Review:

Euler's equations after splitting:

A. ADVECTION:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = 0$$

Solution via Semi-Lagrangian method:

$$\vec{u}(\vec{x}, t^{n+1}) = \vec{u}\left(\vec{x} - \Delta t\, u(\vec{x}, t^n), t^n\right)$$

subject to boundary conditions:

$$\vec{u} \cdot \vec{\eta} = \vec{u}_{solid} \cdot \vec{\eta} \quad \text{at any solid interface}$$

Result of this step $\hat{u}(\vec{x}) := \vec{u}(x, t^n)$

B. BODY FORCES (e.g. gravity)

$$\frac{\partial \vec{u}}{\partial t} = g$$

Solution via Forward Euler:

$$\tilde{u}(\vec{x}) = \hat{u}(\vec{x}) + \Delta t\, \vec{g}$$

C. PRESSURE & INCOMPRESSIBILITY

$$\frac{\partial \vec{u}}{\partial t} + \frac{1}{\rho} \nabla p = 0$$

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

Solution:

→ Discretize time derivative $\frac{\partial \vec{u}}{\partial t}$ using forward difference

$$\frac{\vec{u}^{(n+1)} - \tilde{u}}{\Delta t} + \frac{1}{\rho}\nabla p = 0$$

→ Take the divergence of these quantities

$$\frac{\nabla\cdot\vec{u}^{(n+1)} - \nabla\cdot\tilde{u}}{\Delta t} + \frac{1}{\rho}\nabla\cdot(\nabla p) = 0$$

→ By the incompressibility condition, we dictate $\nabla\cdot\vec{u}^{(n+1)} = 0$, thus:

$$\frac{-\nabla\cdot\tilde{u}}{\Delta t} + \frac{1}{\rho}\nabla\cdot(\nabla p) = 0 \implies \boxed{\Delta p = \frac{\rho}{\Delta t}\nabla\cdot\tilde{u}} \qquad (1)$$

(where $\nabla\cdot(\nabla p) = \Delta p$, the Laplacian).

Discretization

Every discrete equation stemming from (1) "lives naturally" at cell centers (indexed by whole integers i,j,k), i.e.

$$(\Delta p)_{ijk} = \frac{\rho}{\Delta t}(\nabla\cdot\hat{u})_{ijk} \qquad \forall\ i,j,k \text{ s.t. } \vec{x}_{ijk} \in \Omega.$$

→ For the right-hand side we have:

$$(\nabla\cdot\tilde{u})_{ijk} = \frac{\tilde{u}_{i+1/2,jk} - \tilde{u}_{i-1/2,jk}}{\Delta x} + \frac{\tilde{v}_{i,j+1/2,k} - \tilde{v}_{i,j-1/2,k}}{\Delta y} + \frac{\tilde{w}_{ij,k+1/2} - \tilde{w}_{ij,k-1/2}}{\Delta z}$$

(for square cells, $\Delta x = \Delta y = \Delta z = h$)

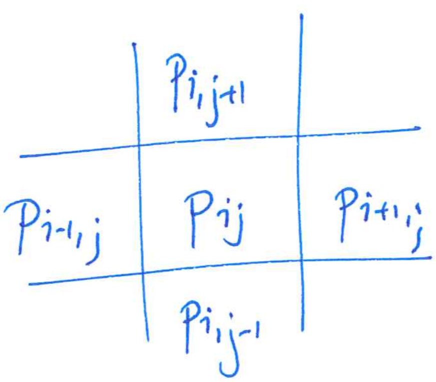
<u>Discretization of the Laplacian</u>

(We demonstrate in 2D, first)

$$(\Delta p)_{ij} = f_{ij} \quad \left(f_{ij} := \frac{\rho}{\Delta t}(\nabla\cdot\tilde{u})_{ij} \right)$$

→ For cells deeply interior

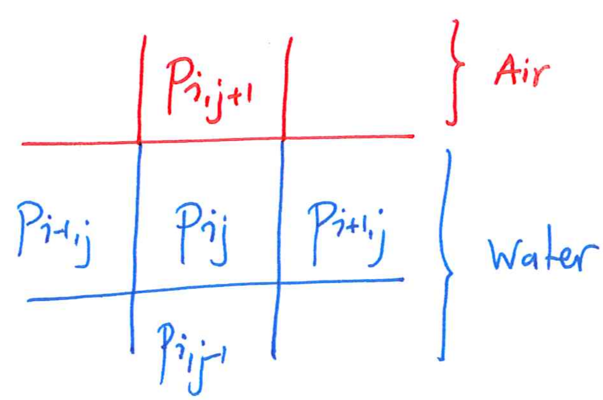
$$(\Delta p)_{ij} = \frac{p_{i-1,j} + p_{i+1,j} + p_{i,j-1} + p_{i,j+1} - 4p_{ij}}{h^2}$$



→ For cells near the air-fluid interface

We implement the boundary condition $\boxed{p=0}$ "Dirichlet" condition:

$$\frac{p_{i-1,j} + p_{i+1,j} + p_{i,j-1} + \overbrace{p_{i,j+1}}^{=0} - 4p_{ij}}{h^2} = f_{ij}$$



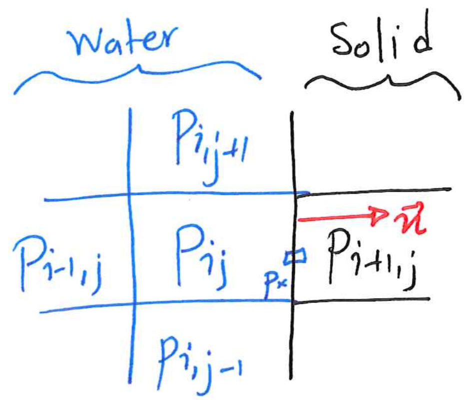
$$\frac{p_{i-1,j} + p_{i+1,j} + p_{i,j-1} - 4p_{ij}}{h^2} = f_{ij}$$

→ For cells near solid boundaries

We implement the boundary condition

$\boxed{\nabla p\cdot\vec{n} = 0}$ "Neumann" condition

Here, $\vec{n} = \begin{pmatrix}1\\0\end{pmatrix}$ e.g. $\nabla p\cdot\vec{n} = \begin{pmatrix}p_x\\p_y\end{pmatrix}\cdot\begin{pmatrix}1\\0\end{pmatrix} = p_x$



The condition $p_x = 0$ is discretized as:

$$\frac{p_{i+1,j} - p_{ij}}{h} = 0 \rightarrow \boxed{p_{i+1,j} - p_{ij} = 0}$$

The discrete equation becomes

$$\frac{p_{i-1,j} + p_{i,j-1} + p_{i,j+1} - 3p_{ij} + \overbrace{(p_{i+1,j} - p_{ij})}^{=0}}{h^2} = f_{ij}$$

Similarly, we handle the following cases:

→ 3D : $(\Delta p)_{ijk} = \dfrac{p_{i-1,jk} + p_{i+1,jk} + p_{i,j-1,k} + p_{i,j+1,k} + p_{ij,k-1} + p_{ij,k+1} - 6p_{ijk}}{h^2}$

→ Cells neighboring more than 1 solid-water or air-water interfaces (or mixtures of the 2).

Ultimately we arrive at a system of equations

$$L\underline{p} = \underline{f}$$

$\underline{p}$ = Vector of all pressures
$\underline{f}$ = Vector of all right-hand-side values

We can show that:

→ L is symmetric

→ L is negative definite

⇒ Thus, we can use CG to solve $(-L)\underline{p} = (-\underline{f})$ which is symmetric and POSITIVE definite

(In fact, we can use CG unmodified; it works for either purely positive or purely negative definite symmetric systems)

Notes:

→ When simulating Smoke, all Boundary conditions are Neumann; in this case 2 things happen:

* p is only computable up to a constant, i.e if $p_0(\vec{x})$ is a solution, $p_0(\vec{x})+c$ is also a solution
* $Lp=f$ is a singular system: It has a 1-dimensional nullspace, the vector of all-constant pressures i.e. if $p^\circ=(c,c,c,\dots,c)$, then $Lp^\circ=0$

CG can be minimally modified to handle this issue, by imposing a pre-determined average pressure $\bar{p}=\frac{1}{N}\sum p_{ijk}=\text{const.}$

→ CG can require several iterations to converge, even to elementary accuracy. As a rule of thumb if the fluid grid is of size $N\times N\times N$, at the bare minimum $2N$-$3N$ iterations are essential, and possibly $\sim 10N$-$20N$ may be required for somewhat acceptable convergence.

* CG can be accelerated with "preconditioning": If P is a good approximation of A^{-1} (but, cheaper to compute), preconditioning instead attempts to solve

$$Ax=b \rightsquigarrow \underbrace{PA}_{\tilde{A}}x=\underbrace{Pb}_{\tilde{b}} \rightsquigarrow \tilde{A}x=\tilde{b}$$

If $P\approx A^{-1}$, then $\tilde{A}\cong I$, and CG is significantly accelerated.

Popular preconditioner: Incomplete Cholesky

General Idea:

LU factorization $A = L \cdot U$ (L/U are lower/upper triangular).

Problem: Even if A is sparse, L, U can be <u>dense</u>

Incomplete factorization:

$A \approx \tilde{L} \cdot \tilde{U}$ (L/U also triangular)

$\tilde{L}$ & $\tilde{U}$ are only allowed to have non-zeros, where A had non-zeros in the first place

$\Rightarrow$ Trade-off between storage-complexity-accuracy.

Typical costs:

→ Construction of I.C. factorization ≈ 20x-200x un-preconditioned CG iterations.

→ Cost of preconditioned CG iteration ≈ 1.5x-5x un-preconditioned iterations

→ Reduction in # required iterations ≈ 5x-20x

<u>Smoke simulation</u>

→ 2 additional physical properties:

T: temperature

s: concentration of smoke particles

$\Rightarrow$ Stored at cell centers, averaged when necessary.

⇒ Buoyancy effects

⇝ Presence of smoke particles makes air "heavier"

⇝ Hot air rises, cold air moves down wards

⇒ Replace $\vec{g}$ (gravity) with Buoyancy term

$$f_b = \left(0, -\alpha s + \beta(T - T_{amb}), 0\right)$$

T = ambient temperature

α, β = constants (tune manually)

⇒ Temperature / Concentration Advection

Assumption: Temperature & concentration only move around, instead of explicitly diminishing (only approximate)

$$\frac{DT}{Dt} = 0 \qquad \frac{Ds}{Dt} = 0$$

Use e.g. Semi-Lagrangian advection.

⇒ Boundary Conditions

⇒ $\nabla \vec{p} \cdot \vec{n} = 0$ at any solid boundaries

$p = 0$ at any container openings (allowing smoke to escape and vanish).

⇒ $T = T_{object}$ at objects (can be hot!)

⇒ $\vec{u} = \vec{u}_{source}$ at smoke sources

⇒ $s = s_{source}$ at smoke sources ($s = 0$ inside non-source objects).

"Advanced" issues with smoke

The combined effects of averaging / Semi-Lagrangian / Large time steps / Large h may lead to smoke appearing

⇒ Blurred, smeared out

⇒ Less energetic than expected (less turbulent)

⇒ Vortices disappearing too fast

Remedies:

* Vorticity confinement: We start by measuring the vorticity:

$$\vec{\omega} = \nabla \times \vec{u} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \partial/\partial x & \partial/\partial y & \partial/\partial z \\ u & v & w \end{vmatrix}$$

* Determine "direction of closest vortex"

$$\vec{N} = \frac{\nabla \|\omega\|}{\| \nabla \|\omega\| \|}$$

* Add back a "rotation force" to increase the amplitude of the twirling effect

$$\vec{f}_{conf} = \varepsilon \cdot h \cdot (\vec{N} \times \vec{\omega})$$

⇒ As $h \to 0$, this extra "push" vanishes, and we get vorticity by means of better resolution.

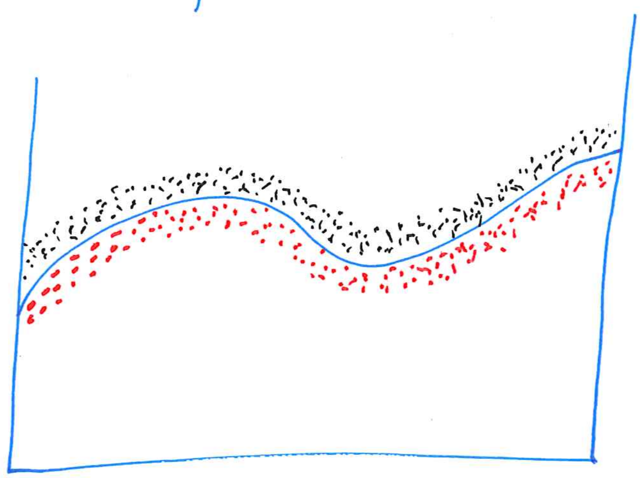
* Semi-Lagrangian often not accurate enough

⇒ Copy/Advect quantities by following <u>curved</u> flow lines.

Water simulation

Where is the water?

A. Use marker particles

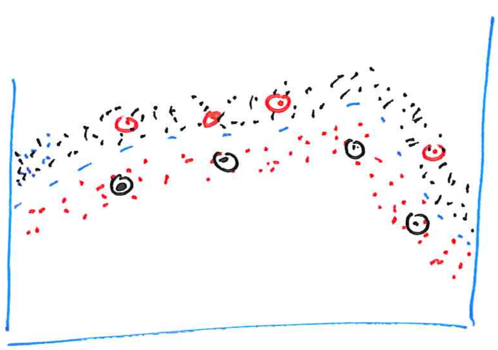


∴ = Air particles
∴ = Water particles

Particles are moved around with the velocity field, i.e.

$$\frac{d\vec{x}}{dt} = \vec{u}(\vec{x})$$

(Note: Use at least trapezoidal rule, or another 2+-order accurate method). May need to take smaller dt for particle advection, than water simulation. Afterwards, reconstruct surface by finding interface between air/water particles

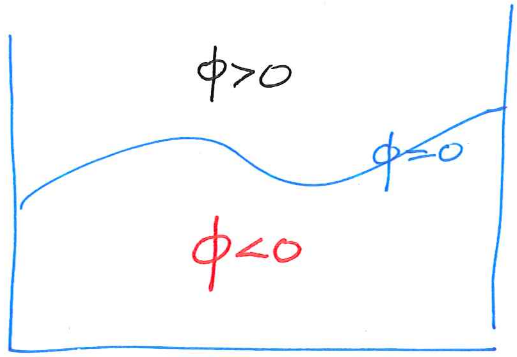


Red outliers in the air region can be rendered as spray droplets! (and then, removed)

Black outliers in the water region can be rendered as foam/micro-bubbles

Resample as necessary!

B. Use level-sets



Levelset values can be advected, too!

(Ideally, we want to move the zero levelset values, to find the new interface location, but we can advect the entire ϕ field with it!)

Advection equation $\frac{D\phi}{Dt}=0 \rightsquigarrow \boxed{\frac{\partial\phi}{\partial t}+\vec{u}\cdot\nabla\phi=0}$

Issues:

→ Even if $\phi^{(n)}(\vec{x})$ is a signed distance function, this property may be invalidated after advection!

The Fast Marching Method re-initializes the ϕ function, without moving the interface $\Gamma_0=\{x : \phi(x)=0\}$. such that the signed distance property is restored. Cost $=O(N\log N)$.

→ For advection algorithms, we may need $\vec{u}$ values where it is not normally present! (i.e. in the air region). If $\vec{x}$ is an air location we can extrapolate $\vec{u}_{ex}(\vec{x})=u(\vec{x}_{closest})$ where $\vec{x}_{closest}$ is the closest point on surface.

This extrapolation can be integrated into the fast marching method.