Notes

- Previous lecture
  - Missing a factor of Δt in pressure rescaling
  - Not really important though - can absorb into rescaling of pressure (along with density etc.)
  - True update is $$u^{n+1} = u^{(3)} - \Delta t \tfrac{1}{\rho} \nabla p$$
  - If you want to use rescaled pressure from last step as initial guess for pressure solve, need to multiply it by $\Delta t_{new}/\Delta t_{old}$
- Homework 6 goes out later today: check web
- Still marking homework 4…

Recall from last class

- Solving incompressible flow by splitting up the equations into easier chunks
  - We handled gravity
  - We handled pressure
  - We handled viscosity
- Today we'll do the missing piece, advection

$$u_t = g$$

$$\boldsymbol{u_t + u \cdot \nabla u = 0}$$

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

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

Time integration again

- And again, we're doing this in time:

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

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

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

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

- We've chosen to use a staggered grid so the last step works well
- Now need to figure out *advect()*

Velocity advection

- This is a nonlinear problem
  - Difficult for implicit methods
  - Let's stay explicit
- Before jumping to advecting velocity, let's look at advecting a scalar q

$$q_t + u \cdot \nabla q = 0$$

The true solution

- Recall the material derivative - what we're solving

$$\frac{Dq}{Dt} = 0$$

- So if we identify a particle in the flow with some value q_0, just move the particle around and don't change q_0
  - Trivial for Lagrangian methods
  - We'll come back to this later
- But let's keep thinking Eulerian, and try to solve the PDE on a grid

Scalar advection in 1D

- Let's simplify even more, to just one dimension: $q_t+uq_x=0$
- Incompressible flow in 1D is just u=constant
- And let's ignore boundary conditions for now
  - E.g. use a periodic boundary
- True solution just translates q around at speed u - shouldn't change shape

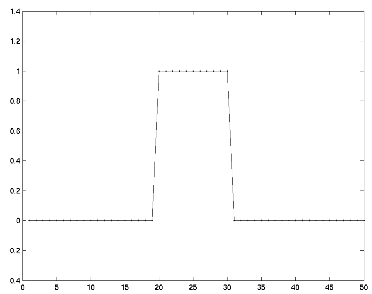
First try: central differences

- Centred-differences give better accuracy
- Example: $$\frac{\partial q_i}{\partial t} = -u\left(\frac{q_{i+1} - q_{i-1}}{2\Delta x}\right)$$
  - 2nd order accurate in space
- Eigenvalues are pure imaginary - rules out Forward Euler and RK2 in time
- But what does the solution look like?

Testing a pulse

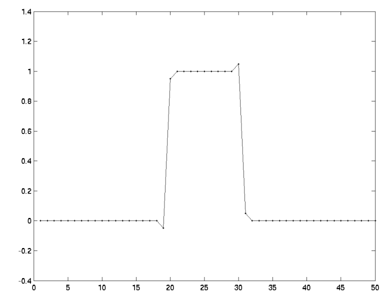
- We know things have to work out nicely in the limit (second order accurate)
  - I.e. when the grid is fine enough
  - What does that mean? -- when the sampled function looks smooth on the grid
- In graphics, it's just redundant to use a grid that fine
  - we can fill in smooth variations with interpolation later
- So we're always concerned about coarse grids == not very smooth data
- Discontinuous pulse is a nice test case

A pulse (initial conditions)



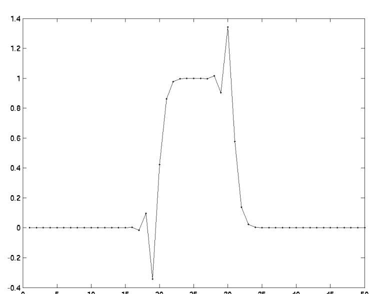
Centered finite differences

- A few time steps (RK4, small Δt) later
  - u=1, so pulse should just move right without changing shape



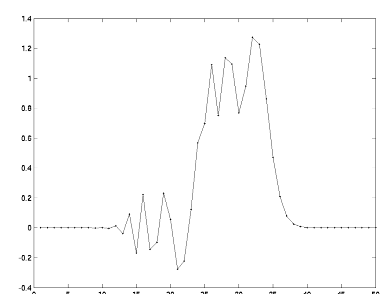
Centred finite differences

- A little bit later…



Centred finite differences

- A fair bit later



What went wrong?

- Lots of ways to interpret this error
- Example - phase analysis
  - Take a look at what happens to a sinusoid wave numerically
  - The amplitude stays constant (good), but the wave speed depends on wave number (bad) - we get dispersion
  - So the sinusoids that initially sum up to be a square pulse move at different speeds and separate out
    - We see the low frequency ones moving faster…
  - But this analysis won't help so much in 3D, variable u…

Modified PDE's

- Another way to interpret error - try to account for it in the physics
- Look at truncation error more carefully:

$$q_{i+1} = q_i + \Delta x \frac{\partial q}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 q}{\partial x^2} + \frac{\Delta x^3}{6}\frac{\partial^3 q}{\partial x^3} + O\left(\Delta x^4\right)$$

$$q_{i-1} = q_i - \Delta x \frac{\partial q}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 q}{\partial x^2} - \frac{\Delta x^3}{6}\frac{\partial^3 q}{\partial x^3} + O\left(\Delta x^4\right)$$

$$\frac{q_{i+1} - q_{i-1}}{2\Delta x} = \frac{\partial q}{\partial x} + \frac{\Delta x^2}{6}\frac{\partial^3 q}{\partial x^3} + O\left(\Delta x^3\right)$$

- Up to high order error, we numerically solve

$$q_t + uq_x = -\frac{u\Delta x^2}{6} q_{xxx}$$

Interpretation

$$q_t + uq_x = -\frac{u\Delta x^6}{6} q_{xxx}$$

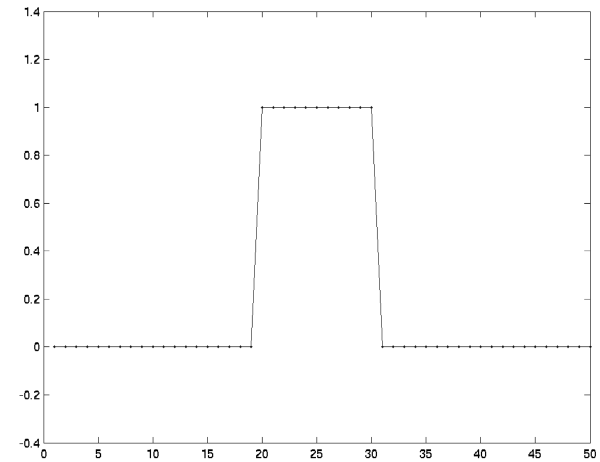
- Extra term is "dispersion"
  - Does exactly what phase analysis tells us
  - Behaves a bit like surface tension…
- We want a numerical method with a different sort of truncation error
  - Any centred scheme ends up giving an odd truncation error --- dispersion
- Let's look at one-sided schemes

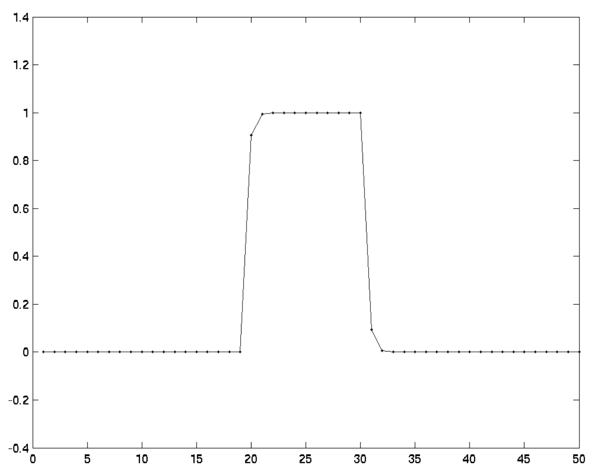
Upwind differencing

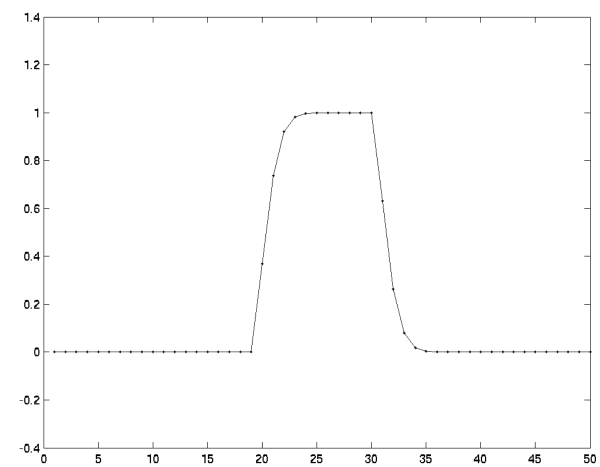
- Think physically:
  - True solution is that q just translates at velocity u
- Information flows with u
- So to determine future values of q at a grid point, need to look "upwind" -- where the information will blow from
  - Values of q "downwind" only have any relevance if we know q is smooth -- and we're assuming it isn't

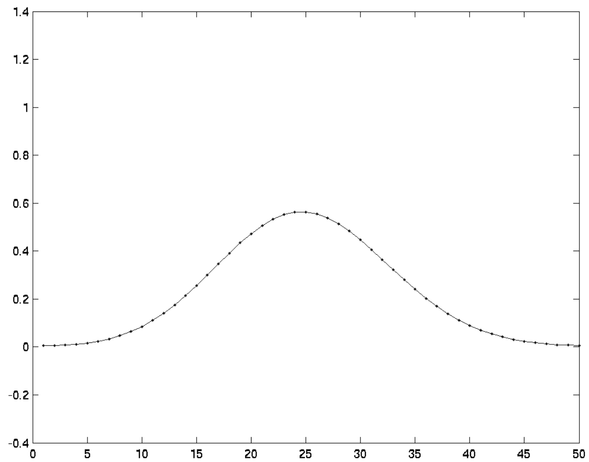
1st order upwind

- Basic idea: look at sign of u to figure out which direction we should get information
- If u<0 then $\partial q/\partial x \approx (q_{i+1}-q_i)/\Delta x$
- If u>0 then $\partial q/\partial x \approx (q_i-q_{i-1})/\Delta x$
- Only 1st order accurate though
  - Let's see how it does on the pulse…









Modified PDE again

- Let's see what the modified PDE is this time

$$q_{i+1} = q_i + \Delta x \frac{\partial q}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 q}{\partial x^2} + O(\Delta x^3)$$

$$\frac{q_{i+1} - q_i}{\Delta x} = \frac{\partial q}{\partial x} + \frac{\Delta x}{2}\frac{\partial^2 q}{\partial x^2} + O(\Delta x^2)$$

- For u<0 then we have $q_t + uq_x = -u\Delta x\, q_{xx}$
- And for u>0 we have (basically flip sign of Δx)

$$q_t + uq_x = u\Delta x\, q_{xx}$$

- Putting them together, 1st order upwind numerical solves (to 2nd order accuracy)

$$q_t + uq_x = |u\Delta x| q_{xx}$$

Interpretation

- The extra term (that disappears as we refine the grid) is **diffusion** or **viscosity**
- So sharp pulse gets blurred out into a hump, and eventually melts to nothing
- It looks a lot better, but still not great
  - Again, we want to pack as much detail as possible onto our coarse grid
  - With this scheme, the detail melts away to nothing pretty fast
- Note: unless grid is really fine, the numerical viscosity is much larger than physical viscosity - so might as well not use the latter

Fixing upwind method

- Natural answer - reduce the error by going to higher order - but life isn't so simple
- High order difference formulas can overshoot in extrapolating
  - If we difference over a discontinuity
  - Stability becomes a real problem
- Go nonlinear (even though problem is linear)
  - "limiters" - use high order unless you detect a (near-)overshoot, then go back to 1st order upwind
  - "ENO" - try a few different high order formulas, pick smoothest

Hamilton-Jacobi Equations

- In fact, the advection step is a simple example of a Hamilton-Jacobi equation (HJ)
  - $q_t+H(q,q_x)=0$
- Come up in lots of places
  - Level sets…
- Lots of research on them, and numerical methods to solve them
- We don't want to get into that complication

Other problems

- Even if we use top-notch numerical methods for HJ, we have problems
  - Time step limit: CFL condition
    - Have to pick time step small enough that information physically moves less than a grid cell: $\Delta t<\Delta x/u$
  - Schemes can get messy at boundaries

Exploiting Lagrangian view

- But wait! This was trivial for Lagrangian (particle) methods!
- We still want to stick an Eulerian grid for now, but somehow exploit the fact that
  - If we know q at some point x at time t, we just follow a particle through the flow starting at x to see where that value of q ends up

$$q\big(x(t+\Delta t),t+\Delta t\big)=q_0$$

$$\frac{dx}{dt}=u\big(x\big),\quad x(t)=x_0$$

Looking backwards

- Problem with tracing particles - we want values at **grid nodes** at the end of the step
  - Particles could end up anywhere
- But… we can look backwards in time

$$q_{ijk}=q\big(x(t-\Delta t),t-\Delta t\big)$$

$$\frac{dx}{dt}=u\big(x\big),\quad x(t)=x_{ijk}$$

- Same formulas as before - but new interpretation
  - To get value of q at a grid point, follow a particle backwards through flow to wherever it started

Semi-Lagrangian method

- Also dubbed “stable fluids” in graphics (reinvention)
- To find the new value of q at a grid point, trace particle backwards from grid point (in velocity field u) for -Δt and interpolate from old values of q
- Two questions
  - How do we trace?
  - How do we interpolate?

Tracing

- The errors we make in tracing backwards aren’t too big a deal
  - We don’t compound them - stability isn’t an issue
  - How accurate we are in tracing doesn’t effect shape of q much, just location
    - Whether we get too much blurring, oscillations, or a nice result is really up to interpolation
- Thus look at “Forward” Euler and RK2

Tracing: 1st order

- We’re at grid node (i,j,k) at position x_{ijk}
- Trace backwards through flow for -Δt

$$x_{old} = x_{ijk} - \Delta t\, u_{ijk}$$

  - Note - using u value at grid point (what we know already) like Forward Euler.
- Then can get new q value (with interpolation)

$$\begin{aligned} q_{ijk}^{n+1} &= q^n\left(x_{old}\right) \\ &= q^n\left(x_{ijk} - \Delta t\, u_{ijk}\right) \end{aligned}$$

Behaviour around vortices

- [draw examples]
  - Forward Euler tracing will grab information from further out
    - If we’re actually advecting velocity itself, vortex will slow down and shrink…
  - RK2 much better, but still a little
  - Backward Euler is the opposite
    - Vortices will grow!
  - Trapezoidal Rule is just right
  - But implicit methods probably way too slow
    - Stability is not an issue for us!
  - Modified Euler (slightly more stable than RK2) is attractive…

Interpolation

- First order accurate: nearest neighbour
  - Just pick q value at grid node closest to x_{old}
  - Doesn't work for slow fluid (small time steps) -- nothing changes!
- Second order accurate: linear
  - Or bilinear (2D), trilinear (3D)
  - Still fast, easy to handle boundary conditions…
  - How well does it work?

Linear interpolation

- Error in linear interpolation is proportional to

$$\Delta x^2 \frac{\partial^2 q}{\partial x^2}$$

- Modified PDE ends up something like…

$$\frac{Dq}{Dt} = k(\Delta t)\Delta x^2 \frac{\partial^2 q}{\partial x^2}$$

  - We have numerical viscosity, things will smear out
  - For reasonable time steps, k looks like 1/Δt ~ 1/Δx
- [Equivalent to 1st order upwind for CFL Δt]
- In practice, too much smearing!

Nice properties of lerping

- But linear interpolation is completely stable
  - Interpolated value of q must lie between the old values of q on the grid
  - Thus with each time step, max(q) cannot increase, and min(q) cannot decrease
- Thus we end up with a fully stable algorithm - take Δt as big as you want
  - Great for interactive applications
    (if the pressure solve is fast enough…)

Cubic interpolation

- To fix the problem of excessive smearing, go to higher order
- E.g. use cubic splines
  - Finding interpolating C^2 cubic spline is a little painful, an alternative is the
  - C^1 Catmull-Rom (cubic Hermite) spline
    - [derive]
- Introduces overshoot problems
  - Stability isn't so easy to guarantee anymore

Min-mod limited Catmull-Rom

- Trick is to check if either slope at the endpoints of the interval has the wrong sign
  - If so, clamp the slope to zero
  - Still use cubic Hermite formulas with more reliable slopes
- This has same stability guarantee as linear interpolation
  - But in smoother parts of flow, higher order accurate
  - Called "high resolution"
- Still has issues with boundary conditions…

Velocity advection

- So far we've concentrated just on advecting some scalar q
- We want to advect velocity
  - But velocity field defines advection: nonlinear!
  - So we ignore it: "lagging"
    - Just use the old velocity field, treat u, v, and w as just scalars to move around like q
  - Again - only 1st order accurate in time, but that's OK.

Staggered grid

- Problem: velocity on a staggered grid
- Simple answer
  - Average velocities to get flow field where you need it, e.g. $u_{ijk}=0.5(u_{i+1/2\ jk} + u_{i-1/2\ jk})$
  - So advect each component of velocity around in averaged velocity field
- Even cheaper
  - Advect averaged velocity field around (with any other quantity you care about) --- reuse interpolation coefficients!
  - But - all that averaging smears u out… more numerical viscosity! [worse for small Δt]

Vorticity confinement

- The interpolation errors behave like viscosity, the averaging from the staggered grid behaves like viscosity…
  - Net effect is that interesting flow structures (vortices) get smeared out
  - Boooooring
- Idea of vorticity confinement - add a fake force that spins vortices faster
  - Try to cancel off some of the numerical viscosity in a stable way

Smoke

- Smoke is a bit more than just a velocity field
- Need temperature (hot air rises) and smoke density (smoke eventually falls)
- Real physics - density depends on temperature, temperature depends on viscosity and thermal conduction, …
  - We'll ignore most of that: small scale effects
  - Boussinesq approximation: ignore density variation except in gravity term, ignore energy transfer except thermal conduction
  - We go a step further and ignore thermal conduction - insignificant vs. numerical dissipation

Smoke concentration

- There's more than just air temperature to consider too
- Smoke weighs more than air - so need to track smoke concentration
  - Also could be used for rendering (though tracing particles can give better results)
  - Point is: physics depends on smoke concentration, not just appearance
- We again ignore effect of this in all terms except gravity force

Buoyancy

- For smoke, where there is no interface, we can add ρgy to pressure (and just solve for the difference) thus cancelling out g term in equation
- All that's left is buoyancy -- variation in vertical force due to density variation
- Density varies because of temperature change and because of smoke concentration
- Assume linear relationship (small variations)

$$f_{bouy} = \left(-\alpha s + \beta T\right)$$

  - T=0 is ambient temperature; α, β depend on g etc.

Smoke equations

- So putting it all together…

$$u_t + u \cdot \nabla u + \nabla p = \left(-\alpha s + \beta T\right)(0,1,0)$$

$$\nabla \cdot u = 0$$

$$T_t + u \cdot \nabla T = 0$$

$$s_t + u \cdot \nabla s = 0$$

- We know how to solve the u part, using old values for s and T
- Advecting s and T around is simple - just scalar advection

Notes on discretization

- Smoke concentration and temperature may as well live in grid cells same as pressure
- But then to add buoyancy force, need to average to get values at staggered positions
- Also, to maintain conservation properties, should only advect smoke concentration and temperature (and any other scalars) in a divergence-free velocity field
  - If you want to do all the advection together, do it before adding buoyancy force
  - I.e. advect; buoyancy; pressure solve; repeat