Representation of fluid state variables

The quantities that participate in Euler's equations

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

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

Consist of:

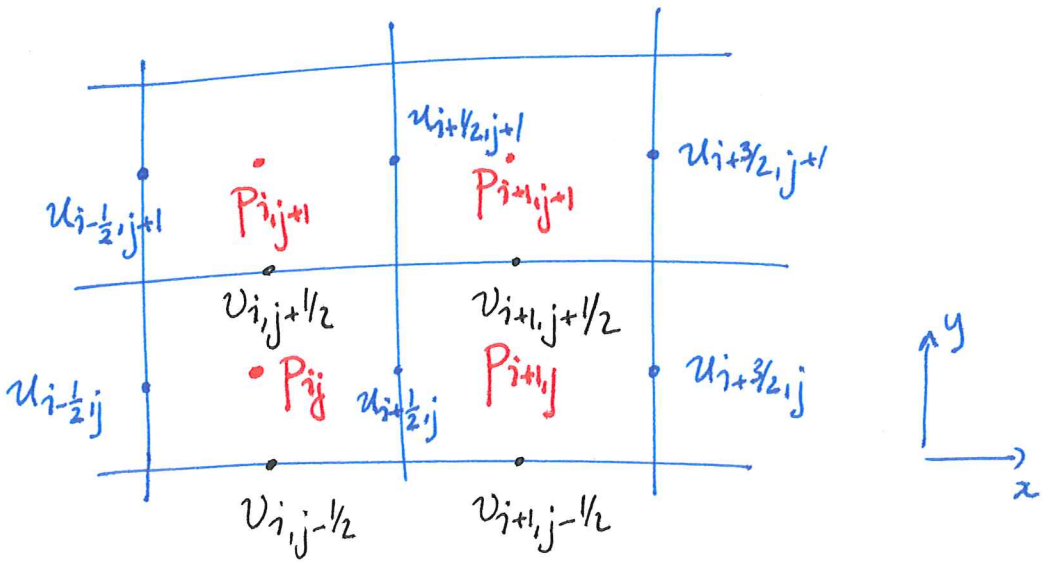
→ The vector valued velocity field $\vec{u}(x,y,z) =$

$$= \begin{pmatrix} u(x,y,z) \\ v(x,y,z) \\ w(x,y,z) \end{pmatrix}$$

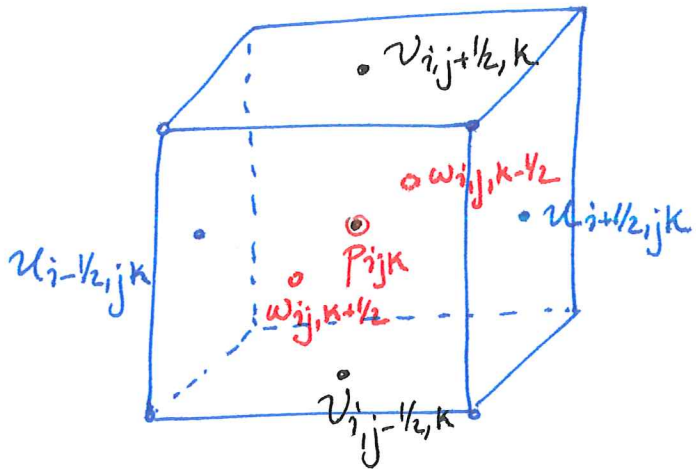
→ The pressure $p(x,y,z)$

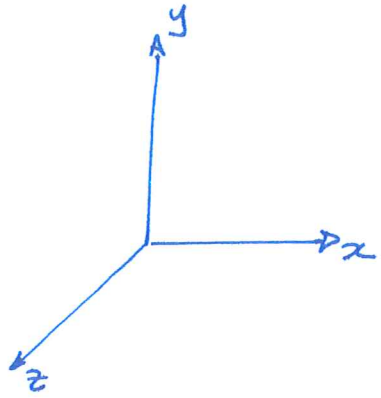
By far, the most popular grid-based representation has been the "MAC-grid" (named after the Marker-And-Cell method of Harlow & Welch, 1965). The unusual aspect is that each of the functions u, v, w, p gets sampled at a different set of locations:

in 2D



In 3D





The reasoning for adopting this type of discretization is often associated with the problems & artifacts caused by <u>not</u> doing so (for the purposes of incompressibility, more on that later). One reason that we can discuss immediately, though is: Derivatives are "naturally" defined at the exact places where they are needed.

Let's justify this... consider the following approximations for the derivative of f:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} + O(h) \quad \text{Forward difference}$$

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} + O(h) \quad \text{Backward difference}$$

$$f'(x) \approx \frac{f\left(x+\frac{h}{2}\right) - f\left(x-\frac{h}{2}\right)}{h} + O(h^2) \quad \text{Central difference}$$

I.e. Taking the difference of 2 function values at 2 points that are an $\frac{h}{2}$ distance apart, and then dividing by h is a 2nd order accurate approximation at the midpoint and only 1st order accurate for f' at either point.

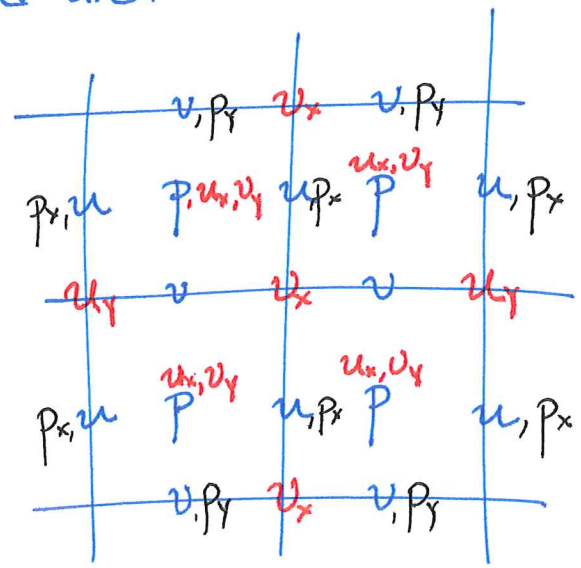
For multivariate functions we would write

$$\frac{\partial f}{\partial x}(x,y,z) = \frac{f(x+\frac{h}{2},y,z) - f(x-\frac{h}{2},y,z)}{h} + O(h^2)$$

$$\frac{\partial f}{\partial y}(x,y,z) = \frac{f(x,y+\frac{h}{2},z) - f(x,y-\frac{h}{2},z)}{h} + O(h^2)$$

$$\vdots$$

Thus it is reasonable to consider the midpoint of 2 data values (that get used in a finite-difference approximation of the derivative) to be the "natural" location of that derivative. Thus, the "natural locations" of derivatives in a 2D MAC grid are:



Thus, the "natural locations" of the following quantities are coincident:

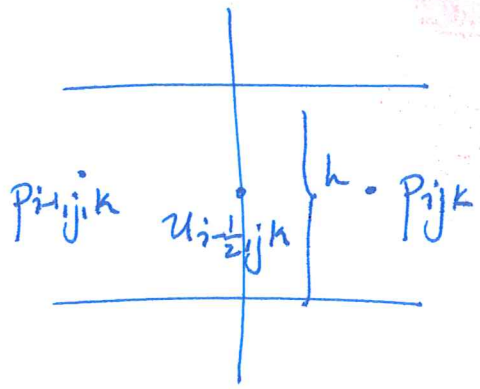
- u (or u_t) and p_x
- v (or v_t) and p_y
- w (or w_t) and p_z
- p and u_x, v_y, w_z

This makes it really convenient to discretize some of the (split) Euler equations, eg:

$$\vec{u}_t + \frac{1}{\rho}\nabla p = g \quad \text{, or} \quad \begin{cases} u_t + \frac{1}{\rho} p_x = g_{(x)} \rightarrow u_t, p_x \text{ collocated!} \\ v_t + \frac{1}{\rho} p_y = g_{(y)} \quad \text{etc} \\ w_t + \frac{1}{\rho} p_z = g_{(z)} \quad \text{etc} \end{cases}$$

Example of discretization:

$$u_t + \frac{1}{\rho} p_x = g_{(x)}$$



$$\Downarrow$$

$$\frac{u_{i-1/2,j,k}(t+\Delta t) - u_{i-1/2,j,k}(t)}{\Delta t} + \frac{1}{\rho}\frac{p_{i,j,k} - p_{i-1,j,k}}{h} = g_{(x)}$$

etc.

Similarly, for the incompressibility condition.

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

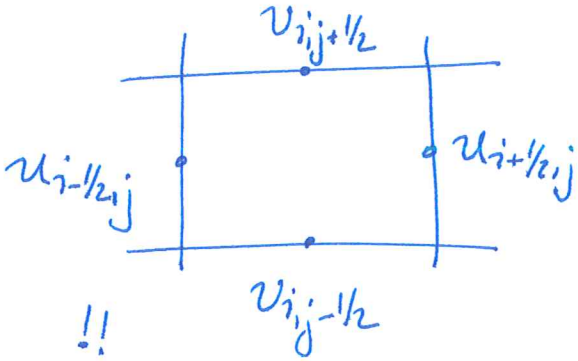
or

$$u_x + v_y = 0$$

or

$$\frac{u_{i+1/2,j} - u_{i-1/2,j}}{h} + \frac{v_{i,j+1/2} - v_{i-1/2,j}}{h} = 0$$

!!



We will revisit the staggered grid properties, later!

Advection

The 1st split equation is

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = 0 \quad \rightarrow \text{3 equations, really! (2 in 2D)}$$

Let's take the 1st of these equations: (in 2D)

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = 0.$$

A "straightforward" discretization would be:

$$\frac{u(x,y;t+\Delta t) - u(x,y;t)}{\Delta t} + u(x,y;t) \frac{u(x+\frac{h}{2},y) - u(x-\frac{h}{2},y)}{h} + v(x,y;t) \frac{u(x,y+h/2) - u(x,y-h/2)}{h}$$

The problem with this, is that this method is unconditionally unstable! (will blow up with any Δt)

Interestingly, the stable method that is commonly used in graphics leverages the duality between particle-based and location-based descriptions of advection:

$$0 = \frac{\partial u}{\partial t} + \vec{u}\cdot\nabla u = \frac{Du}{Dt} = \frac{d}{dt}\, u(\vec{x}(t), t)$$

... and, by approximating this last equation with a finite difference:

$$0 = \frac{Du}{Dt} \approx \frac{u(\vec{x}(t+\Delta t), t+\Delta t) - u(\vec{x}(t), t)}{\Delta t} = 0 \qquad (1)$$

As a last step, we approximate:

$$\vec{x}(t+\Delta t) \approx \vec{x}(t) + \Delta t\, \vec{v}(t),$$

or $$\vec{x}(t) = \vec{x}(t+\Delta t) - \Delta t\, \vec{v}(t) \qquad (2)$$

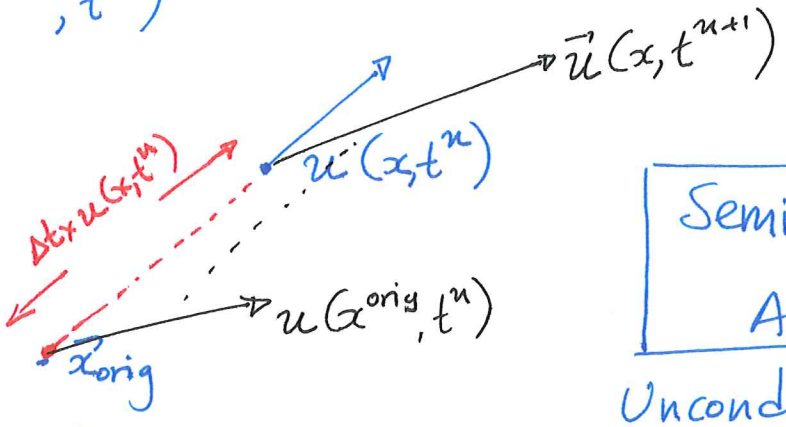
Using (1) & (2):

$$u(\vec{x}, t+\Delta t) \approx u(\vec{x} - \Delta t\, u(\vec{x}, t), t)$$

or $$\boxed{u(\vec{x}, t^{n+1}) \approx u(\vec{x} - \Delta t\, \vec{u}(x, t^n), t^n)}$$

In other words:

→ To generate the value of $u(x, t^{n+1})$, trace back the location $\vec{x}^{orig} = \vec{x} - \Delta t\, u(x, t^n)$, and read the value from $\vec{u}(\vec{x}^{orig}, t^n)$

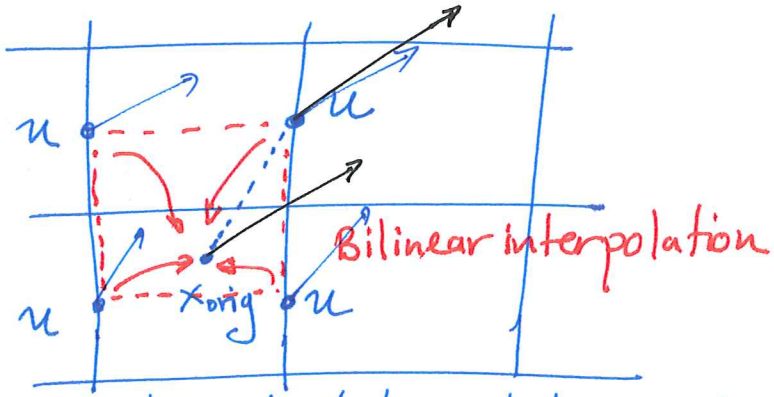


Semi-Lagrangian Advection

Unconditionally Stable!

Implementation notes:

→ Since we store velocities on a staggered grid, the location $\vec{x}^{orig}$ need not coincide with a location where velocities reside ⇒ Use bilinear (2D) or trilinear (3D) interpolation



→ In practice, we do not take arbitrarily large Δt (although stability allows us to). To generate good visual results, we typically impose $\|\Delta t\, u(x,t)\| < k \cdot h$ ($k = 1..3$ typically).

or $\Delta t < \frac{k \cdot h}{\|u\|}$ (i.e. Semi-Lagrangian ray travels k-cells max).

Pressure & Incompressibility

As stated before, we split the Euler equations as follows:

A. ADVECTION

$$\left(\frac{D\vec{u}}{Dt} = \right) \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = 0$$

⇒ Using Semi-Lagrangian method:

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

↳ Intermediate velocity #1.

B. GRAVITY (or body forces)

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

⇒ Using Forward Euler:

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

C. PRESSURE & INCOMPRESSIBILITY

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

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

Input: $\tilde{u}$, $p^{(n)}$

Output: $\vec{u}^{(n+1)}$, $p^{(n+1)}$

Note: As a result of the advection & body force step, the intermediate velocity might have lost its incompressible nature (i.e. $\nabla \cdot \tilde{u} \neq 0$), since we never enforced this constraint. This gives us the opportunity to use this last step to "correct" any volume-changing aspects of the velocity field and enforce $\nabla \cdot \vec{u}^{(n+1)} = 0$!

(Essentially, we use the pressure to generate "impulses" that jolt the velocity field back to an incompressible state)

This approach, called a pressure projection, or divergence-free projection is described in the discretization:

$$\frac{\partial \vec{u}}{\partial t} + \frac{1}{\rho}\nabla p = 0 \quad \rightsquigarrow \quad \frac{\vec{u}^{(n+1)} - \tilde{u}}{\Delta t} + \frac{1}{\rho}\nabla p^{(n+1)} = 0 \qquad (3)$$

$$\nabla \cdot \vec{u} = 0 \quad \rightsquigarrow \quad \nabla \cdot \vec{u}^{(n+1)} = 0 \qquad (4)$$

We manipulate this, as follows:

(3) $\vec{u}^{(n+1)} - \tilde{u} = -\frac{\Delta t}{\rho}\nabla p^{(n+1)}$ Take div of both sides! $\Rightarrow$

$$\Rightarrow \underbrace{\nabla \cdot \vec{u}^{(n+1)}}_{=0 \ (4)} - \nabla \cdot \tilde{u} = -\frac{\Delta t}{\rho}\nabla \cdot \left(\nabla p^{(n+1)}\right)$$

$$\Rightarrow \nabla \cdot \left(\nabla p^{(n+1)}\right) = \frac{\rho}{\Delta t}\nabla \cdot \tilde{u}$$

The quantity on the LHS is, in fact

$$\nabla \cdot (\nabla p) = \nabla \cdot \begin{pmatrix} \partial p/\partial x \\ \partial p/\partial y \\ \partial p/\partial z \end{pmatrix} =$$

$$= \frac{\partial}{\partial x}\left(\frac{\partial p}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\partial p}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\partial p}{\partial z}\right)$$

$$= \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right) p$$

$$= \Delta p.$$

Where Δ is the Laplacian operator $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$

Likewise, the RHS becomes:

$$\frac{\rho}{\Delta t} \nabla \cdot \tilde{u} = \frac{\rho}{\Delta t}\left(\frac{\partial \tilde{u}}{\partial x} + \frac{\partial \tilde{v}}{\partial y} + \frac{\partial \tilde{w}}{\partial z}\right)$$

If we have employed the MAC-Grid staggered discretization this equation is discretized as

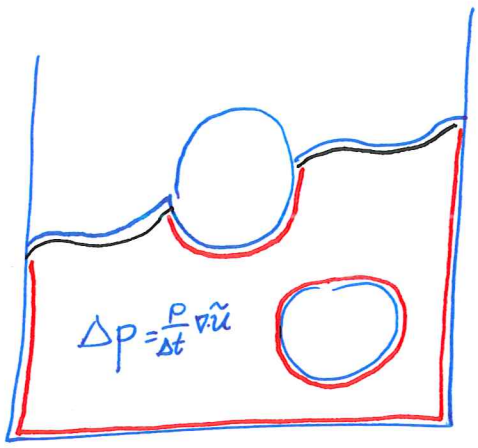
$$\Delta p = (\rho/\Delta t)\, \nabla \cdot \tilde{u} \Rightarrow$$

$$\Rightarrow \frac{p_{i+1,jk} + p_{i-1,jk} + p_{i,j+1,k} + p_{i,j-1,k} + p_{ij,k+1} + p_{ij,k-1} - 6\, p_{ijk}}{h^2}$$

$$= \frac{\rho}{\Delta t}\left(\left(\tilde{u}_{i+1/2,jk} - \tilde{u}_{i-1/2,jk}\right) + \left(\tilde{v}_{i,j+1/2,k} - \tilde{v}_{i,j-1/2,k}\right) + \left(\tilde{w}_{ij,k+1/2} - \tilde{w}_{ij,k-1/2}\right)\right)$$

Boundary Conditions

In order to solve this Poisson equation, we need Boundary conditions at the following locations



$\sim$ = "Dirichlet" Boundaries

$p = 0$

$\Rightarrow$ At the air-water interface: we fix $\boxed{p=0}$ as previously justified (called a Dirichlet condition)

$\sim$ = "Neumann" Boundaries

$\nabla p \cdot \vec{n} = 0$

$\Rightarrow$ At the interface with objects (moving or stationary) or along container walls: We impose $\nabla p \cdot \vec{n} = 0$ (i.e. normal component of the pressure gradient).

Justification: During the advection/gravity steps, we impose $\hat{u} \cdot \vec{\eta} = \tilde{u} \cdot \vec{\eta} = \vec{u}^{solid} \cdot \vec{\eta}$ at all solid-fluid interfaces. This will be the correct normal velocity, and we dont want the pressure/incompressibility step to change that

Thus: $(\vec{u}^{(n+1)} - \tilde{u}) \cdot \vec{\eta} = \underbrace{\left(-\frac{\Delta t}{\rho} \nabla p\right) \cdot \vec{\eta}}_{=0} \Rightarrow \vec{u}^{n+1} \cdot \vec{\eta} = \tilde{u} \cdot \vec{\eta} = \vec{u}^{solid} \cdot \vec{\eta}$!

Incorporating boundary conditions into the discrete equations

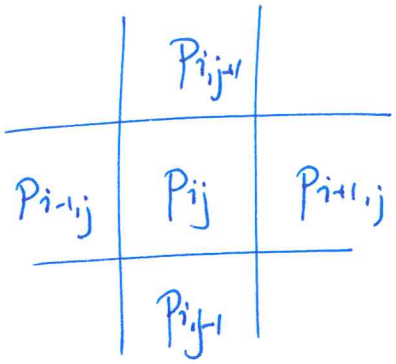
For simplicity consider the 2D case, and the equation

$$\Delta p = f$$

(Here, $f = \rho/\Delta t \, \nabla \cdot \vec{u}$).

At an interior point:

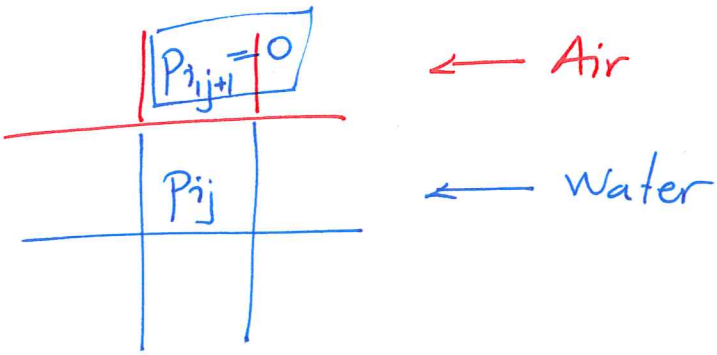
$$\frac{p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - 4p_{ij}}{h^2} = f_{ij}$$



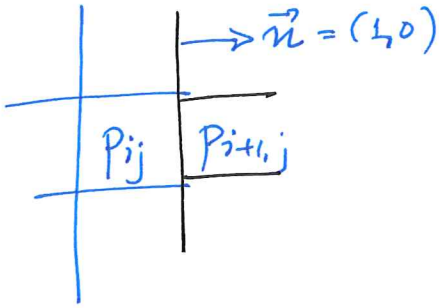
Near a Dirichlet (air) boundary

Equation Becomes:

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



Near a Neumann (solid) boundary



We have $\nabla p \cdot \vec{n} = 0 \Rightarrow$

$$\Rightarrow \left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}\right) \cdot (1,0) = 0$$

$$\Rightarrow \frac{\partial p}{\partial x} = 0 \Rightarrow \frac{p_{i+1,j} - p_{ij}}{h} = 0 \Rightarrow \boxed{p_{i+1,j} - p_{ij} = 0}$$

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

It turns out that the resulting system of equations is:

⇒ Symmetric!

⇒ Negative-Definite!

↳ We can just negate all equations ⇒ System becomes symmetric & positive definite

Thus CG is applicable, and can be used to compute the pressure field.

Technical issues (read R. Bridson's notes)

⇒ The "standard" CG algorithm might take too long to converge (at a minimum we must take as many iterations as the diameter (by the Manhattan distance) of our voxelized domain). Preconditioned CG reduces the number of iterations substantially (sometimes by 0.1x or more) at the expense of making each iteration more expensive

⇒ When simulating smoke, we typically only use Neumann conditions. In such a case p is only known up to a constant and the discrete Laplacian has a rank-1 nullspace. CG can still handle this, but with a careful minor modification.