Today:
- Multi-dimensional unsplit methods
- Donor Cell and Corner Transport Upwind
- Variable coefficient advection
- Stream functions
- aux arrays and b4step2.

Monday:
- Multi-dimensional acoustics and elasticity

Reading: Chapter 21

2d finite volume method for \( q_t + f(q)_x + g(q)_y = 0 \)

Evolution of total mass due to fluxes through cell edges:

\[
\frac{d}{dt} \int_{C_i} q(x, y, t) \, dx \, dy = \int_{y_{j-1/2}}^{y_{j+1/2}} f(q(x_{i+1/2}, y, t)) \, dy - \int_{y_{j-1/2}}^{y_{j+1/2}} f(q(x_{i-1/2}, y, t)) \, dy + \int_{x_{i-1/2}}^{x_{i+1/2}} g(q(x, y_{j+1/2}, t)) \, dx - \int_{x_{i-1/2}}^{x_{i+1/2}} g(q(x, y_{j-1/2}, t)) \, dx.
\]

Suggests:

\[
\frac{\Delta x \Delta y Q_{ij}^{n+1} - \Delta x \Delta y Q_{ij}^n}{\Delta t} = -\Delta y[F_{i+1/2,j}^n - F_{i-1/2,j}^n] - \Delta x[G_{i,j+1/2}^n - G_{i,j-1/2}^n].
\]

2d finite volume method for \( q_t + f(q)_x + g(q)_y = 0 \)

\[
\Delta x \Delta y Q_{ij}^{n+1} = \Delta x \Delta y Q_{ij}^n - \Delta t \Delta y[F_{i+1/2,j}^n - F_{i-1/2,j}^n] - \Delta t \Delta x[G_{i,j+1/2}^n - G_{i,j-1/2}^n].
\]

Where we define numerical fluxes:

\[
F_{i-1/2,j}^n \approx \frac{1}{\Delta t \Delta y} \int_{x_{i-1/2}}^{x_{i+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} f(q(x_{i-1/2}, y, t)) \, dy \, dt,
\]

\[
G_{i,j-1/2}^n \approx \frac{1}{\Delta t \Delta x} \int_{y_{j-1/2}}^{y_{j+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} g(q(x, y_{j-1/2}, t)) \, dx \, dt.
\]

Rewrite by dividing by \( \Delta x \Delta y \):

\[
Q_{ij}^{n+1} = Q_{ij}^n - \frac{\Delta t}{\Delta x}[F_{i+1/2,j}^n - F_{i-1/2,j}^n] - \frac{\Delta t}{\Delta y}[G_{i,j+1/2}^n - G_{i,j-1/2}^n].
\]
2d finite volume method

\[ Q_{ij}^{n+1} = Q_{ij}^n - \frac{\Delta t}{\Delta x} [F_{i+1/2,j}^n - F_{i-1/2,j}^n] - \frac{\Delta t}{\Delta y} [G_{i,j+1/2}^n - G_{i,j-1/2}^n]. \]

Fluctuation form:

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

The \( \tilde{F} \) and \( \tilde{G} \) are correction fluxes to go beyond Godunov’s upwind method.

Incorporate approximations to second derivative terms in each direction (\( q_{xx} \) and \( q_{yy} \)) and mixed term \( q_{xy} \).

Advection: Donor Cell Upwind

With no correction fluxes, Godunov’s method for advection is

**Donor Cell Upwind:**

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

Stable only if \( \left| \frac{u \Delta t}{\Delta x} \right| + \left| \frac{v \Delta t}{\Delta y} \right| \leq 1 \).

Advection: Corner Transport Upwind (CTU)

Correction fluxes can be added to advect waves correctly.

**Corner Transport Upwind:**

Stable for \( \max \left( \left| \frac{u \Delta t}{\Delta x} \right|, \left| \frac{v \Delta t}{\Delta y} \right| \right) \leq 1 \).
Advection: Corner Transport Upwind (CTU)

Need to transport triangular region from cell $(i,j)$ to $(i,j+1)$:

\[
\text{Area } = \frac{1}{2}(u \Delta t)(v \Delta t) \implies \left( \frac{1}{2}uv(\Delta t)^2 \right) (Q_{ij} - Q_{i-1,j}).
\]

Accomplished by correction flux:

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

\[
\frac{\Delta t}{\Delta y}(\tilde{G}_{i,j+1/2} - \tilde{G}_{i,j-1/2}) \text{ gives approximation to } \frac{1}{2} \Delta t^2 uv q_{xy}.
\]

\[
\frac{\Delta t}{\Delta x}(\tilde{F}_{i+1/2,j} - \tilde{F}_{i-1/2,j}) \text{ gives similar approximation.}
\]

---

Wave propagation algorithms in 2D

Clawpack requires:

- Normal Riemann solver `rpn2.f`
  Solves 1d Riemann problem \( q_t + A q_x = 0 \)
  Decomposes \( \Delta Q = Q_{ij} - Q_{i-1,j} \) into \( A^+ \Delta Q \) and \( A^- \Delta Q \).
  For \( q_t + A q_x + B q_y = 0 \), split using eigenvalues, vectors:
    \[
    A = R \Lambda R^{-1} \implies A^- = RA^- R^{-1}, A^+ = RA^+ R^{-1}
    \]
  Input parameter \( i_{xy} \) determines if it’s in \( x \) or \( y \) direction.
  In latter case splitting is done using \( B \) instead of \( A \).
  This is all that’s required for dimensional splitting.

- Transverse Riemann solver `rpt2.f`
  Decomposes \( A^+ \Delta Q \) into \( B^- A^+ \Delta Q \) and \( B^+ A^+ \Delta Q \) by splitting this vector into eigenvectors of \( B \).
  (Or splits vector into eigenvectors of \( A \) if \( i_{xy}=2 \).)

Wave propagation algorithm for \( q_t + A q_x + B q_y = 0 \)

Decompose \( A = A^+ + A^- \) and \( B = B^+ + B^- \).

For \( \Delta Q = Q_{ij} - Q_{i-1,j} \):

---
Variable-coefficient advection

Assume incompressible: $u_x + v_y = 0$.

Same formulas work, but replace $u$ and $v$ by

$$u_{i-1/2,j} = \frac{1}{\Delta y} \int_{y_{i-1/2}}^{y_{i+1/2}} u(x_{i-1/2}, y) \, dy,$$

$$v_{i,j-1/2} = \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} v(x, y_{j-1/2}) \, dx.$$

These satisfy discrete divergence-free property:

$$\frac{1}{\Delta x}(u_{i+1/2,j} - u_{i-1/2,j}) + \frac{1}{\Delta y}(v_{i,j+1/2} - v_{i,j-1/2}) = 0$$

Stream function: $\psi(x, y)$ such that $u = \psi_y, \ v = -\psi_x$.

Then $u_x + v_y = 0$ and contours of $\psi$ are streamlines.

The flux per unit time across any curve $C$ in $x$-$y$ plane is

$$\int_C \nabla \psi(x(s), y(s)) \cdot ((x'(s), y'(s)) \, dx$$

In particular,

$$u_{i-1/2,j} = \frac{1}{\Delta y}(\psi(x_{i-1/2}, y_{j+1/2}) - \psi(x_{i-1/2}, y_{j-1/2})),$$

$$v_{i,j-1/2} = -\frac{1}{\Delta x}(\psi(x_{i+1/2}, y_{j-1/2}) - \psi(x_{i-1/2}, y_{j-1/2})).$$
**Solid body rotation**

Stream function: $\psi(x, y) = \omega(x^2 + y^2)$.

Streamlines are circles about origin.

Velocity field: $u(x, y) = 2\omega y$, $v(x, y) = -2\omega x$.

Solution is periodic with period $\pi/\omega$.

See Figures 20.5, 20.6.

**Swirling flow**

Stream function: $\psi(x, y, t) = \cos(2\pi t)(\sin^2(\pi x) + \cos^2(\pi y))/\pi$.

Variation in time causes reversal of flow.

See $\text{CLAW/apps/advection/2d/swirl}$

**Storing data in aux arrays**

In Clawpack, $q(i,j,m)$, $m=1,\ldots,m\text{eqn}$ holds the solution.

Often there is spatially varying data that describes the problem:

- Edge velocities for advection,
- Density $\rho_0(x,y)$, bulk modulus $K_0(x,y)$ for acoustics,
- Topography or bathymetry for shallow water,
- Edge lengths, angles, and cell areas for mapped grids,

These can be stored in $\text{aux}(i,j,m)$, $m=1,2,\ldots,m\text{aux}$.

The Fortran function $\text{setaux}$ is called every time a new grid is created (when AMR is used).

To use this, copy library version (which does nothing) to application directory and modify this file and Makefile.
The `setaux` function is only called when grids are created.

The `b4stepN` function (in N dimensions) is called before each time step.

Can use this for example to:
- Change aux arrays for time-dependent velocities,
- Print something out every time step (e.g. total mass),

To use this, copy library version (which does nothing) to application directory and modify this file and `Makefile`.

See:
- `$CLAW/apps/advection/2d/swirl/b4step2.f`
- `$CLAW/apps/advection/2d/swirl/setaux.f`
- `$CLAW/apps/advection/2d/swirl/psi.f`