Slide 1

FYS-GEO 4500   02 Nov 2009

Before we start:
Questions over the reading?

The problem set

Slide 2

Our schedule

	date		Topic	Chapter in LeVeque
1	17 Aug 2009	Monday 13.15-15.00	introduction to conservation laws, Clawpack	1 & 2 & 5
2	24 Aug 2009	Monday 13.15-15.00	the Riemann problem, characteristics	3
3	**28 Aug 2009**	**Friday 13.15-15.00**	finite volume methods for linear systems	4
4	**8 Sep 2009**	**Tuesday 13.15-15.00**	high resolution methods	6
5	21 Sep 2009	Monday 13.15-15.00	boundary conditions, accuracy, variable coeff.	7,8, part of 9
6	**29 Sep 2009**	**Tuesday 13.15-15.00**	nonlinear conservation laws, finite volume methods	11 & 12
7	5 Oct 2009	Monday 13.15-15.00	nonlinear equations & systems	13 & 14
8	12 Oct 2009	Monday 13.15-15.00	finite volume methods for nonlinear systems	15
9	19 Oct 2009	Monday 13.15-15.00	source terms and multidimensions	16,17,18,19
10	26 Oct 2009	Monday 13.15-15.00	multidimensional systems	20 & 21
11	2 Nov 2009	Monday 13.15-15.00	any other topics; project planning	
12	16 Nov 2009	Monday 13.15-15.00	applications: tsunamis, pockmarks, venting, impacts	
13	23 Nov 2009	Monday 13.15-15.00	applications: volcanic jets, pyroclastic flows, lahars	
14	30 Nov 2009	Monday 13.15-15.00	review; progress, problems & projects	
16	7 Dec 2009	Monday 13.15-15.00	FINAL PROJECT REPORTS DUE	

Slide 3

Riemann solvers: exact vs approximate?

Whether to use a Roe solver, or other approximate Riemann solver, as opposed to an iterative exact solver is under dispute.

Exact solvers are typically costly in time and storage

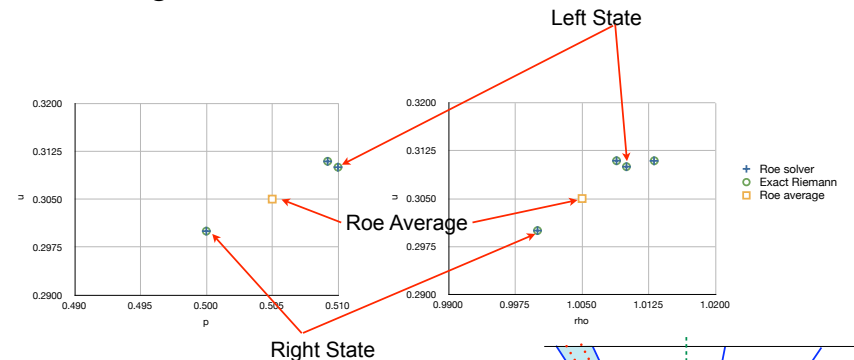You don't need all the information generated

However, if you use a Roe solver:

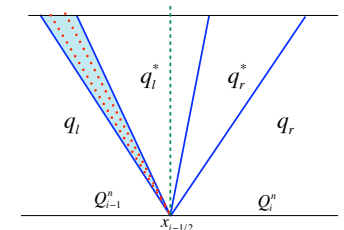You don't get the full structure of the rarefaction wave

In certain circumstances, the approximation may be poor

As computers and methods improve, more people may prefer exact iterative solvers.
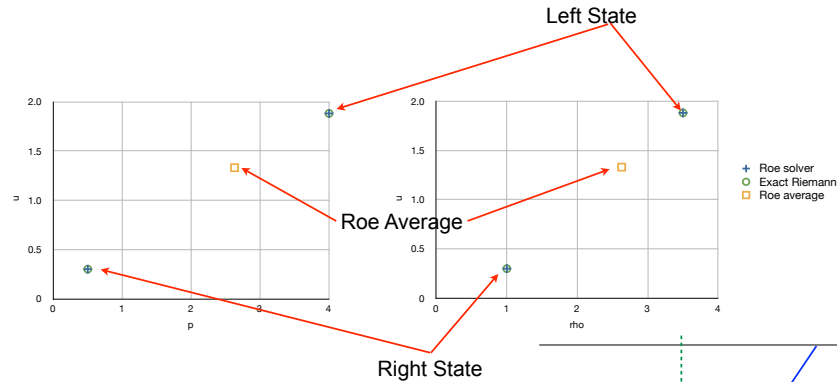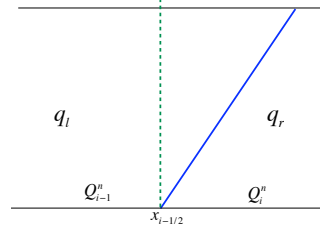
Slide 4

How good is the Roe solver?

When the left and right states are close together, the Roe solver is very good

Slide 1

How good is the Roe solver?

Left State

Right State

+ Roe solver
○ Exact Riemann
□ Roe average

Roe Average

When the left and right states are connected by a single shock, there are no intermediate states, and the Roe solver is exact

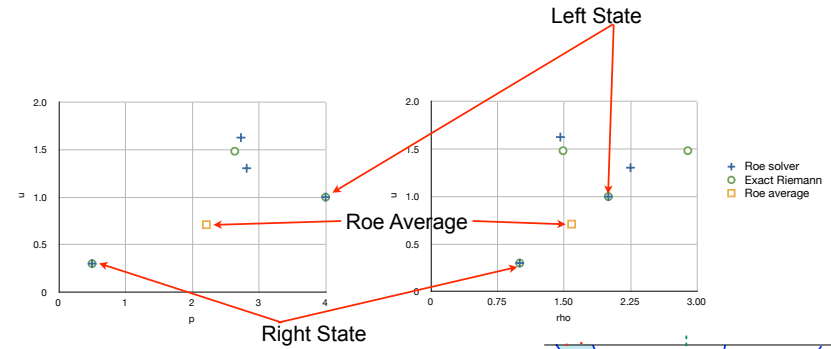q_l q_r

Q_{i-1}^n Q_i^n

$x_{i-1/2}$

Slide 2

How good is the Roe solver?
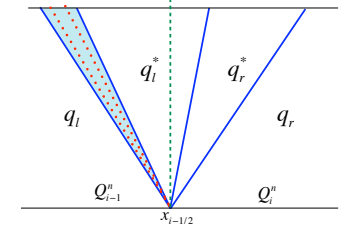
Left State

+ Roe solver
○ Exact Riemann
□ Roe average

Roe Average

Right State

For arbitrary right and left states, the Roe solver is definitely inaccurate.

If the resolution is sufficiently good, this circumstance should not occur in practice.

But in the other two cases, one or two iterations in the exact solve are enough!

q_l^* q_r^*

q_l q_r

Q_{i-1}^n Q_i^n

$x_{i-1/2}$

Slide 3

Multidimensional Scalar Equations
(Chapter 20 in Leveque)

Slide 4

High-resolution methods for scalar hyperbolic equations in two dimensions

Start once again with the constant-velocity advection equation, this time with a velocity $\vec{u} = [u,v]$ with both x and y components. We consider both positive for now.

$$q_t + uq_x + vq_y = 0$$

Then of course the solution is the unchanged original function, simply translated with time at the velocity $\vec{u}$. But now it's moving at an angle.

A Taylor series expansion gives us

$$q(x,y,t_{n+1}) = q(x,y,t_n) - u\Delta t q_x - v\Delta t q_y$$

$$+ \frac{1}{2}(\Delta t)^2 \left[u^2 q_{xx} + vu q_{xy} + uv q_{yx} + v^2 q_{yy} \right] + \dots$$

Donor-cell upwind method

The donor-cell upwind method is simply

$$Q_{ij}^{n+1} = Q_{ij}^n - \frac{\Delta t}{\Delta x}\Big[u^+\big(Q_{i,j}^n - Q_{i-1,j}^n\big) + u^-\big(Q_{i+1,j}^n - Q_{i,j}^n\big)\Big]$$
$$- \frac{\Delta t}{\Delta y}\Big[v^+\big(Q_{i,j}^n - Q_{i,j-1}^n\big) + v^-\big(Q_{i,j+1}^n - Q_{i,j}^n\big)\Big]$$

with $u^+ = \max(u,0),\ u^- = \min(u,0)$ as before. This can be written in the fluctuation form

$$Q_{ij}^{n+1} = Q_{ij}^n - \frac{\Delta t}{\Delta x}\Big(\mathcal{A}^+\Delta Q_{i-1/2,j} + \mathcal{A}^-\Delta Q_{i+1/2,j}\Big) - \frac{\Delta t}{\Delta y}\Big(\mathcal{B}^+\Delta Q_{i,j-1/2} + \mathcal{B}^-\Delta Q_{i,j+1/2}\Big)$$
$$- \frac{\Delta t}{\Delta x}\Big(\tilde{F}_{i+1/2,j} - \tilde{F}_{i-1/2,j}\Big) - \frac{\Delta t}{\Delta y}\Big(\tilde{G}_{i,j+1/2} - \tilde{G}_{i,j-1/2}\Big)$$
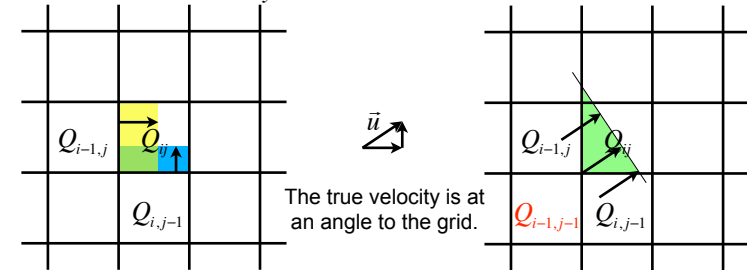
with

$$\mathcal{A}^+\Delta Q_{i-1/2,j} = u^\pm\big(Q_{ij} - Q_{i-1,j}\big), \quad \mathcal{B}^+\Delta Q_{i,j-1/2} = v^\pm\big(Q_{i,j} - Q_{i,j-1}\big)$$

and without the corrective limited fluxes, i.e. $\tilde{F} = \tilde{G} = 0$

The naive donor-cell upwind method misses the contribution from the corner

$$Q_{ij}^{n+1} = Q_{ij}^n - \frac{\Delta t}{\Delta x}\Big[u^+\big(Q_{i,j}^n - Q_{i-1,j}^n\big) + u^-\big(Q_{i+1,j}^n - Q_{i,j}^n\big)\Big]$$
$$- \frac{\Delta t}{\Delta y}\Big[v^+\big(Q_{i,j}^n - Q_{i,j-1}^n\big) + v^-\big(Q_{i,j+1}^n - Q_{i,j}^n\big)\Big]$$

The true velocity is at an angle to the grid.

This method will have stability problems. Dimensional splitting (doing an intermediate update with a sweep in one direction, followed by a sweep in the other direction) may help.

A better approach is to use the REA technique

The Godunov REA scheme can be easily extended to two (or three) dimensions:

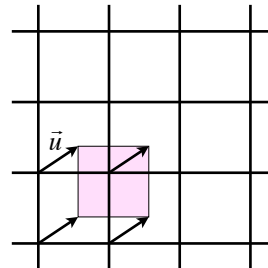1. Reconstruct a piece-wise linear function from the cell averages.

$$q^n(x,y,t_n) = Q_{ij}^n \text{ for } x,y \text{ in cell } i,j$$

2. Evolve the advection equation exactly with these data to obtain a later-time function.

$$\tilde{q}^n(x,y,t_{n+1})$$

3. Average this function over each grid cell to obtain new cell averages.

$$Q_{ij}^{n+1} = \frac{1}{\Delta x \Delta y}\int_{y_{j-1/2}}^{y_{j+1/2}}\int_{x_{i-1/2}}^{x_{i+1/2}}\tilde{q}^n(x,t_{n+1})\,dx\,dy$$

## Colella's corner-transport upwind method

Following the REA procedure results in:

$$Q_{ij}^{n+1} = Q_{ij}^n - \frac{u\Delta t}{\Delta x}\big(Q_{i,j}^n - Q_{i-1,j}^n\big) - \frac{v\Delta t}{\Delta y}\big(Q_{i,j}^n - Q_{i,j-1}^n\big)$$
$$+ \frac{uv\Delta t^2}{\Delta x \Delta y}\big(Q_{i,j}^n - Q_{i-1,j}^n - Q_{i,j-1}^n + Q_{i-1,j-1}^n\big)$$

This expression is rearranged and simplified from Leveque's Eqn 20.10

This is still just first-order accurate, but can be improved by introducing limited slopes, just as we did for 1-dimension.

Let's look at it in still a different way.
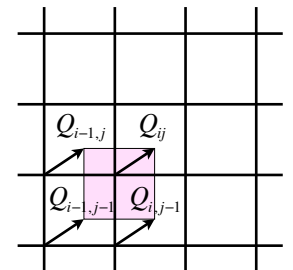
Wave-propagation implementation of CTU

In a sense, we turn CTU inside out.

Consider transport from the edges (faces), not the corners.

From the interface between $(i-1,j)$ and (i,j) a wave goes into both (i,j) and $(i,j+1)$: and four distinct waves affect (i,j).

You can work out how much each wave contributes to each cell from simple geometry. Just sum up the areas of the little triangles and squares!
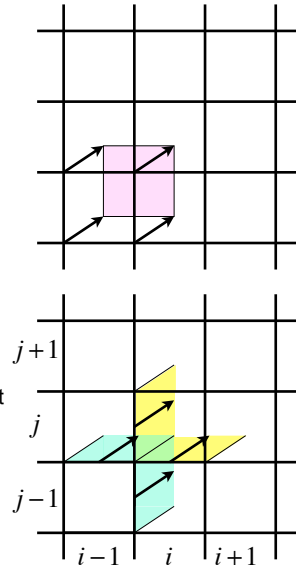
This interpretation leads to the fluctuation format

$$Q_{ij}^{n+1} = Q_{ij}^{n} - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^{+}\Delta Q_{i-1/2,j} + \mathcal{A}^{-}\Delta Q_{i+1/2,j}\right) - \frac{\Delta t}{\Delta y}\left(\mathcal{B}^{+}\Delta Q_{i,j-1/2} + \mathcal{B}^{-}\Delta Q_{i,j+1/2}\right)$$

$$-\frac{\Delta t}{\Delta x}\left(\tilde{F}_{i+1/2,j} - \tilde{F}_{i-1/2,j}\right) - \frac{\Delta t}{\Delta y}\left(\tilde{G}_{i,j+1/2} - \tilde{G}_{i,j-1/2}\right)$$

with the fluctuations defined as for Lax-Wendroff:

$$\mathcal{A}^{\pm}\Delta Q_{i-1/2,j} = u^{\pm}\left(Q_{ij} - Q_{i-1,j}\right), \quad \mathcal{B}^{\pm}\Delta Q_{i,j-1/2} = v^{\pm}\left(Q_{i,j} - Q_{i,j-1}\right)$$

but now with the correction fluxes:

$$\tilde{F}_{i-1/2,j} = -\frac{1}{2}\frac{\Delta t}{\Delta y}uv(Q_{i-1,j} - Q_{i-1,j-1}),$$

$$\tilde{F}_{i+1/2,j} = -\frac{1}{2}\frac{\Delta t}{\Delta y}uv(Q_{ij} - Q_{i,j-1}),$$

$$\tilde{G}_{i,j-1/2} = -\frac{1}{2}\frac{\Delta t}{\Delta x}uv(Q_{i,j-1} - Q_{i-1,j-1}),$$

$$\tilde{G}_{i,j+1/2} = -\frac{1}{2}\frac{\Delta t}{\Delta x}uv(Q_{ij} - Q_{i-1,j}).$$

Generalise to variable-velocity advection

The fluctuations generalise easily, using the cell-interface velocities:

$$\mathcal{A}^{\pm}\Delta Q_{i-1/2,j} = u_{i-1/2,j}^{\pm}\left(Q_{ij} - Q_{i-1,j}\right), \quad \mathcal{B}^{\pm}\Delta Q_{i,j-1/2} = v_{i,j-1/2}^{\pm}\left(Q_{i,j} - Q_{i,j-1}\right)$$

But now the fluxes must be rewritten for the transverse velocities, taking into account that the velocities may be either negative or positive. The correction fluxes are set to zero to start each time step, then built up from consideration of the signs of the velocities. There are four possibilities in each direction.

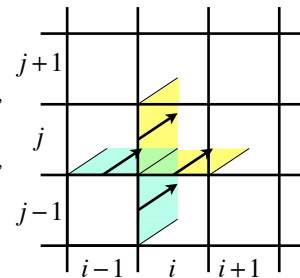First set $\quad \tilde{F}_{i-1/2,j} := 0$ and $\tilde{G}_{i,j-1/2} := 0 \quad \forall i,j$

Then do an x-sweep to update

$$\tilde{G}_{i-1,j-1/2} := \tilde{G}_{i-1,j-1/2} - \frac{1}{2}\frac{\Delta t}{\Delta x}u_{i-1/2,j}^{-}v_{i-1,j-1/2}^{-}(Q_{ij} - Q_{i-1,j}),$$

$$\tilde{G}_{i-1,j+1/2} := \tilde{G}_{i-1,j+1/2} - \frac{1}{2}\frac{\Delta t}{\Delta x}u_{i-1/2,j}^{-}v_{i-1,j+1/2}^{+}(Q_{ij} - Q_{i-1,j}),$$

$$\tilde{G}_{i,j-1/2} := \tilde{G}_{i,j-1/2} - \frac{1}{2}\frac{\Delta t}{\Delta x}u_{i-1/2,j}^{+}v_{i,j-1/2}^{-}(Q_{ij} - Q_{i-1,j}),$$

$$\tilde{G}_{i,j+1/2} := \tilde{G}_{i,j+1/2} - \frac{1}{2}\frac{\Delta t}{\Delta x}u_{i-1/2,j}^{+}v_{i,j+1/2}^{+}(Q_{ij} - Q_{i-1,j}).$$

Transverse fluxes are also needed

A transverse Riemann solver takes the right- and left-going fluctuations

$$\mathcal{A}^{\pm}\Delta Q_{i-1/2,j} = u_{i-1/2,j}^{\pm}(Q_{ij} - Q_{i-1,j})$$

and makes the four transverse fluctuations

$$\mathcal{B}^{-}\mathcal{A}^{+}\Delta Q_{i-1/2,j} = v_{i,j-1/2}^{-}u_{i-1/2,j}^{+}\left(Q_{ij} - Q_{i-1,j}\right)$$

$$\mathcal{B}^{+}\mathcal{A}^{+}\Delta Q_{i-1/2,j} = v_{i,j+1/2}^{+}u_{i-1/2,j}^{+}\left(Q_{ij} - Q_{i,j-1}\right)$$

$$\mathcal{B}^{-}\mathcal{A}^{-}\Delta Q_{i-1/2,j} = v_{i,j-1/2}^{-}u_{i-1/2,j}^{-}\left(Q_{ij} - Q_{i-1,j}\right)$$

$$\mathcal{B}^{+}\mathcal{A}^{-}\Delta Q_{i-1/2,j} = v_{i,j+1/2}^{+}u_{i-1/2,j}^{-}\left(Q_{ij} - Q_{i,j-1}\right)$$

These are then used to update the correction fluxes $\tilde{G}$

Similarly a y-sweep to update the $\tilde{F}$ terms, also including the transverse fluxes.

Then to High-Resolution

For the extension to high-resolution, all that is needed now is to add additional correction terms to $\tilde{F}$ and $\tilde{G}$:

$$\tilde{F}_{i-1/2,j} := \tilde{F}_{i-1/2,j} + \frac{1}{2}|u_{i-1/2,j}|\left(1 - \frac{\Delta t}{\Delta x}|u_{i-1/2,j}|\right)\tilde{\mathcal{W}}_{i-1/2,j}$$

$$\tilde{G}_{i,j-1/2} := \tilde{G}_{i,j-1/2} + \frac{1}{2}|v_{i,j-1/2}|\left(1 - \frac{\Delta t}{\Delta y}|v_{i,j-1/2}|\right)\tilde{\mathcal{W}}_{i,j-1/2}$$

Once again, $\tilde{\mathcal{W}}_{i,j-1/2}$ represents the appropriately slope-limited version of the wave.

For the advection equation the waves are simply

$$\mathcal{W}_{i-1/2,j} = Q_{ij} - Q_{i-1,j}$$

$$\mathcal{W}_{i,j-1/2} = Q_{ij} - Q_{i,j-1}$$

Then to nonlinear conservation laws:

The scheme just developed works for nonlinear conservation laws as well. The waves are

$$\mathcal{W}_{i-1/2,j} = Q_{ij} - Q_{i-1,j}$$

$$\mathcal{W}_{i,j-1/2} = Q_{ij} - Q_{i,j-1}$$

the speeds are

$$s_{i-1/2,j} = \begin{cases} \left[f(Q_{ij}) - f(Q_{i-1,j})\right]/(Q_{ij} - Q_{i-1,j}) & \text{if } Q_{i-1,j} \neq Q_{ij} \\ f'(Q_{ij}) & \text{if } Q_{i-1,j} = Q_{ij} \end{cases}$$

$$s_{i,j-1/2} = \begin{cases} \left[g(Q_{ij}) - g(Q_{i,j-1})\right]/(Q_{ij} - Q_{i,j-1}) & \text{if } Q_{i,j-1} \neq Q_{ij} \\ g'(Q_{ij}) & \text{if } Q_{i,j-1} = Q_{ij} \end{cases}$$

the fluctuations are

$$\mathcal{A}^{\pm}\Delta Q_{i-1/2,j} = s^{\pm}_{i-1/2,j}\mathcal{W}_{i-1/2,j}, \quad \mathcal{B}^{\pm}\Delta Q_{i,j-1/2} = s^{\pm}_{i,j-1/2}\mathcal{W}_{i,j-1/2}$$

and the second-order correction terms are

$$\tilde{F}_{i-1/2,j} := \tilde{F}_{i-1/2,j} + \frac{1}{2}|s_{i-1/2,j}|\left(1 - \frac{\Delta t}{\Delta x}|s_{i-1/2,j}|\right)\tilde{\mathcal{W}}_{i-1/2,j}$$

$$\tilde{G}_{i,j-1/2} := \tilde{G}_{i,j-1/2} + \frac{1}{2}|s_{i,j-1/2}|\left(1 - \frac{\Delta t}{\Delta y}|s_{i,j-1/2}|\right)\tilde{\mathcal{W}}_{i,j-1/2}$$

with a similar, but slightly different recipe for the transverse fluxes…

Multidimensional Systems of Equations
(Chapter 21 in Leveque)

Constant coefficient linear system

We look first at the linear system, with A and B constant (though noncommutative) matrices:

$$q_t + Aq_x + Bq_y = 0$$

and we will adopt the fluctuation form for the update:

$$Q_{ij}^{n+1} = Q_{ij}^n - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^+\Delta Q_{i-1/2,j} + \mathcal{A}^-\Delta Q_{i+1/2,j}\right) - \frac{\Delta t}{\Delta y}\left(\mathcal{B}^+\Delta Q_{i,j-1/2} + \mathcal{B}^-\Delta Q_{i,j+1/2}\right)$$

$$- \frac{\Delta t}{\Delta x}\left(\tilde{F}_{i+1/2,j} - \tilde{F}_{i-1/2,j}\right) - \frac{\Delta t}{\Delta y}\left(\tilde{G}_{i,j+1/2} - \tilde{G}_{i,j-1/2}\right)$$

so we will need the fluctuations themselves and the correction fluxes.

Wave-propagation approach

The algorithm goes as follows:

1. Initialize $\tilde{F}_{i-1/2,j} := 0$ and $\tilde{G}_{i,j-1/2} := 0 \qquad \forall i,j$

2. Sweep through in x, solving each Riemann problem. Get the waves and speeds and compute the fluctuations. For the constant-coefficient linear case, the fluctuations are

$$\mathcal{A}^{\pm}\Delta Q_{i-1/2,j} = \sum_{p=1}^{m}\left(s^{p}_{i-1/2,j}\right)^{\pm}\mathcal{W}^{p}_{i-1/2,j} = A^{\pm}\Delta Q_{i-1/2,j}$$

3. With slope-limited waves, the correction fluxes are updated:

$$\tilde{F}_{i-1/2,j} := \tilde{F}_{i-1/2,j} + \frac{1}{2}\sum_{p=1}^{m}\left|s^{p}_{i-1/2,j}\right|\left(1 - \frac{\Delta t}{\Delta x}\left|s^{p}_{i-1/2,j}\right|\right)\tilde{\mathcal{W}}^{p}_{i-1/2,j}$$

4. As in the scalar case, right-going transverse fluctuations are defined:

$$\mathcal{B}^{\pm}\mathcal{A}^{+}\Delta Q_{i-1/2,j} = B^{\pm}A^{+}(Q_{ij} - Q_{i-1,j})$$

Wave-propagation approach, continued

5. Use the fluctuations $\mathcal{B}^{\pm}\mathcal{A}^{+}\Delta Q_{i-1/2,j}$ to update the correction fluxes above and below the current cell:

$$\tilde{G}_{i,j+1/2} := \tilde{G}_{i,j+1/2} - \frac{\Delta t}{2\Delta x}\mathcal{B}^{+}\mathcal{A}^{+}\Delta Q_{i-1/2,j}, \; \tilde{G}_{i,j-1/2} := \tilde{G}_{i,j-1/2} - \frac{\Delta t}{2\Delta x}\mathcal{B}^{-}\mathcal{A}^{+}\Delta Q_{i-1/2,j}$$

6. Now left-going transverse fluctuations are defined and used to update the correction fluxes above and below the *previous* cell:

$$\tilde{G}_{i-1,j+1/2} := \tilde{G}_{i-1,j+1/2} - \frac{\Delta t}{2\Delta x}\mathcal{B}^{+}\mathcal{A}^{-}\Delta Q_{i-1/2,j}, \; \tilde{G}_{i-1,j-1/2} := \tilde{G}_{i-1,j-1/2} - \frac{\Delta t}{2\Delta x}\mathcal{B}^{-}\mathcal{A}^{-}\Delta Q_{i-1/2,j}$$

7. Sweep through in y, solving each Riemann problem as in steps 2-6. Get the waves and speeds and compute the fluctuations $\mathcal{B}^{\pm}\Delta Q_{i,j-1/2}$ Split these waves transversely to modify the $\tilde{F}$ correction fluxes.

8. Finally, apply the updating formula:

$$Q^{n+1}_{ij} = Q^{n}_{ij} - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^{+}\Delta Q_{i-1/2,j} + \mathcal{A}^{-}\Delta Q_{i+1/2,j}\right) - \frac{\Delta t}{\Delta y}\left(\mathcal{B}^{+}\Delta Q_{i,j-1/2} + \mathcal{B}^{-}\Delta Q_{i,j+1/2}\right)$$

$$- \frac{\Delta t}{\Delta x}\left(\tilde{F}_{i+1/2,j} - \tilde{F}_{i-1/2,j}\right) - \frac{\Delta t}{\Delta y}\left(\tilde{G}_{i,j+1/2} - \tilde{G}_{i,j-1/2}\right)$$

Så enkelt er det!

We're basically done.

Leveque then goes on to describe how the two-dimensional algorithm is implemented in Clawpack, and illustrates examples for the acoustic equations (as a linear system) and the shallow-water equations (as a nonlinear system).

There are some examples you can run in the code you have already, and I'll attempt some more conversions from Clawpack 4.3 in the next few days.

There's just one small thing more to be concerned with, before we do our projects, and that's some new aspects of boundary conditions that apply in multi-dimensional systems.
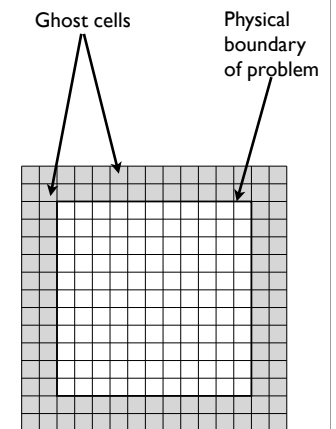
Boundary conditions

Boundary conditions are handled in two or more dimensions in much the same way as they are in one dimension - by using ghost cells in all dimensions. Note that the corners need to be taken care of and properly filled.

Periodic: simply copy the data from the other side of the grid (doubly periodic maps a torus!).

For **Solid Walls**, it is the velocity *normal* to the wall that is negated in the ghost cells (the x velocity along the right and left edges and the y velocity along the bottom and top edges) while the other quantities are copied from the neighboring interior cells. In particular, any *tangential* velocity (*free slip*) is allowed.

These boundary conditions, and simple extrapolation, are implemented in `clawpack/2d/lib/bc2.f`
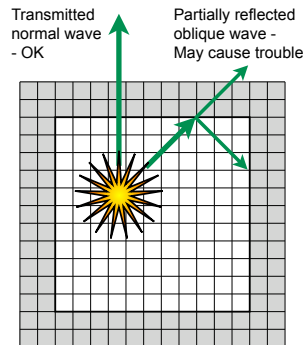
Ghost cells   Physical boundary of problem

A two-dimensional grid with a border of two ghost cells on all boundaries

Boundary conditions - Extrapolation

Extrapolation of outgoing waves is also included in `clawpack/2d/lib/bc2.f`, and often works fairly well. It works perfectly for waves moving normal to the boundaries, but may cause spurious reflections for waves moving *obliquely*, especially in the corners. See the discussion in Leveque section 21.8.5.

The simplest solution is to make the computational box large enough that reflections don't affect the region of interest.

Transmitted normal wave - OK

Partially reflected oblique wave - May cause trouble
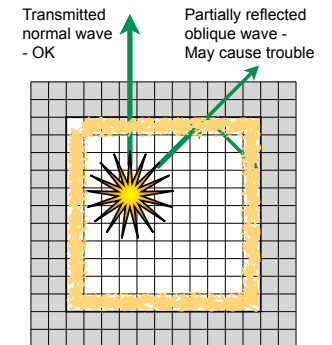

Boundary conditions - Extrapolation

Extrapolation of outgoing waves is also included in `clawpack/2d/lib/bc2.f`, and often works fairly well. It works perfectly for waves moving normal to the boundaries, but may cause spurious reflections for waves moving *obliquely*, especially in the corners. See the discussion in Leveque section 21.8.5.

The simplest solution is to make the computational box large enough that reflections don't affect the region of interest.

Transmitted normal wave - OK

Partially reflected oblique wave - May cause trouble

Another possibility is to fill a portion of the interior adjacent to the boundary with material of high impedance so that waves passing through are attenuated. This is tricky, and may result in loss of hyperbolicity. Also this absorbing material will diffuse into the interior and may affect the solution.

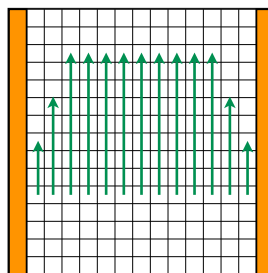## No-Slip Boundaries

Real fluids experience wall friction and form a thin *boundary layer* next to a wall in which the tangential velocity rapidly approaches zero. In this boundary layer, viscosity is important, and there is therefore a loss of hyperbolicity in this region. You can implement no-slip conditions by negating the tangential velocity in the ghost cells, but be careful to watch for non-physical spreading of the resultant vorticity beyond the boundary layer.

No-slip boundaries

What we've accomplished

We started with the one-dimensional advection equation, and developed a Godunov-style REA approach tailored to it;

We realised that systems of linear hyperbolic partial differential equations could be broken down into independent advection equations;

We learned how Riemann problems can be used to advance the solution of such problems from one time step to the next;

We learned that the procedure can be generalised to nonlinear hyperbolic equations and systems of equations;

We learned (have we?) to accept the compromise of *approximate* Riemann solvers for nonlinear systems;

And we've extended what we've learned in one dimension to (at least) two.

Now a tiny bit more about the practicalities of the code -

Project recipe:

Write down all the equations of the problem in conservative form, including closure relations (equations of state, for example). You should also prepare an entropy equation that will be calculable should transonic conditions (or centred rarefactions) arise.

Find the Jacobian of the corresponding quasilinear system, and calculate its eigenvalues and eigenvectors.

For an arbitrary pair of right and left states, solve the Riemann problem, either exactly or approximately. Then write down general formulas for the waves, $\mathcal{W}_{i-1/2}^{p}$, the wave speeds $s^{\pm}$, the fluctuations $\mathcal{A}^{\pm}\Delta Q_{i-1/2}$ and the entropy fix for the transonic case.

Project recipe, continued

Under $CLAW/myclaw, prepare a new directory for the routines you must write. Write the Riemann solver (**rp1.f** or **rp2n.f** and **rp2t.f**) in Fortran in this directory. Copy over $CLAW/util/(**testrp1.f** or **testrp2n.f**), modify it and use it to test your Riemann solver.

Figure out what special work space you need, what boundary conditions, source terms, and other things that you want, and what special variables you need to input or initialise. Then write (or copy and modify) the appropriate routines (**driver**, **setprob**, setaux, **qinit**, bcN, b4stepN, srcN) in the same directory. Write (or copy and modify) a **Makefile** that points to these files, and construct **setrun.py** and **setplot.py** to fill the data files and make the plots.

Finally: compile, run, and check your results.

Examples of potential projects using Clawpack

Euler equations:
    Explosive volcanic eruptions
    High-energy meteor impacts

Shallow-water equations:
    Tsunami in a fjord system or in a basin of varied bathymetry

Dusty gas equations:
    Fluidisation and hydrothermal venting
    Geysers
    Volcanic jets
    Pyroclastic flows
    DeLaval nozzle in a dusty gas

Airy-wave equations:
    Normal (deep or intermediate) water waves
    Pockmarks
    Atmospheric dispersal of contaminants
    Climate patterns

Elastic equations:
    Seismic waves and deformations following impacts or severe earthquakes

Assignment for next time

CAMBRIDGE TEXTS IN APPLIED MATHEMATICS
Finite Volume Methods for Hyperbolic Problems
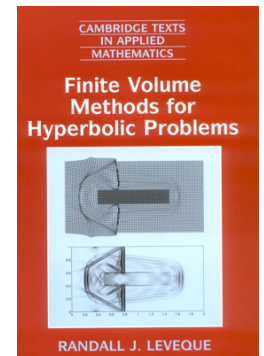RANDALL J. LEVEQUE

Read Chapters 20 & 21.

Based on the project drafts you hand me today, I will prepare some practical advice that I can give you during the course of this week and at next Monday's lecture.

Come talk to me before Thursday regarding anything you would specifically like me to cover (or review) at next Monday's lecture (I don't have anything prepared, yet!); particularly if you have any doubts about how to proceed with your project.

I will be gone 3 - 14 November, so there will be no lecture on 9 November. I will try to remain accessible by email during that time.

At our class meeting on 16 November, I expect to see significant progress on your projects, and will ask several of you to present your progress and difficulties for general discussion. Our last two class meetings, on 23 and 30 November will proceed in similar fashion, as needed.

Final project reports will be due on 7 Dec.

Next: Projects and Reviews