AMath 574 February 4, 2011

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:

$$\begin{aligned}
\frac{d}{dt}\iint_{\mathcal{C}_{ij}} q(x,y,t)\,dx\,dy = &\int_{y_{j-1/2}}^{y_{j+1/2}} f(q(x_{i+1/2},y,t)\,dy \\
&- \int_{y_{j-1/2}}^{y_{j+1/2}} f(q(x_{i-1/2},y,t)\,dy \\
&+ \int_{x_{i-1/2}}^{x_{i+1/2}} g(q(x,y_{j+1/2},t)\,dx \\
&- \int_{x_{i-1/2}}^{x_{i+1/2}} g(q(x,y_{j-1/2},t)\,dx.
\end{aligned}$$

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

Evolution of total mass due to fluxes through cell edges:

$$
\begin{aligned}
\frac{d}{dt}\iint_{\mathcal{C}_{ij}} q(x,y,t)\,dx\,dy = &\int_{y_{j-1/2}}^{y_{j+1/2}} f(q(x_{i+1/2},y,t)\,dy \\
&- \int_{y_{j-1/2}}^{y_{j+1/2}} f(q(x_{i-1/2},y,t)\,dy \\
&+ \int_{x_{i-1/2}}^{x_{i+1/2}} g(q(x,y_{j+1/2},t)\,dx \\
&- \int_{x_{i-1/2}}^{x_{i+1/2}} g(q(x,y_{j-1/2},t)\,dx.
\end{aligned}
$$

Suggests:

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

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

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

Where we define numerical fluxes:

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

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

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:

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

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

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

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

Fluctuation form:

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

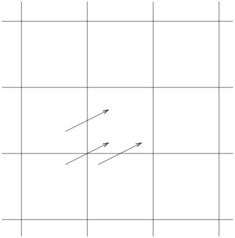
The $\tilde{F}$ and $\tilde{G}$ are correction fluxes to go beyond Godunov's upwind method.

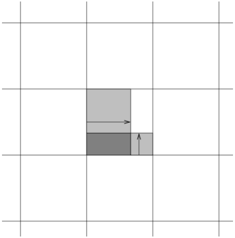
Incorporate approximations to second derivative terms in each direction (q_{xx} and q_{yy}) and mixed term q_{xy}.

Advection: Donor Cell Upwind

With no correction fluxes, Godunov's method for advection is Donor Cell Upwind:

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

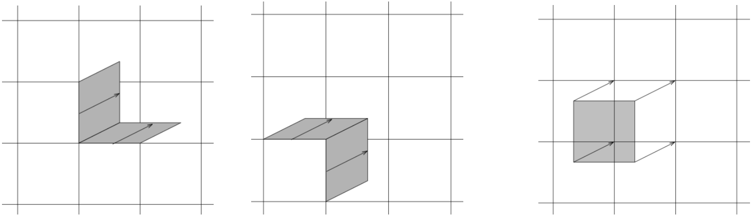




Stable only if $\left|\frac{u\Delta t}{\Delta x}\right| + \left|\frac{v\Delta t}{\Delta y}\right| \leq 1.$

Correction fluxes can be added to advect waves correctly.

Corner Transport Upwind:



Stable for $\max\left(\left|\frac{u\Delta t}{\Delta x}\right|, \left|\frac{v\Delta t}{\Delta y}\right|\right) \leq 1.$

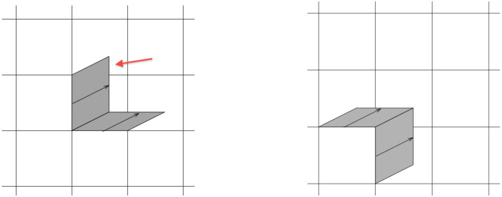
Advection: Corner Transport Upwind (CTU)

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

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

Accomplished by correction flux:

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



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

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

Wave propagation algorithms in 2D

Clawpack requires:

Normal Riemann solver `rpn2.f`
Solves 1d Riemann problem $q_t + Aq_x = 0$
Decomposes $\Delta Q = Q_{ij} - Q_{i-1,j}$ into $\mathcal{A}^+\Delta Q$ and $\mathcal{A}^-\Delta Q$.
For $q_t + Aq_x + Bq_y = 0$, split using eigenvalues, vectors:

$$A = R\Lambda R^{-1} \implies A^- = R\Lambda^- R^{-1}, A^+ = R\Lambda^+ R^{-1}$$

Input parameter `ixy` 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.

Wave propagation algorithms in 2D

Clawpack requires:

Normal Riemann solver `rpn2.f`
Solves 1d Riemann problem $q_t + Aq_x = 0$
Decomposes $\Delta Q = Q_{ij} - Q_{i-1,j}$ into $\mathcal{A}^+\Delta Q$ and $\mathcal{A}^-\Delta Q$.
For $q_t + Aq_x + Bq_y = 0$, split using eigenvalues, vectors:

$$A = R\Lambda R^{-1} \implies A^- = R\Lambda^- R^{-1}, A^+ = R\Lambda^+ R^{-1}$$

Input parameter `ixy` 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.

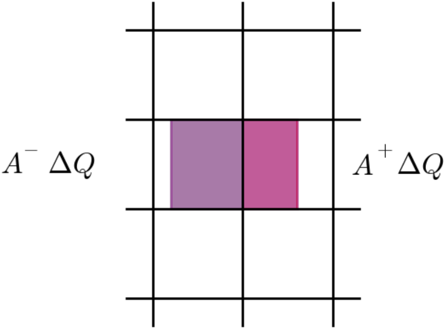
Transverse Riemann solver `rpt2.f`
Decomposes $\mathcal{A}^+\Delta Q$ into $\mathcal{B}^-\mathcal{A}^+\Delta Q$ and $\mathcal{B}^+\mathcal{A}^+\Delta Q$ by splitting this vector into eigenvectors of B.

(Or splits vector into eigenvectors of A if `ixy=2`.)

Wave propagation algorithm for $q_t + Aq_x + Bq_y = 0$

Decompose $A = A^+ + A^-$ and $B = B^+ + B^-$.

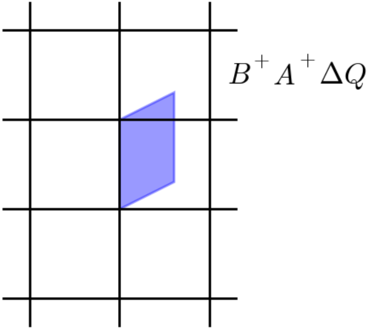
For $\Delta Q = Q_{ij} - Q_{i-1,j}$:



Wave propagation algorithm for $q_t + Aq_x + Bq_y = 0$

Decompose $A = A^+ + A^-$ and $B = B^+ + B^-$.

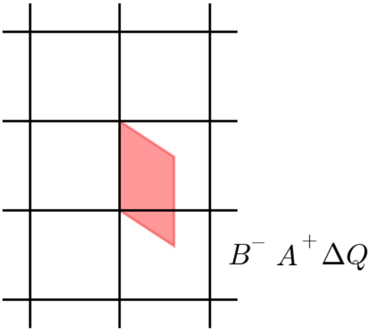
For $\Delta Q = Q_{ij} - Q_{i-1,j}$:



Wave propagation algorithm for $q_t + Aq_x + Bq_y = 0$

Decompose $A = A^+ + A^-$ and $B = B^+ + B^-$.

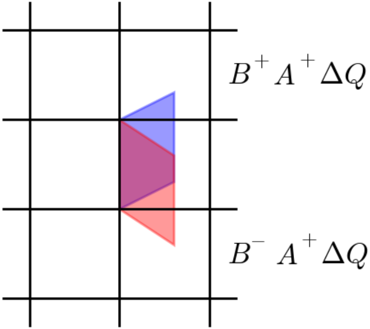
For $\Delta Q = Q_{ij} - Q_{i-1,j}$:



Wave propagation algorithm for $q_t + Aq_x + Bq_y = 0$

Decompose $A = A^+ + A^-$ and $B = B^+ + B^-$.

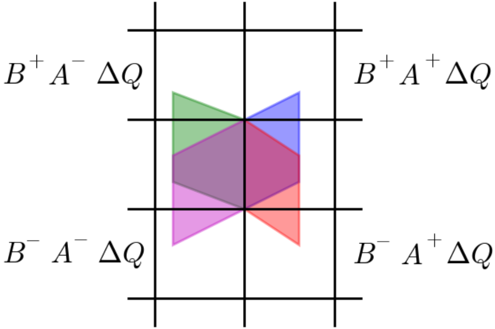
For $\Delta Q = Q_{ij} - Q_{i-1,j}$:



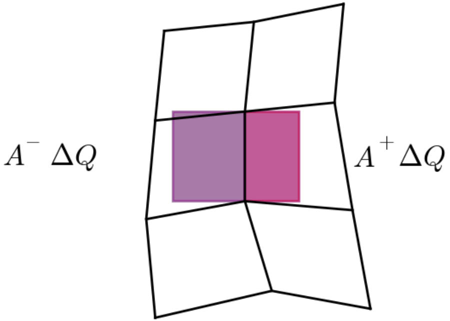
Wave propagation algorithm for $q_t + Aq_x + Bq_y = 0$

Decompose $A = A^+ + A^-$ and $B = B^+ + B^-$.

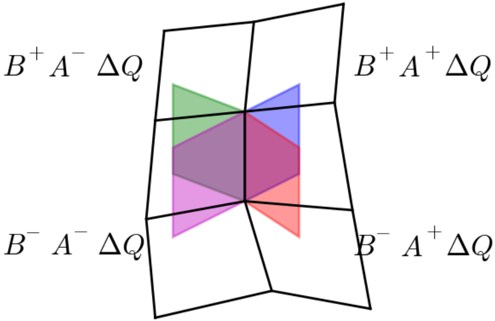
For $\Delta Q = Q_{ij} - Q_{i-1,j}$:



Wave propagation algorithm on a quadrilateral grid



Wave propagation algorithm on a quadrilateral grid



Variable-coefficient advection

Assume incompressible: $u_x + v_y = 0$.

Same formulas work, but replace u and v by

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

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

These satisfy discrete divergence-free property:

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

Variable-coefficient advection

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

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

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

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

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 π/ω.

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 **$CLAW/apps/advection/2d/swirl**

Storing data in `aux` arrays

In Clawpack, `q(i,j,m), m=1,...,meqn` 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,

Storing data in `aux` arrays

In Clawpack, `q(i,j,m), m=1,...,meqn` 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 `aux(i,j,m), m=1,2,...,maux`.

The Fortran function `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`.

Using `b4stepN.f`

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