Notes

- Please read
  - Fedkiw, Stam, Jensen, "Visual simulation of smoke", SIGGRAPH '01
- I'll try to have homework 3 back to you later today
  - and homework 4 by Thursday
- Reminder: there is a project in this course, you should be thinking about it
  - Homework 6 is going to be smoke simulation (with extensions such as turning it into water)

Going to 3D

- Today we'll solve full 3D fluid equations
  - Incompressible flow only - don't care about sound waves
  - We may not need viscosity, except for really goopy fluids
    - A lot of CFD is all about overcoming numerical viscosity…
- But without interfaces - simplifies life!
  - Prototypical example: smoke
  - We have fluid moving, and either particles or smoke concentration field moving passively with the flow
  - Passive == velocity doesn't care about smoke

Operator Splitting

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

$$u_t = g$$

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

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

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

Time integration

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

$$u^{(1)} = u^n + \Delta t g$$

$$u^{(2)} = advect(u^{(1)}, \Delta t)$$

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

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

- Need to explain the last 3 steps
- Start off at the end though
  - In some ways hardest part

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: $\nabla \cdot \left(u^{(3)} - \Delta t \frac{1}{\rho} \nabla p \right) = 0$
  - Rearrange: $\nabla \cdot \left(\Delta t \frac{1}{\rho} \nabla p \right) = \nabla \cdot u^{(3)}$
  - Solve this Poisson problem (often density is constant and you can rescale p by it, also Δt)
    - Make this assumption from now on:

$$\nabla^2 p = \nabla \cdot u^{(3)}$$
$$u^{n+1} = u^{(3)} - \nabla p$$

Boundary conditions

- Delicate issue: what BC's for pressure equation? (for both $\nabla \cdot \frac{1}{\rho} \nabla p$ and $\nabla \cdot u^{(3)}$)
- Closed no-slip boundaries (frictional walls) u=0
  - And pressure gradient shouldn't change it: ∂p/∂n=0
- For closed no-stick boundaries u•n=0
  - And same ∂p/∂n=0
- For open, controlled boundaries, same but u≠0
  - But make sure compatible: ∫ u•n=0
- For open, uncontrolled boundaries, specify p
  - u may be changed as incompressibility dictates
  - If we don't know u outside, take ∂u/∂n=0

Approximate projection

- Can now directly discretize Poisson equation on a grid

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

- Central differences - 2nd order, no bias

Problems

- Becomes exact in limit for smooth u
- But isn't good on coarse grids, rough u
  - Particularly around boundaries!
    (e.g. after adding gΔt everywhere, then setting u=0 at ground)
  - See the errors as weird noise, oscillations
- 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

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

- So want $\frac{u^{n+1}_{i+1\,jk} - u^{n+1}_{i-1\,jk}}{2\Delta x} + \frac{v^{n+1}_{ij+1k} - v^{n+1}_{ij-1k}}{2\Delta y} + \frac{w^{n+1}_{ijk+1} - w^{n+1}_{ijk-1}}{2\Delta z} = 0$
- Update with discrete gradient of p $u^{n+1} = u^{(3)} - \nabla p$

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

- Plug in update formula to solve for p

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

Problems

- Pressure problem decouples into 8 independent subproblems
- "Checkerboard" instability
- Weird stuff around boundaries…
- Can be fixed with some effort, but there is a better way
- Really want to avoid differences over 2 grid points, but still want centred
- Thus use a staggered grid

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
- [draw diagram]

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) 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=0, replace p_{ij+1k} with p_{ijk} (so p stays the same as you move across the boundary)

(continued)

- Same kind of boundary treatment for velocity when taking the divergence
- 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)
  - Maybe use infinity norm (max)
  - Play around with tol (maybe 1e-4 is good enough?)

CG

- Already mentioned this when talking about implicit time integration for elasticity problems
- All CG needs to know is how to multiply matrix with a vector (and basic vector operations)
- Repeating a slide from before…

CG again (relabeled)

- r=b-Ap (p is initial guess)
- $\rho=r^T r$, check if already solved
- s=r
- Loop:
  - t=As
  - $\alpha= \rho/(s^T t)$
  - x+= αs, r-= αt, check for convergence
  - $\rho_{new}=r^T r$
  - $\beta= \rho_{new} /\rho$
  - s=r+ βs
  - $\rho=\rho_{new}$

Speeding it up

- This generally takes as many iterations as your grid is large
  - E.g. if 30x70x40 expect to take 70 iterations (or proportional to it)
  - Though a good initial guess may reduce that a lot
- Basic issue: pressure is globally coupled - information needs to travel from one end of the grid to the other
  - Each step of CG can only go one grid point
- Idea of a "preconditioner": if we can get a routine which approximately computes A^{-1}, call it M, then solve MAx=Mb

Preconditioners

- Technically, convergence of PCG (preconditioned congugate gradient) depends on eigenvalue distribution of MA
- Lots and lots of work on how to pick an M
- Examples: FFT, SSOR, ADI, multigrid, sparse approximate inverses
- We'll take a look at Incomplete Cholesky factorization
- But first, how do we change CG to take account of M?
  - M has to be SPD, but MA might not be…

PCG

- r=b-Ap, z=Mr, s=z
- $\rho=z^Tr$, check if already solved
- Loop:
  - t=As
  - $\alpha= \rho/(s^Tt)$
  - x+= αs, r-= αt , check for convergence
  - z=Mr
  - $\rho_{new}=z^Tr$
  - $\beta= \rho_{new} /\rho$
  - s=z+ βs
  - $\rho=\rho_{new}$

Cholesky

- True Gaussian elimination, which is called Cholesky factorization in the SPD case, gives $A=LL^T$
- L is a lower triangular matrix
- Then solving Ap=b can be done by
  - Lx=p, $L^Tp=x$
  - Each solve is easy to do - triangular
  - [reminder]
- But can't do that here since L has many more nonzeros than A -- EXPENSIVE!

Incomplete Cholesky

- We only need approximate result for preconditioner
- So do Cholesky factorization, but throw away new nonzeros (set them to zero)
- Result is not exact, but pretty good
  - Instead of O(n) iterations (for an n^3 grid) we get $O(n^{1/2})$ iterations
- Can actually do better:
  - Modified Incomplete Cholesky
  - Same algorithm, only when we throw away nonzeros, we add them to the diagonal
  - Gets us down to $O(n^{1/4})$ iterations

IC(0)

- Incomplete Cholesky level 0: IC(0) is where we make sure L=0 wherever A=0
- For this A (7-point Laplacian) with the regular grid ordering, things are nice
- Write $A=F+D+F^T$ where F is strictly lower triangular and D is diagonal
- Then IC(0) ends up being of the form $L=(FE^{-1}+E)$ where E is diagonal
  - We only need to store E^{-1} (when doing the triangular solves, only care about inverses of diagonal elements)

Computing IC(0)

- Need to find diagonal E so that $(LL^T)_{ij}=A_{ij}$ wherever $A_{ij}\neq 0$
- Expand out:
  - $LL^T=F+F^T+E^2+FE^{-2}F^T$
- Again, for this special case, can show that last term only contributes to diagonal and elements where $A_{ij}=0$
- So we get the off-diagonal correct for free
- Let's take a look at diagonal entry for grid point ijk

Diagonal Entry

- Assume we order increasing in i, j, k
- Note F=A for lower diagonal elements

$$\left(LL^T\right)_{ijk,ijk} = E_{ijk}^2 + A_{ijk,i-1\,jk}^2 E_{i-1\,jk}^2 + A_{ijk,ij-1k}^2 E_{ij-1k}^2 + A_{ijk,ijk-1}^2 E_{ijk-1}^2$$

- Want this to match A's diagonal
  Then solving for next E_{ijk} in terms of previously determined ones:

$$E_{ijk} = \sqrt{A_{ijk,ijk} - A_{ijk,i-1\,jk}^2 E_{i-1\,jk}^2 - A_{ijk,ij-1k}^2 E_{ij-1k}^2 - A_{ijk,ijk-1}^2 E_{ijk-1}^2}$$

Practicalities

- Again, actually only want to store inverse of E
- Note that for values of A or E off the grid, substitute zero in formula
  - In particular, can start at $E_{000,000}=\sqrt{A_{000,000}}$
- Modified Incomplete Cholesky looks very similar, except instead of matching diagonal entries, we match row sums
- Can squeeze out a little more performance with the “Eisenstat trick”

Viscosity

- The viscosity update (if we really need it - highly viscous fluids) is just Backwards Euler:

$$\left(I - \Delta t \nu \nabla^2\right) u^{(3)} = u^{(2)}$$

- Boils down to almost the same linear system to solve!
  - Or rather, 3 similar linear systems to solve - one for each component of velocity
    (NOTE: solve separately, not together!)
  - Actually, gets easier to solve the smaller $\nu\Delta t$ is
  - Again use PCG with Incomplete Cholesky

Advection

- All that we haven’t yet talked about is how to do advection
- This is the tricky part