Clawpack Tutorial
Part I

Randall J. LeVeque
Applied Mathematics
University of Washington

Conservation Laws Package

www.clawpack.org

Slides posted at
http://www.clawpack.org/links/tutorials
http://faculty.washington.edu/rjl/tutorials

(green indicates links)

Outline

Part 1:

- Hyperbolic problems, Finite volume methods
- Riemann problems and Godunov's method
- Downloading and installing
- Makefile, running the code
- Setting parameters in `setrun.py`
- Riemann solvers

Part 2:

- Specifying boundary conditions
- Plotting with the Python modules, `setplot.py`
- Two space dimensions

Part 3:

- GeoClaw for tsunami modeling
- Adaptive mesh refinement

Some links

Clawpack: **http://www.clawpack.org**

If that's not reachable, try:

http://faculty.washington.edu/rjl/clawpack

These slides:

http://faculty.washington.edu/rjl/tutorials

Textbook with many Clawpack examples:
Finite Volume Methods for Hyperbolic Problems, Cambridge University Press, 2003.

http://www.clawpack.org/doc/book

Slides from recent 3-week short course:

http://faculty.washington.edu/rjl/ipde

Some collaborators

Marsha Berger, NYU (AMR, GeoClaw)
David George, USGS CVO
(Tsunamis, debris flows, dam breaks)
Kyle Mandli, UW $\rightarrow$ UT-Austin
(PyClaw, GeoClaw, storm surges)
David Ketcheson, KAUST
(PyClaw, SharpClaw, PetClaw)

Jan Olav Langseth, FFI, Oslo
Donna Calhoun, Boise State
Christiane Helzel, Bochum
Sorin Mitran, UNC

Funded in part by: NSF, DOE, NCAR, NIH, ONR, AFOSR
Founders Term Professorship

First order hyperbolic PDE in 1 space dimension

Linear: $q_t + Aq_x = 0, \qquad q(x,t) \in \mathbb{R}^m,\ A \in \mathbb{R}^{m \times m}$

Conservation law: $q_t + f(q)_x = 0, \qquad f : \mathbb{R}^m \to \mathbb{R}^m$ (flux)

Quasilinear form: $q_t + f'(q)q_x = 0$

Hyperbolic if A or $f'(q)$ is diagonalizable with real eigenvalues.

Models wave motion or advective transport.

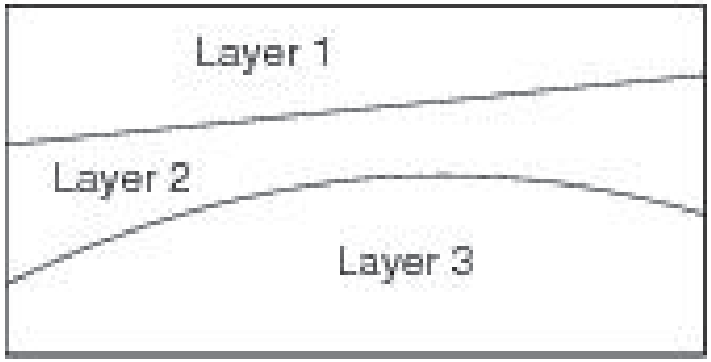
Eigenvalues are wave speeds.

Balance law (source terms): $q_t + f(q)_x = \psi(q)$

Some applications where CLAWPACK has been used

- Aerodynamics, supersonic flows
- Seismic waves, tsunamis, flow on the sphere
- Volcanic flows, dusty gas jets, pyroclastic surges
- Ultrasound, lithotripsy, shock wave therapy
- Plasticity, nonlinear elasticity
- Chemotaxis and pattern formation
- Semiconductor modeling
- Multi-fluids, multi-phase flows, bubbly flow
- Combustion, detonation waves
- Astrophysics: binary stars, planetary nebulae, jets,
- Magnetohydrodynamics, plasmas, relativistic flow
- Numerical relativity — gravitational waves, cosmology

Seismic waves in layered earth



Layers 1 and 3: $\rho = 2, \quad \lambda = 1, \quad \mu = 1, \quad c_p \approx 1.2, \quad c_s \approx 0.7$

Layer 2: $\rho = 5, \quad \lambda = 10, \quad \mu = 5, \quad c_p = 2.0, \quad c_s = 1$

Impulse at top surface at $t = 0$.

Solved on uniform Cartesian grid (600×300).

Cell average of material parameters used in each grid cell.

Extrapolation at computational boundaries.

Equations of linear elasticity (in 2d)

$$
\begin{aligned}
\sigma_t^{11} &- (\lambda + 2\mu)u_x - \lambda v_y &= 0 \\
\sigma_t^{22} &- \lambda u_x - (\lambda + 2\mu)v_y &= 0 \\
\sigma_t^{12} &- \mu(v_x + u_y) &= 0 \\
\rho u_t &- \sigma_x^{11} - \sigma_y^{12} &= 0 \\
\rho v_t &- \sigma_x^{12} - \sigma_y^{22} &= 0
\end{aligned}
$$

where $\lambda(x, y)$ and $\mu(x, y)$ are Lamé parameters.

This has the form $q_t + Aq_x + Bq_y = 0$.

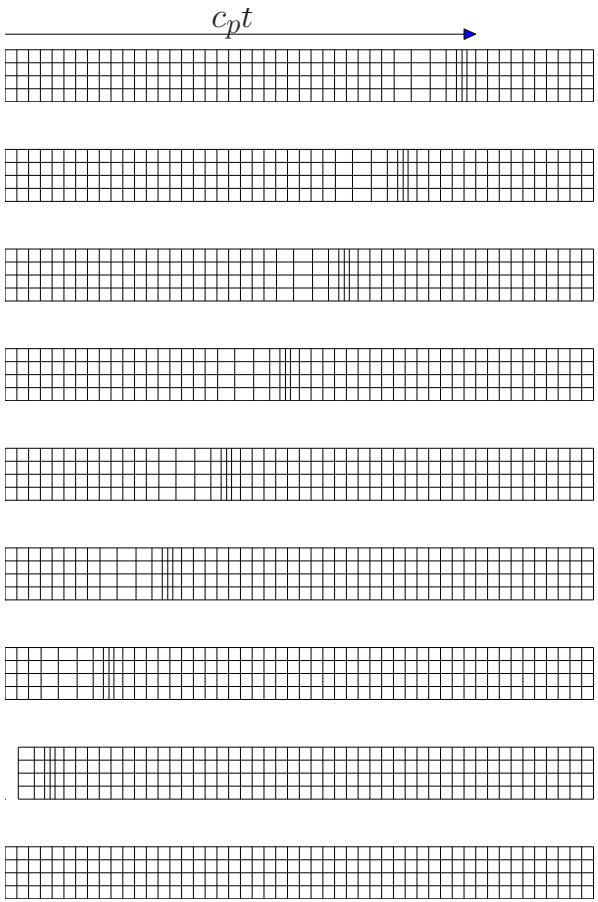
The matrix $(A\cos\theta + B\sin\theta)$ has eigenvalues $-c_p, \quad -c_s, \quad 0, \quad c_s, \quad c_p$

P-wave (dilatational) speed: $c_p = \sqrt{\frac{\lambda+2\mu}{\rho}}$

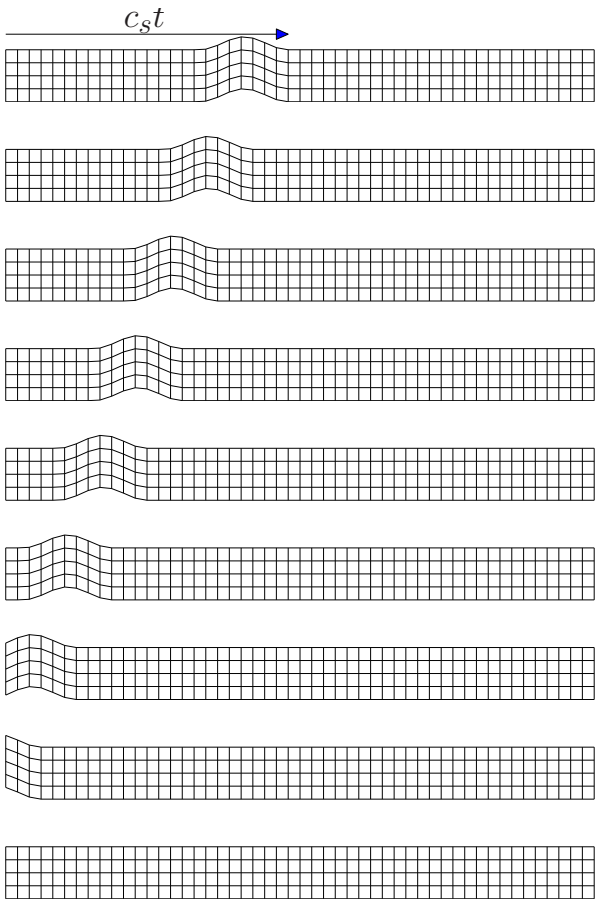
S-wave (shear) speed: $c_s = \sqrt{\frac{\mu}{\rho}}$

Elastic waves

P-waves

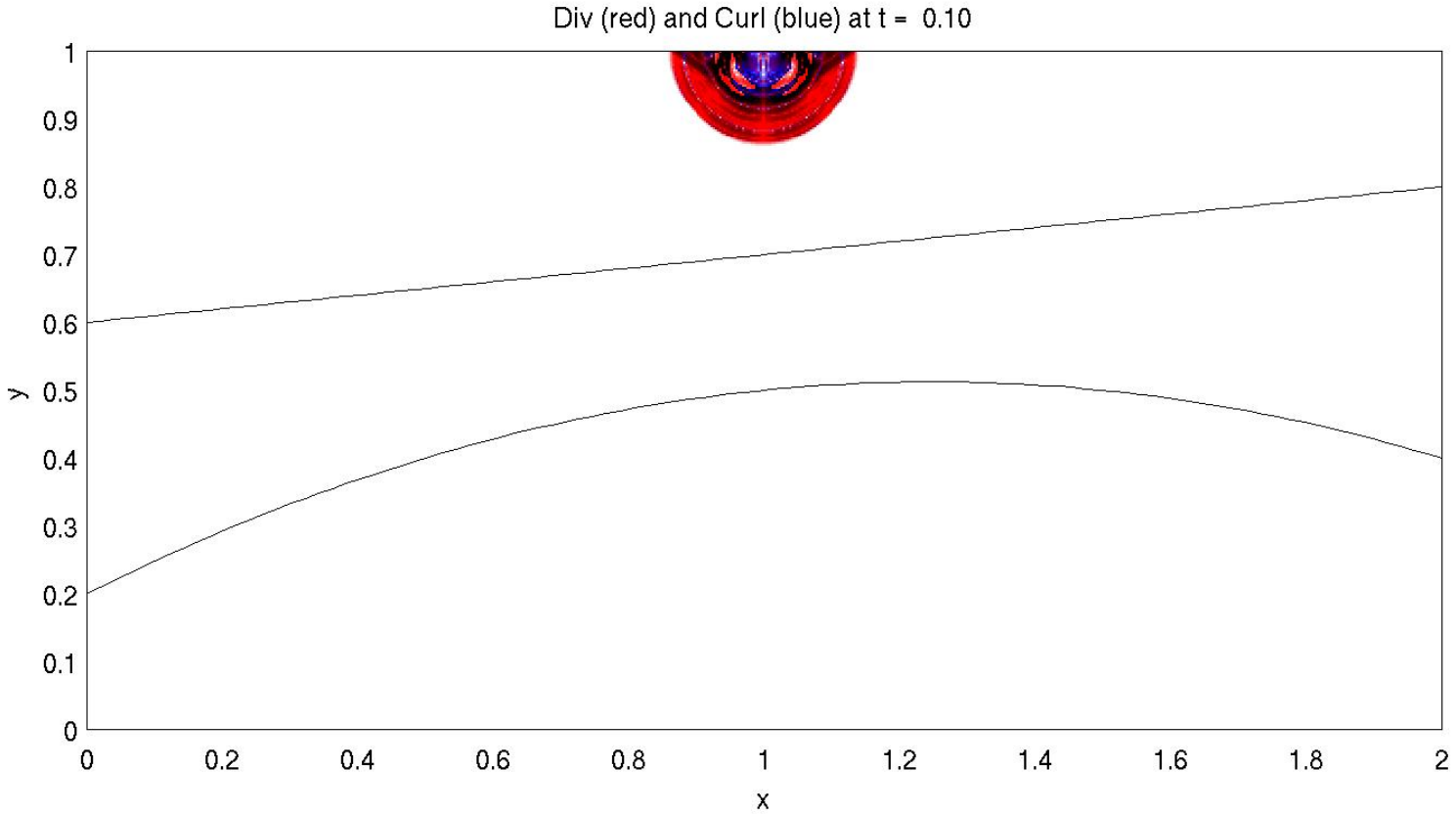


S-waves



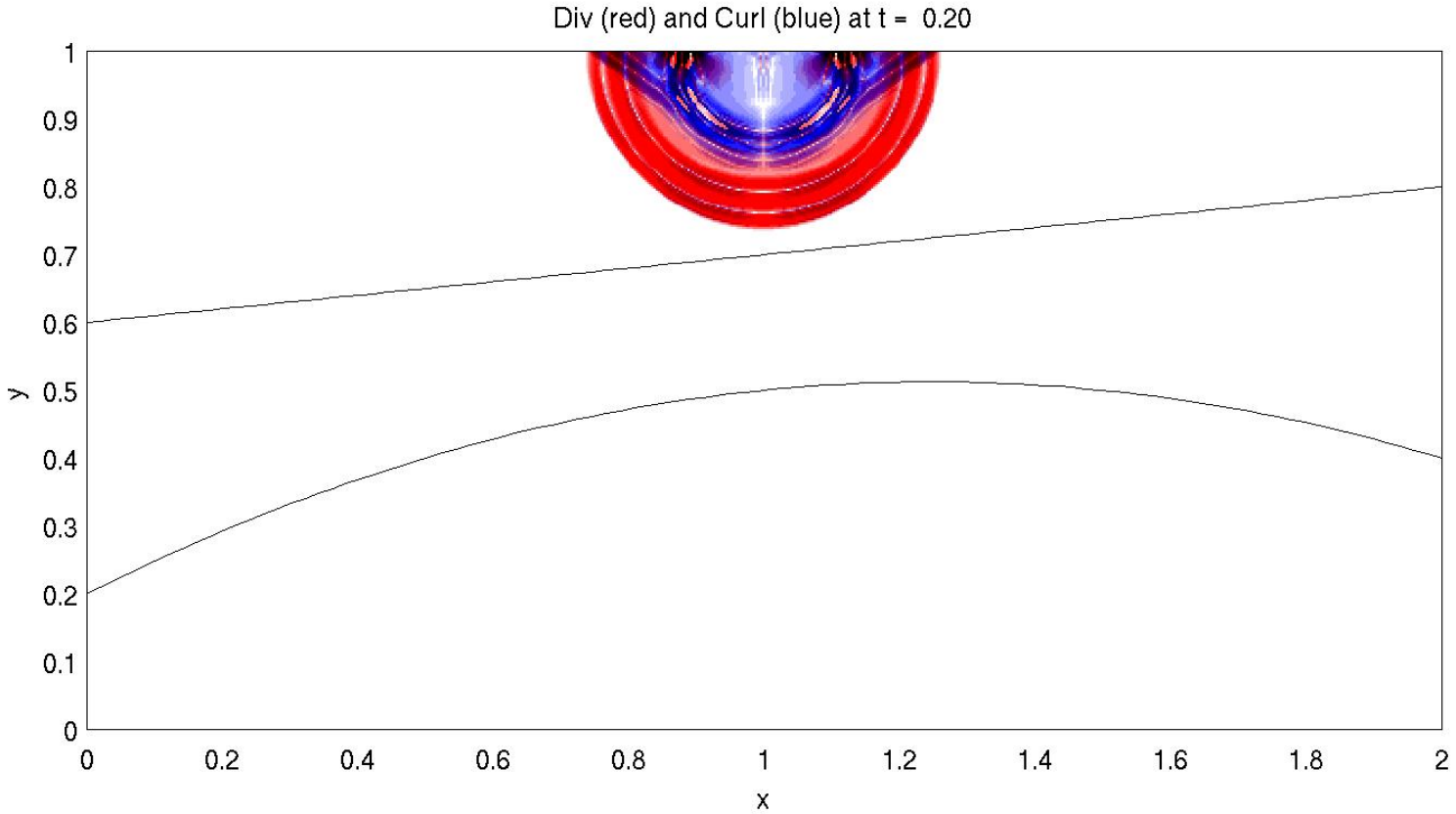
Seismic wave in layered medium

Red = div(u) [P-waves], Blue = curl(u) [S-waves]



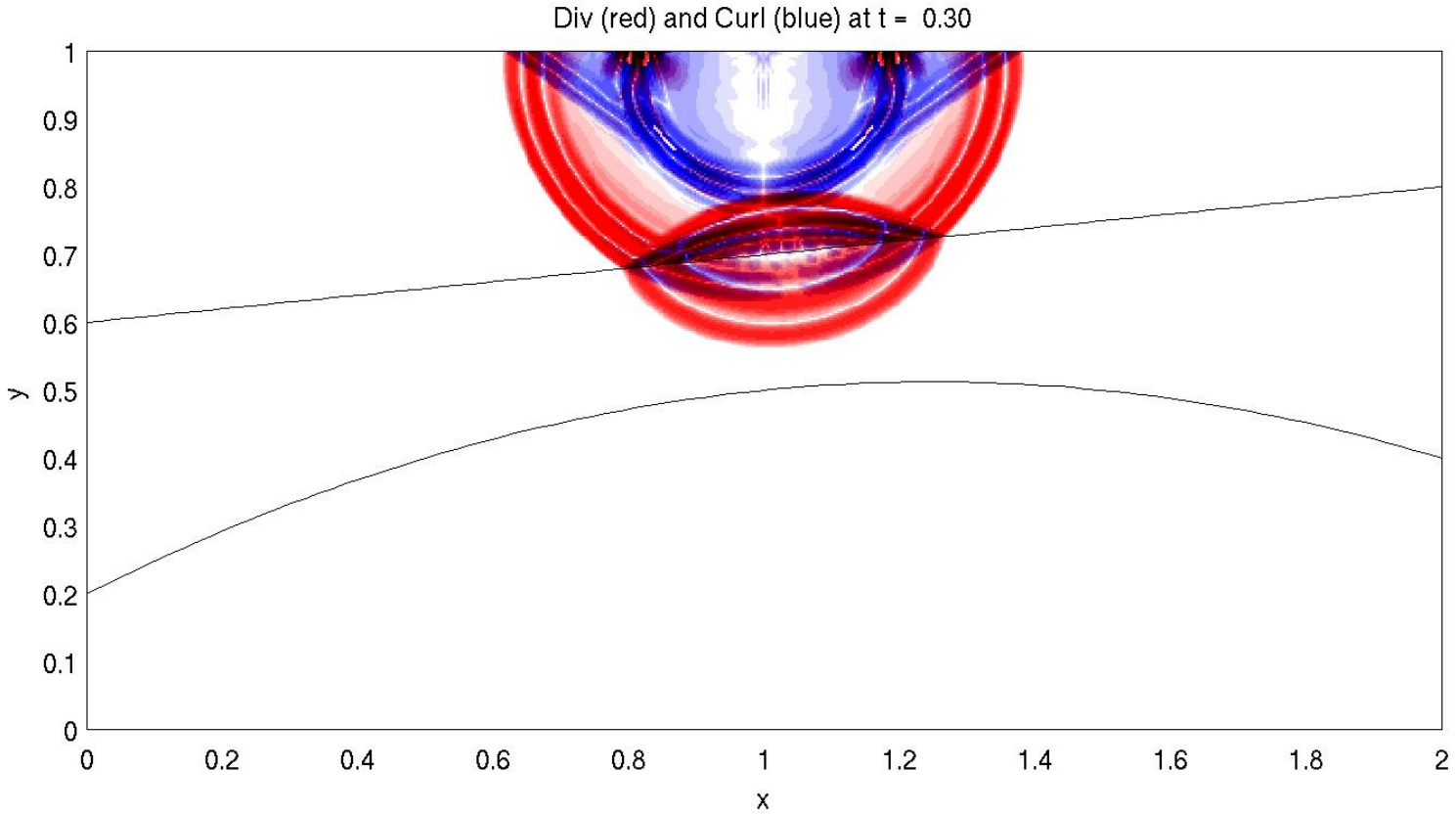
Seismic wave in layered medium

Red = div(u) [P-waves], Blue = curl(u) [S-waves]



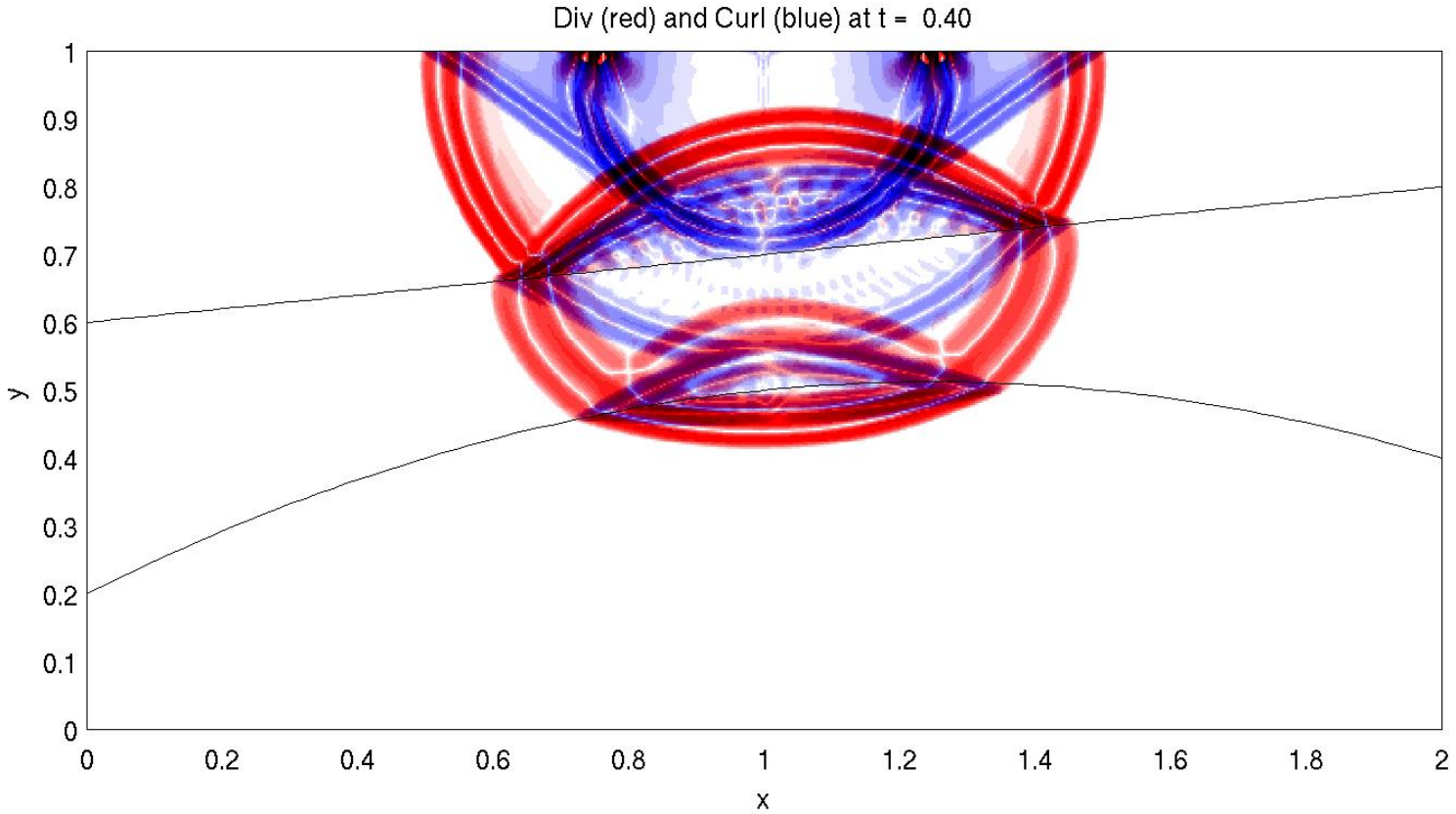
Seismic wave in layered medium

Red = div(u) [P-waves], Blue = curl(u) [S-waves]



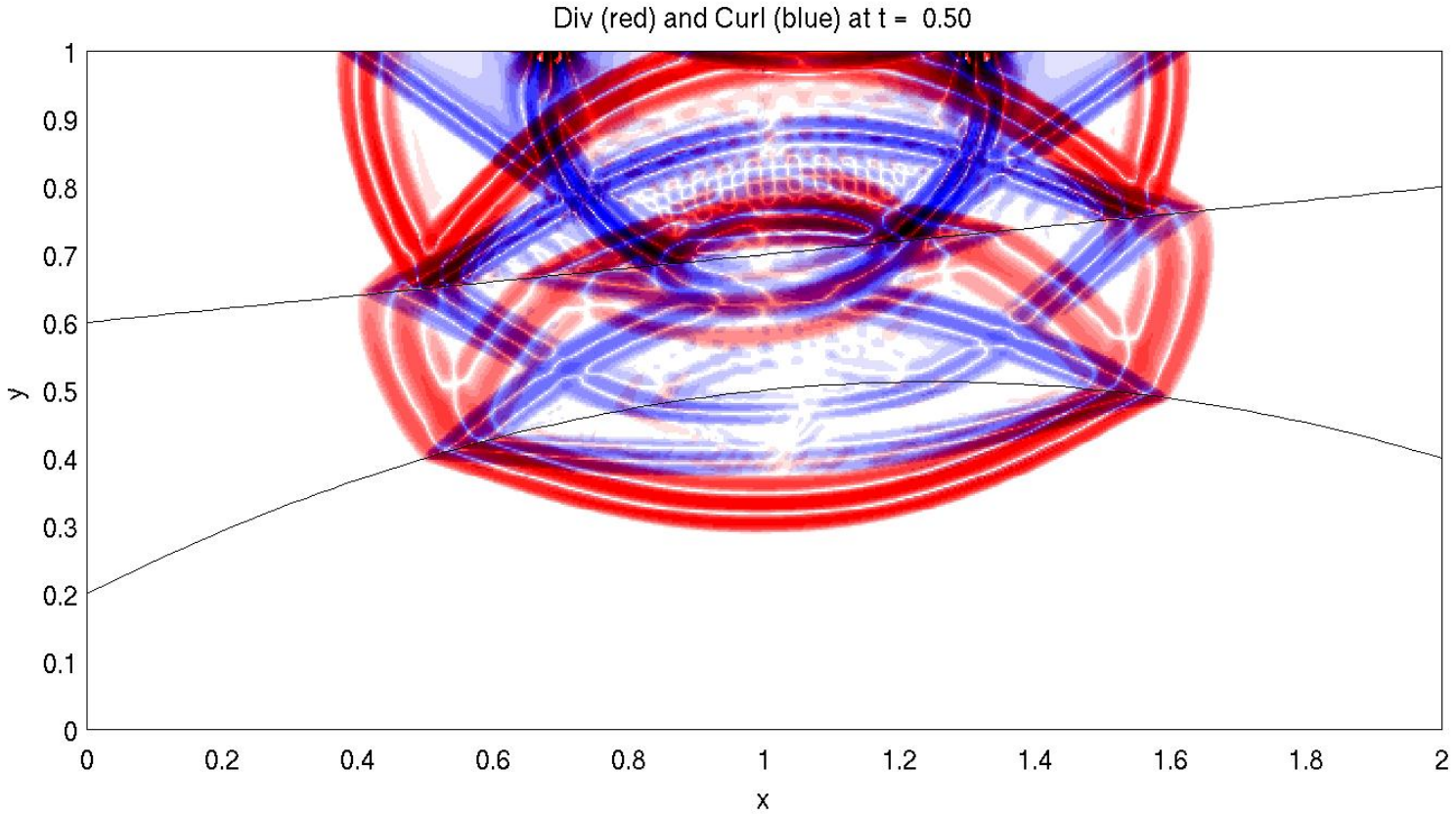
Seismic wave in layered medium

Red = div(u) [P-waves], Blue = curl(u) [S-waves]



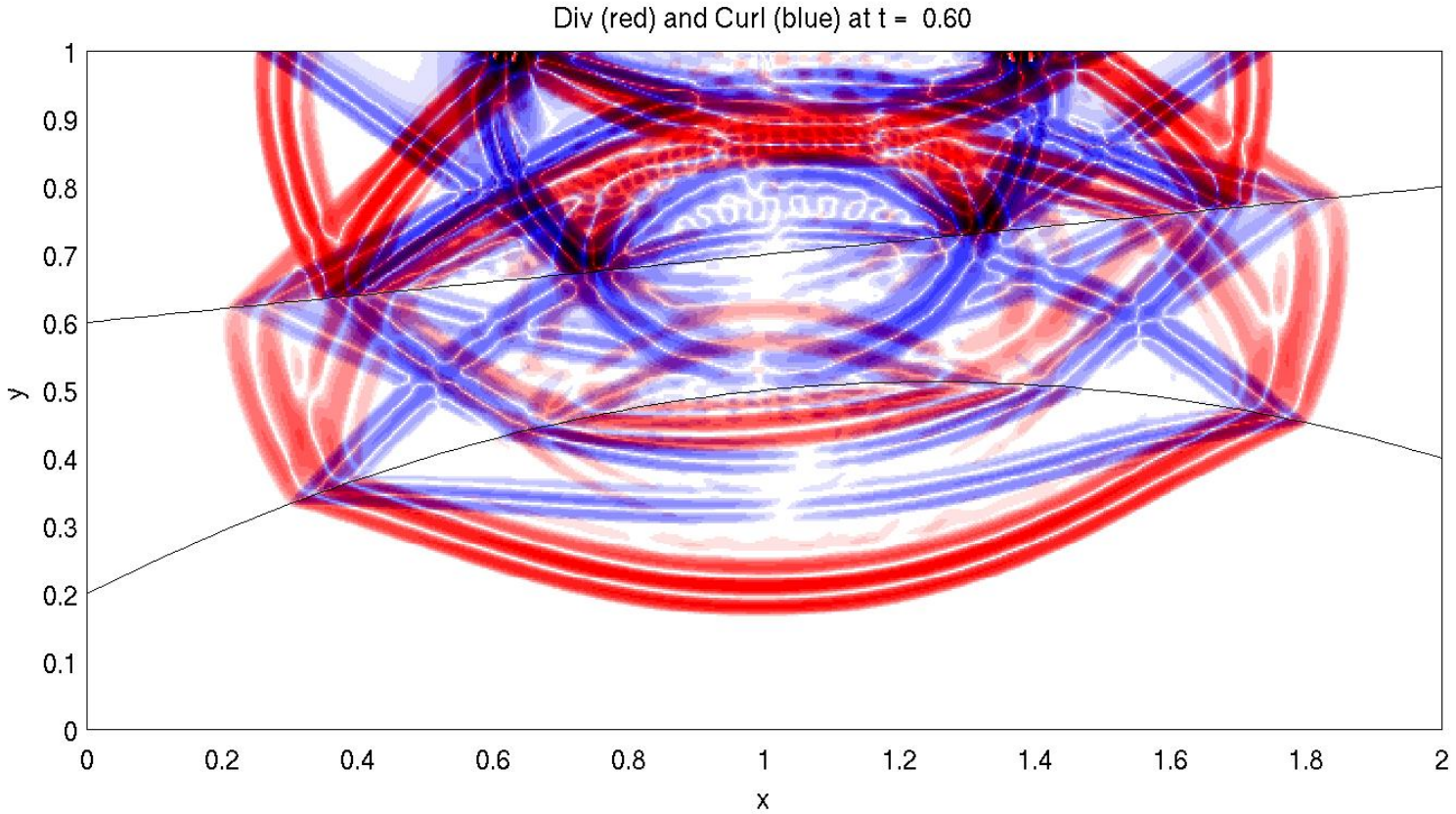
Seismic wave in layered medium

Red = div(u) [P-waves], Blue = curl(u) [S-waves]



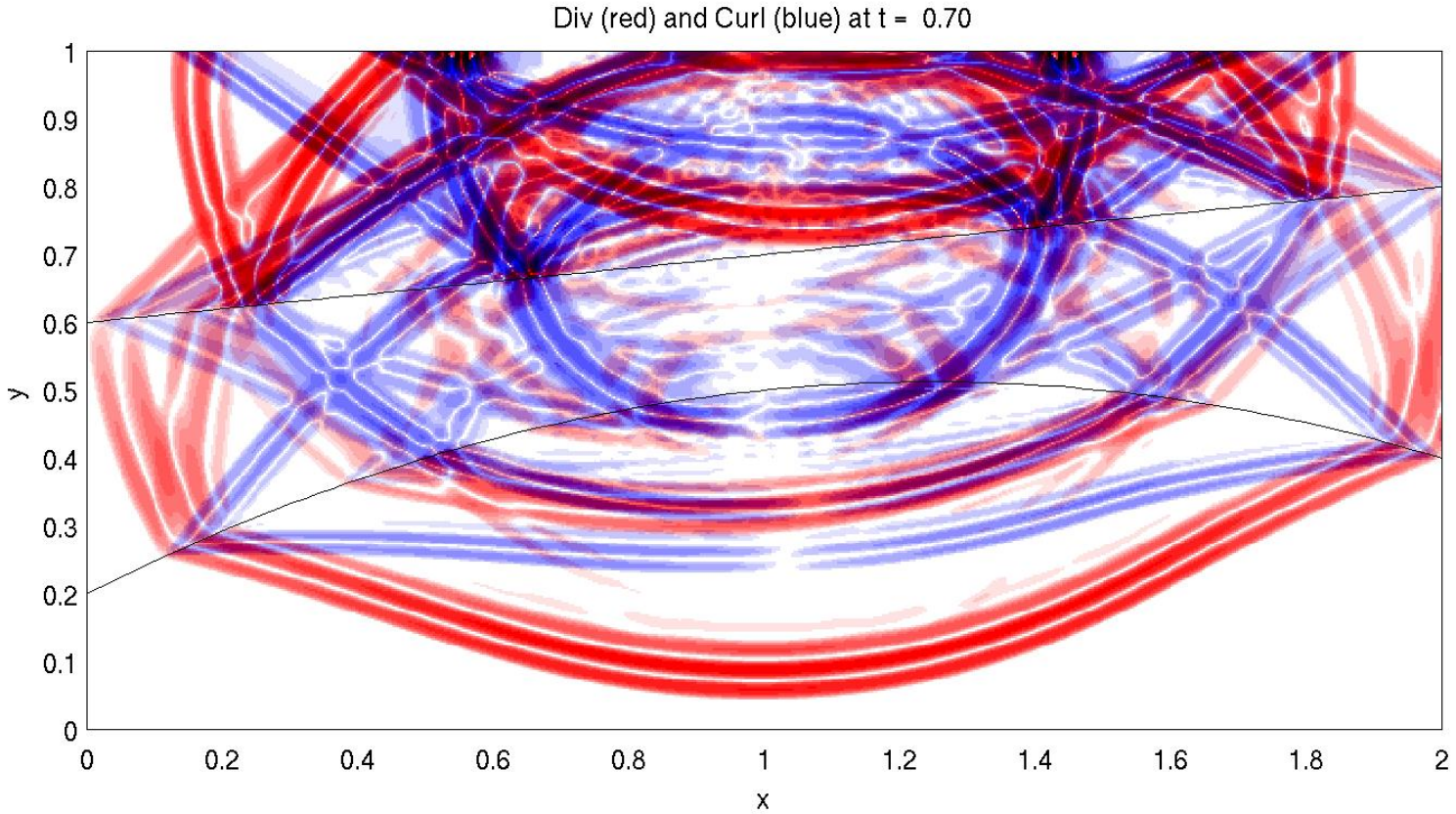
Seismic wave in layered medium

Red = div(u) [P-waves], Blue = curl(u) [S-waves]



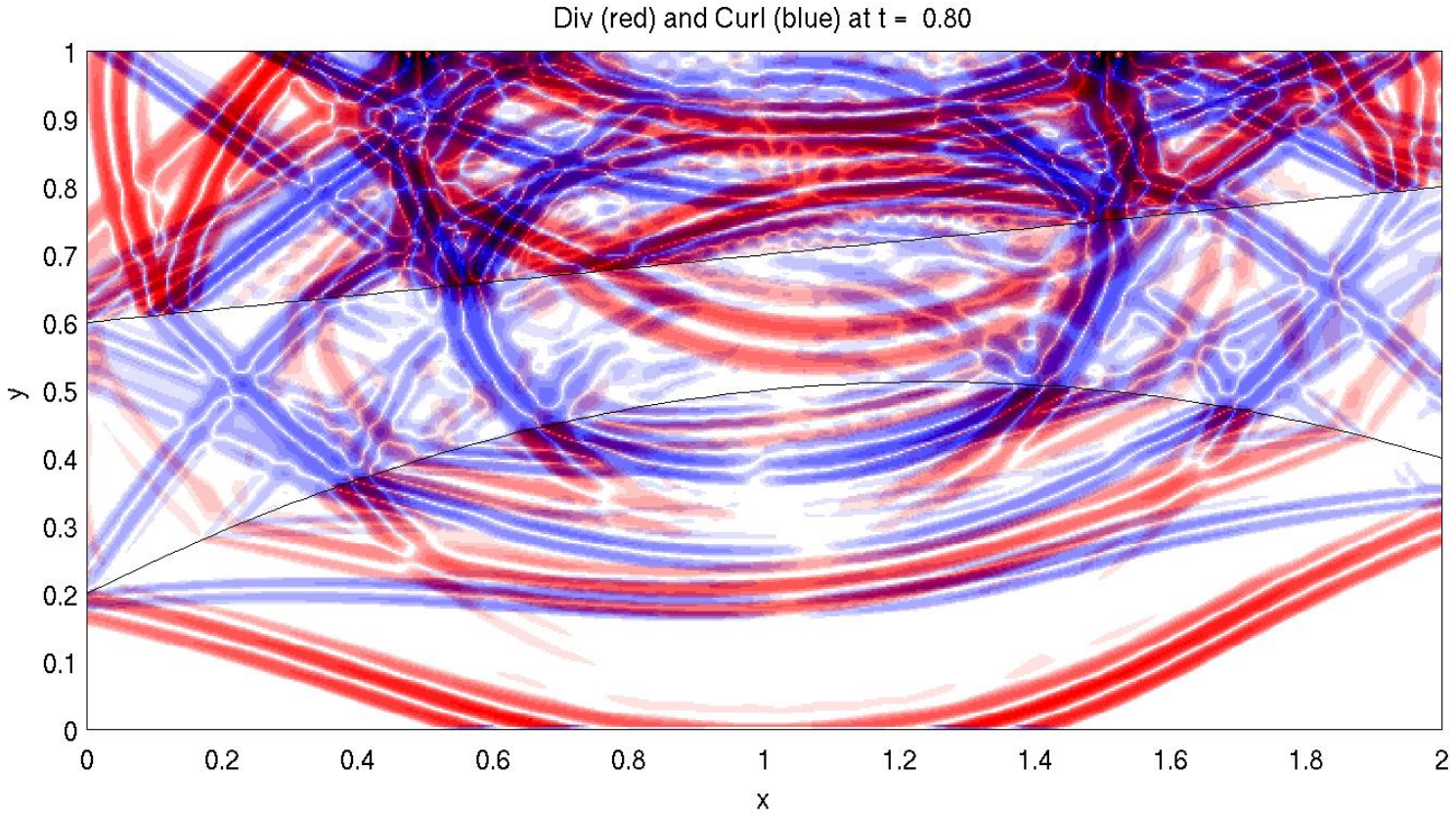
Seismic wave in layered medium

Red = div(u) [P-waves], Blue = curl(u) [S-waves]



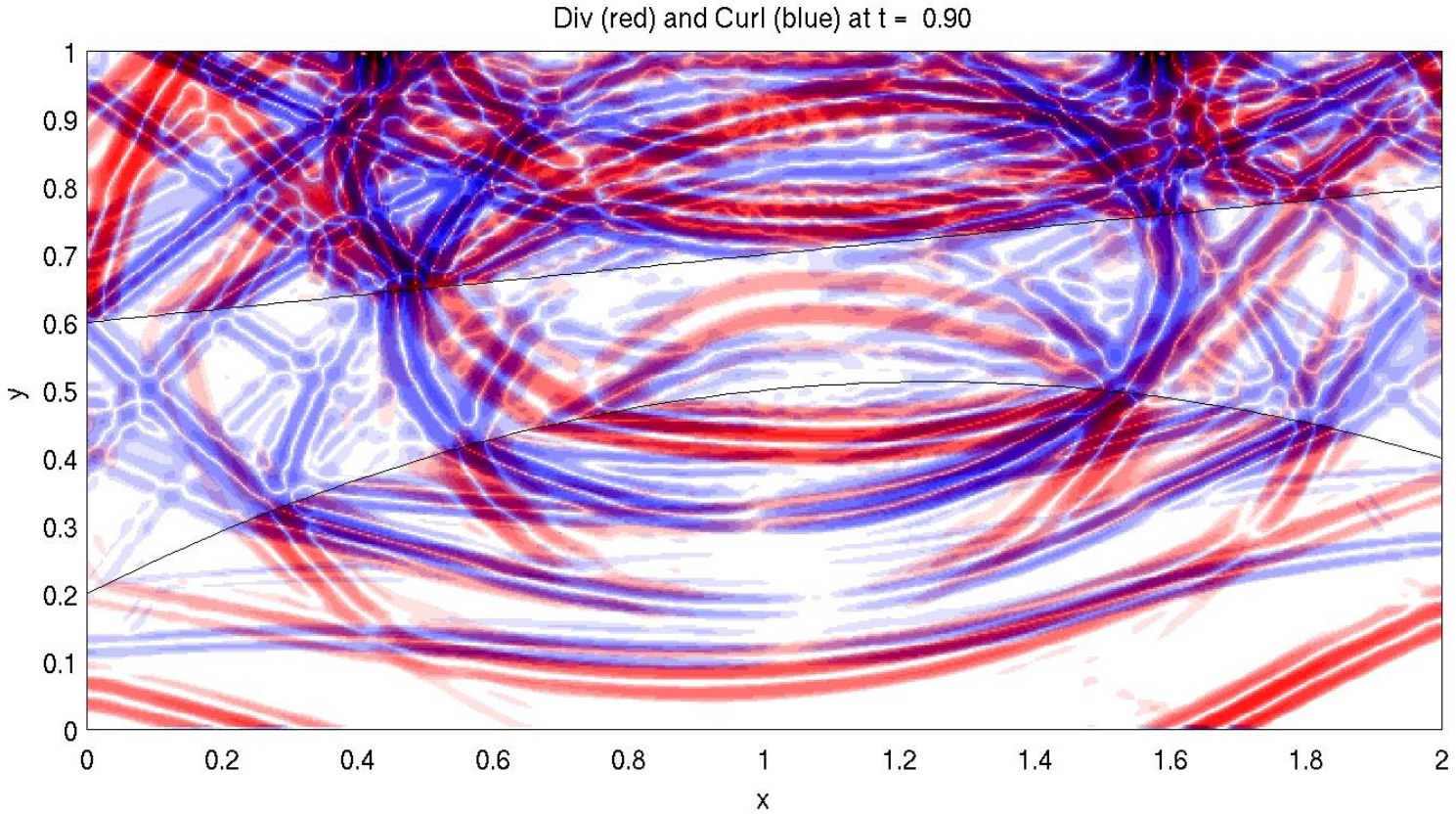
Seismic wave in layered medium

Red = div(u) [P-waves], Blue = curl(u) [S-waves]



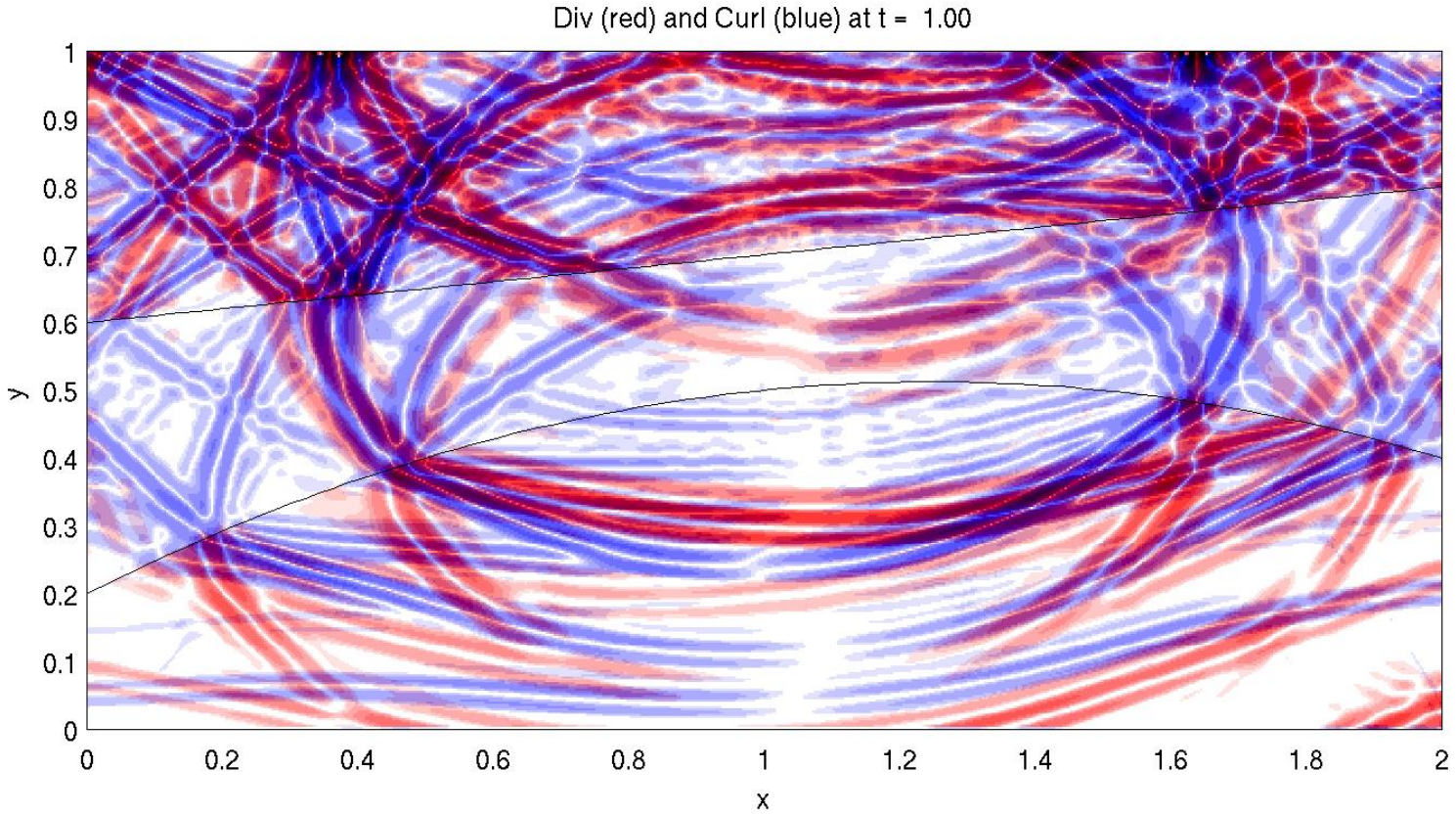
Seismic wave in layered medium

Red = div(u) [P-waves], Blue = curl(u) [S-waves]



Seismic wave in layered medium

Red = div(u) [P-waves], Blue = curl(u) [S-waves]



Adaptive Mesh Refinement (AMR)

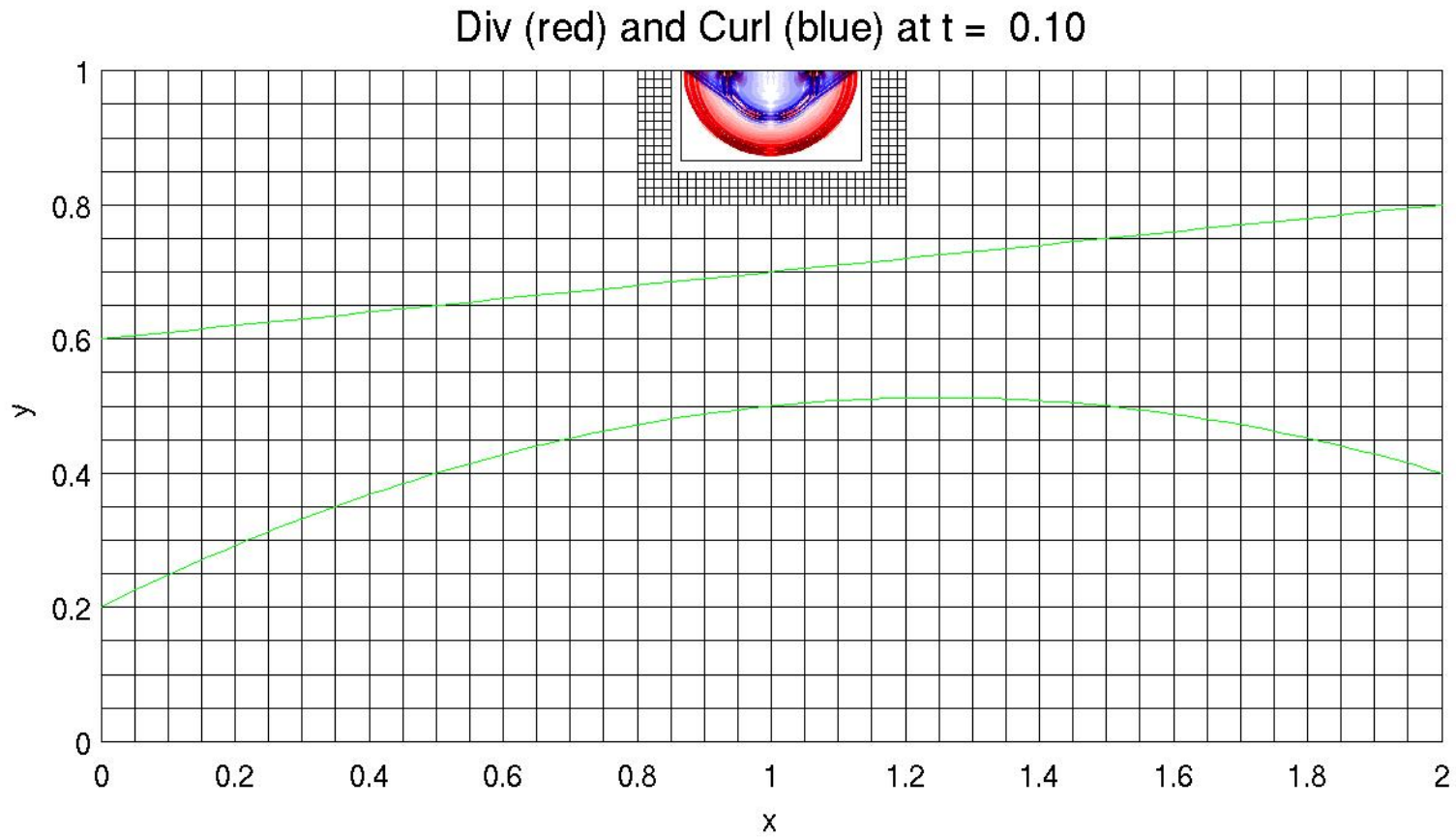
- Cluster grid points where needed
- Automatically adapt to solution
- Refined region moves in time-dependent problem

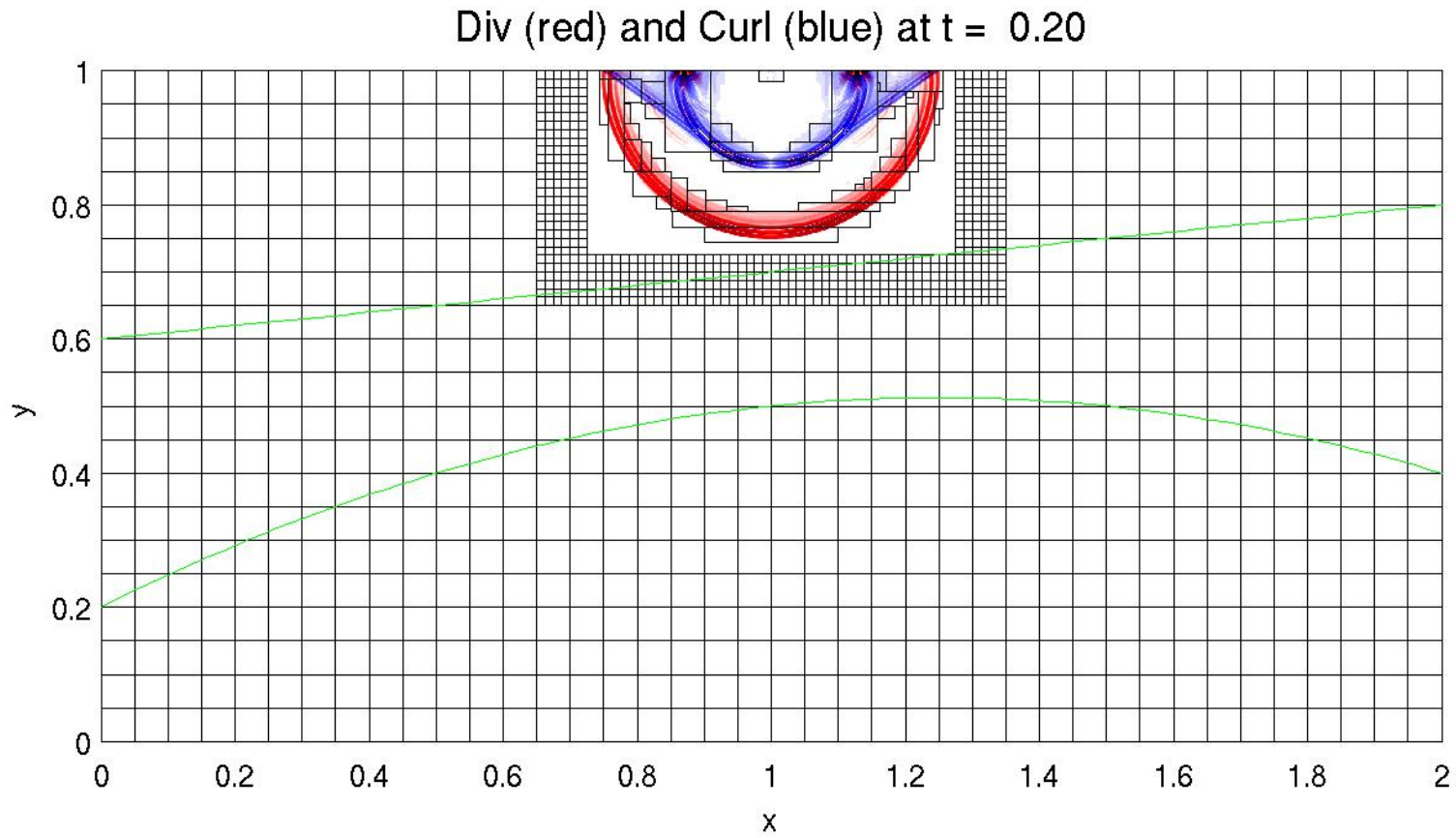
Basic approaches:

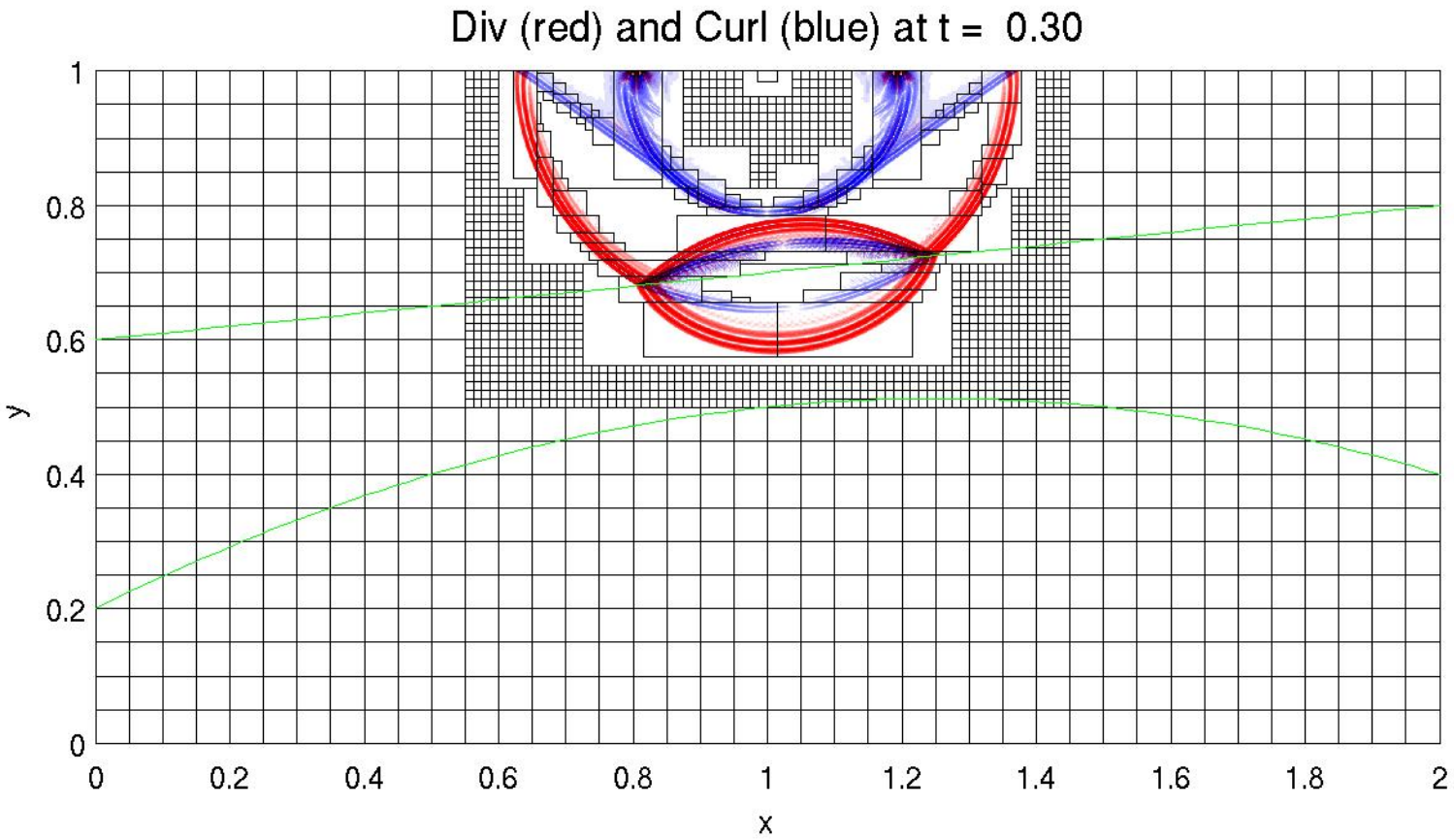
- Cell-by-cell refinement
  Quad-tree or Oct-tree data structure
  Structured or unstructured grid
- Refinement on “rectangular” patches
  Berger-Colella-Oliger style

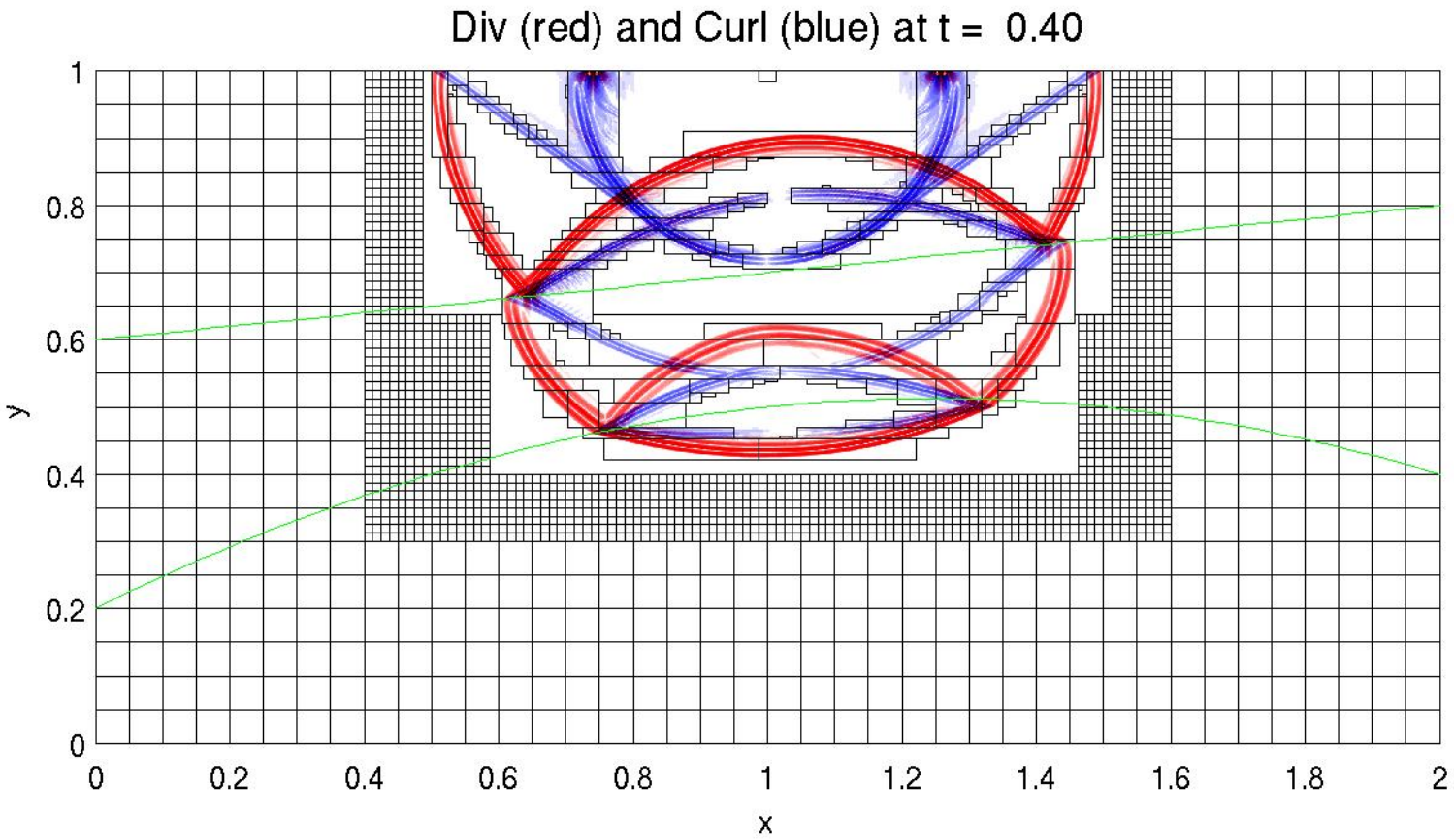
Clawpack Software:

Explicit — Work of Marsha Berger, NYU
Implicit — Current work of Jonathan Claridge


Div (red) and Curl (blue) at t = 0.10
y
x
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Div (red) and Curl (blue) at t = 0.20
y
x
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Div (red) and Curl (blue) at t = 0.30
y
x
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Div (red) and Curl (blue) at t = 0.40
y
x
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

Finite differences vs. finite volumes

Finite difference Methods

- Pointwise values $Q_i^n \approx q(x_i, t_n)$
- Approximate derivatives by finite differences
- Assumes smoothness

Finite volume Methods

- Approximate cell averages: $Q_i^n \approx \dfrac{1}{\Delta x} \displaystyle\int_{x_{i-1/2}}^{x_{i+1/2}} q(x, t_n)\, dx$
- Integral form of conservation law,

$$\frac{\partial}{\partial t} \int_{x_{i-1/2}}^{x_{i+1/2}} q(x,t)\, dx \quad = \quad f(q(x_{i-1/2}, t)) - f(q(x_{i+1/2}, t))$$

leads to conservation law $q_t + f_x = 0$ but also directly to numerical method.

CLAWPACK — www.clawpack.org

- Open source, 1d, 2d, (3d in V4.3, soon to be ported)
- Originally f77 with Matlab graphics (V4.3).
- Now use Python for user interface, graphics
- Adaptive mesh refinement, GeoClaw.
- Coming: OpenMP, better f90 version.

User supplies:

- Riemann solver, splitting data into waves and speeds (Need not be in conservation form)
- Boundary condition routine to extend data to ghost cells Standard `bc1.f` routine includes many standard BC's
- Initial conditions — `qinit.f`
- Source terms — `src1.f`

The Riemann problem

The Riemann problem consists of the hyperbolic equation under study together with initial data of the form

$$q(x,0) = \begin{cases} q_l & \text{if } x < 0 \\ q_r & \text{if } x \geq 0 \end{cases}$$

Piecewise constant with a single jump discontinuity from q_l to q_r.

The Riemann problem is fundamental to understanding

- The mathematical theory of hyperbolic problems,
- Godunov-type finite volume methods

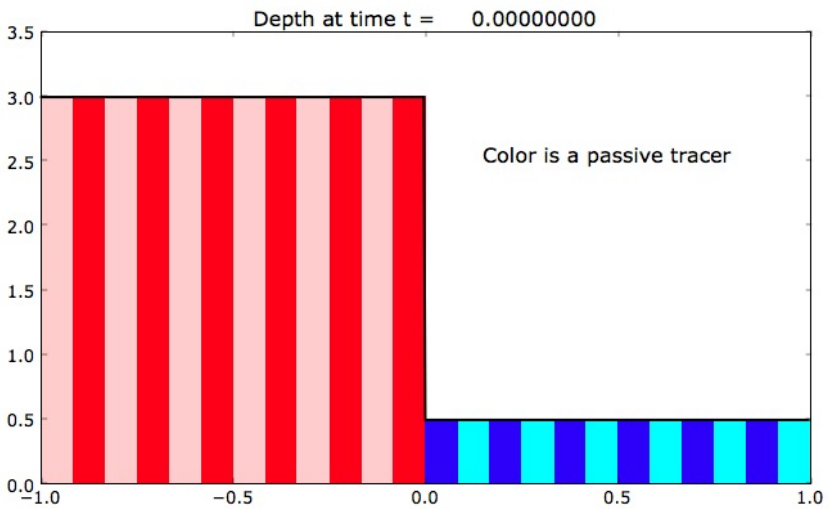
Why? Even for nonlinear systems of conservation laws, the Riemann problem can often be solved for general q_l and q_r, and consists of a set of waves propagating at constant speeds.

The Riemann problem

Dam break problem for shallow water equations

$$h_t + (hu)_x = 0$$

$$(hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = 0$$

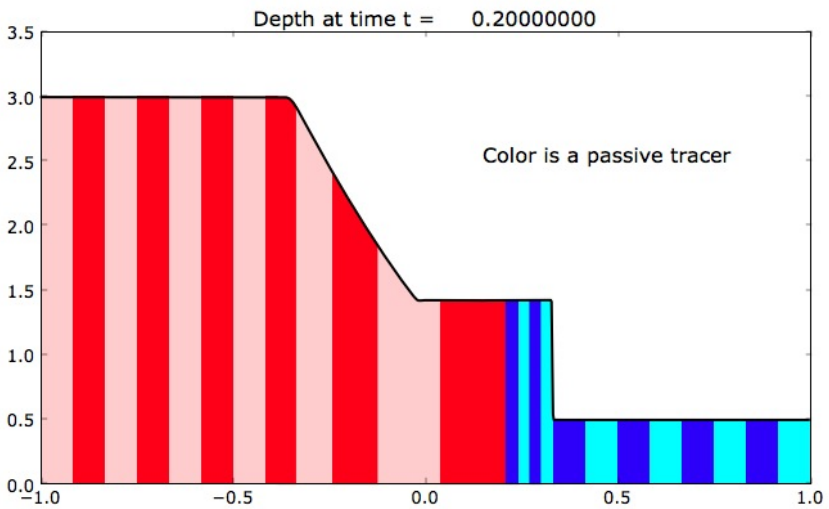


The Riemann problem

Dam break problem for shallow water equations

$$h_t + (hu)_x = 0$$

$$(hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = 0$$

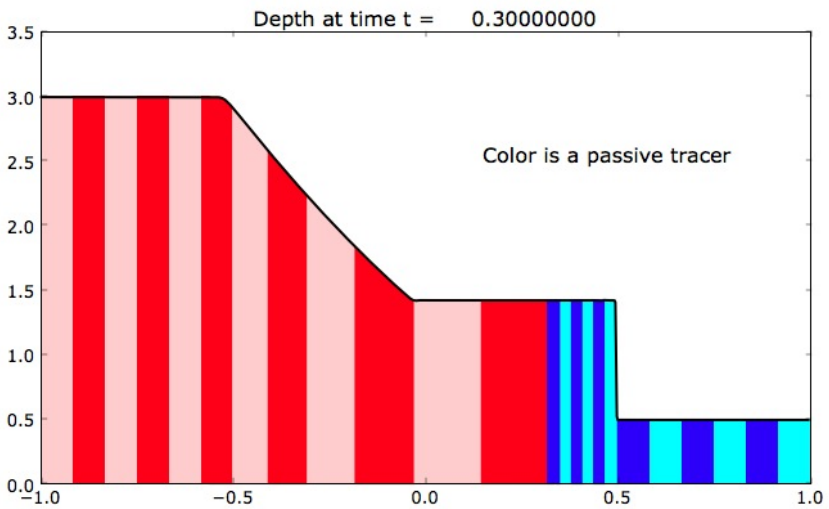


The Riemann problem

Dam break problem for shallow water equations

$$h_t + (hu)_x = 0$$

$$(hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = 0$$

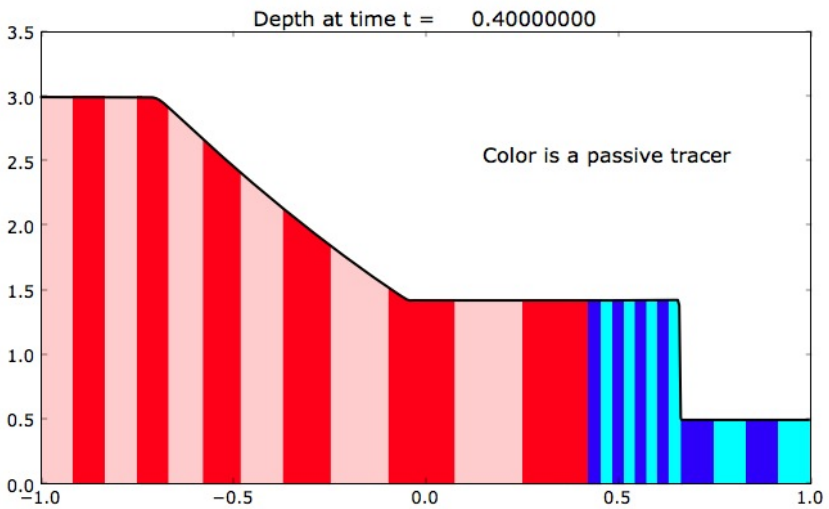


The Riemann problem

Dam break problem for shallow water equations

$$h_t + (hu)_x = 0$$

$$(hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = 0$$

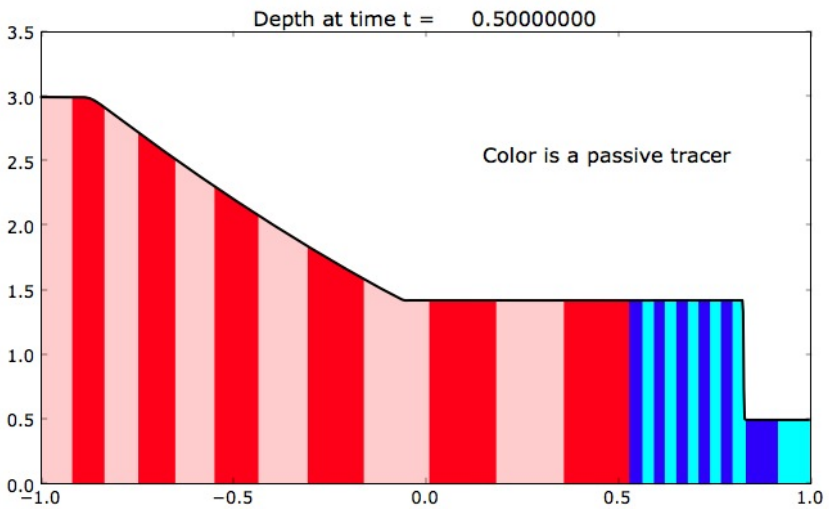


The Riemann problem

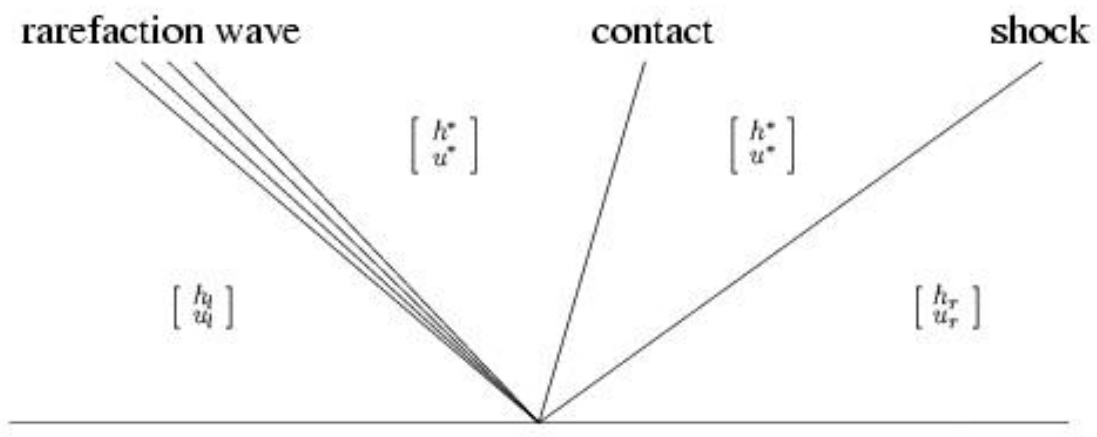
Dam break problem for shallow water equations

$$h_t + (hu)_x = 0$$

$$(hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = 0$$



Riemann solution for the SW equations



The Roe solver uses the solution to a linear system

$$q_t + \hat{A}_{i-1/2} q_x = 0, \qquad \hat{A}_{i-1/2} = f'(q_{\text{ave}}).$$

All waves are simply discontinuities.

Typically a fine approximation if jumps are approximately correct.

Options for using Clawpack

1. Install from tar file: **Instructions**.

   Requires some **prerequisites:** Fortran, Python modules.

2. Use the **VirtualClaw** virtual machine.

3. For some applications, use **EagleClaw**
   (Easy Access Graphical Laboratory for Exploring Conservation Laws)

Documentation: `http://www.clawpack.org/doc/`

Also perhaps useful:

AMath 583 Class notes on Python, Fortran, version control, etc.

Installation

Acquire `clawpack-4.6.1.tar.gz` from www.clawpack.org

```
$ tar -zxf clawpack-4.6.1.tar.gz

$ cd clawpack-4.6.1
$ python setenv.py
$ source setenv.bash
```

Commands in `setenv.bash` can be put in `.bashrc` file.

These define environment variables `CLAW`, `PYTHONPATH`, etc.

E.g. to go to main directory:

```
$ cd $CLAW
```

Virtualization

Clawpack requires:

- Unix/Linux
- gfortran (and OpenMP, MPI)
- Python (preferably IPython)
- Plotting modules
- Sphinx for documentation

Virtualization

Clawpack requires:

- Unix/Linux
- gfortran (and OpenMP, MPI)
- Python (preferably IPython)
- Plotting modules
- Sphinx for documentation

VirtualClaw: Simple way to provide complete OS and software.

VM image for VirtualBox, runs on Linux / Windows / Mac.

Download: **www.clawpack.org/VM**

Compiling, running, plotting

Applications directories contain a Makefile.

`$ make help` For list of options

`$ make .data` Uses `setrun.py` to make Fortran data

`$ make .exe` Compiles Fortran codes

`$ make .output` Runs code, produces `_output`

`$ make .plots` Plots results, produces `_plots`

`$ make .htmls` Produces html versions of source files

Application `Makefile`

Documentation: **www.clawpack.org/doc/makefiles.html**
For example, see **$CLAW/apps/acoustics/1d/example2/Makefile**

Several variables are set, e.g.
where to find setrun function and where to put output:

```
CLAW_setrun_file = setrun.py
CLAW_OUTDIR = _output
```

where to find setplot function and where to put plots:

```
CLAW_setplot_file = setplot.py
CLAW_PLOTDIR = _plots
```

Usually these do not need to be changed.

Application `Makefile` (cont.)

List of local Fortran files:

```
CLAW_SOURCES = \
  driver.f \
  qinit.f \
  rp1.f \
  setprob.f
```

List of library files:

```
# Clawpack library to be used:
CLAW_LIB = $(CLAW)/clawpack/1d/lib

CLAW_LIBSOURCES = \
  $(CLAW_LIB)/claw1ez.f \
  $(CLAW_LIB)/bc1.f \
  etc.
```

Application `Makefile` (cont.)

Ends with...

```
# Include Makefile containing standard
# definitions and make options:
CLAWMAKE = $(CLAW)/util/Makefile.common
include $(CLAWMAKE)
```

The file **$CLAW/util/Makefile.common** contains rules for various targets.

For possible targets, type

```
$ make help
```

Documentation: **www.clawpack.org/doc/makefiles.html**

Setting runtime parameters

The file `setrun.py` contains a function setrun
that returns an object rundata of class ClawRunData.

Never need to write from scratch...
Modify an existing example!

Don't need to know much if anything about Python!

Lots of comments in the sample versions.

Documentation: **www.clawpack.org/doc/setrun.html**

For example, see **$CLAW/apps/acoustics/1d/example2/setrun.py**

Copying and modifying an example

Find an example similar to the one you want to create and copy the directory, e.g.

```
$ cd $CLAW/apps/acoustics/1d
$ cp -r example2 $CLAW/myclaw/newexample
```

Copying and modifying an example

Find an example similar to the one you want to create and copy the directory, e.g.

```
$ cd $CLAW/apps/acoustics/1d
$ cp -r example2 $CLAW/myclaw/newexample
```

Warning: If you have a Subversion copy of Clawpack...
This will copy `.svn` subdirectory too.
Better way:

```
$ cd $CLAW/apps/acoustics/1d
$ svn export example2 $CLAW/myclaw/newexample
```

Documentation: **www.clawpack.org/doc/newapp.html**

Linear acoustics

Example: Linear acoustics in a 1d gas tube

$$q = \begin{bmatrix} p \\ u \end{bmatrix} \qquad \begin{aligned} p(x,t) &= \text{pressure perturbation} \\ u(x,t) &= \text{velocity} \end{aligned}$$

Equations:

$$\begin{aligned} p_t + \kappa u_x &= 0 \\ \rho u_t + p_x &= 0 \end{aligned} \qquad \begin{aligned} &\text{Change in pressure due to compression} \\ &\text{Newton's second law, } F = ma \end{aligned}$$

where $K =$ bulk modulus, and $\rho =$ unperturbed density of gas.

Hyperbolic system:

$$\begin{bmatrix} p \\ u \end{bmatrix}_t + \begin{bmatrix} 0 & \kappa \\ 1/\rho & 0 \end{bmatrix} \begin{bmatrix} p \\ u \end{bmatrix}_x = 0.$$

Linear acoustics

$$\begin{bmatrix} p \\ u \end{bmatrix}_t + \begin{bmatrix} 0 & \kappa \\ 1/\rho & 0 \end{bmatrix} \begin{bmatrix} p \\ u \end{bmatrix}_x = 0.$$

This has the form $q_t + Aq_x = 0$ with

eigenvalues: $\lambda^1 = -c, \qquad \lambda^2 = +c,$

where $c = \sqrt{\kappa/\rho} =$ speed of sound.

eigenvectors: $r^1 = \begin{bmatrix} -Z \\ 1 \end{bmatrix}, \qquad r^2 = \begin{bmatrix} Z \\ 1 \end{bmatrix}$

where $Z = \rho c = \sqrt{\rho\kappa} =$ impedance.

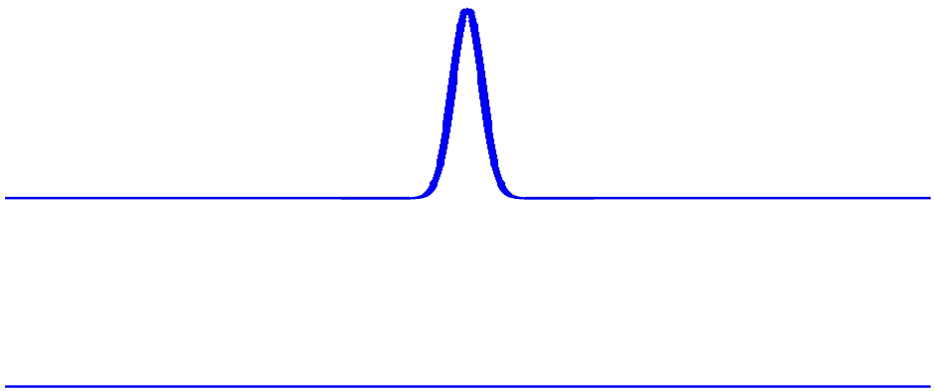
$$R = \begin{bmatrix} -Z & Z \\ 1 & 1 \end{bmatrix}, \qquad R^{-1} = \frac{1}{2Z} \begin{bmatrix} -1 & Z \\ 1 & Z \end{bmatrix}.$$

Acoustic waves

$$q(x,0) = \begin{bmatrix} p(x) \\ 0 \end{bmatrix} = -\frac{p(x)}{2Z}\begin{bmatrix} -Z \\ 1 \end{bmatrix} + \frac{p(x)}{2Z}\begin{bmatrix} Z \\ 1 \end{bmatrix}$$

$$= w^1(x,0)r^1 + w^2(x,0)r^2$$

$$= \begin{bmatrix} p(x)/2 \\ -p(x)/(2Z) \end{bmatrix} + \begin{bmatrix} p(x)/2 \\ p(x)/(2Z) \end{bmatrix}.$$

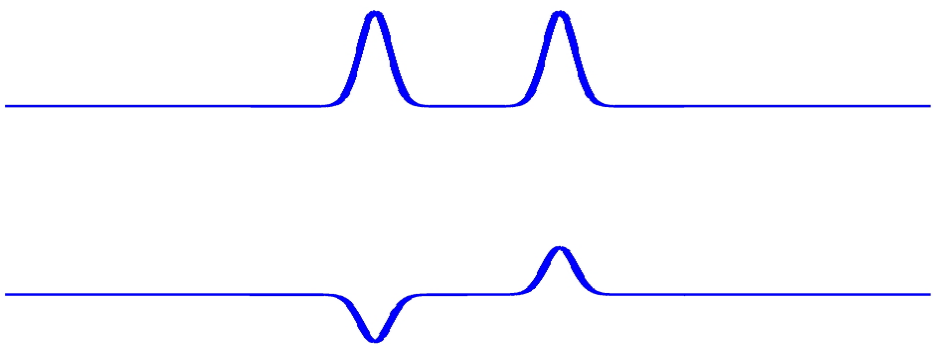


Acoustic waves

$$q(x,0) = \begin{bmatrix} p(x) \\ 0 \end{bmatrix} = -\frac{p(x)}{2Z}\begin{bmatrix} -Z \\ 1 \end{bmatrix} + \frac{p(x)}{2Z}\begin{bmatrix} Z \\ 1 \end{bmatrix}$$

$$= w^1(x,0)r^1 + w^2(x,0)r^2$$

$$= \begin{bmatrix} p(x)/2 \\ -p(x)/(2Z) \end{bmatrix} + \begin{bmatrix} p(x)/2 \\ p(x)/(2Z) \end{bmatrix}.$$

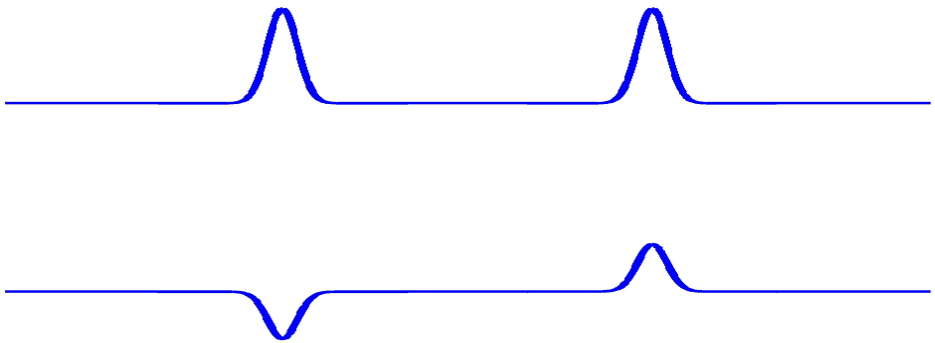


Acoustic waves

$$q(x,0) = \begin{bmatrix} p(x) \\ 0 \end{bmatrix} = -\frac{p(x)}{2Z}\begin{bmatrix} -Z \\ 1 \end{bmatrix} + \frac{p(x)}{2Z}\begin{bmatrix} Z \\ 1 \end{bmatrix}$$

$$= w^1(x,0)r^1 + w^2(x,0)r^2$$

$$= \begin{bmatrix} p(x)/2 \\ -p(x)/(2Z) \end{bmatrix} + \begin{bmatrix} p(x)/2 \\ p(x)/(2Z) \end{bmatrix}.$$

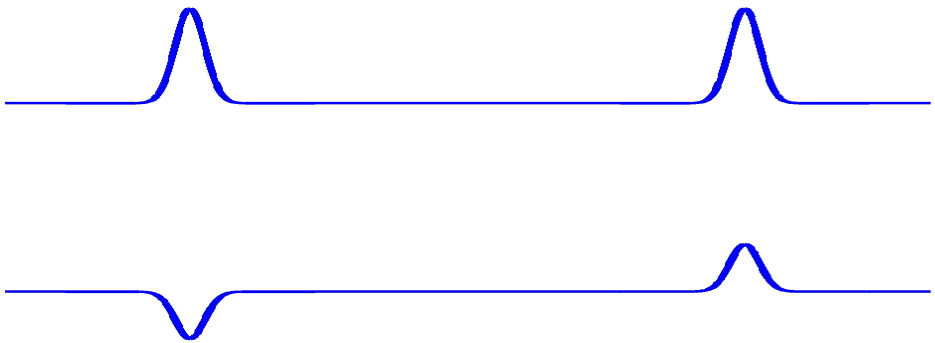


Acoustic waves

$$q(x,0) = \begin{bmatrix} p(x) \\ 0 \end{bmatrix} = -\frac{p(x)}{2Z}\begin{bmatrix} -Z \\ 1 \end{bmatrix} + \frac{p(x)}{2Z}\begin{bmatrix} Z \\ 1 \end{bmatrix}$$

$$= w^1(x,0)r^1 + w^2(x,0)r^2$$

$$= \begin{bmatrix} p(x)/2 \\ -p(x)/(2Z) \end{bmatrix} + \begin{bmatrix} p(x)/2 \\ p(x)/(2Z) \end{bmatrix}.$$

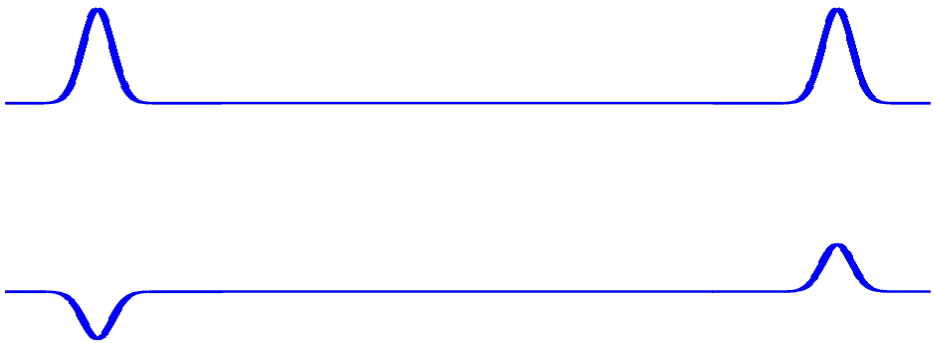


Acoustic waves

$$q(x,0) = \begin{bmatrix} p(x) \\ 0 \end{bmatrix} = -\frac{p(x)}{2Z}\begin{bmatrix} -Z \\ 1 \end{bmatrix} + \frac{p(x)}{2Z}\begin{bmatrix} Z \\ 1 \end{bmatrix}$$

$$= w^1(x,0)r^1 + w^2(x,0)r^2$$

$$= \begin{bmatrix} p(x)/2 \\ -p(x)/(2Z) \end{bmatrix} + \begin{bmatrix} p(x)/2 \\ p(x)/(2Z) \end{bmatrix}.$$



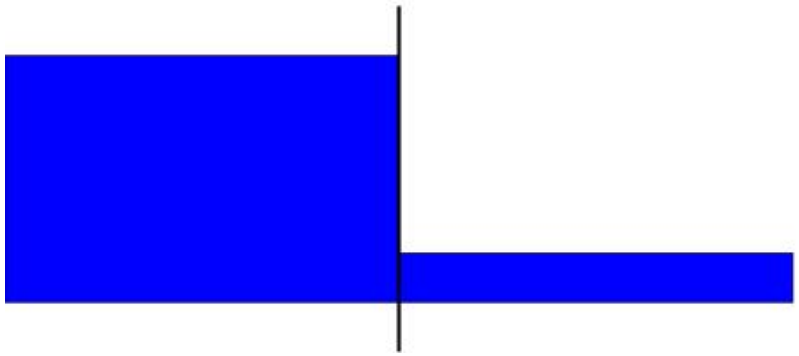
Riemann Problem

Special initial data:

$$q(x,0) = \begin{cases} q_l & \text{if } x < 0 \\ q_r & \text{if } x > 0 \end{cases}$$

Example: Acoustics with bursting diaphram

Pressure:



Acoustic waves propagate with speeds $\pm c$.

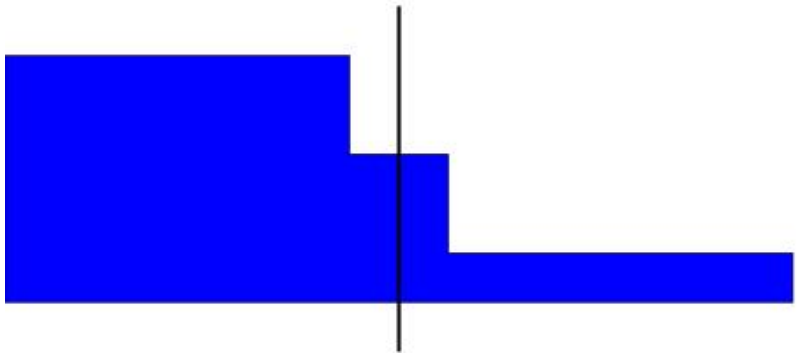
Riemann Problem

Special initial data:

$$q(x,0) = \begin{cases} q_l & \text{if } x < 0 \\ q_r & \text{if } x > 0 \end{cases}$$

Example: Acoustics with bursting diaphram

Pressure:



Acoustic waves propagate with speeds $\pm c$.

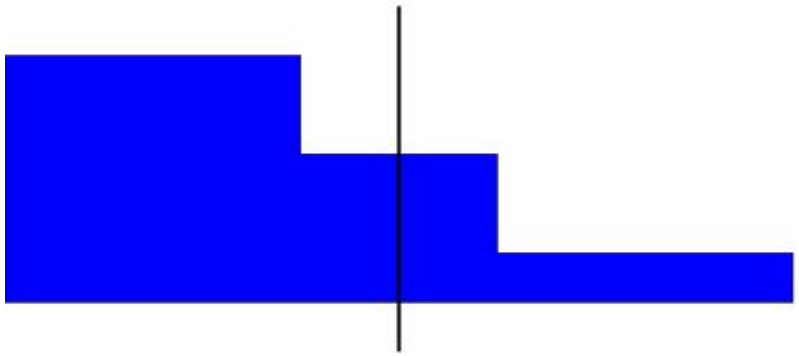
Riemann Problem

Special initial data:

$$q(x,0) = \begin{cases} q_l & \text{if } x < 0 \\ q_r & \text{if } x > 0 \end{cases}$$

Example: Acoustics with bursting diaphram

Pressure:



Acoustic waves propagate with speeds $\pm c$.

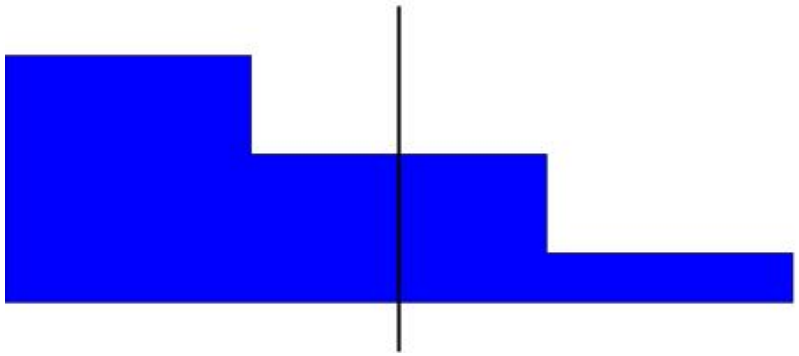
Riemann Problem

Special initial data:

$$q(x,0) = \begin{cases} q_l & \text{if } x < 0 \\ q_r & \text{if } x > 0 \end{cases}$$

Example: Acoustics with bursting diaphram

Pressure:



Acoustic waves propagate with speeds $\pm c$.

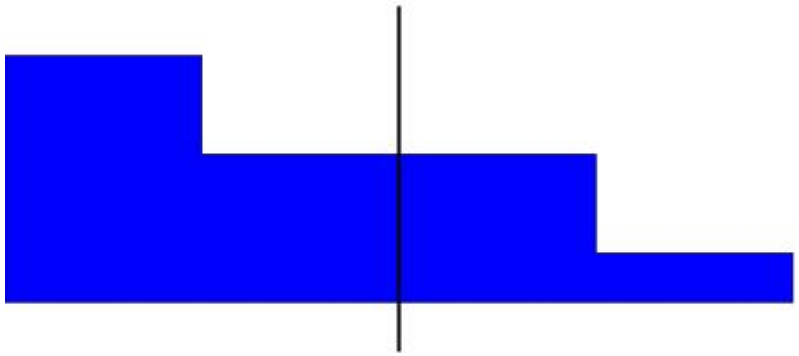
Riemann Problem

Special initial data:

$$q(x,0) = \begin{cases} q_l & \text{if } x < 0 \\ q_r & \text{if } x > 0 \end{cases}$$

Example: Acoustics with bursting diaphram

Pressure:



Acoustic waves propagate with speeds $\pm c$.

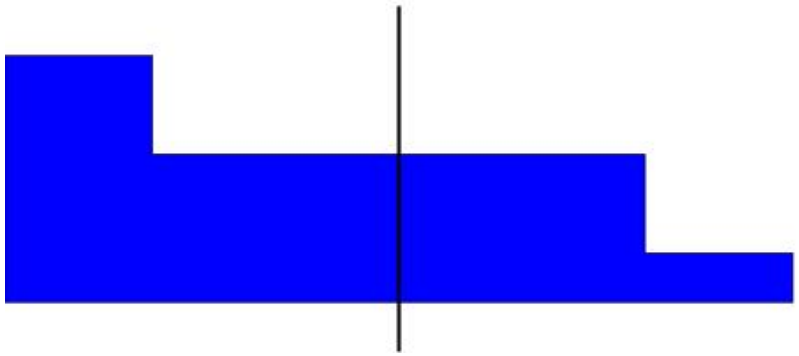
Riemann Problem

Special initial data:

$$q(x,0) = \begin{cases} q_l & \text{if } x < 0 \\ q_r & \text{if } x > 0 \end{cases}$$

Example: Acoustics with bursting diaphram

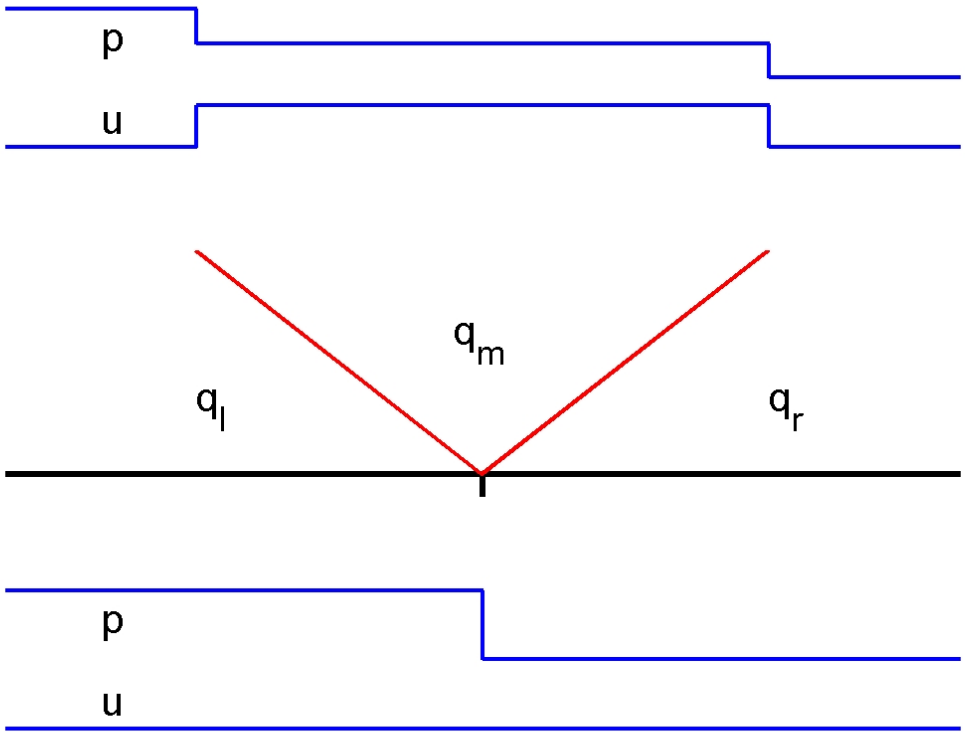
Pressure:



Acoustic waves propagate with speeds $\pm c$.

Riemann Problem for acoustics

Waves propagating in x–t space:



Left-going wave $\mathcal{W}^1 = q_m - q_l$ and
right-going wave $\mathcal{W}^2 = q_r - q_m$ are eigenvectors of A.

Riemann solution for a linear system

Linear hyperbolic system: $q_t + Aq_x = 0$ with $A = R\Lambda R^{-1}$.
General Riemann problem data $q_l,\ q_r \in \mathbb{R}^m$.

Decompose jump in q into eigenvectors:

$$q_r - q_l = \sum_{p=1}^{m} \alpha^p r^p$$

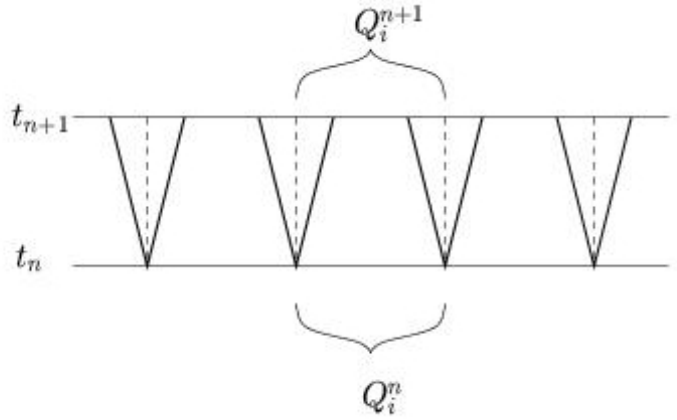
Note: the vector α of eigen-coefficients is

$$\alpha = R^{-1}(q_r - q_l) = R^{-1}q_r - R^{-1}q_l = w_r - w_l.$$

Riemann solution consists of m waves $\mathcal{W}^p \in \mathbb{R}^m$:

$$\mathcal{W}^p = \alpha^p r^p, \qquad \text{propagating with speed } \ s^p = \lambda^p.$$

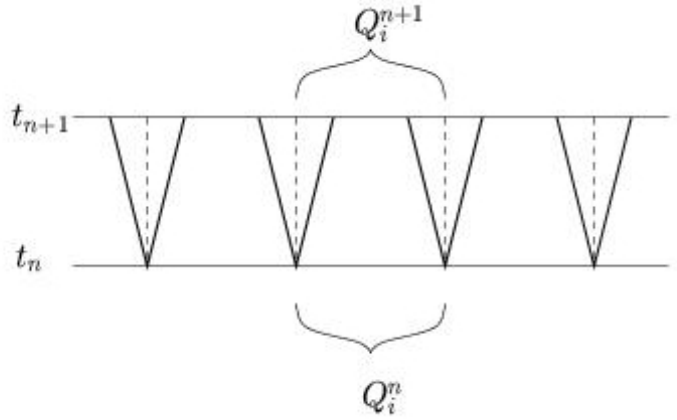
Godunov's Method for $q_t + f(q)_x = 0$



Then either:

1. Compute new cell averages by integrating over cell at t_{n+1},

Godunov's Method for $q_t + f(q)_x = 0$

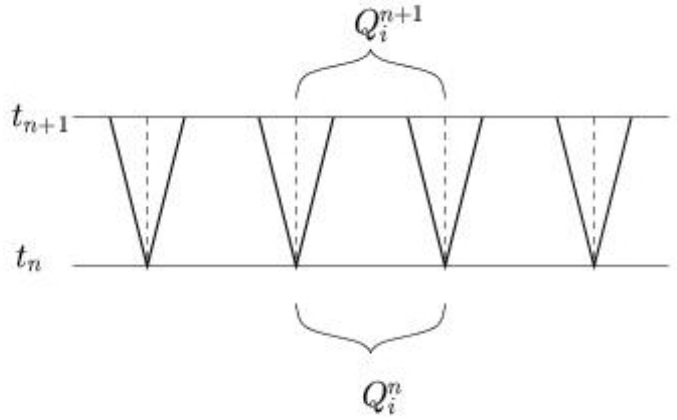


Then either:

1. Compute new cell averages by integrating over cell at t_{n+1},
2. Compute fluxes at interfaces and flux-difference:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}[F_{i+1/2}^n - F_{i-1/2}^n]$$

Godunov's Method for $q_t + f(q)_x = 0$



Then either:

1. Compute new cell averages by integrating over cell at t_{n+1},
2. Compute fluxes at interfaces and flux-difference:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}[F_{i+1/2}^n - F_{i-1/2}^n]$$

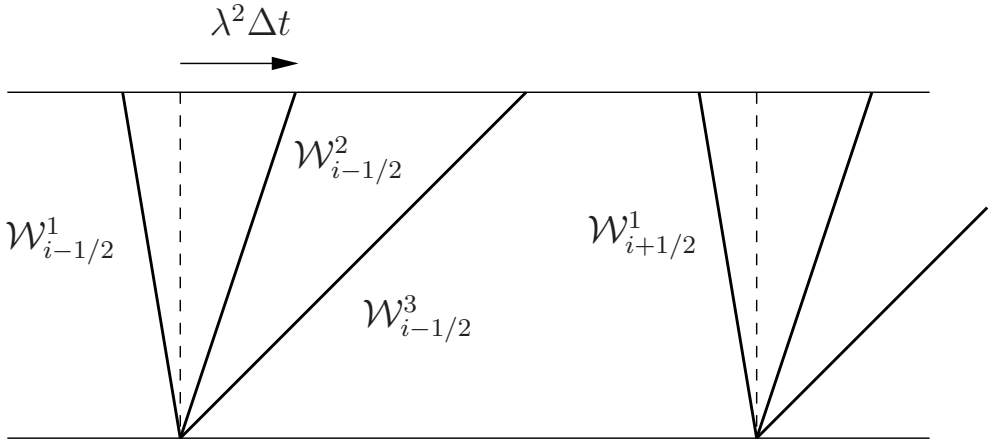
3. Update cell averages by contributions from all waves entering cell:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}[\mathcal{A}^+ \Delta Q_{i-1/2} + \mathcal{A}^- \Delta Q_{i+1/2}]$$

where $\mathcal{A}^{\pm} \Delta Q_{i-1/2} = \sum_{i=1}^{m} (s_{i-1/2}^p)^{\pm} \mathcal{W}_{i-1/2}^p$.

Wave-propagation viewpoint

For linear system $q_t + Aq_x = 0$, the Riemann solution consists of waves $\mathcal{W}^p$ propagating at constant speed λ^p.



$$Q_i - Q_{i-1} = \sum_{p=1}^{m} \alpha^p_{i-1/2} r^p \equiv \sum_{p=1}^{m} \mathcal{W}^p_{i-1/2}.$$

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left[\lambda^2 \mathcal{W}^2_{i-1/2} + \lambda^3 \mathcal{W}^3_{i-1/2} + \lambda^1 \mathcal{W}^1_{i+1/2}\right].$$

CLAWPACK Riemann solver

The hyperbolic problem is specified by the Riemann solver

- Input: Values of q in each grid cell
- Output: Solution to Riemann problem at each interface.
  - Waves $\mathcal{W}^p \in \mathbb{R}^m$, $p = 1,\ 2,\ \ldots,\ M_w$
  - Speeds $s^p \in \mathbb{R}$, $p = 1,\ 2,\ \ldots,\ M_w$,
  - Fluctuations $\mathcal{A}^- \Delta Q,\ \mathcal{A}^+ \Delta Q \in \mathbb{R}^m$

Note: Number of waves M_w often equal to m (length of q), but could be different (e.g. HLL solver has 2 waves).

CLAWPACK Riemann solver

The hyperbolic problem is specified by the Riemann solver

- Input: Values of q in each grid cell
- Output: Solution to Riemann problem at each interface.
  - Waves $\mathcal{W}^p \in \mathbb{R}^m$, $p = 1,\ 2,\ \ldots,\ M_w$
  - Speeds $s^p \in \mathbb{R}$, $p = 1,\ 2,\ \ldots,\ M_w$,
  - Fluctuations $\mathcal{A}^- \Delta Q,\ \mathcal{A}^+ \Delta Q \in \mathbb{R}^m$

Note: Number of waves M_w often equal to m (length of q), but could be different (e.g. HLL solver has 2 waves).

Fluctuations:

$\mathcal{A}^- \Delta Q =$ Contribution to cell average to left,

$\mathcal{A}^+ \Delta Q =$ Contribution to cell average to right

For conservation law, $\mathcal{A}^- \Delta Q + \mathcal{A}^+ \Delta Q = f(Q_r) - f(Q_l)$

CLAWPACK Riemann solver

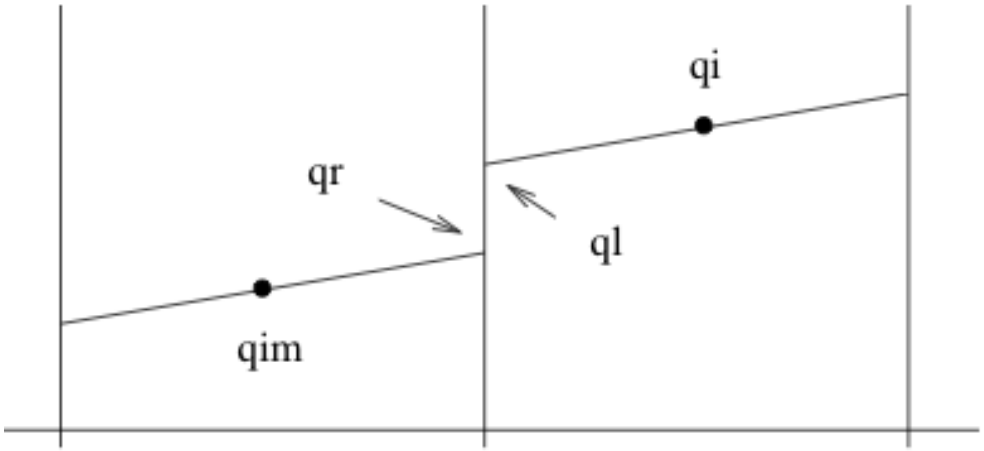
Inputs to `rp1` subroutine:

`ql(i,1:m)` = Value of q at left edge of ith cell,

`qr(i,1:m)` = Value of q at right edge of ith cell,

Warning: The Riemann problem at the interface between cells $i-1$ and i has left state `qr(i-1,:)` and right state `ql(i,:)`.

`rp1` is normally called with `ql = qr = q`,
but designed to allow other methods:



CLAWPACK Riemann solver

Outputs from `rp1` subroutine:

for system of m equations
with `mw` ranging from 1 to M_w =# of waves

`s(i,mw)` = Speed of wave # `mw` in ith Riemann solution,

`wave(i,1:m,mw)` = Jump across wave # `mw`,

`amdq(i,1:m)` = Left-going fluctuation, updates Q_{i-1}
`apdq(i,1:m)` = Right-going fluctuation, updates Q_i

Acoustics examples

- **$CLAW/apps/acoustics/1d/example2/README.html**
- **Acoustics in EagleClaw**