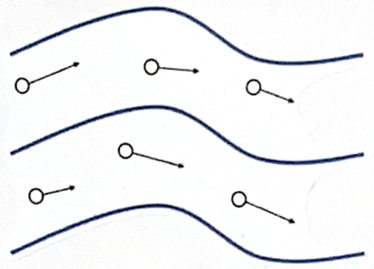
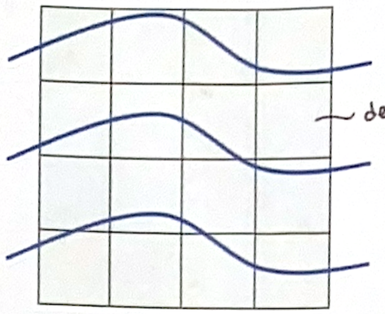
- Fluid Dynamic

Unlike other bodies, fluids exhibit highly volatile behaviors. As a result, it's difficult to come up with a single, efficient way for simulating all fluid effect.



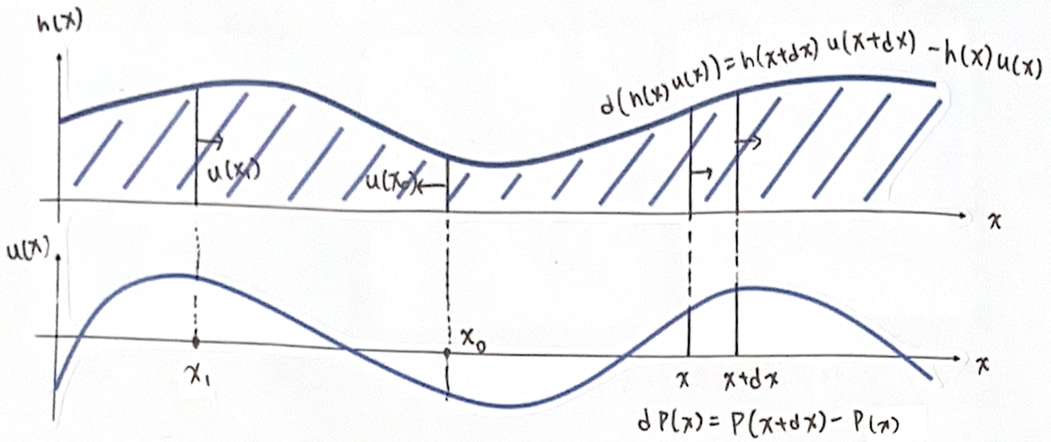
-※ Simulation Approaches

• Lagrangian	• Eulerian
(dynamic particles or mesh) Node movement carries physical quantities (mass, velocity ...)	(static grid or mesh) Grid/Mesh doesn't move. Store physical quantities change.
	

★ Height Field Model

· In 2D, a (1.5D) height field is a height function $h(x)$:

$$\frac{dh(x)}{dt} + \frac{d(h(x)u(x))}{dx} = 0$$



· The velocity is also a function $u(x)$: $\frac{du(x)}{dt} = -u(x)\frac{du(x)}{dx} - \frac{1}{\rho}\frac{dP(x)}{dx} + a(x)$

advection　　external

→ ignoring advection and external acceleration, we get a simple form

$\frac{du(x)}{dt} = -\frac{1}{\rho}\frac{dP(x)}{dx}$, where ρ is density and $P(x)$ is pressure.

- $u(x) \downarrow$, when $P(x) > 0$
- $u(x) \uparrow$, when $P(x) < 0$

For shallow wave, ($\frac{dh}{dx} \approx 0$, $\frac{dh}{dx}$ is small)

$$\begin{cases} \frac{dh}{dt} + \frac{d(hu)}{dx} = 0 \rightarrow \frac{dh}{dt} + u\frac{dh}{dx} + h\frac{du}{dx} = 0 \xrightarrow{dt} \frac{d^2h}{dt^2} + h\frac{du}{dxdt} = 0 \\ \frac{du}{dt} = -\frac{1}{\rho}\frac{dP}{dx} \xrightarrow{dx} \frac{du}{dtdx} = -\frac{1}{\rho}\frac{d^2P}{dx^2} \end{cases}$$

→ then eliminate u and formulate the shallow wave equation: $\frac{d^2h}{dt^2} = -\frac{1}{\rho}\frac{d^2P}{dx^2}$

→ $\frac{h_i^{n+1} - 2h_i^n + h_i^{n-1}}{\Delta t^2} = \frac{h_i^n}{\rho}\frac{P_{i+1}^n - 2P_i^n + P_{i-1}^n}{\Delta x^2}$ → i 表格點, n 表時間

→ $h_i^{n+1} = 2h_i^n - h_i^{n-1} + \frac{\Delta t^2 h_i^n}{\Delta x^2 \rho}(P_{i+1}^n - 2P_i^n + P_{i-1}^n)$

▲ Volume Preservation

The volume of fluid should stay the same when simulation. Suppose that $\sum_i h_i^n = \sum_i h_i^{n+1} = \sum_i h_i^{n-1} = V$.

By summation the difference equation,

$$\sum_i h_i^{n+1} = 2\sum_i h_i^n - \sum_i h_i^{n-1} + \sum_i \frac{\Delta t^2 h_i^n}{\Delta x^2 \rho}(P_{i+1}^n - 2P_i^n + P_{i-1}^n)$$

$$= V + \sum_i \underbrace{\frac{\Delta t^2 h_i^n}{\Delta x^2 \rho}(P_{i+1}^n - 2P_i^n + P_{i-1}^n)}_{\text{may not be zero}}$$

· Solution 1: modify difference equation to

$$h_i^{n+1} = 2h_i^n - h_i^{n-1} + \frac{\Delta t^2}{\Delta x^2 \rho}\left(\left(\frac{h_{i-1}^n + h_i^n}{2}\right)(P_{i-1}^n - P_i^n) + \left(\frac{h_{i+1}^n + h_i^n}{2}\right)(P_{i+1}^n - P_i^n)\right)$$

Sum → $\sum_i h_i^{n+1} = V + \frac{\Delta t^2}{\Delta x^2 \rho}\sum_i \left(\left(\frac{h_{i-1}^n + h_i^n}{2}\right)(P_{i-1}^n - P_i^n) + \left(\frac{h_{i+1}^n + h_i^n}{2}\right)(P_{i+1}^n - P_i^n)\right)$

This is because water exchanges between h_i^n and h_{i+1}^n

用 sum 展開後前後項會互相抵消

· Solution 2: assume h_i^n is constant, so

$$h_i^{n+1} = 2h_i^n - h_i^{n-1} + \frac{\Delta t^2 H}{\Delta x^2 \rho}(P_{i+1}^n - 2P_i^n + P_{i-1}^n)$$

Sum → $\sum_i h_i^{n+1} = V + \frac{\Delta t^2 H}{\Delta x^2 \rho}\sum_i \underbrace{\left((P_{i-1}^n - P_i^n) + (P_{i+1}^n - P_i)\right)}_{\text{must be zero}}$

▲ Pressure: related to the water height $P_i^n = \rho g h_i^n$

→ $h_i^{n+1} = 2h_i^n - h_i^{n-1} + \frac{\Delta t^2 H g}{\Delta x^2}(h_{i+1}^n - 2h_i^n + 2h_{i-1}^n)$

replace by constant α

▲ Viscosity: like damping, trying to slow down the wave.

→ $h_i^{n+1} = h_i^n + \underbrace{\beta}_{\text{Viscosity constant}}(\underbrace{h_i^n - h_i^{n-1}}_{\text{momentum}}) + \alpha(h_{i+1}^n - 2h_i^n + 2h_{i-1}^n)$

Shallow wave simulator

for every cell i

$h_i^{new} = h_i + \beta(h_i - h_i^{old}) + \alpha(h_{i-1} - h_i) + \alpha(h_{i+1} - h_i)$

for every cell

$h_i^{old} = h_i$

$h_i = h_i^{new}$

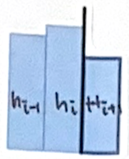
▲ Boundary Conditions

• Dirichlet B.C.: boundary height H_{i+1} is constant.

It is considered as an open boundary.

• Neumann B.C.: specifies the boundary derivative.

It is considered as a closed boundary



⇓

No water exchange through the boundary when $h' = 0$.

Shallow wave simulator with Neumann B.C.

for every cell i

$h_i^{new} = h_i + \beta(h_i - h_i^{old})$

if h_{i-1} exists, then $h_i^{new} = h_i^{new} + \alpha(h_{i-1} - h_i)$

if h_{i+1} exists, then $h_i^{new} = h_i^{new} + \alpha(h_{i+1} - h_i)$

for every cell i

$h_i^{old} = h_i$

$h_i = h_i^{new}$

Shallow wave simulator with Neumann B.C. in 3D

for every cell i, j

$h_{i,j}^{new} = h_{i,j} + \beta(h_{i,j} - h_{i,j}^{old})$

if $h_{i-1,j}$, $h_{i+1,j}$, $h_{i,j-1}$, $h_{i,j+1}$ exists, then $h_{i,j}^{new} = h_{i,j}^{new} + \alpha($ $h_{i-1,j}$, $h_{i+1,j}$, $h_{i,j-1}$, $h_{i,j+1}$ $- h_{i,j})$

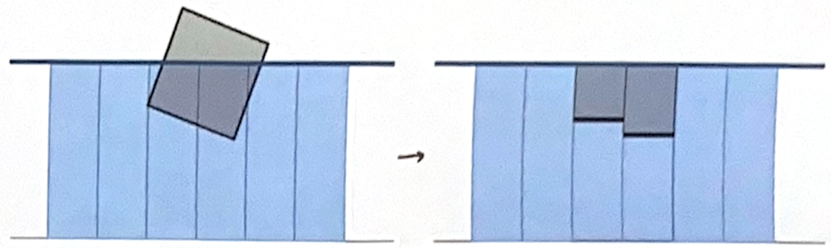
for every cell i, j

$h_{i,j}^{old} = h_{i,j}$

$h_{i,j} = h_{i,j}^{new}$

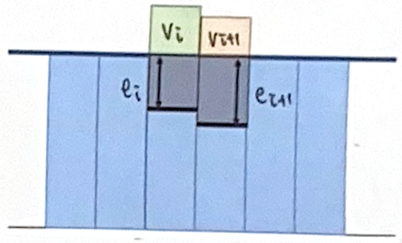
★ Two-Way Coupling

• The coupling between a solid and a liquid should be two way, i.e., liquid → solid and solid → liquid.



How to expel water out of the gray cell regions?

Set up a virtual height v_i, so that $h_i^{real\_new} = h_i - e_i$



$$\rightarrow \begin{cases} h_i - e_i = h_i + \beta(h_i - h_i^{old}) + \alpha(\underbrace{V_{i+1} + h_{i+1}}_{\text{modify by } V} + h_{i-1} - 2\underbrace{V_i - 2h_i}) = h_i^{new} + \alpha(V_{i+1} - 2V_i) \\ h_{i+1} - e_{i+1} = h_{i+1} + \beta(h_{i+1} - h_{i+1}^{old}) + \alpha(h_{i+2} + \underbrace{V_i + h_i}_{} - 2\underbrace{V_{i+1} - 2h_{i+1}}) = h_{i+1}^{new} + \alpha(V_i - 2V_{i+1}) \end{cases}$$

$$\rightarrow \begin{cases} 2V_i - V_{i+1} = \frac{1}{\alpha}(h_i^{new} - h_i + e_i) = b_i \\ -V_i + 2V_{i+1} = \frac{1}{\alpha}(h_{i+1}^{new} - h_{i+1} + e_{i+1}) = b_{i+1} \end{cases}$$

$$\rightarrow \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} V_i \\ V_{i+1} \end{bmatrix} = \begin{bmatrix} b_i \\ b_{i+1} \end{bmatrix}$$

→ more general in programing $\begin{bmatrix} 1 & & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & & 1 \end{bmatrix} \begin{bmatrix} V_{i-1} \\ V_i \\ V_{i+1} \\ V_{i+2} \end{bmatrix} = \begin{bmatrix} 0 \\ b_i \\ b_{i+1} \\ 0 \end{bmatrix}$

Shallow wave simulator with solid object in 3D

for every cell i,j

$$h_{i,j}^{new} = h_{i,j} + \beta(h_{i,j} - h_{i,j}^{old})$$

if $\begin{matrix} h_{i-1,j} \\ h_{i+1,j} \\ h_{i,j-1} \\ h_{i,j+1} \end{matrix}$ exists, then $h_{i,j}^{new} = h_{i,j}^{new} + \alpha\left(\begin{matrix} h_{i-1,j} \\ h_{i+1,j} \\ h_{i,j-1} \\ h_{i,j+1} \end{matrix} - h_{i,j}\right)$

// Get v

for every cell i,j

if in contact with solid

$$b_{i,j} = \frac{1}{\alpha}(h_{i,j}^{new} - h_{i,j} + e_{i,j})$$

$tag_{i,j} = true$

else

$v_{i,j} = 0$

$tag_{i,j} = false$

(mask)

PCG_solver (v, b, tag) → 共軛梯度法

for every cell i,j.

if $\begin{matrix} h_{i-1,j} \\ h_{i+1,j} \\ h_{i,j-1} \\ h_{i,j+1} \end{matrix}$ exists, then $h_{i,j}^{new} = h_{i,j}^{new} + \alpha\gamma\left(\begin{matrix} v_{i-1,j} \\ v_{i+1,j} \\ v_{i,j-1} \\ v_{i,j+1} \end{matrix} - v_{i,j}\right)$

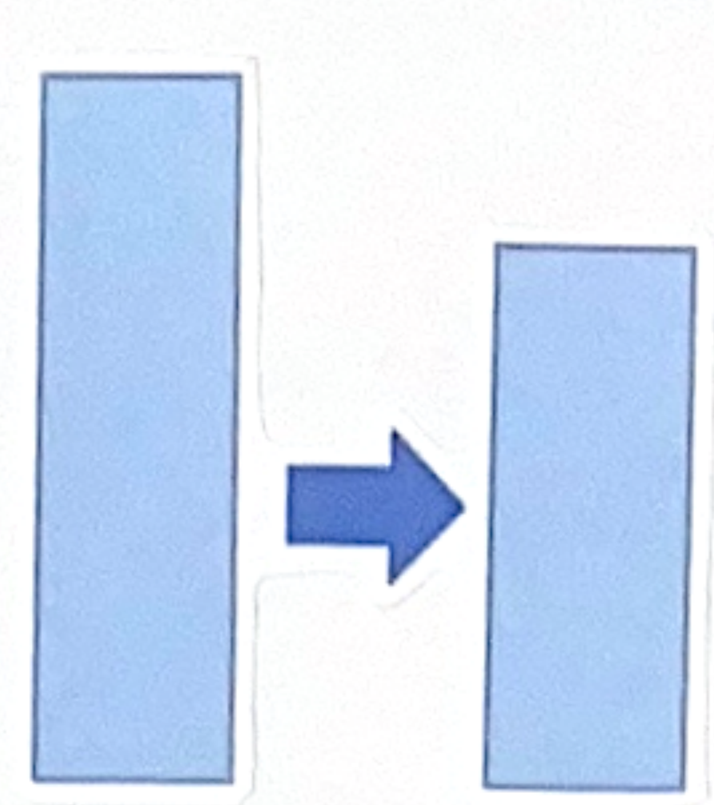
γ: relaxation factor, 玩家移動物体太快的話水浪會很大, because this is explicit method

$h_{i,j}^{old} = h_{i,j}$

$h_{i,j} = h_{i,j}^{new}$

- Rigid Body Update

Estimate the floating force by the actual water expelled in every column by the buoyancy explained by Archimede's principle.



In 2D: $f_i = \rho g \Delta x (h_i - h_i^{new})$

In 3D: $f_i = \rho g \Delta A (h_{i,j} - h_{i,j}^{new})$

再把物体下所有Δx, ΔA造成的浮力加總

* Incompressible, Viscous Navier-Stokes Equations

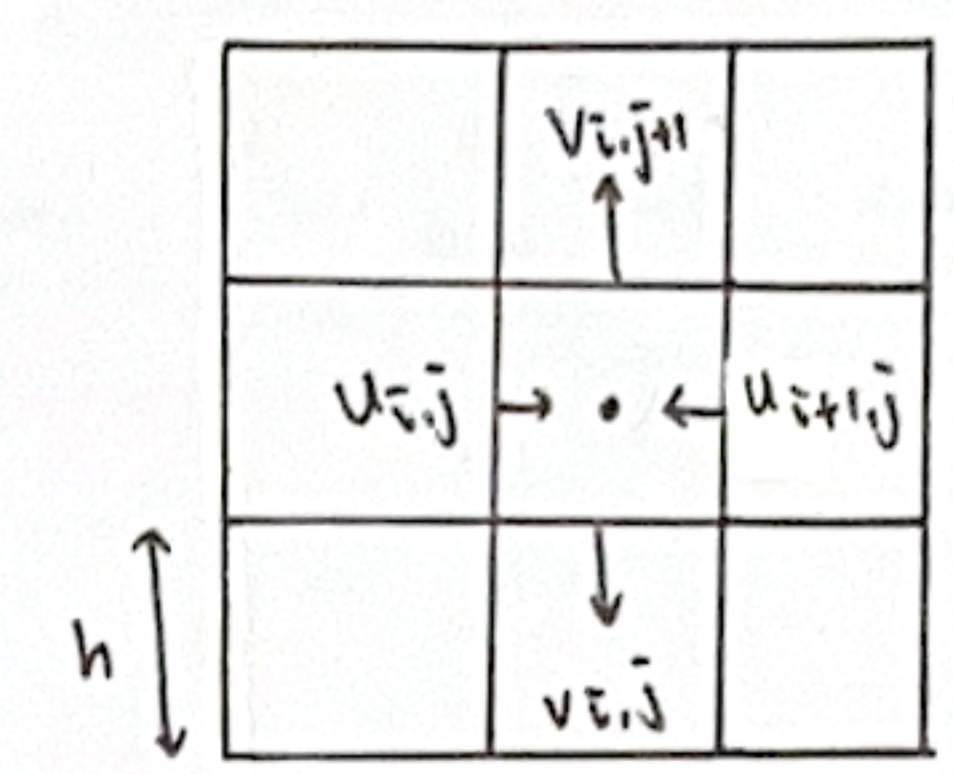
Equation formula:

- Incompressibility: $\nabla \cdot \vec{u} = 0$
- Momentum: $\frac{\partial \vec{u}}{\partial t} = \vec{g} - (\vec{u} \cdot \nabla)\vec{u} + \mu \nabla^2 \vec{u} - \nabla \vec{p}$

$\vec{g}$: external acceleration (ex. gravity); $(\vec{u} \cdot \nabla)\vec{u}$: advection; $\mu \nabla^2 \vec{u}$: (Laplacian) diffusion; $\nabla \vec{p}$: internal pressure

Using method of characteristics to solve a long partial differential equation in steps.

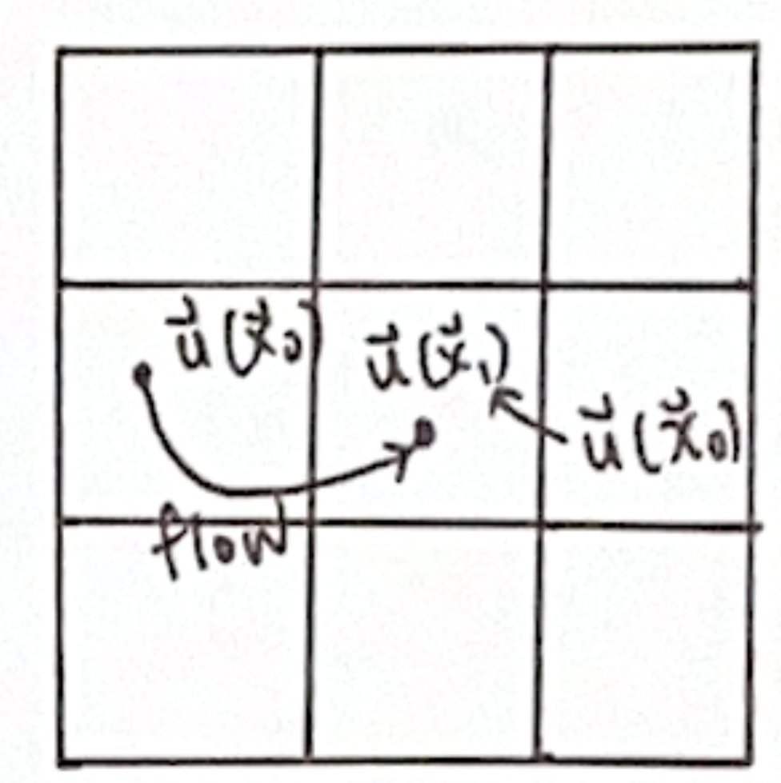
- Step 1: external acceleration



The update of $\vec{u}$ by $\frac{\partial \vec{u}}{\partial t} = \vec{g}$ is straightforward, just add acceleration to u and v.

Take gravity for example: $v_{i,j}^{new} = v_{i,j} + \Delta t g$

- Step 2: advection

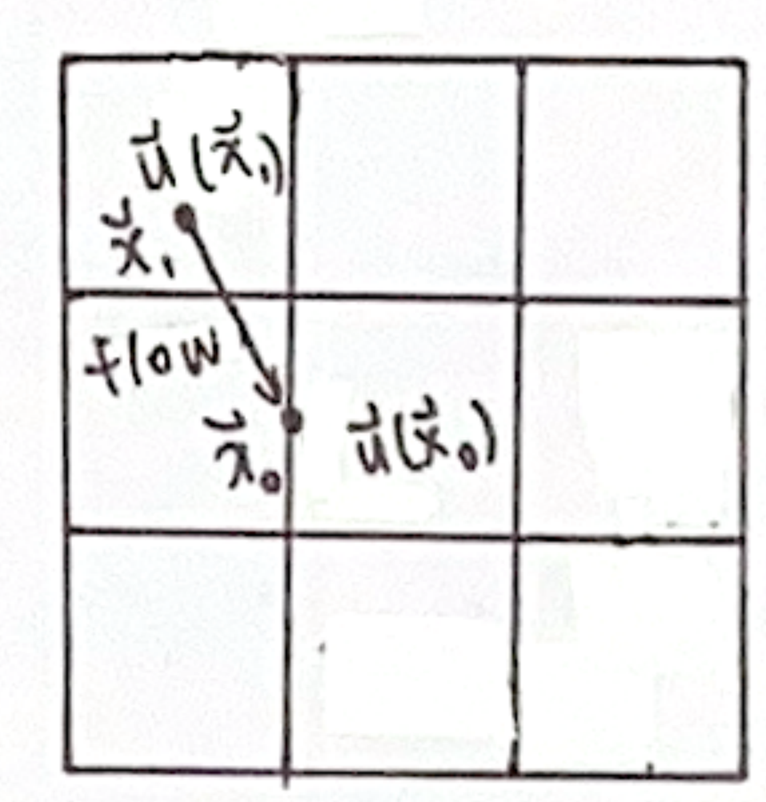


Update $\vec{u}$ by solving $\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} = -u\frac{\partial u}{\partial x} - v\frac{\partial v}{\partial y}$.

However, solving this in an Eularian way can be a source of instability.

To solve this problem, we come to realize that advection means to carry physical quantities by velocity. The solution is to trace a virtual particle backward over time. The Semi-Lagrangian Method:

a. One step:



Define $x_0 = (i-0.5, j)$

compute $u(x_0)$

$x_1 = x_0 - \Delta t\, \vec{u}(\vec{x}_0)$

compute $u(x_1)$

$u_{i,j}^{new} = u(x_1)$

Define $y_0 = (i, j-0.5)$

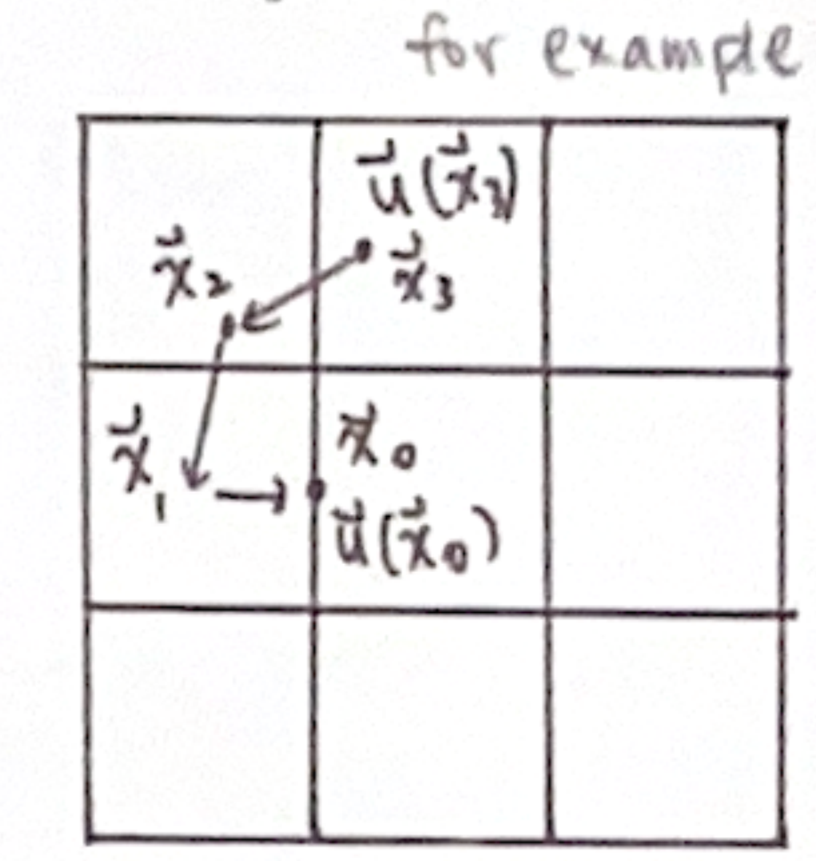
compute $v(y_0)$

$y_1 = y_0 - \Delta t\, v(y_0)$

compute $v(y_1)$

$v_{i,j}^{new} = v(y_1)$

b. Three steps: (take x-direction for example)



"→" means flow

Define $x_0 = (i-0.5, j)$

compute $u(x_0)$

$x_1 = x_0 - \frac{1}{3}\Delta t\, u(x_0)$

compute $u(x_1)$

$x_2 = x_1 - \frac{1}{3}\Delta t\, u(x_1)$

compute $u(x_2)$

$x_3 = x_2 - \frac{1}{3}\Delta t\, u(x_2)$

compute $u(x_3)$

$u_{i,j}^{new} = u(x_3)$

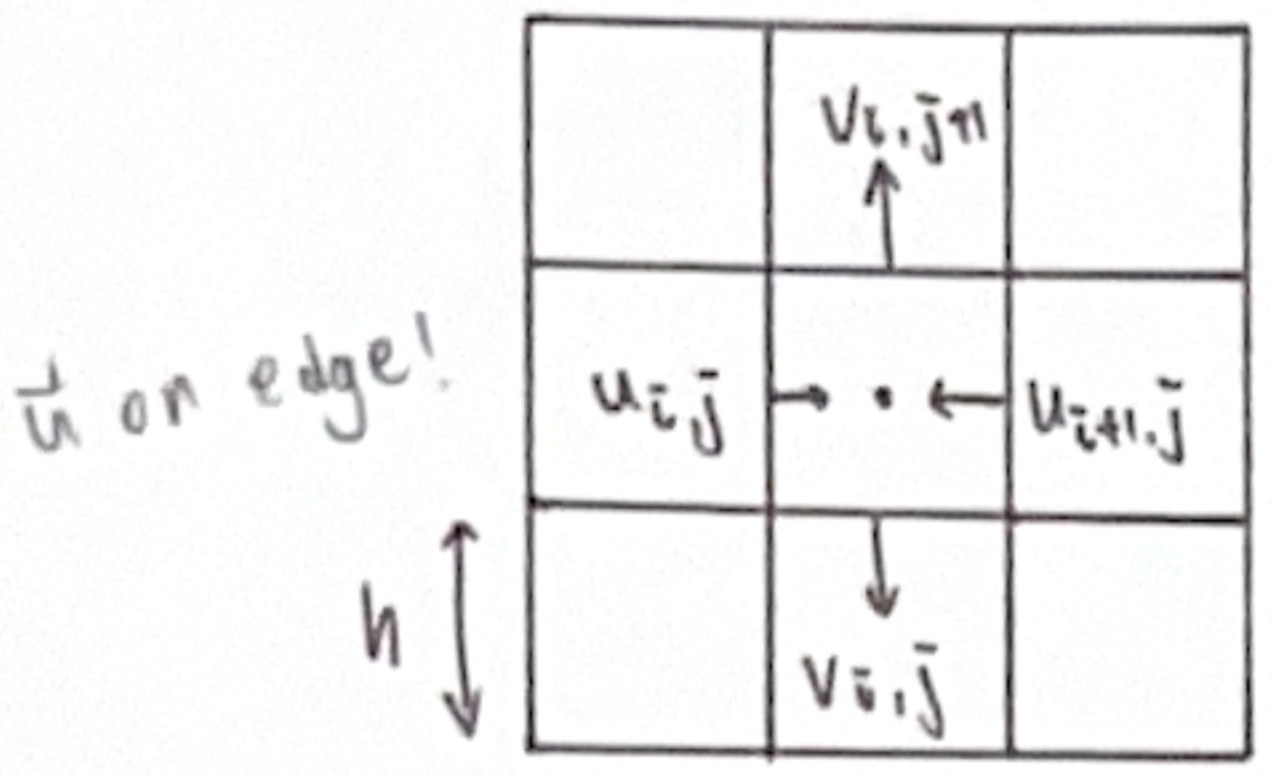
• Step 3: diffusion

Update $\vec{u}$ by solving $\frac{\partial \vec{u}}{\partial t} = \mu \Delta \vec{u}$

a. One step:

$$u^{new}_{i,j} = u_{i,j} + \mu \Delta t \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}}{h^2}$$

$$v^{new}_{i,j} = v_{i,j} + \mu \Delta t \frac{v_{i-1,j} + v_{i+1,j} + v_{i,j-1} + v_{i,j+1}}{h^2}$$



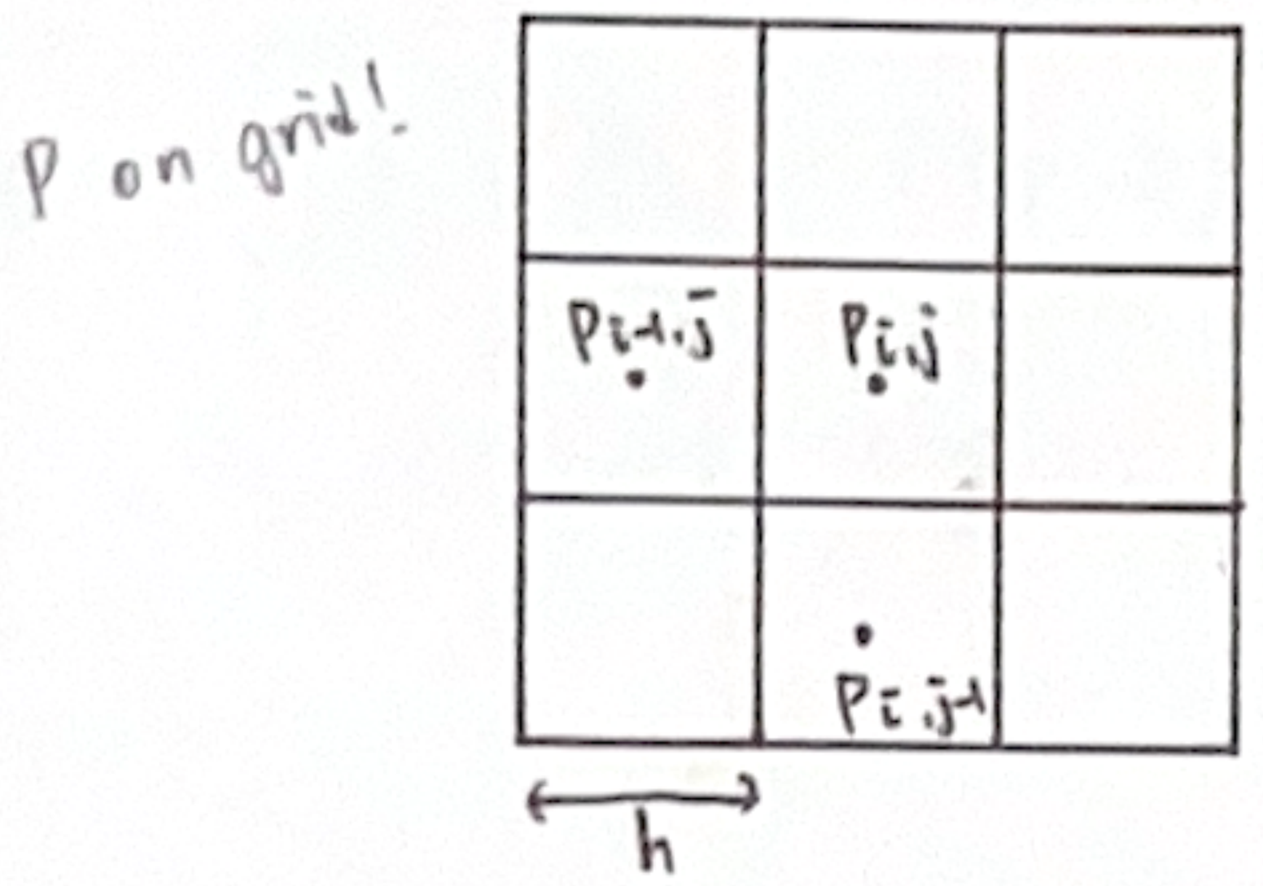
If $\mu \Delta t$ is large, the system would be unstable, so we could also use smaller sub-steps

→ b. Two steps: (take x-direction for example)

$$u^{temp}_{i,j} = u_{i,j} + \mu \frac{\Delta t}{2} \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}}{h^2}$$

$$u^{new}_{i,j} = u^{temp}_{i,j} + \mu \frac{\Delta t}{2} \frac{u^{temp}_{i-1,j} + u^{temp}_{i+1,j} + u^{temp}_{i,j-1} + u^{temp}_{i,j+1}}{h^2}$$

• Step 4: Pressure projection



Update $\vec{u}$ by solving $\frac{\partial \vec{u}}{\partial t} = -\nabla \vec{p}$

Staggered grid makes this easy:

$$\begin{cases} u^{new}_{i,j} = u_{i,j} - \frac{1}{h}(p_{i,j} - p_{i-1,j}) \\ v^{new}_{i,j} = v_{i,j} - \frac{1}{h}(p_{i,j} - p_{i,j-1}) \end{cases}$$

What is $\vec{p}$? It is the pressure caused by incompressibility, which means that after updating from this step, the velocity field should achieve

$$\nabla \cdot \vec{u}^{new} = 0$$

which means $u^{new}_{i+1,j} + v^{new}_{i,j+1} - u^{new}_{i,j} - v^{new}_{i,j} = 0$ (4 walls' velocity on a grid)

$$\rightarrow u_{i+1,j} - \frac{1}{h}(p_{i+1,j} - p_{i,j}) + v_{i,j+1} - \frac{1}{h}(p_{i,j+1} - p_{i,j})$$
$$- u_{i,j} + \frac{1}{h}(p_{i,j} - p_{i-1,j}) - v_{i,j} + \frac{1}{h}(p_{i,j} - p_{i,j-1}) = 0$$

$$\rightarrow 4p_{i,j} - p_{i+1,j} - p_{i,j+1} - p_{i-1,j} - p_{i,j-1} = h(u_{i,j} + v_{i,j} - u_{i+1,j} - v_{i,j+1})$$

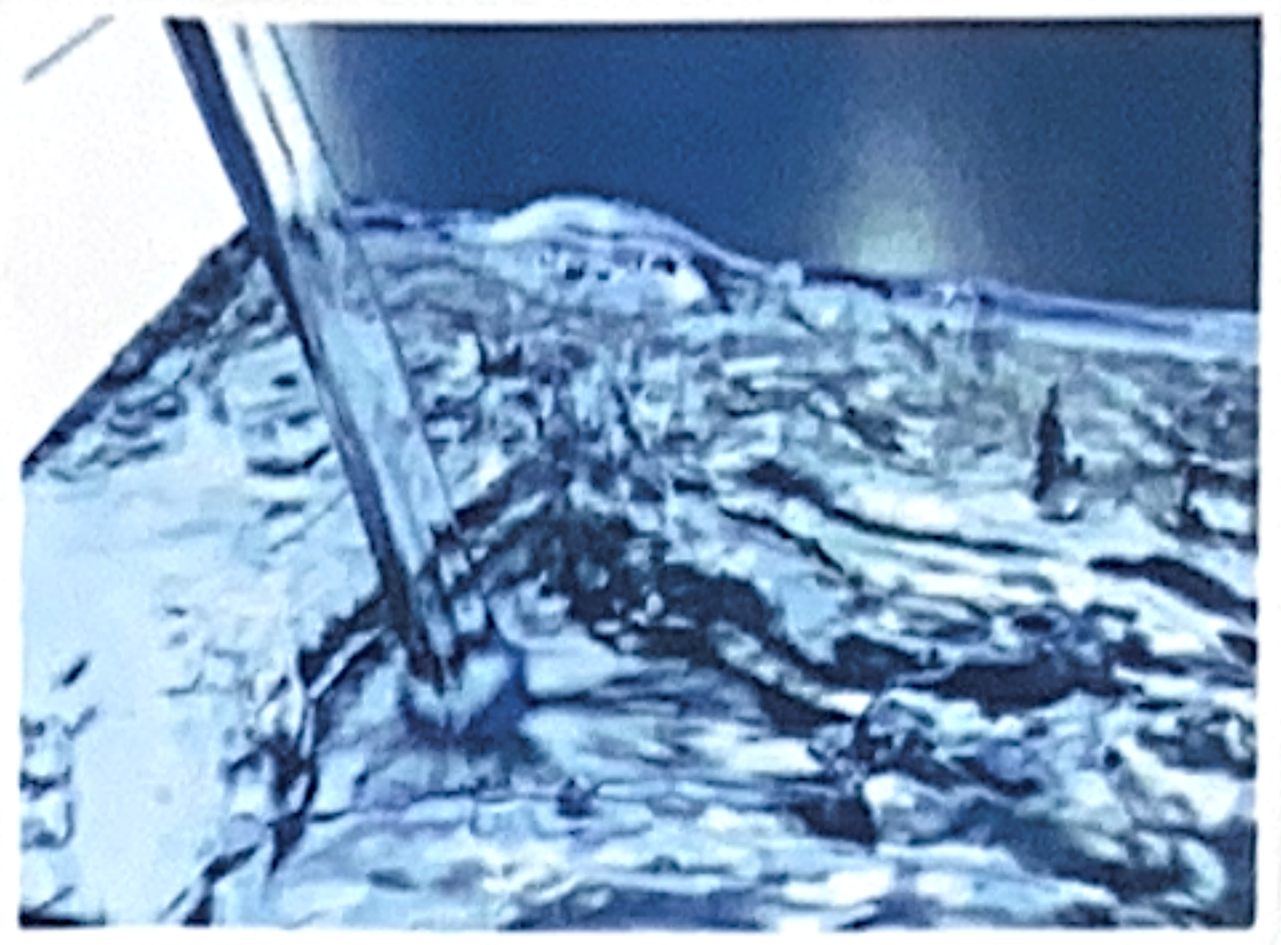
with Boundary conditions
- Dirichlet (open) $p_{i-1,j} = P$
- Neumann (close) $p_{i-1,j} = p_{i,j}$

→ Once we solve $\vec{p}$, update $\vec{u}$ and done.

• Air and Smoke

- Air simulation is done in two steps.
  1. update the flow (the velocity field) $\vec{u}$.
  2. use Semi-Lagrangian advect all of the other physical quantities, i.e., density, temperature ......
- Typically we use Dirichlet boundaries for an open space; Neumann boundaries for a container.
- It can be used to simulate underwater as well.

• Water Simulation



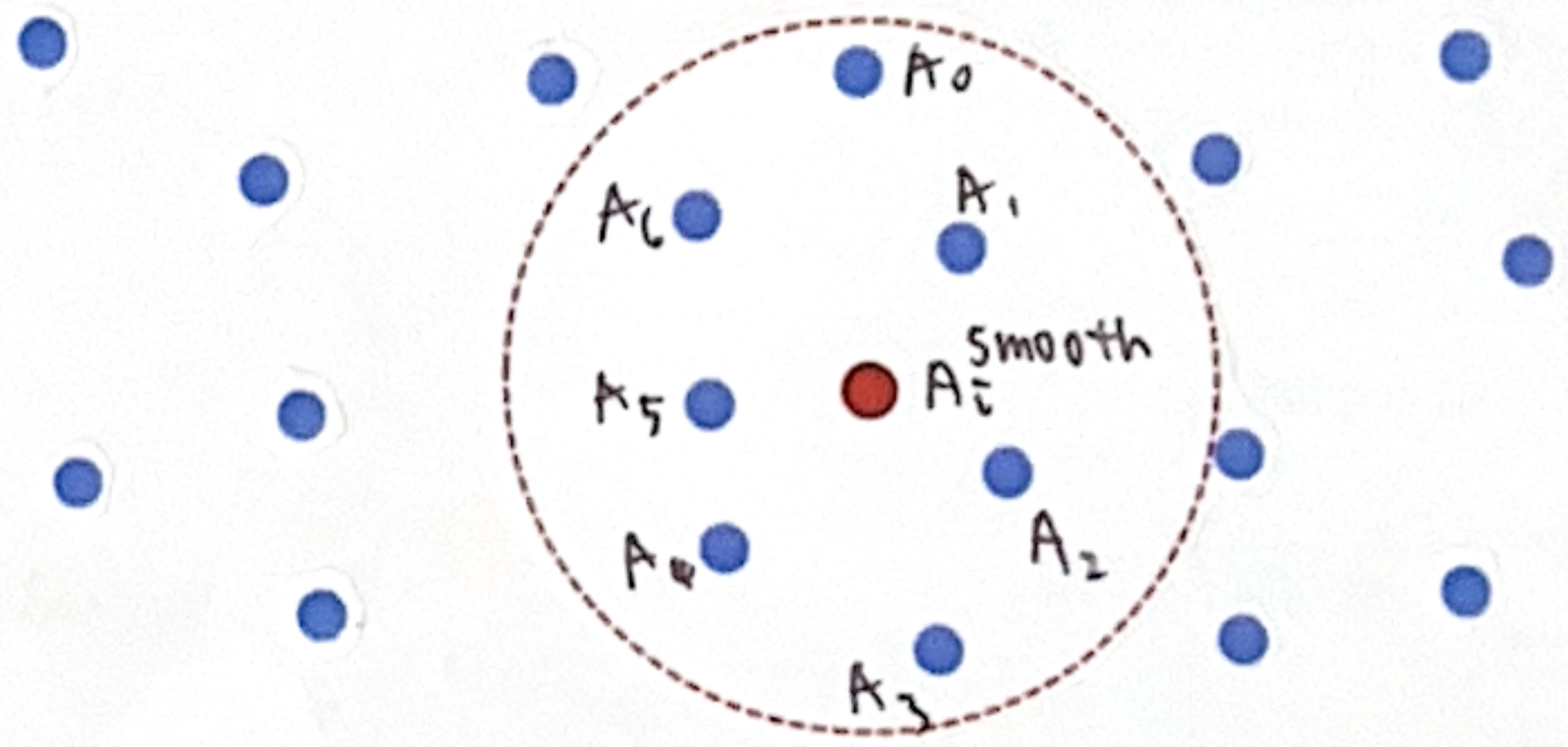
- Two presentations
  - Volume-of-fluid
  - signed distance function defined over the grid
- when implementing advection, both Semi-Lagragian and level set method cause volume loss. → need corrections.

• Smoothed Particle Hydrodynamics (SPH)

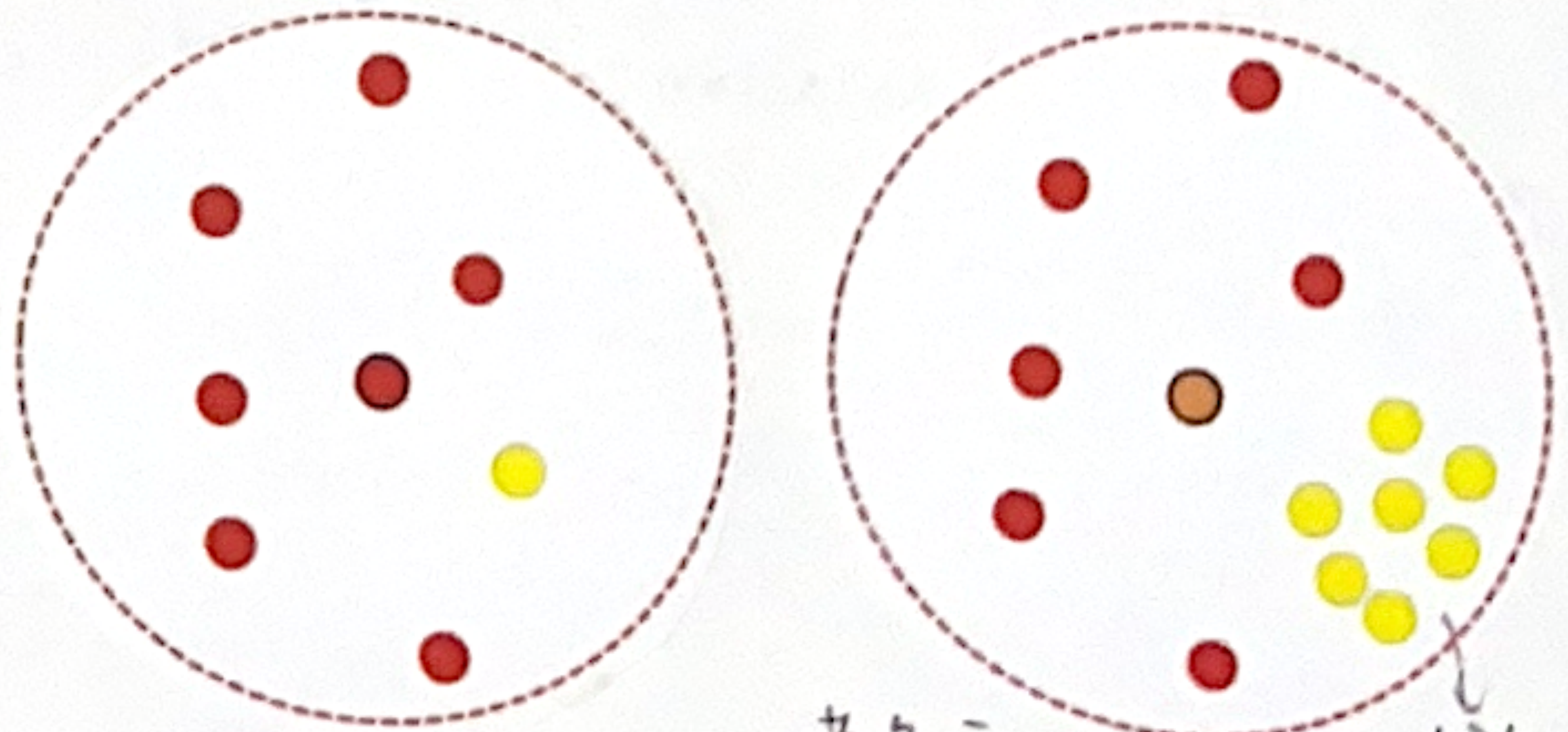
★ SPH Model

Consider a Lagrangian particle system: each water molecule is a particle with physical quantities attached, such as position $\vec{x}_i$, velocity $\vec{v}_i$, and mass m_i .... generally write as A_i.

• Simple model: $A_i^{smooth} = \frac{1}{n} \sum_j A_j$, for $\|\vec{x}_i - \vec{x}_j\| < R$



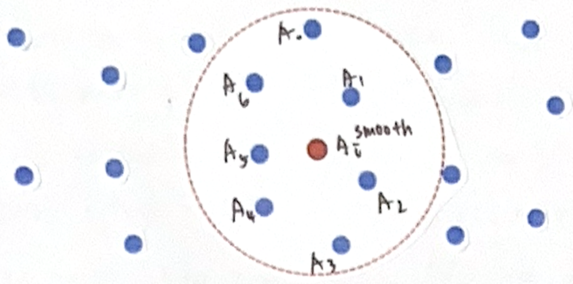
- problem: only considered the quantity of particles. Didn't consider the distribution of each particle.



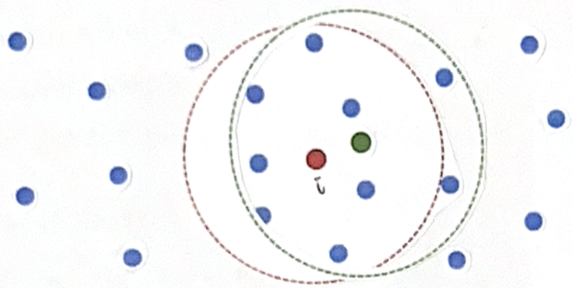
黃色空間小但影響大!

- Better Model: assume each particle has a volume V_j

$$\rightarrow A_i^{smooth} = \frac{1}{n}\sum_j V_j A_j \quad , \text{for } \|\vec{x}_i - \vec{x}_j\| < R$$



▲ Problem: not smooth when moving the selection area! ($\eta \rightarrow a$)



- Final Model: $A_i^{smooth} = \sum_j V_j A_j W_{ij}$ for $\|\vec{x}_i - \vec{x}_j\| < R$

where W_{ij} is the smoothing kernal. $\begin{cases} \|\vec{x}_i - \vec{x}_j\| \text{ large}, & W_{ij} \text{ small} \\ \|\vec{x}_i - \vec{x}_j\| \text{ small}, & W_{ij} \text{ large} \end{cases}$

▲ Volume of particle? $V_i = \frac{m_i}{\rho_i} \rightarrow V_i = \frac{m_i}{\rho_i^{smooth}} = \frac{m_i}{\sum_j m_j W_{ij}}$

(By the final model, $\rho_i^{smooth} = \sum_j V_j \rho_j W_{ij} = \sum_j m_j W_{ij}$)

Therefore, the actual solution is

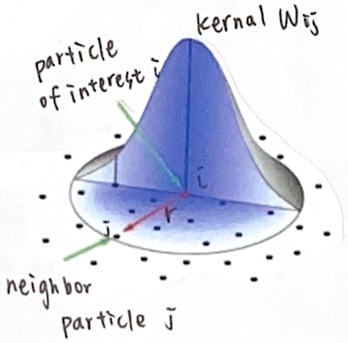
$$A_i^{smooth} = \sum_j \frac{m_j}{\sum_k m_k W_{jk}} A_j W_{ij}$$

- The derivative is easy to compute:

Gradient: $\nabla A_i^{smooth} = \sum_j V_j A_j \nabla W_{ij}$

Laplacian: $\nabla^2 A_i^{smooth} = \sum_j V_j A_j \nabla^2 W_{ij}$

- There are a lot of kernal model to calculate W_{ij}. For example →



* SPH-Based Fluids

Modeled fluid dynamics by applying three forces on particle i.

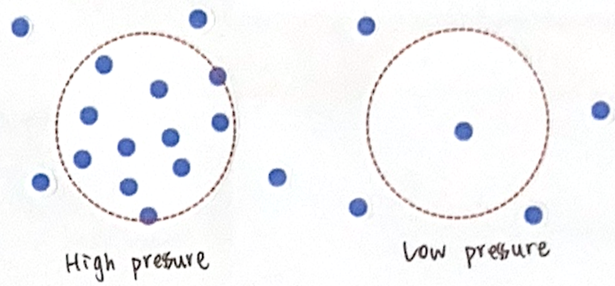
1. Gravity Force: $\vec{F}_i^{gravity} = m_i g$

2. Pressure Force: related to the density

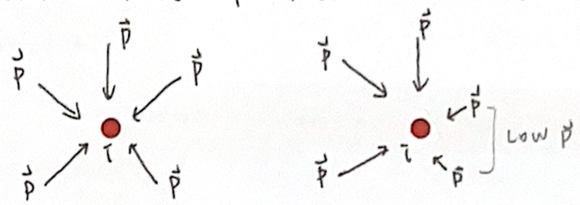
a. compute density of particle i: $\rho_i = \sum_j m_j W_{ij}$

b. convert it into pressure (by some empirical function):

(ex) $P_i = k\left(\left(\frac{\rho_i}{\rho_{constant}}\right)^7 - 1\right)$



c. Pressure force depends on the difference of pressure:

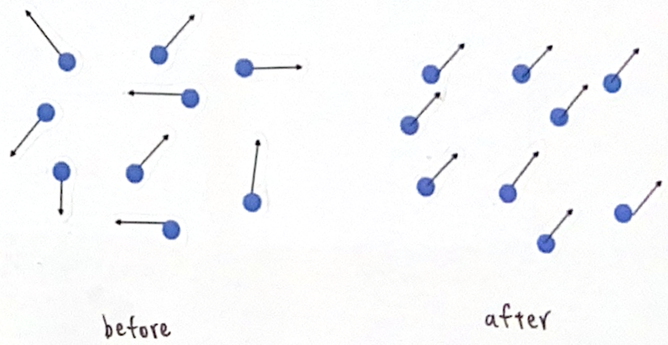


→ No pressure force! → Pressure force

Therefore, mathematically, the difference of pressure is gradient of pressure: $\vec{F}_i^{pressure} = -V_i \nabla_i P^{smooth} = -V_i \sum_j V_j P_j \nabla_i W_{ij}$

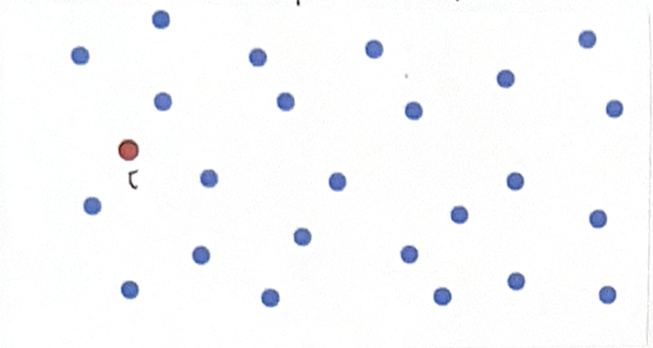
3. Viscosity Force: in order to minimize the difference between the particle velocity and the velocities of its neighbors.

$$\vec{F}_i^{viscosity} = -V m_i \nabla_i^2 \vec{v}^{smooth} = -V m_i \sum_j V_j \vec{v}_j \nabla_i^2 W_{ij}$$

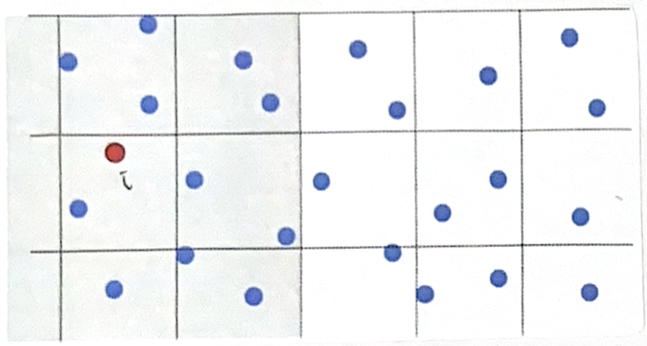


▲ Problem: exhaustive neighborhood search

To search over every particle pair, the time complexity is $O(N^2)$.

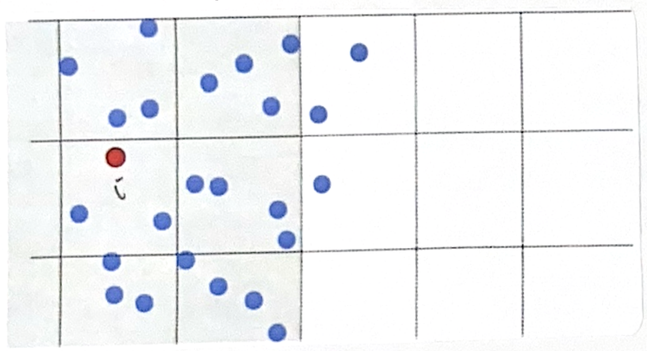


Solution: Spatial Partition



- Separate the space into cell
- Each cell stores the particles in it
- To find the neighborhood of i, just look at the surrounding.

If particles are not uniformly distributed:



- Octree
- Binary Spatial Partitioning tree

▲ Fluid display

Reconstruct the water surface from particles.

representation

typical visualization