Notes

- Please read
  - Fedkiw, Stam, Jensen, “Visual simulation of smoke”, SIGGRAPH ‘01

Solving Full Equations

- Let’s now solve the full incompressible Euler or Navier-Stokes equations
- We’ll avoid interfaces (e.g. free surfaces)
- Think smoke

Operator Splitting

- Generally a bad idea to treat incompressible flow as conservation laws with constraints
- Instead: split equations up into easy chunks, just like Shallow Water

$$u_t + u \cdot \nabla u = 0$$

$$u_t = \nu \nabla^2 u$$

$$u_t = g$$

$$u_t + \tfrac{1}{\rho} \nabla p = 0 \qquad (\nabla \cdot u = 0)$$

Time integration

- Don’t mix the steps at all - 1st order accurate

$$u^{(1)} = advect(u^n, \Delta t)$$

$$u^{(2)} = u^{(1)} + \nu \Delta t \nabla^2 u^{(2)}$$

$$u^{(3)} = u^{(2)} + \Delta t g$$

$$u^{n+1} = u^{(3)} - \Delta t \tfrac{1}{\rho} \nabla p$$

- We’ve already seen how to do the advection step
- Often can ignore the second step (viscosity)
- Let’s focus for now on the last step (pressure)

Advection boundary conditions

- But first, one last issue
- Semi-Lagrangian procedure may need to interpolate from values of u outside the domain, or inside solids
  - Outside: no correct answer. Extrapolating from nearest point on domain is fine, or assuming some far-field velocity perhaps
  - Solid walls: velocity should be velocity of wall (usually zero)
    - Technically only normal component of velocity needs to be taken from wall, in absence of viscosity the tangential component may be better extrapolated from the fluid

Continuous pressure

- Before we discretize in space, last step is to take $u^{(3)}$, figure out the pressure p that makes u^{n+1} incompressible:
  - Want $\nabla \cdot u^{n+1} = 0$
  - Plug in pressure update formula: $\nabla \cdot \left(u^{(3)} - \Delta t \tfrac{1}{\rho} \nabla p\right) = 0$
  - Rearrange: $\nabla \cdot \left(\Delta t \tfrac{1}{\rho} \nabla p\right) = \nabla \cdot u^{(3)}$
  - Solve this Poisson problem (often density is constant and you can rescale p by it, also Δt)
    - Make this assumption from now on:

$$\nabla^2 p = \nabla \cdot u^{(3)}$$

$$u^{n+1} = u^{(3)} - \nabla p$$

Pressure boundary conditions

- Issue of what to do for p and u at boundaries in pressure solve
- Think in terms of control volumes: subtract pn from u on boundary so that integral of u•n is zero
- So at closed boundary we end up with

$$u^{n+1} \cdot n = 0$$
$$u^{n+1} \cdot n = u^{(3)} \cdot n - \frac{\partial p}{\partial n}$$

Pressure BC's cont'd.

- At closed wall boundary have two choices:
  - Set u•n=0 first, then solve for p with ∂p/∂n=0, don't update velocity at boundary
  - Or simply solve for p with ∂p/∂n=u•n and update u•n at boundary with -∂p/∂n
  - Equivalent, but the second option make sense in the continuous setting, and generally keeps you more honest
- At open (or free-surface) boundaries, no constraint on u•n, so typically pick p=0

Approximate projection

- Can now directly discretize Poisson equation on a grid

$$\left(\nabla^2 p\right)_{ijk} = \left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2}\right)$$
$$\approx \frac{p_{i+1jk} - 2p_{ijk} + p_{i-1jk}}{\Delta x^2} + \frac{p_{ij+1k} - 2p_{ijk} + p_{ij-1k}}{\Delta y^2} + \frac{p_{ijk+1} - 2p_{ijk} + p_{ijk-1}}{\Delta z^2}$$
$$\left(\nabla \cdot u\right)_{ijk} = \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)_{ijk}$$
$$\approx \frac{u_{i+1jk} - u_{i-1jk}}{2\Delta x} + \frac{v_{ij+1k} - v_{ij-1k}}{2\Delta y} + \frac{w_{ijk+1} - w_{ijk-1}}{2\Delta z}$$
$$\left(\nabla p\right)_{ijk} \approx \left[\frac{p_{i+1jk} - p_{i-1jk}}{2\Delta x}, \frac{p_{ij+1k} - p_{ij-1k}}{2\Delta y}, \frac{p_{ijk+1} - p_{ijk-1}}{2\Delta z}\right]$$

- Central differences - 2nd order, no bias

Issues

- On the plus side: simple grid, simple discretization, becomes exact in limit for smooth u…
- But it doesn't work
  - Divergence part of equation can't "see" high frequency compression waves
  - Left with high frequency oscillatory error
  - Need to filter this out - smooth out velocity field before subtracting off pressure gradient
  - Filtering introduces more numerical viscosity, eliminates features on coarse grids
- Also: doesn't exactly make u incompressible
  - Measuring divergence of result gives nonzero
- So let's look at exactly enforcing the incompressibility constraint

Exact projection (1st try)

- Connection
  - use the discrete divergence as a hard constraint to enforce, pressure p turns out to be the Lagrange multipliers…
- Or let's just follow the route before, but discretize divergence and gradient first
  - First try: use centred differences as before
  - u and p all "live" on same grid: u_{ijk}, p_{ijk}
  - This is called a "collocated" scheme

Exact collocated projection

- So want $\left(\nabla \cdot u^{n+1}\right)_{ijk} = 0$

$$\frac{u^{n+1}_{i+1jk} - u^{n+1}_{i-1jk}}{2\Delta x} + \frac{v^{n+1}_{ij+1k} - v^{n+1}_{ij-1k}}{2\Delta y} + \frac{w^{n+1}_{ijk+1} - w^{n+1}_{ijk-1}}{2\Delta z} = 0$$

- Update with discrete gradient of p $u^{n+1} = u^{(3)} - \nabla p$

$$u^{n+1}_{ijk} = u^{(3)}_{ijk} - \left[\frac{p_{i+1jk} - p_{i-1jk}}{2\Delta x}, \frac{p_{ij+1k} - p_{ij-1k}}{2\Delta y}, \frac{p_{ijk+1} - p_{ijk-1}}{2\Delta z}\right]$$

- Plug in update formula to solve for p

$$\frac{p_{i+2jk} - 2p_{ijk} + p_{i-2jk}}{4\Delta x^2} + \frac{p_{ij+2k} - 2p_{ijk} + p_{ij-2k}}{4\Delta y^2} + \frac{p_{ijk+2} - 2p_{ijk} + p_{ijk-2}}{4\Delta z^2} =$$
$$\frac{u^{(3)}_{i+1jk} - u^{(3)}_{i-1jk}}{2\Delta x} + \frac{v^{(3)}_{ij+1k} - v^{(3)}_{ij-1k}}{2\Delta y} + \frac{w^{(3)}_{ijk+1} - w^{(3)}_{ijk-1}}{2\Delta z}$$

Problems

- Pressure problem decouples into 8 independent subproblems
- "Checkerboard" instability
  - Divergence still doesn't see high-frequency compression waves
- Really want to avoid differences over 2 grid points, but still want centred
- Thus use a staggered MAC grid, as in last class

Staggered grid

- Pressure p lives in centre of cell, p_{ijk}
- u lives in centre of x-faces, $u_{i+1/2,j,k}$
- v in centre of y-faces, $v_{i,j+1/2,k}$
- w in centre of z-faces, $w_{i,j,k+1/2}$
- Whenever we need to take a difference (grad p or div u) result is where it should be

- Works beautifully with "stair-step" boundaries
  - Not so simple to generalize to other boundary geometry

Exact staggered projection

- Do it discretely as before, but now want

$$\left(\nabla \cdot u^{n+1}\right)_{ijk} = 0$$

$$\frac{u^{n+1}_{i+1/2\,jk} - u^{n+1}_{i-1/2\,jk}}{\Delta x} + \frac{v^{n+1}_{ij+1/2k} - v^{n+1}_{ij-1/2k}}{\Delta y} + \frac{w^{n+1}_{ijk+1/2} - w^{n+1}_{ijk-1/2}}{\Delta z} = 0$$

- And update is

$$u^{n+1}_{i+1/2\,jk} = u^{(3)}_{i+1/2\,jk} - \frac{p_{i+1\,jk} - p_{ijk}}{\Delta x}$$

$$v^{n+1}_{ij+1/2k} = v^{(3)}_{ij+1/2k} - \frac{p_{ij+1k} - p_{ijk}}{\Delta y}$$

$$w^{n+1}_{ijk+1/2} = w^{(3)}_{ijk+1/2} - \frac{p_{ijk+1} - p_{ijk}}{\Delta z}$$

(Continued)

- Plugging in to solve for p

$$\frac{p_{i+1\,jk} - 2p_{ijk} + p_{i-1\,jk}}{\Delta x^2} + \frac{p_{ij+1k} - 2p_{ijk} + p_{ij-1k}}{\Delta y^2} + \frac{p_{ijk+1} - 2p_{ijk} + p_{ijk-1}}{\Delta z^2} = \frac{u^{(3)}_{i+1/2\,jk} - u^{(3)}_{i-1/2\,jk}}{\Delta x} + \frac{v^{(3)}_{ij+1/2k} - v^{(3)}_{ij-1/2k}}{\Delta y} + \frac{w^{(3)}_{ijk+1/2} - w^{(3)}_{ijk-1/2}}{\Delta z}$$

- This is for all i,j,k: gives a linear system to solve -Ap=d

Pressure solve simplified

- Assume for simplicity that Δx=Δy=Δz=h
- Then we can actually rescale pressure (again - already took in density and Δt) to get

$$6p_{ijk} - p_{i+1\,jk} - p_{i-1\,jk} - p_{ij+1k} - p_{ij-1k} - p_{ijk+1} - p_{ijk-1} = -u^{(3)}_{i+1/2\,jk} + u^{(3)}_{i-1/2\,jk} - v^{(3)}_{ij+1/2k} + v^{(3)}_{ij-1/2k} - w^{(3)}_{ijk+1/2} + w^{(3)}_{ijk-1/2}$$

- At boundaries where p is known, replace (say) p_{i+1jk} with known value, move to right-hand side (be careful to scale if not zero!)
- At boundaries where (say) ∂p/∂y=v, replace p_{ij+1k} with p_{ijk}+v (so finite difference for ∂p/∂y is correct at boundary)

Solving the Linear System

- So we're left with the problem of efficiently finding p
- Luckily, linear system Ap=-d is symmetric positive definite
- Incredibly well-studied A, lots of work out there on how to do it fast

How to solve it

- Direct Gaussian Elimination does not work well
  - This is a large sparse matrix - will end up with lots of fill-in (new nonzeros)
- If domain is square with uniform boundary conditions, can use FFT
  - Fourier modes are eigenvectors of the matrix A, everything works out
- But in general, will need to go to iterative methods
  - Luckily - have a great starting guess! Pressure from previous time step [appropriately rescaled]

Convergence

- Need to know when to stop iterating
- Ideally - when error is small
- But if we knew the error, we'd know the solution
- We can measure the residual for Ap=b: it's just r=b-Ap
  - Related to the error: Ae=r
- So check if norm(r)<tol*norm(b)
  - Play around with tol (maybe 1e-4 is good enough?)
- For smoke, may even be enough to just take a fixed number of iterations

Conjugate Gradient

- Standard iterative method for solving symmetric positive definite systems
- For a fairly exhaustive description, read
  - "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain", by J. R. Shewchuk
- Basic idea: steepest descent

Plain vanilla CG

- r=b-Ap (p is initial guess)
- $\rho=r^Tr$, check if already solved
- s=r (first search direction)
- Loop:
  - t=As
  - $\alpha=\rho/(s^Tt)$ (optimum step size)
  - $x += \alpha s$, $r -= \alpha t$, check for convergence
  - $\rho_{new}=r^Tr$
  - $\beta=\rho_{new}/\rho$
  - $s=r+\beta s$ (updated search direction)
  - $\rho=\rho_{new}$