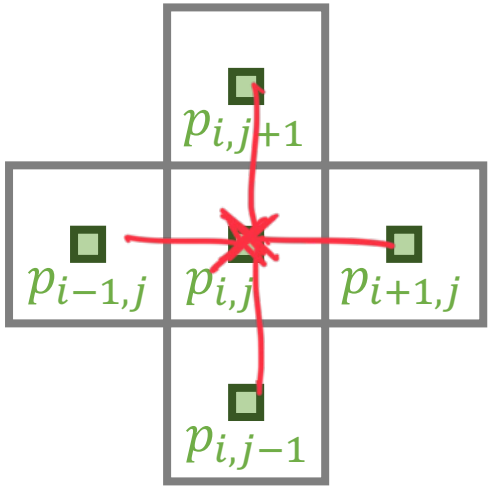


Laplace Operator

$$\mathrm{Lp} = \Delta P = \nabla \cdot \nabla P$$

$$= \frac{d^2}{dx^2} p + \frac{d^2}{dy^2} p$$

$$= \frac{p_{i-1,j} - 2p_{i,j} + p_{i+1,j}}{\Delta x^2} + \frac{p_{i,j-1} - 2p_{i,j} + p_{i,j+1}}{\Delta y^2}$$



$\Delta x = \Delta y$

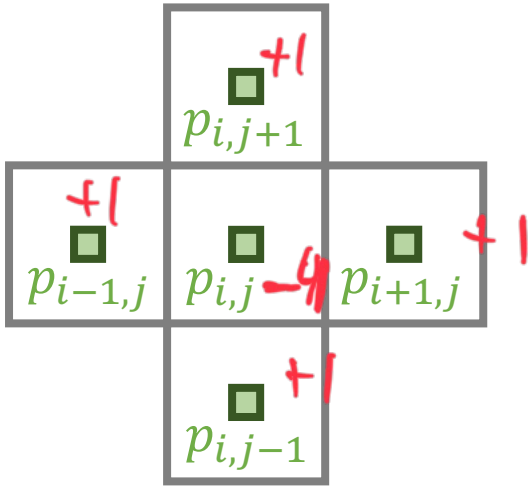
Laplace Operator

$$\mathrm{Lp} = \Delta P = \nabla \cdot \nabla P$$

$$= \frac{d^2}{dx^2}p + \frac{d^2}{dy^2}p$$

$$= \frac{p_{i-1,j} - 2p_{i,j} + p_{i+1,j}}{\Delta x^2} + \frac{p_{i,j-1} - 2p_{i,j} + p_{i,j+1}}{\Delta y^2}$$

$$= \frac{-4p_{i,j} + p_{i-1,j} + p_{i+1,j} + p_{i,j-1} + p_{i,j+1}}{\Delta x^2}$$

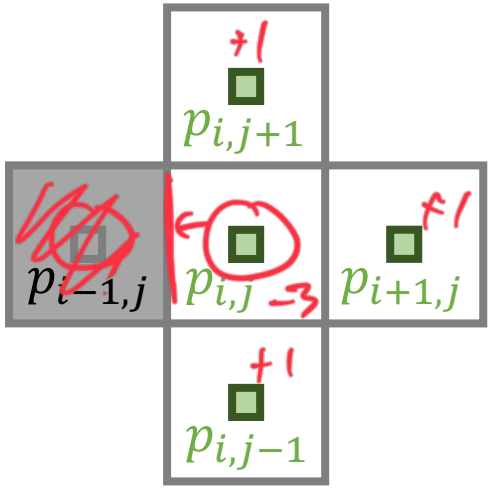


Laplace Operator - Solid

$$\mathrm{Lp} = \Delta P = \nabla \cdot \nabla P$$

make

$$p_{i-1,j} = p_{i,j}$$



$$= \frac{d^2}{dx^2}p + \frac{d^2}{dy^2}p$$

$$= \frac{p_{i,j} - 2p_{i,j} + p_{i+1,j}}{\Delta x^2} + \frac{p_{i,j-1} - 2p_{i,j} + p_{i,j+1}}{\Delta y^2}$$

$$= \frac{-3p_{i,j} + p_{i+1,j} + p_{i,j-1} + p_{i,j+1}}{\Delta x^2}$$

this coefficient changes and the $p_{i-1,j}$ term disappears

coeff in general = #neighbours - #solid

Laplace Operator - Air

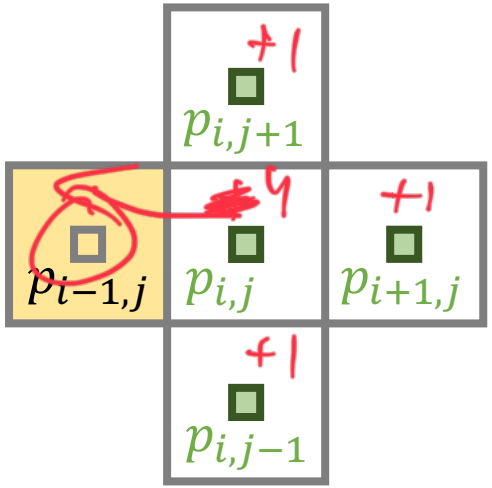
$$\mathrm{Lp} = \Delta P = \nabla \cdot \nabla P$$

$$= \frac{d^2}{dx^2}p + \frac{d^2}{dy^2}p$$

$$= \frac{0 - 2p_{i,j} + p_{i+1,j}}{\Delta x^2} + \frac{p_{i,j-1} - 2p_{i,j} + p_{i,j+1}}{\Delta y^2}$$

$$= \frac{-4p_{i,j} + p_{i+1,j} + p_{i,j-1} + p_{i,j+1}}{\Delta x^2}$$

make $p_{i-1,j} = 0$



this coefficient stays the same, but the $p_{i-1,j}$ term still disappears

Gradient Operator

$$\mathrm{Gp} = \nabla P$$

$$= \left(\frac{dp}{dx}, \frac{dp}{dy}\right)$$

Gradient Operator

$$\mathrm{Gp} = \nabla P$$

$$= \left(\frac{dp}{dx}, \frac{dp}{dy}\right)$$

stacked vector!

$$= \left(\frac{p_{i,j} - p_{i+1,j}}{\Delta x}, \frac{p_{i,j} - p_{i,j+1}}{\Delta y}\right)$$

$$\begin{bmatrix} \frac{\partial p}{\partial x} \\ \frac{\partial p}{\partial y} \end{bmatrix}$$

Gradient Operator

$$\mathrm{Gp} = \nabla P$$

$$= \left(\frac{dp}{dx}, \frac{dp}{dy}\right)$$

stacked vector!

$$= \left(\frac{p_{i,j} - p_{i+1,j}}{\Delta x}, \frac{p_{i,j} - p_{i,j+1}}{\Delta y}\right)$$

$$= -\mathrm{D}^{\mathrm{T}}\mathrm{p}$$

[n × 3n]

Div:

vector field

↓

scalar field

Grad

scalar field → vector field

[3n × n]

Advection

Life of Pi (2012)

Advection

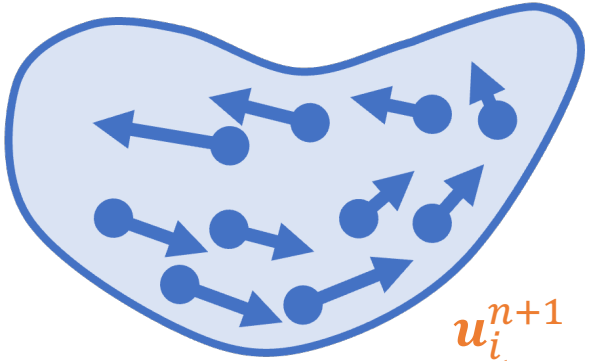
$$\frac{D\boldsymbol{u}}{Dt} = 0$$

"transport without change"

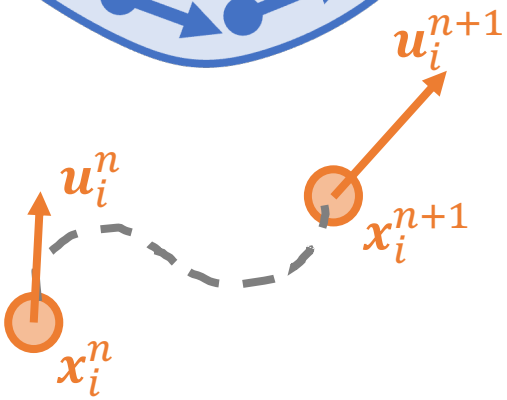
Advection

$$\frac{D\boldsymbol{u}}{Dt} = 0$$

"transport without change"



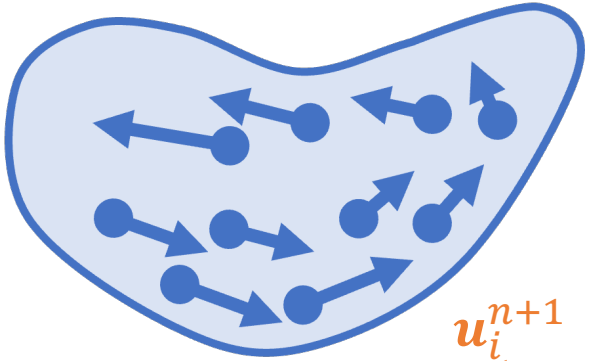
Particles can move forward and just keep their current $(\boldsymbol{u}, p)$ state



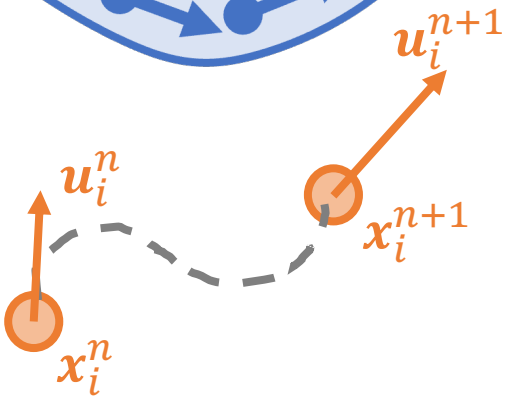
Advection

$$\frac{D\boldsymbol{u}}{Dt} = 0$$

"transport without change"

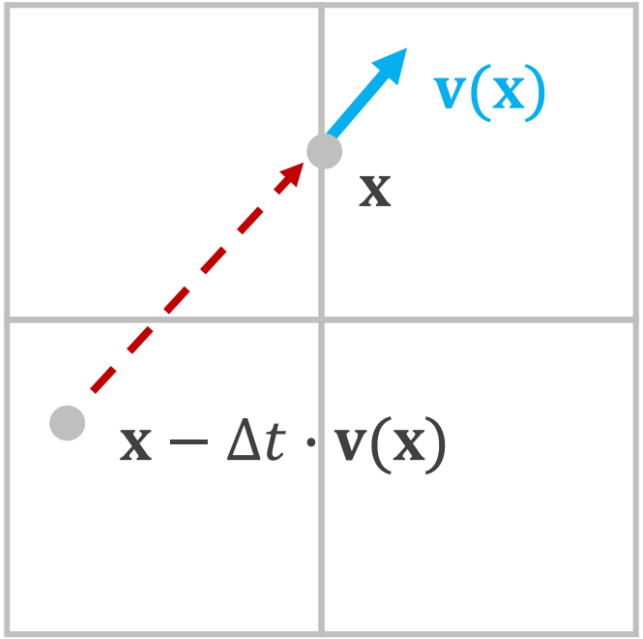


- Particles can move forward and just keep their current $(\boldsymbol{u}, p)$ state
- How to reconcile this with the grid
  - interpolation problem



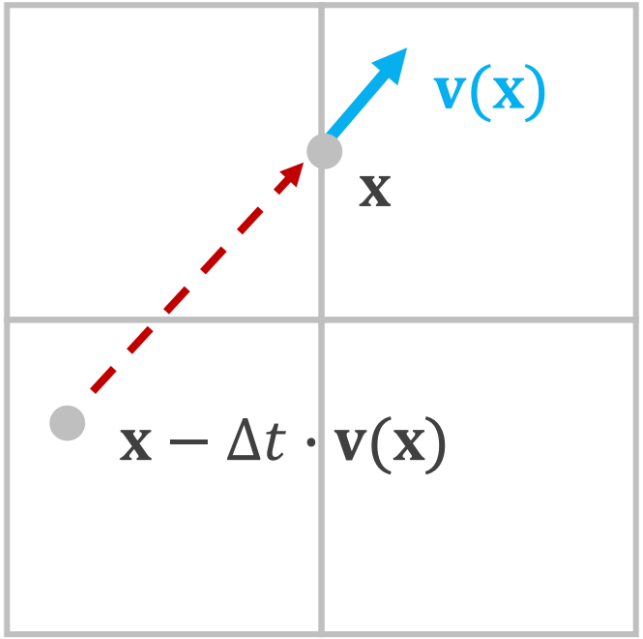
Advection – Backwards Semi-Lagrangian

trace backwards and interpolate



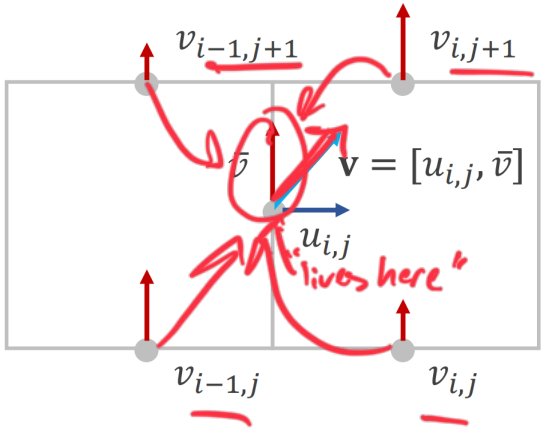
Advection – Backwards Semi-Lagrangian

trace backwards and interpolate



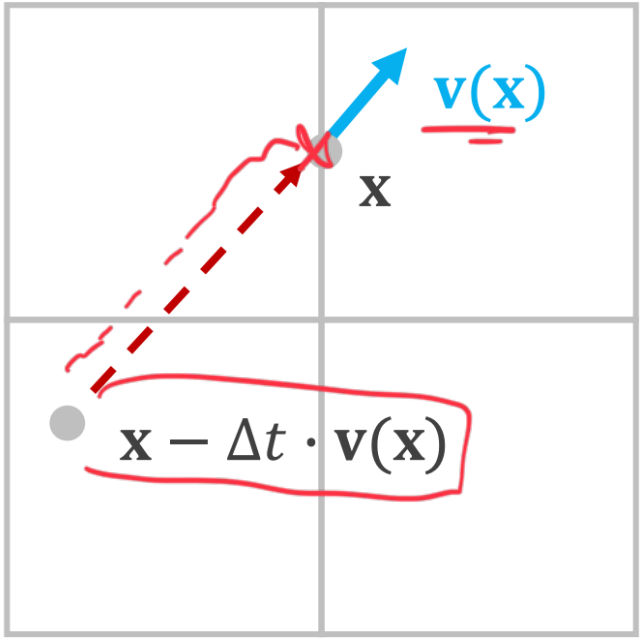
Step 1: Find velocity at the face

Find the parallel component by averaging



Advection – Backwards Semi-Lagrangian

trace backwards and interpolate



Step 1: Find velocity at the face

Find the parallel component by averaging

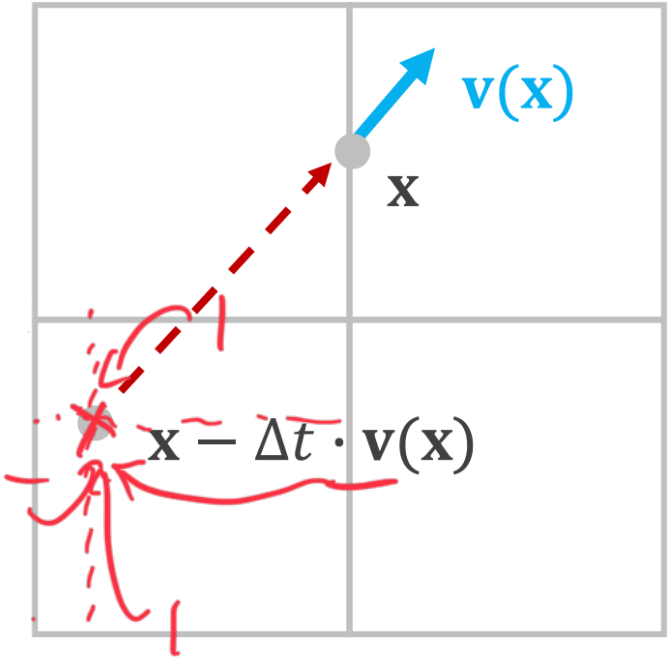
Step 2: Trace backwards

not +

$$\boldsymbol{x}^{n-1} = \boldsymbol{x}^{n} - \Delta t \boldsymbol{u}$$

Advection – Backwards Semi-Lagrangian

trace backwards and interpolate



Step 1: Find velocity at the face

Find the parallel component by averaging

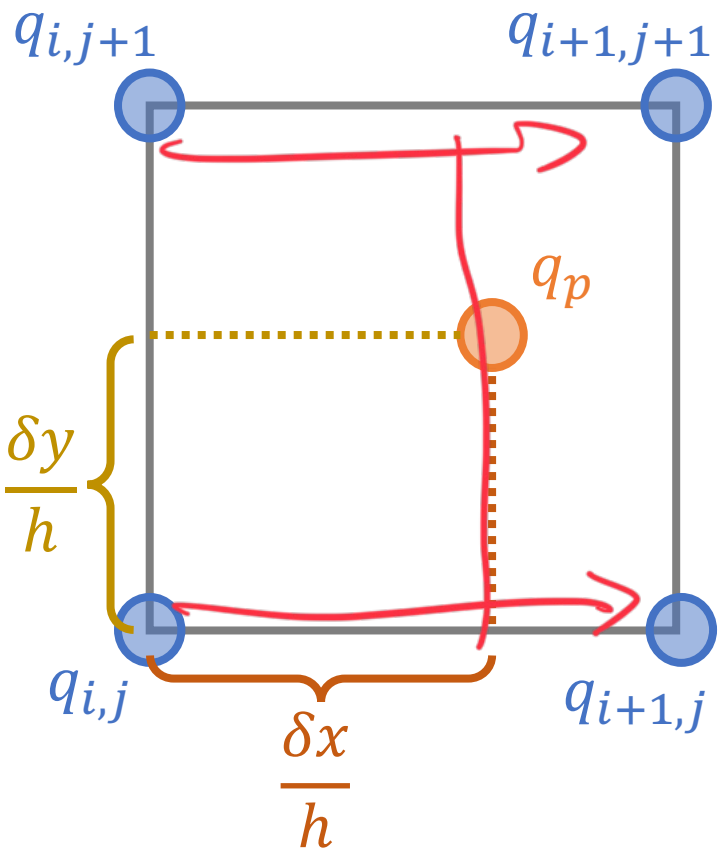
Step 2: Trace backwards

$$\boldsymbol{x}^{n-1} = \boldsymbol{x}^{n} - \Delta t \boldsymbol{u}$$

Step 3: Interpolate

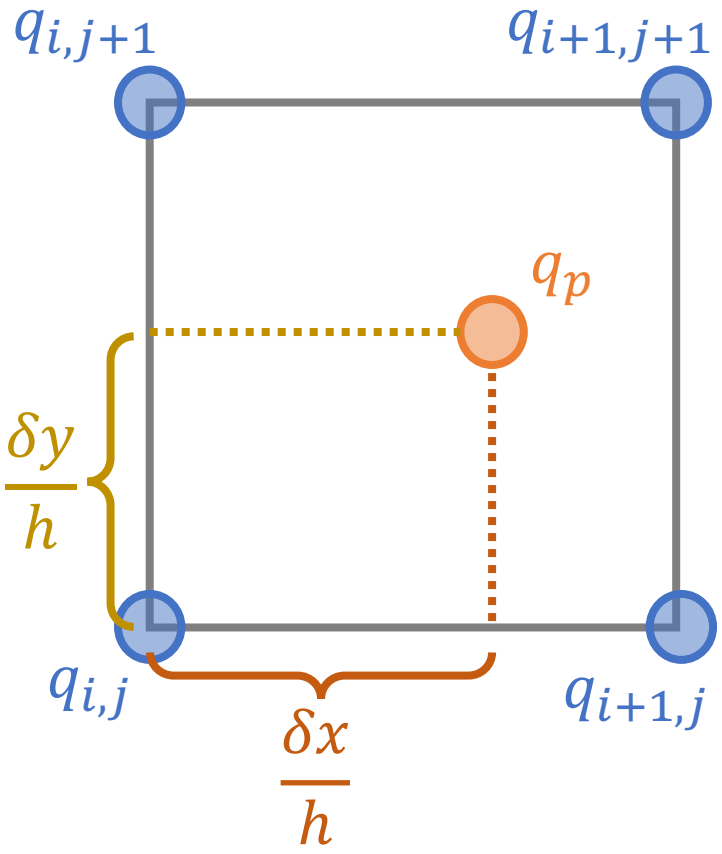
Bilinear

Bilinear Interpolation

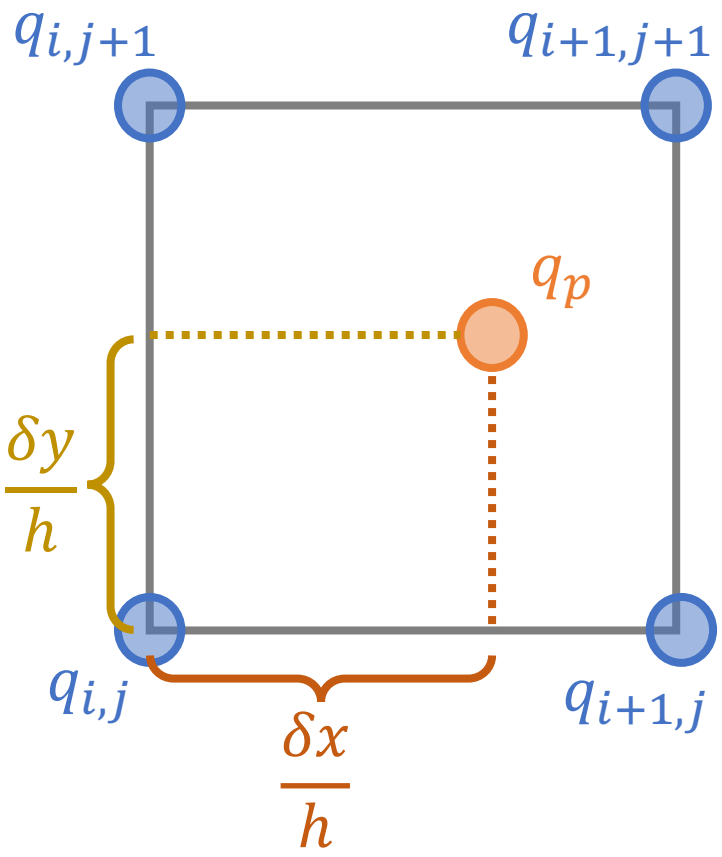


Bilinear Interpolation

$$w_{i,j} = \underbrace{\left(1 - \frac{\delta x}{h}\right)}_{\text{lerp } x}\underbrace{\left(1 - \frac{\delta y}{h}\right)}_{\text{lerp } y}$$



Bilinear Interpolation



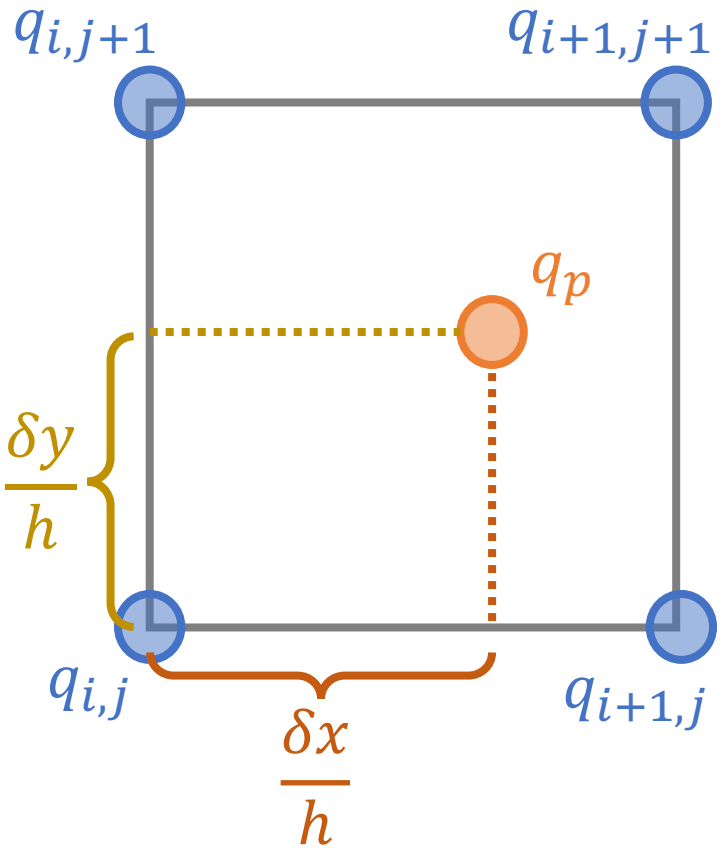
$$w_{i,j} = \left(1 - \frac{\delta x}{h}\right)\left(1 - \frac{\delta y}{h}\right)$$

$$w_{i+1,j} = \left(\frac{\delta x}{h}\right)\left(1 - \frac{\delta y}{h}\right)$$

$$w_{i,j+1} = \left(1 - \frac{\delta x}{h}\right)\left(\frac{\delta y}{h}\right)$$

$$w_{i+1,j+1} = \left(\frac{\delta x}{h}\right)\left(\frac{\delta y}{h}\right)$$

Bilinear Interpolation



$$w_{i,j} = \left(1 - \frac{\delta x}{h}\right)\left(1 - \frac{\delta y}{h}\right)$$

$$w_{i+1,j} = \left(\frac{\delta x}{h}\right)\left(1 - \frac{\delta y}{h}\right)$$

$$w_{i,j+1} = \left(1 - \frac{\delta x}{h}\right)\left(\frac{\delta y}{h}\right)$$

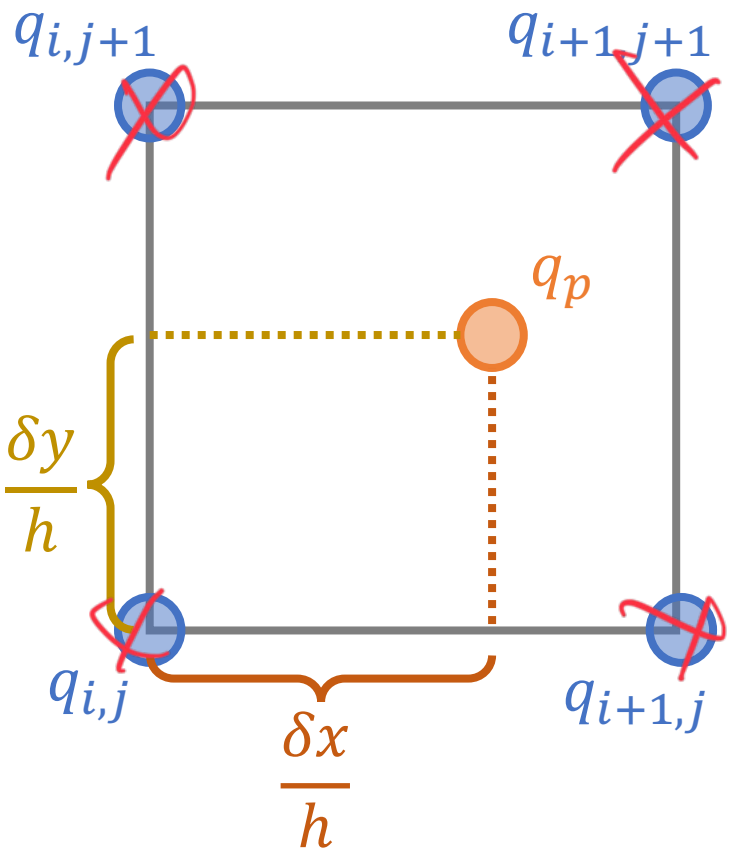
$$w_{i+1,j+1} = \left(\frac{\delta x}{h}\right)\left(\frac{\delta y}{h}\right)$$

$$\boxed{q_p = \frac{\sum_n w_n q_n}{\sum_n w_n}}$$

Just a "weighted sum"

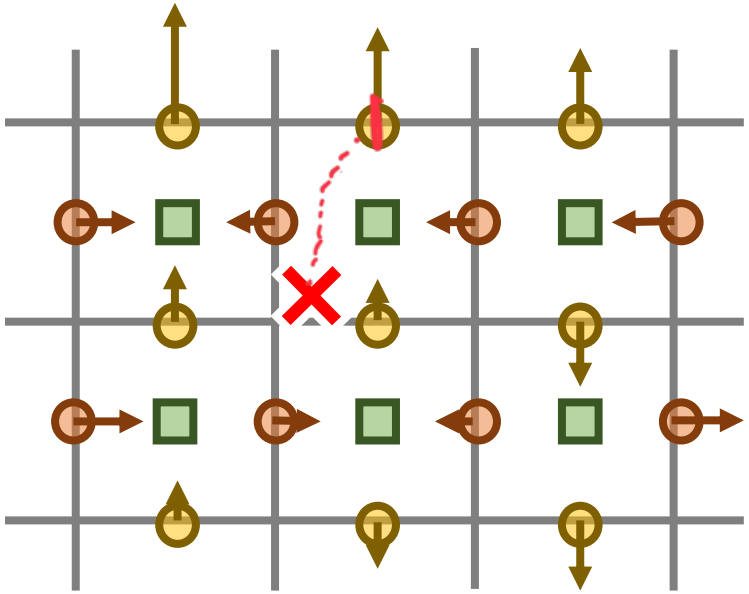
n are all grid samples inside $2h \times 2h$ box

Bilinear Interpolation

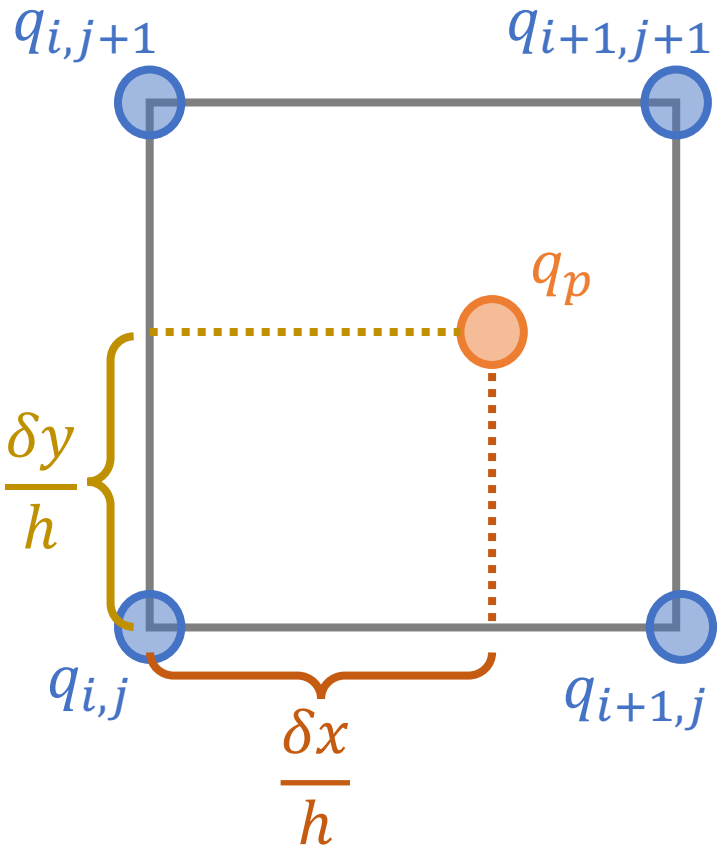


h distance?

remember 3 different grids

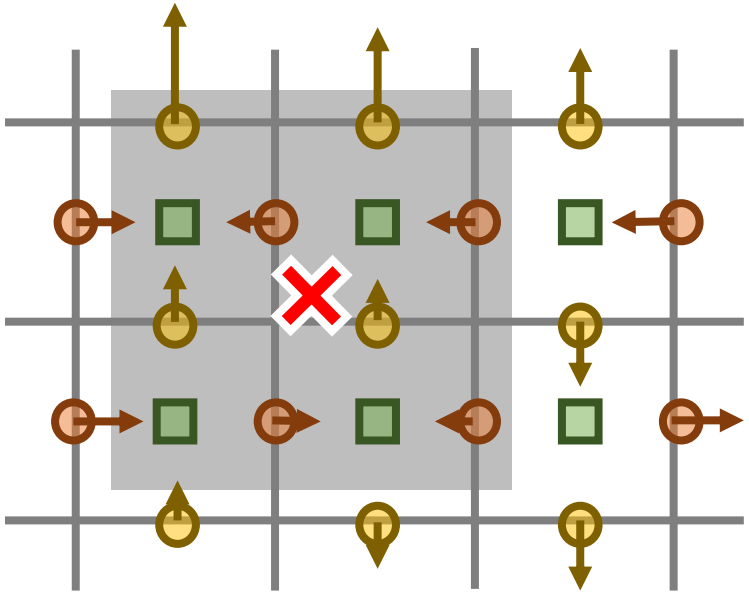


Bilinear Interpolation

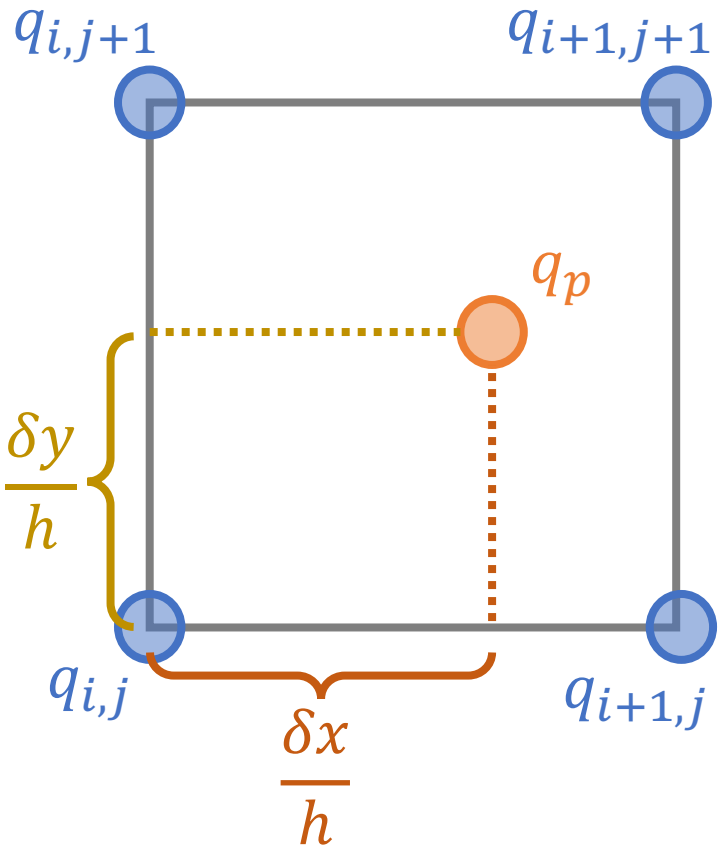


h distance?

remember 3 different grids

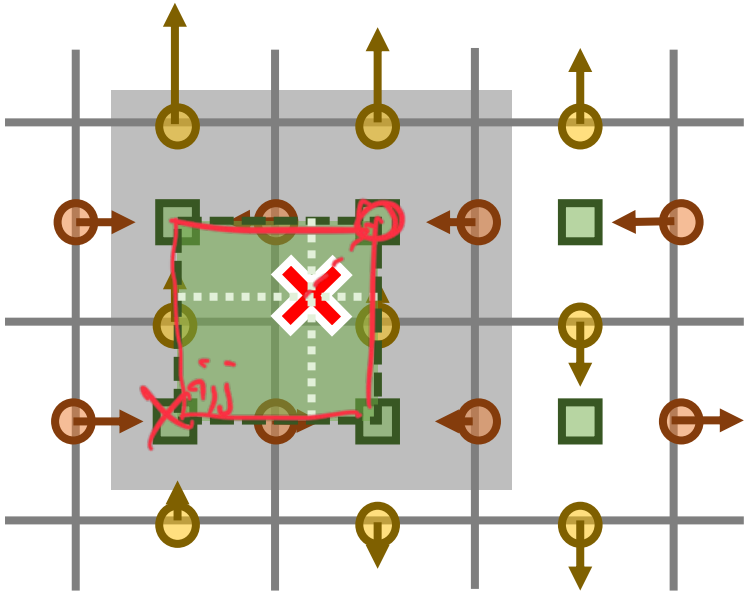


Bilinear Interpolation

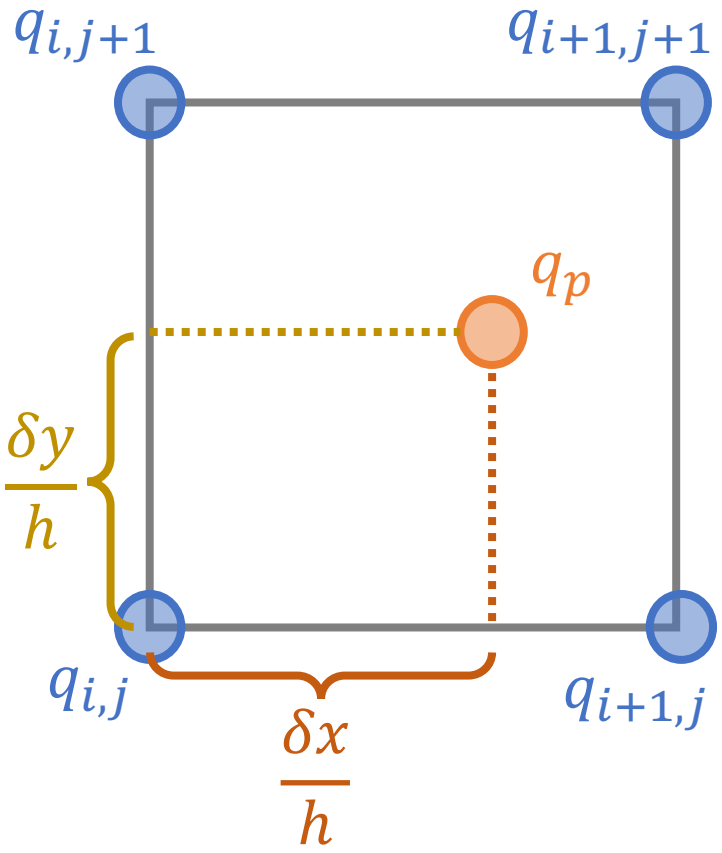


h distance?

remember 3 different grids

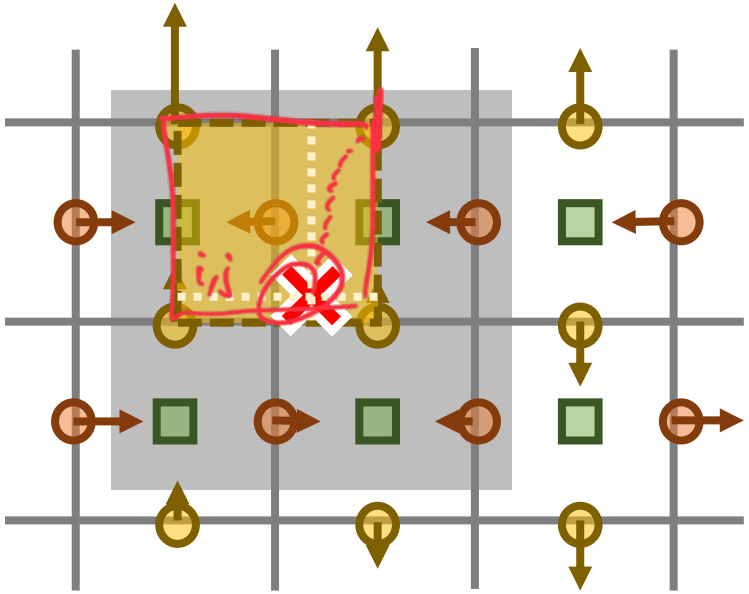


Bilinear Interpolation

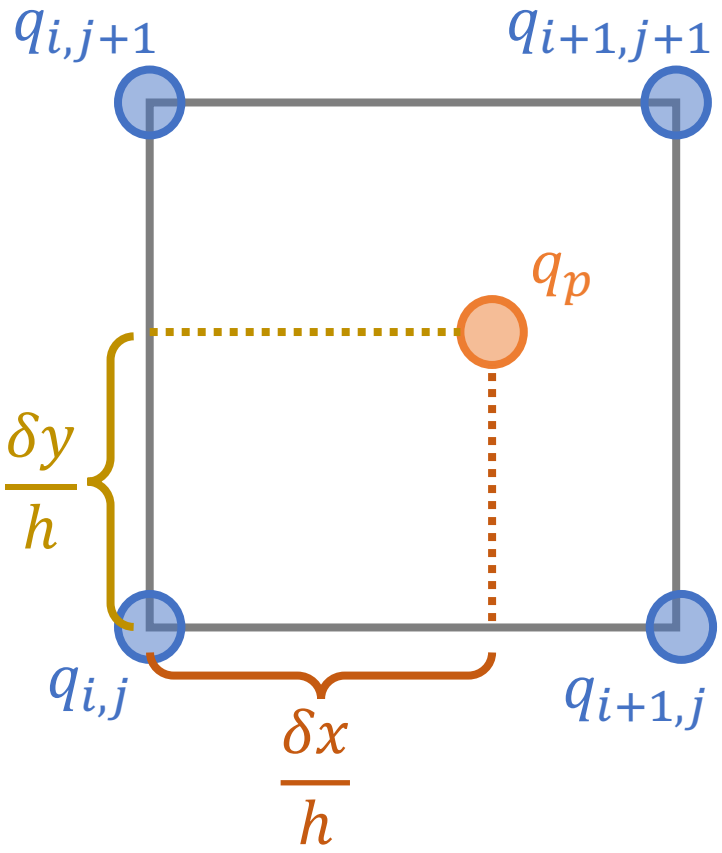


h distance?

remember 3 different grids

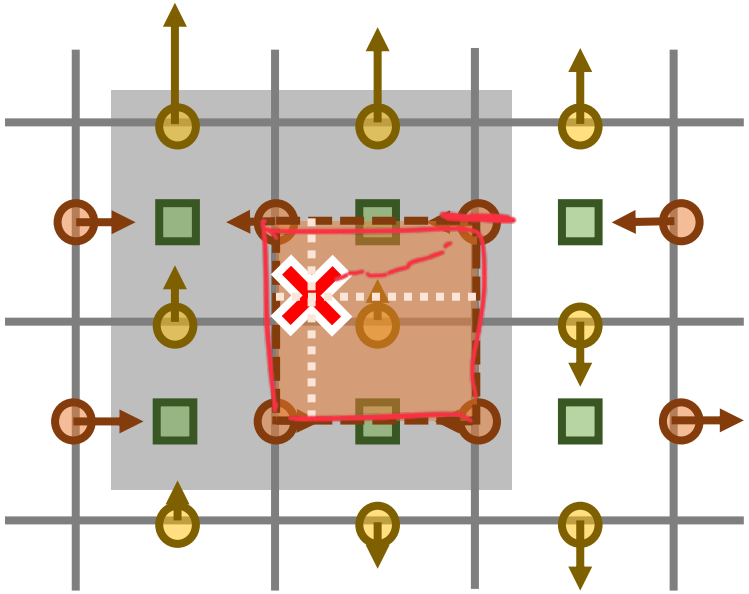


Bilinear Interpolation



h distance?

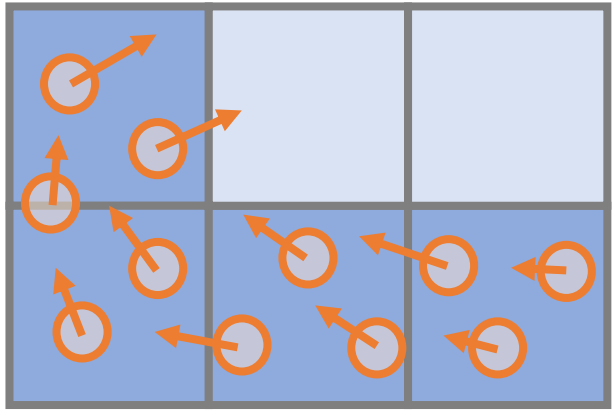
remember 3 different grids

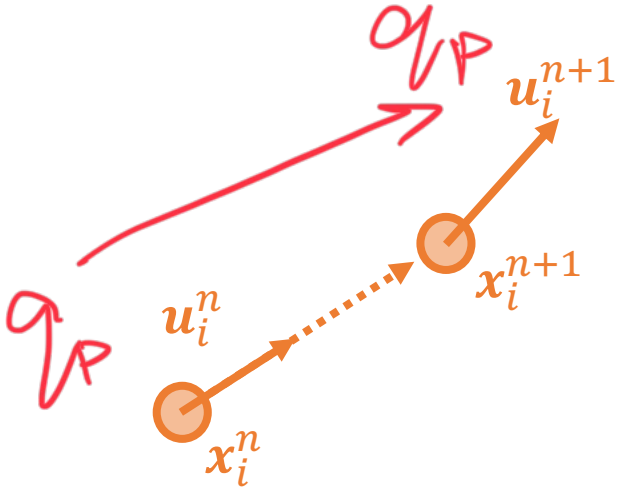


Particles?
The Mummy (1999)

Tracker Particles

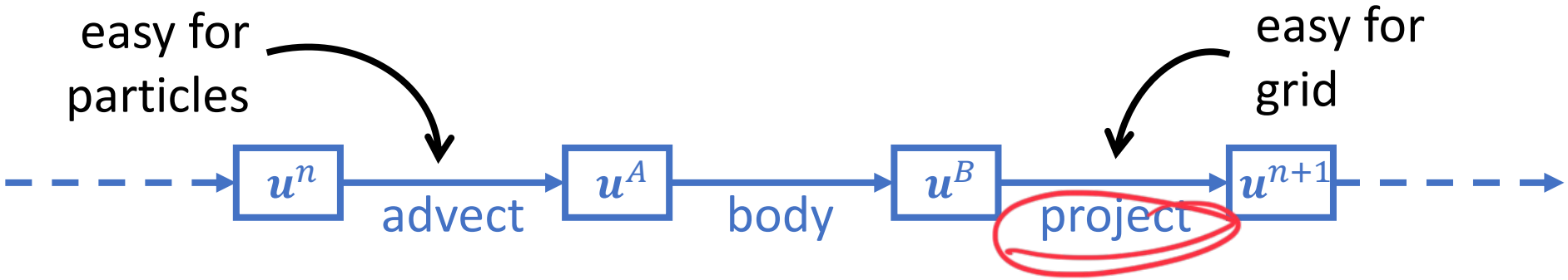
- add particles that exist independently of the grid
- advection is just forward Euler



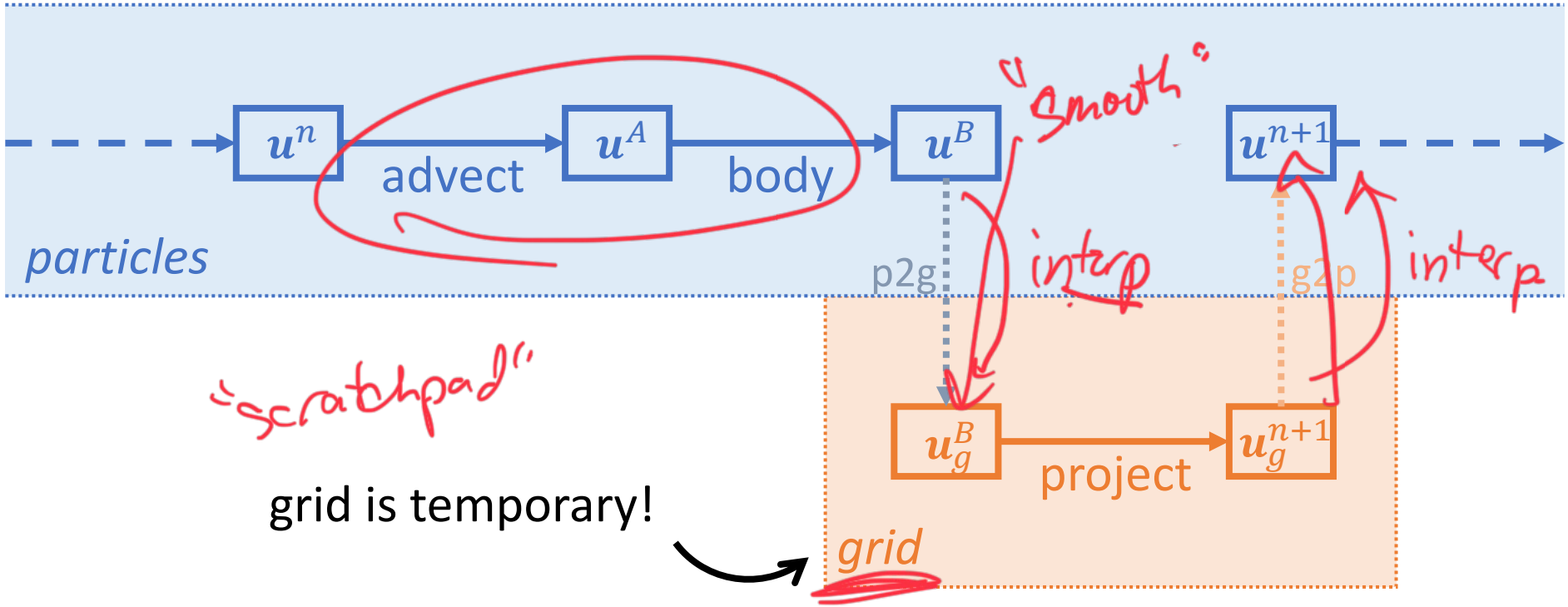


$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \Delta t \boldsymbol{u}^n$$

A New Pipeline



A New Pipeline – Particle-in-Cell



grid is temporary!

Particle-in-Cell – Grid2Particle

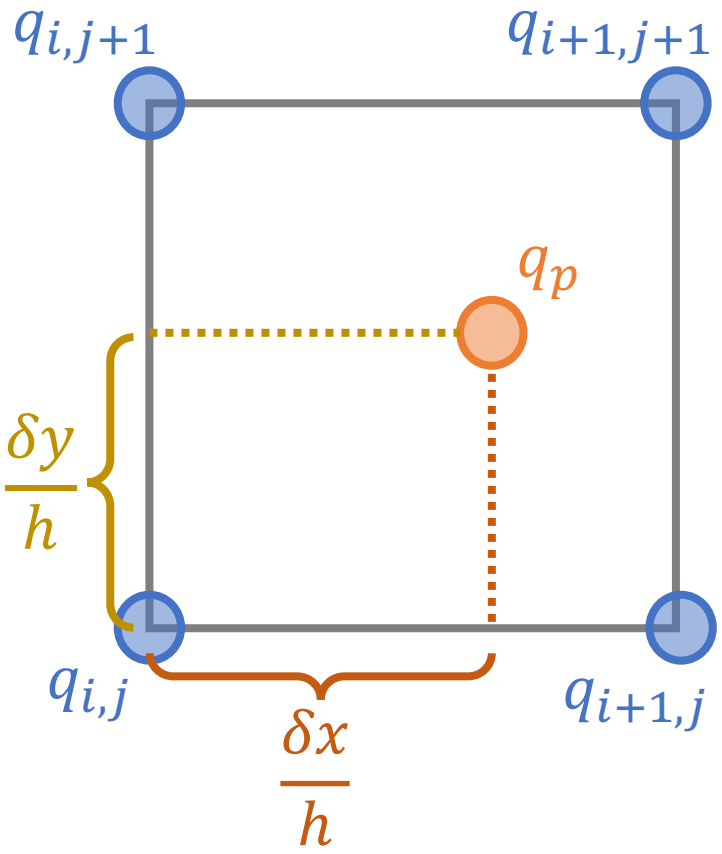
we already know this!

Particle-in-Cell – Grid2Particle

we already know this!

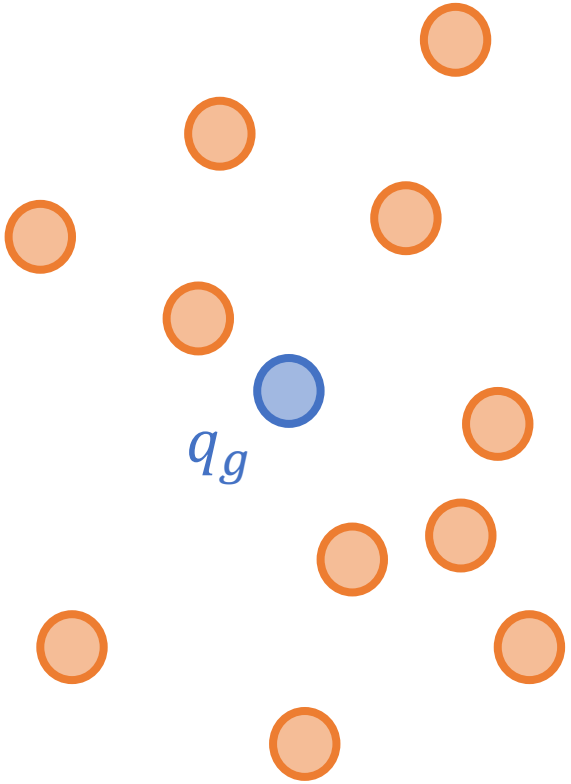
exactly bilinear interpolation
(weighted sum of grid samples)

$$q_p = \frac{\sum_n w_n q_n}{\sum_n w_n}$$



Particle-in-Cell – Particle2Grid

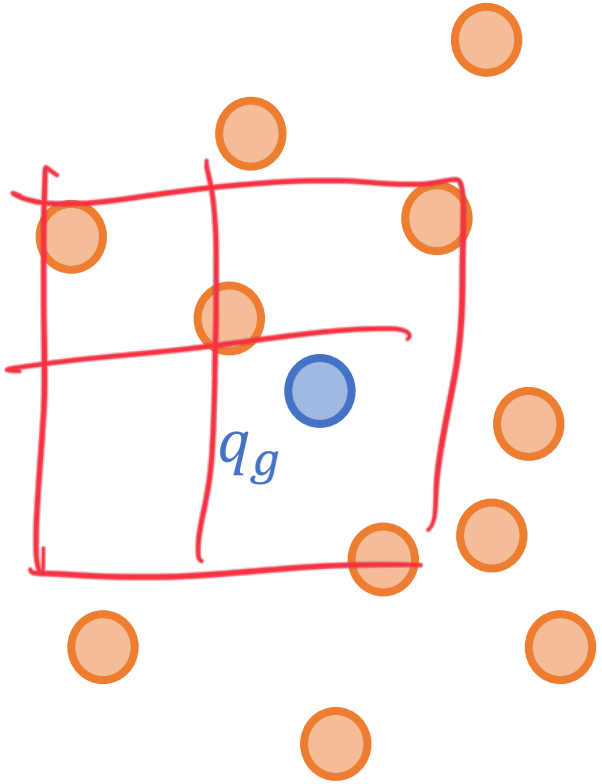
can we also use a weighted sum?



Particle-in-Cell – Particle2Grid

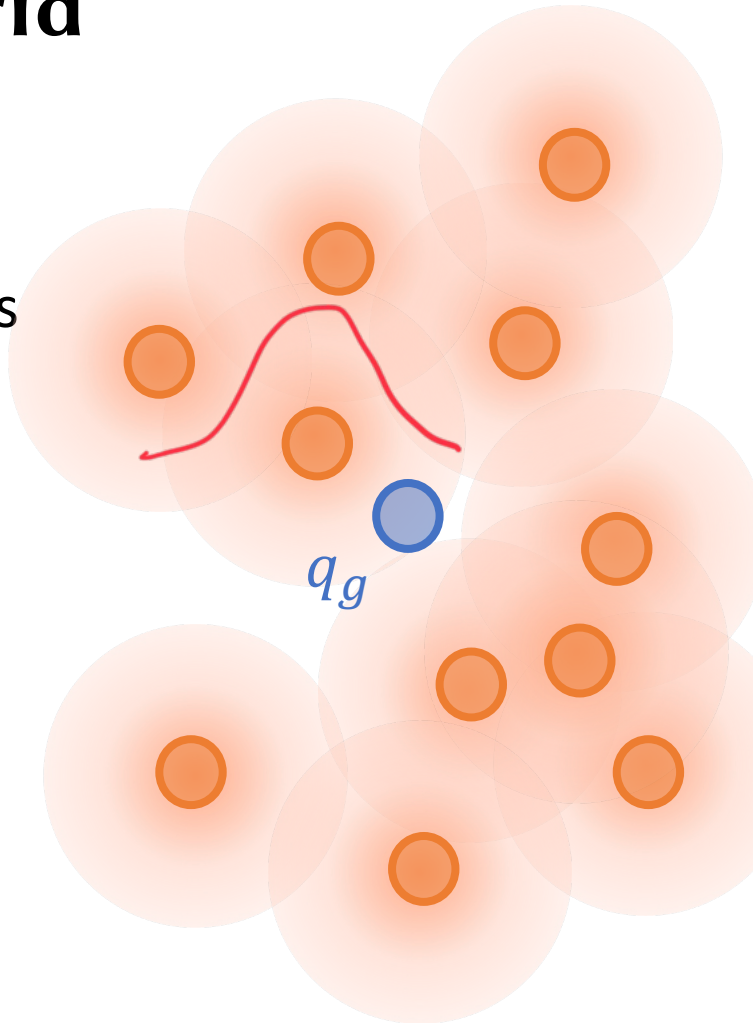
can we also use a weighted sum?

yes! but… there's a lot of particles



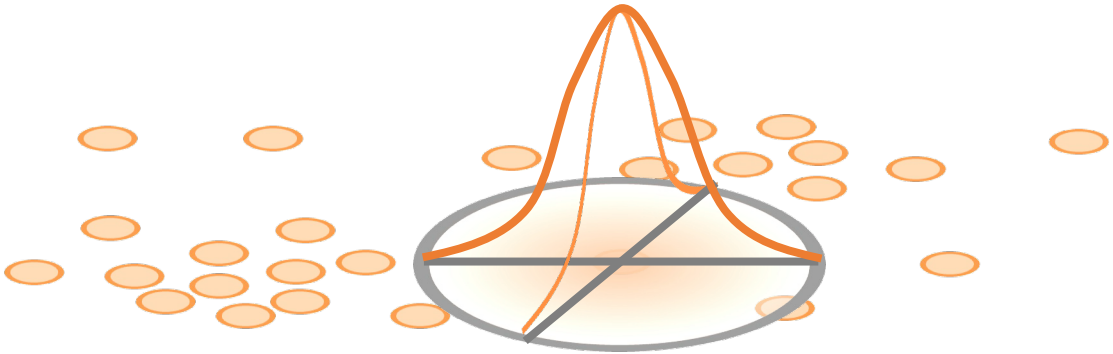
Particle-in-Cell – Particle2Grid

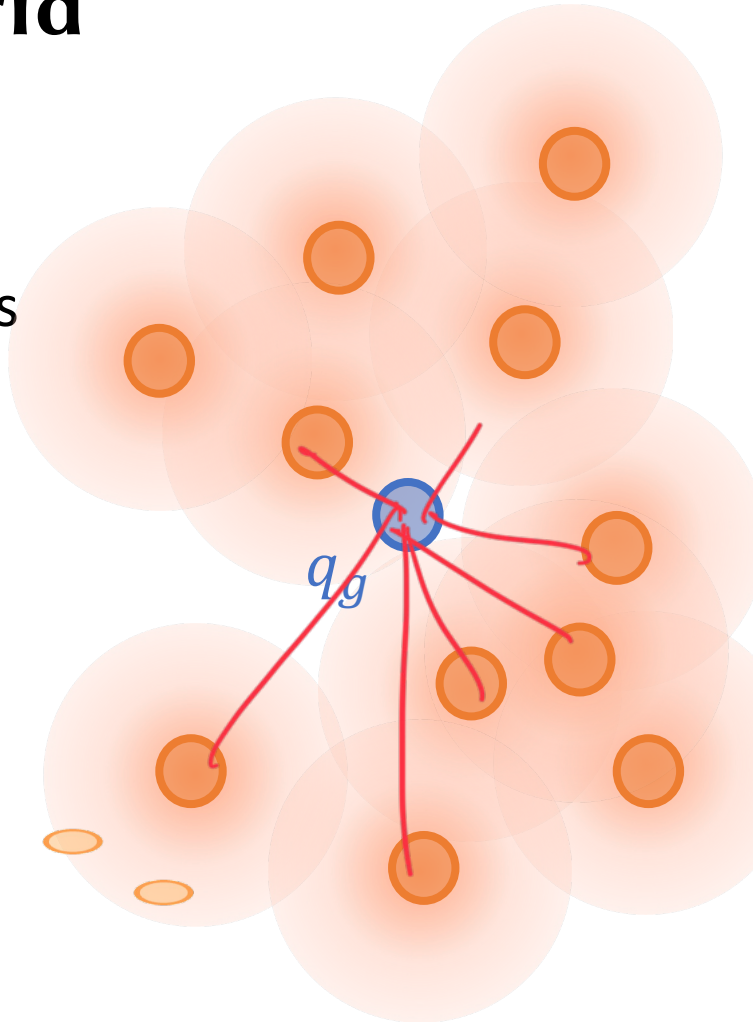
- can we also use a weighted sum?
  - yes! but… there's a lot of particles
  - idea: "blur" each particle



Particle-in-Cell – Particle2Grid

- can we also use a weighted sum?
  - yes! but… there's a lot of particles
  - idea: "blur" each particle
  - gaussian-like kernel





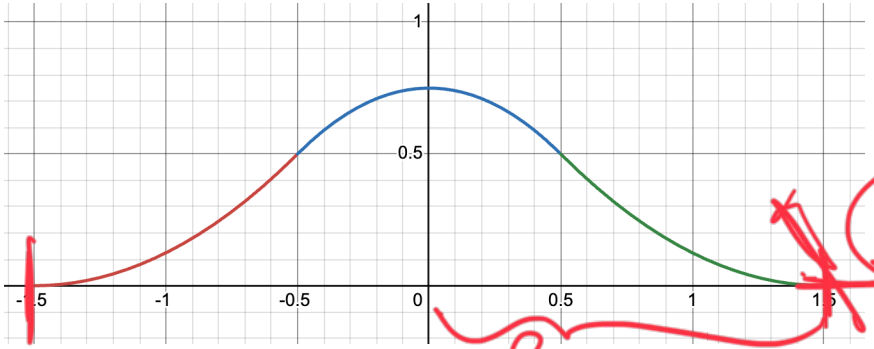
Particle-in-Cell – Particle2Grid

$$w_p(\delta x, \delta y) = QBS\left(\frac{\delta x}{h}\right) QBS\left(\frac{\delta y}{h}\right)$$

Particle-in-Cell – Particle2Grid

$$w_p(\delta x, \delta y) = QBS\left(\frac{\delta x}{h}\right) QBS\left(\frac{\delta y}{h}\right)$$

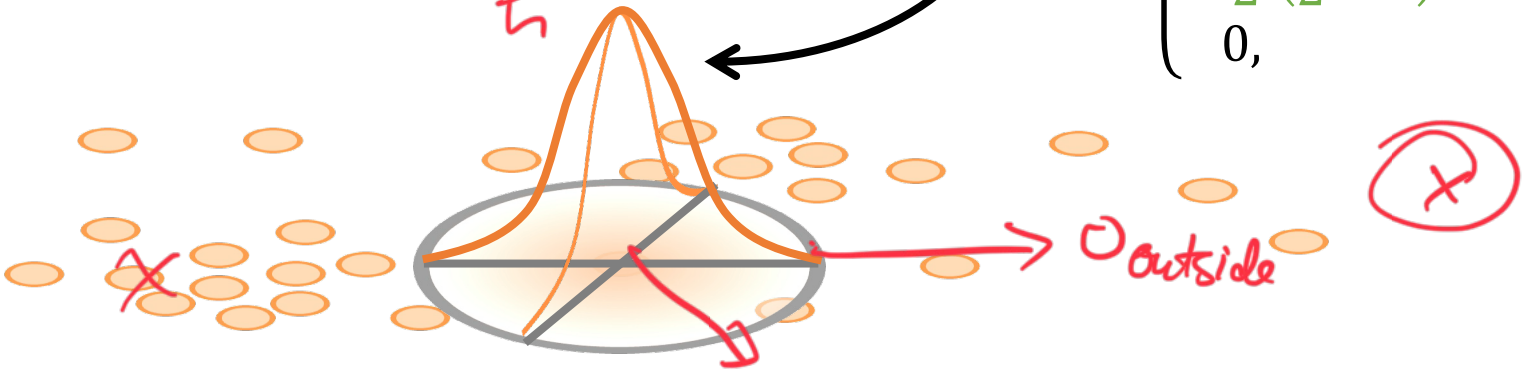
"Quadratic B-spline"



more radius h

$$QBS(r) = \begin{cases} \frac{1}{2}\left(r + \frac{3}{2}\right)^2, & -\frac{3}{2} \le r < -\frac{1}{2} \\ \frac{3}{4} - r^2, & -\frac{1}{2} \le r < \frac{1}{2} \\ \frac{1}{2}\left(\frac{3}{2} - r\right)^2, & \frac{1}{2} \le r < \frac{3}{2} \\ 0, & otherwise \end{cases}$$

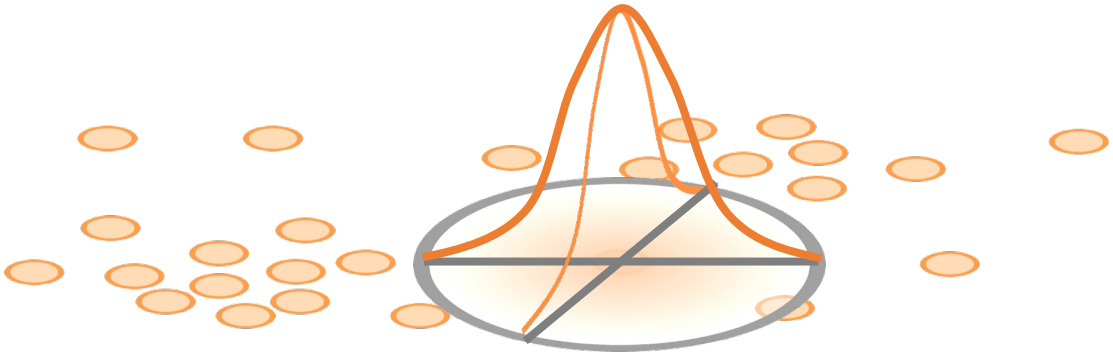
h

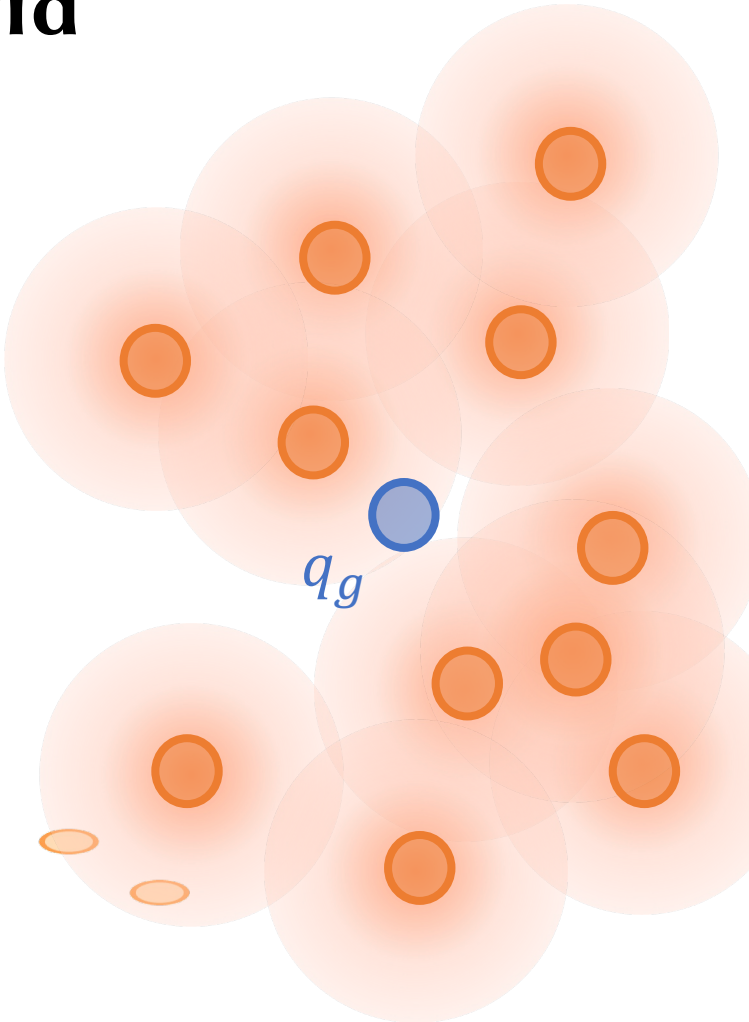


Particle-in-Cell – Particle2Grid

$$w_p(\delta x, \delta y) = QBS\left(\frac{\delta x}{h}\right) QBS\left(\frac{\delta y}{h}\right)$$

now that we have weights, just sum





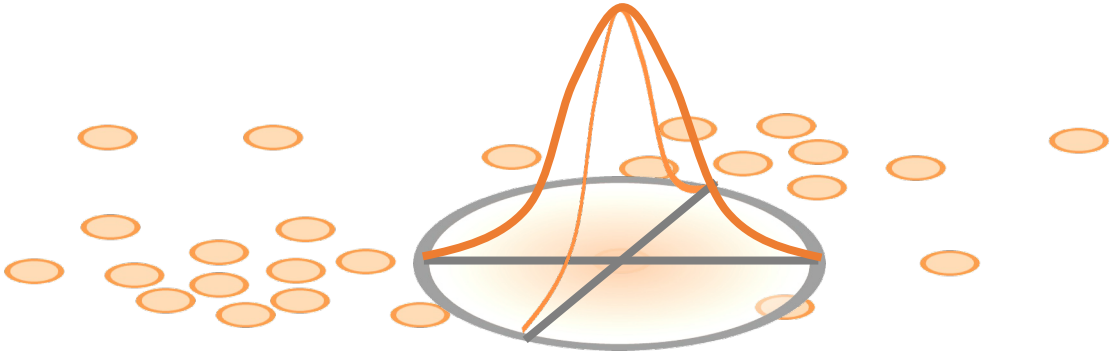
Particle-in-Cell – Particle2Grid

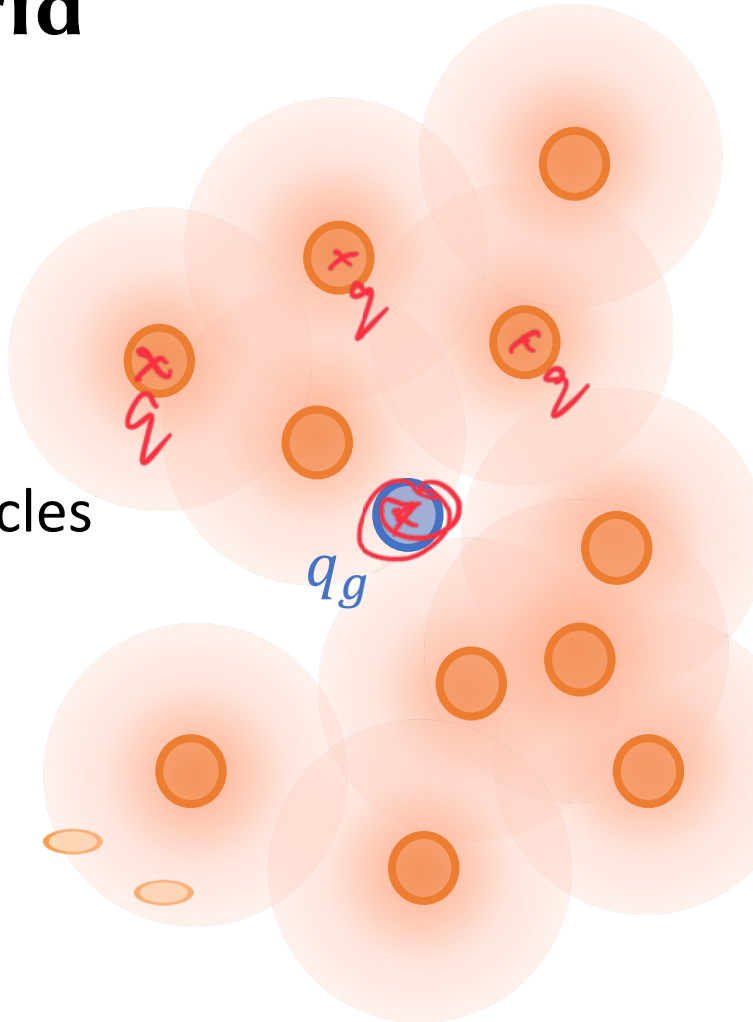
$$w_p(\delta x, \delta y) = QBS\left(\frac{\delta x}{h}\right) QBS\left(\frac{\delta y}{h}\right)$$

now that we have weights, just sum

$$q_p = \frac{\sum_n w_n q_n}{\sum_n w_n}$$

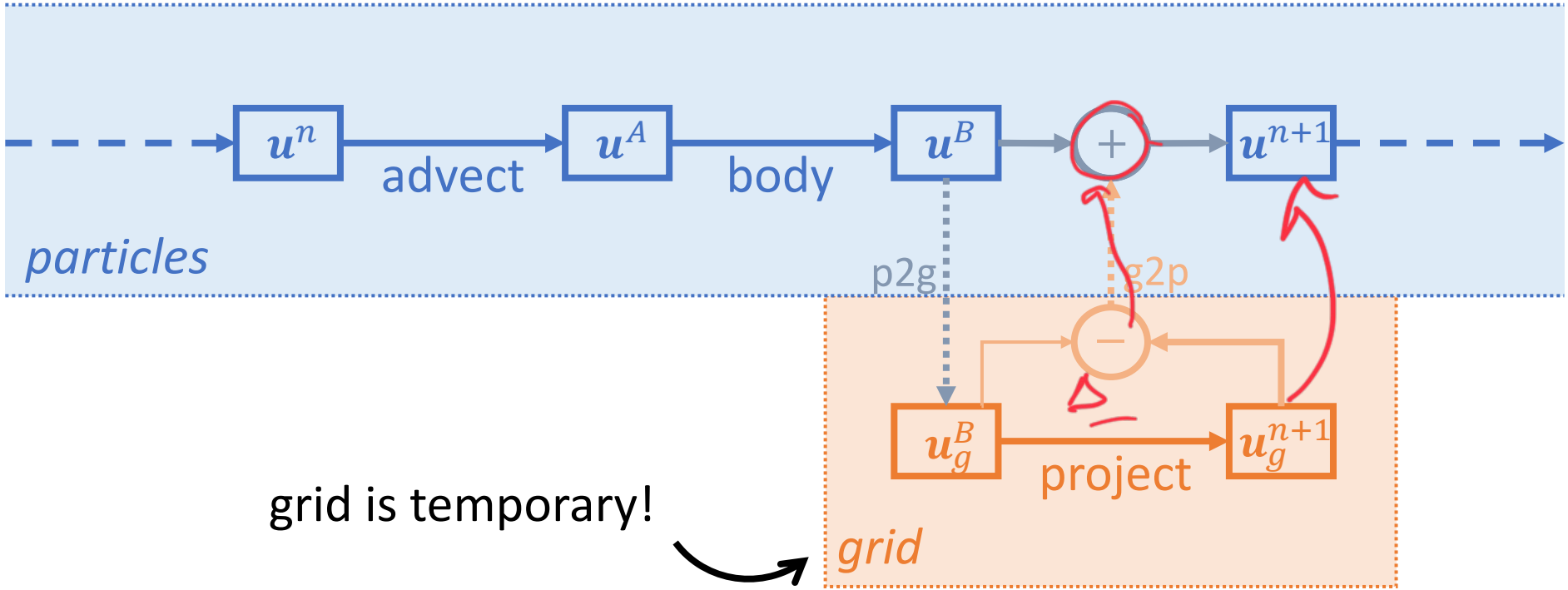
again, n are all particles inside $2h \times 2h$ box





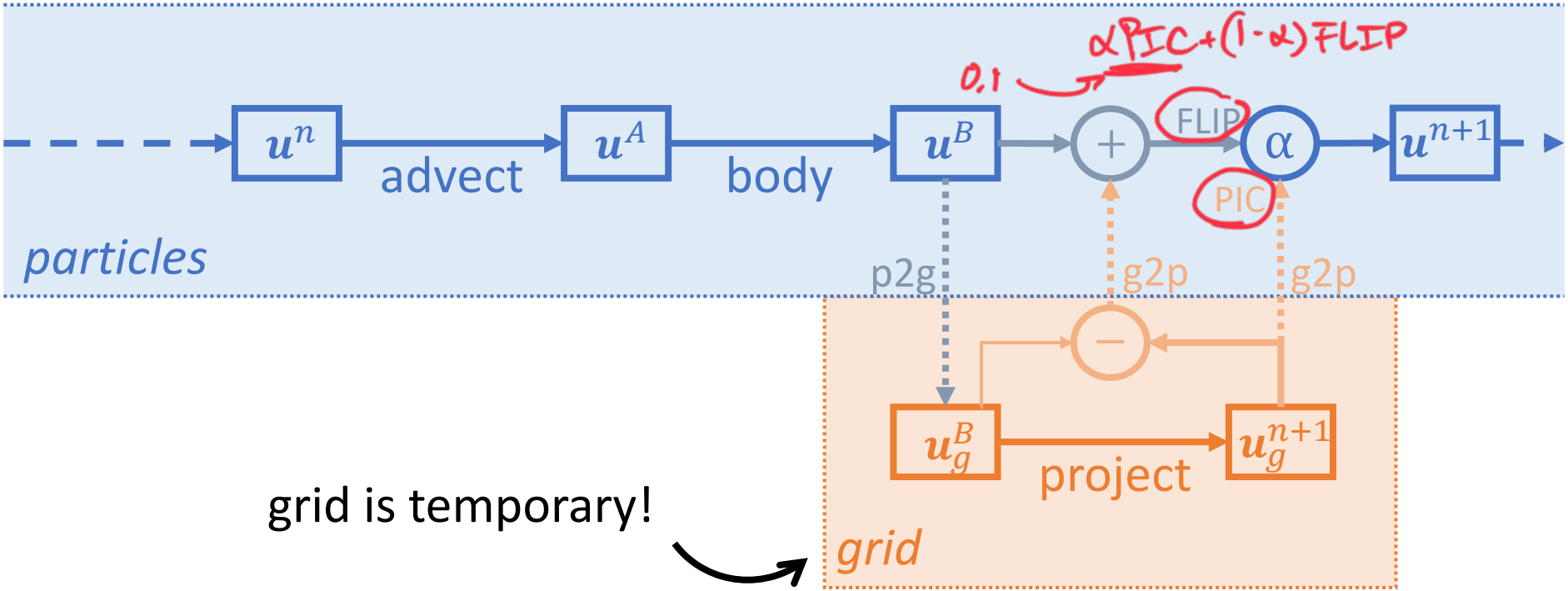
Blade Runner 2049 (2017)

A New Pipeline – Fluid-Implicit Particle



grid is temporary!

A New Pipeline – PIC-FLIP Blend



grid is temporary!