FYS-GEO 4500 21 Sep 2009

Before we start:
Questions over the reading?

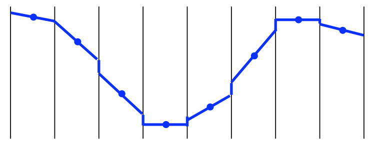
The problem set

Review of High-Resolution Methods

We improve the first-order upwind method by introducing corrections, and writing:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^+\Delta Q_{i-1/2} + \mathcal{A}^-\Delta Q_{i+1/2}\right) - \frac{\Delta t}{\Delta x}\left(\widetilde{F}_{i+1/2} - \widetilde{F}_{i-1/2}\right)$$

We derive the corrections by considering piece-wise linear (instead of piece-wise constant) reconstructions.



Our syllabus - converging

	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, part of 12
7	5 Oct 2009	Monday 13.15-15.00	nonlinear equations & systems; shallow-water eqn	part of 12,13
8	12 Oct 2009	Monday 13.15-15.00	gas dynamics, Euler equations, methods	14, part of 15
9	19 Oct 2009	Monday 13.15-15.00	varying flux functions, source terms	part of 16, 17
10	26 Oct 2009	Monday 13.15-15.00	multidimensional hyperbolic problems & methods	18 & 19
11	2 Nov 2009	Monday 13.15-15.00	systems & applications; project planning	20 & 21
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	

Review of High-Resolution Methods

Taking the basic Lax-Wendroff formula:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{2\Delta x}A\left(Q_{i+1}^n - Q_{i-1}^n\right) + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 A^2\left(Q_{i+1}^n - 2Q_i^n + Q_{i-1}^n\right)$$

we re-write it in the flux form

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left(F_{i+1/2}^n - F_{i-1/2}^n\right)$$

with

$$F_{i-1/2}^n = \frac{1}{2}A\left(Q_i^n + Q_{i-1}^n\right) - \frac{1}{2}\frac{\Delta t}{\Delta x}A^2\left(Q_i^n - Q_{i-1}^n\right)$$

Then making use of the divided matrices $A^{\pm}$ we can write this as

$$F_{i-1/2}^n = \left(A^- Q_i^n + A^+ Q_{i-1}^n\right) + \frac{1}{2}|A|\left(I - \frac{\Delta t}{\Delta x}|A|\right)\left(Q_i^n - Q_{i-1}^n\right)$$

Review of High-Resolution Methods

$$F_{i-1/2}^n = \left(A^- Q_i^n + A^+ Q_{i-1}^n\right) + \frac{1}{2}|A|\left(I - \frac{\Delta t}{\Delta x}|A|\right)\left(Q_i^n - Q_{i-1}^n\right)$$

This version of the Lax-Wendroff formula has a correction term that can be limited, if we choose, to avoid oscillations around extrema.

For a one-equation system (the advection equation), we can apply a simple functional limiter to the slope:

$$\sigma_i^n = \left(\frac{Q_{i+1}^n - Q_i^n}{\Delta x}\right)\phi_i^n$$

Examples of limiters:

Lax-Wendroff: $\phi(\theta) = 1$

minmod: $\phi(\theta) = \text{minmod}(1,\theta)$

superbee: $\phi(\theta) = \max(0, \min(1,2\theta), \min(2,\theta))$

MC: $\phi(\theta) = \max(0, \min((1+\theta)/2, 2, 2\theta))$

vanLeer: $\phi(\theta) = \dfrac{(\theta + |\theta|)}{(1+|\theta|)}$

Boundary Conditions (Chapter 7 in Leveque)

Review of High-Resolution Methods

For a system of equations, we use limiters on the waves. The wave-propagation form for a high-resolution version of Lax-Wendroff is:

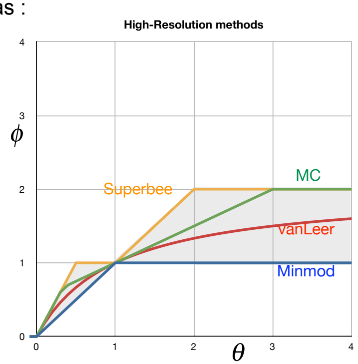
$$F_{i-1/2}^n = \left(A^- Q_i^n + A^+ Q_{i-1}^n\right) + \frac{1}{2}\sum_{p=1}^{m}\left|s_{i-1/2}^p\right|\left(1 - \frac{\Delta t}{\Delta x}\left|s_{i-1/2}^p\right|\right)\widetilde{\mathcal{W}}_{i-1/2}^p$$

with the limited version of the waves defined as :

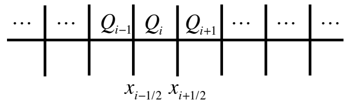
$$\widetilde{\mathcal{W}}_{i-1/2}^p = \alpha_{i-1/2}^p \phi(\theta_{i-1/2}^p) r^p$$

and a generalised wave speed

$$s_{i-1/2}^p = \lambda^p$$

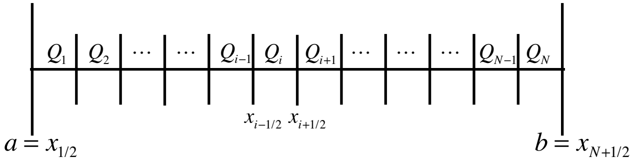


Real problems have boundaries

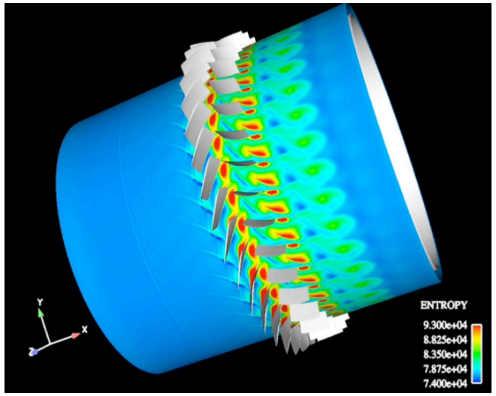


In computing the values Q_i^{n+1} for the next time step, we have assumed the availability of data from Q_{i-1}^n and Q_{i+1}^n . But we can never compute on an infinite grid, and in practice we must often deal with physical boundaries such as solid walls or materials with different properties.

Real problems have boundaries



In computing the values Q_i^{n+1} for the next time step, we have assumed the availability of data from Q_{i-1}^n and Q_{i+1}^n . But we can never compute on an infinite grid, and in practice we must often deal with physical boundaries such as solid walls or materials with different properties.



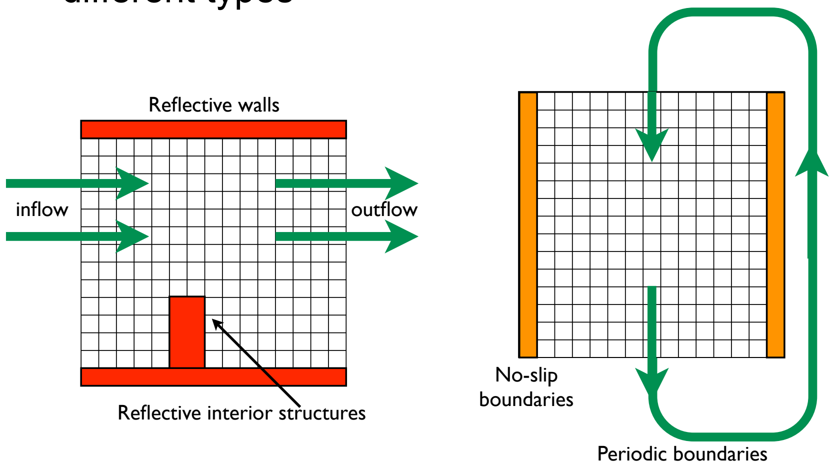
Calculation from the Stanford Center for Turbulence Research of a portion of the main compressor turbine in a modern jet engine. There are inflow and outflow boundaries as well as internal reflective boundaries for the turbine fan blades. The coordinate system rotates with the turbine, which starts slowly and approaches cruising speed.

This calculation couples to a separate calculation for the combustor, which uses the outflow from the compressor, and supplies the energy to rotate the turbine.

Calculation from the Stanford Center for Turbulence Research of a portion of the main compressor turbine in a modern jet engine. There are inflow and outflow boundaries as well as internal reflective boundaries for the turbine fan blades. The coordinate system rotates with the turbine, which starts slowly and approaches cruising speed.

This calculation couples to a separate calculation for the combustor, which uses the outflow from the compressor, and supplies the energy to rotate the turbine.

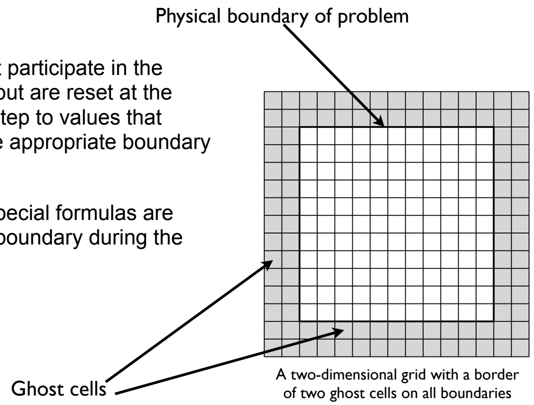
Boundary conditions may be of several different types



One common approach is to use *ghost cells*

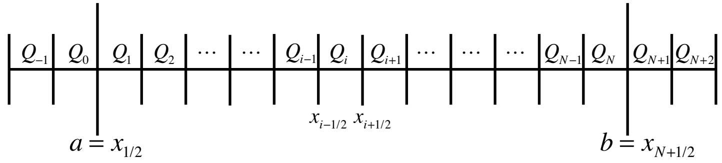
Ghost cells are cells that participate in the computational scheme, but are reset at the beginning of each time step to values that effectively implement the appropriate boundary condition.

If this is done, then no special formulas are needed on the physical boundary during the computational step.



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

Boundary conditions in one dimension.

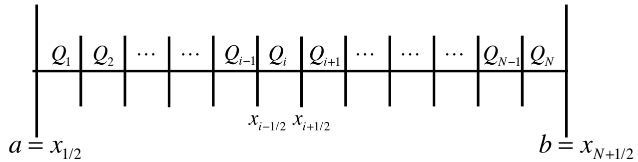


The solution in the domain $[a,b]$ is the *interior solution*, and is computed by the wave-propagation procedure, solving Riemann problems at each interface.

To compute the appropriate fluxes to update cells 1 and N, we need values from the neighbouring ghost cells (0,-1) and (N+1, N+2). These are provided by the boundary-condition procedure.

This boundary-condition procedure will depend on the nature of the boundary conditions being used.

Boundary conditions in one dimension.

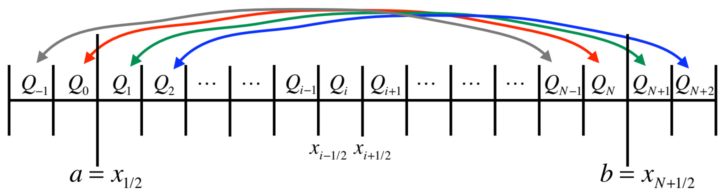


The solution in the domain $[a,b]$ is the *interior solution*, and is computed by the wave-propagation procedure, solving Riemann problems at each interface.

To compute the appropriate fluxes to update cells 1 and N, we need values from the neighbouring ghost cells (0,-1) and (N+1, N+2). These are provided by the boundary-condition procedure.

This boundary-condition procedure will depend on the nature of the boundary conditions being used.

Periodic boundary conditions



For periodic boundary conditions, we simply set

$$Q_{-1} = Q_{N-1}, \quad Q_0 = Q_N, \quad Q_{N+1} = Q_1, \quad Q_{N+2} = Q_2.$$

at the beginning of each time step.

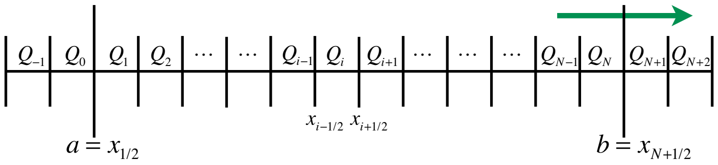
In Clawpack, the work is done in the Fortran routine

```
$CLAW/clawpack/1d/lib/bc1.f
```

and you invoke it in `setrun.py` by setting

```
clawdata.mthbc_xlower = 2
clawdata.mthbc_xupper = 2
```

Outflow boundary conditions



For a pure upwind method, we don't need any outflow boundary condition. But we do for higher-order methods, and if waves are travelling in both directions.

The simplest and most effective procedure is simply to extrapolate the solution from the last physical cell. If the outflow boundary is at right, we set:

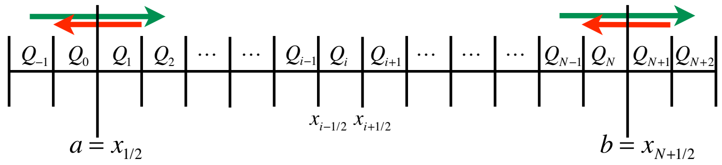
$$Q_{N+1} = Q_N\,, \quad Q_{N+2} = Q_N$$

for a zero-order or constant extrapolation, or

$$Q_{N+1} = 2Q_N - Q_{N-1},\ Q_{N+2} = 3Q_N - 2Q_{N-1}$$

for a first-order or linear extrapolation. In most circumstances, constant extrapolation gives better stability.

Boundary conditions for systems of equations



For systems with both positive and negative eigenvalues, each boundary may have both incoming and outgoing characteristics.

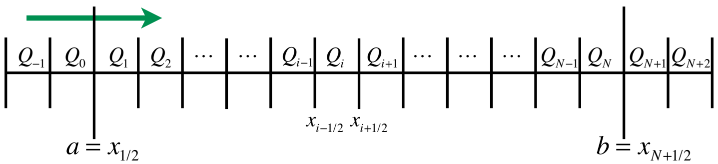
Then you treat the boundary cells with the normal Riemann solution, for example

$$Q_1 - Q_0 = \sum_{p=1}^{m} \mathcal{W}_{1/2}^{p},$$

but the incoming waves have strength zero. The outgoing waves may have nonzero strength, but it doesn't matter since the ghost cells will be reset at the start of the next cycle.

It should be possible to apply the appropriate boundary conditions for the separate characteristics, with some knowledge of the appropriate physics.

Inflow boundary conditions



For inflow conditions it is sufficient to reset the ghost cells on the inflow boundary to the desired source values, which may be time-dependent.

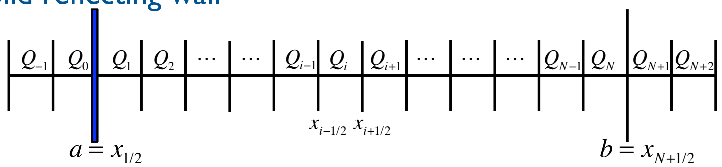
For the advection equation with velocity u, and the inflow boundary at right, set

$$Q_0 = Q_{source}\left(t + \frac{\Delta x}{2u}\right), \quad Q_{-1} = Q_{source}\left(t + \frac{3\Delta x}{2u}\right).$$

Note that we advance the time by the travel time from the centre of each of these cells to the physical boundary.

An example from the acoustics equation

solid reflecting wall



Recall the acoustics equations

$$p_t + K u_x = 0$$
$$\rho_0 u_t + p_x = 0$$

To implement a solid reflecting wall in this system, we reverse the velocity in the ghost cells and make the pressure the same:

$$\text{for } Q_0: \quad p_0 = p_1, \quad u_0 = -u_1$$
$$\text{for } Q_{-1}: \quad p_{-1} = p_2, \quad u_{-1} = -u_2$$

Set the boundary conditions in setrun.py

```
# --------------------
# Boundary conditions:
# --------------------

# Number of ghost cells (usually 2)
clawdata.mbc = 2

# Choice of BCs at xlower and xupper:
#   0 => user specified (must modify bcN.f to use this option)
#   1 => extrapolation (non-reflecting outflow)
#   2 => periodic (must specify this at both boundaries)
#   3 => solid wall for systems where q(2) is normal velocity

clawdata.mthbc_xlower = 1
clawdata.mthbc_xupper = 1
```

For inflow and other options, you must modify bc1.f

Accuracy, stability, and convergence: how do we ensure confidence in the code?

Numerical computation is still an art; there are many things that can go wrong in a calculation.

It is important to be suspicious of peculiar behaviour, but even if things *look* right, they may not *be* right!

Code systems are subject to a variety of tests to ensure correct behaviour:

Verification: test the code solutions against analytical solutions for simple problems.

Validation: test the code solutions against experimental results.

Convergence: prove that the code solution converges to the correct solution as the grid is refined.

Uncertainty Quantification: develop confidence measures that tell us how much we can rely on the results.

Accuracy and Convergence (Chapter 8 in Leveque)

Sources of error

Sources of error

Truncation error: Your functional representation is incomplete; you can't include all the terms in the Taylor series, for example.

Sources of error

Truncation error: Your functional representation is incomplete; you can't include all the terms in the Taylor series, for example.

Roundoff error: The computer's representation of a real number is imprecise; least significant digits are dropped. *Beware subtracting numbers of similar magnitude!*

Numerical diffusion, numerical viscosity: The finite resolution of a gridded calculation fails to resolve fine physical structures, which then diffuse on the scale of the grid resolution.

Sources of error

Truncation error: Your functional representation is incomplete; you can't include all the terms in the Taylor series, for example.

Roundoff error: The computer's representation of a real number is imprecise; least significant digits are dropped. *Beware subtracting numbers of similar magnitude!*

Sources of error

Truncation error: Your functional representation is incomplete; you can't include all the terms in the Taylor series, for example.

Roundoff error: The computer's representation of a real number is imprecise; least significant digits are dropped. *Beware subtracting numbers of similar magnitude!*

Numerical diffusion, numerical viscosity: The finite resolution of a gridded calculation fails to resolve fine physical structures, which then diffuse on the scale of the grid resolution.

Imperfect boundary conditions: Outgoing waves may partially reflect back into the grid, causing distortions.

Sources of error

Truncation error: Your functional representation is incomplete; you can't include all the terms in the Taylor series, for example.

Roundoff error: The computer's representation of a real number is imprecise; least significant digits are dropped. *Beware subtracting numbers of similar magnitude!*

Numerical diffusion, numerical viscosity: The finite resolution of a gridded calculation fails to resolve fine physical structures, which then diffuse on the scale of the grid resolution.

Imperfect boundary conditions: Outgoing waves may partially reflect back into the grid, causing distortions.

The most obvious, and easiest to deal with are the following:

Errors in setup, incorrect initial conditions or material properties
Coding errors (bugs)

The global error: norms

The global error $E^N = Q^N - q^N$ is computed from the errors at each grid cell by using an appropriate norm. A p-norm is defined by:

$$\|E\|_p = \left(\Delta x \sum_i |E_i|^p \right)^{1/p}.$$

The most useful norms are the 1-norm (corresponding to the solution integrals) the 2-norm (the root-mean-square error) and the infinity-norm, or maximum overall error, which measures point-wise convergence:

$$\|E\|_\infty = \max_i |E_i|.$$

Point-wise convergence is generally impossible when the solutions are not smooth.

Measures of error

In a finite volume method, the numerical solution value Q_i^n in grid cell i is an approximation to the cell integral of the true solution

$$q_i^n = \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} q(x, t_n)\, dx.$$

If the solution is sufficiently smooth, this is also reasonably close to the value $q(x_i, t_n)$ at the cell centre. We may choose to compare Q_i^n with either of q_i^n or $q(x_i, t_n)$.

Errors tend to grow with time, so we need to specify the time at which the error comparison of the computed solution against the calculated solution is made. If the comparison is done at time $T = N\Delta t$, we denote the global error as

$$E^N = Q^N - q^N.$$

We will assume that the ratio $\frac{\Delta t}{\Delta x}$ is constant as we refine the grid.

Convergence and Accuracy

There's some divergence of terminology in this area.

Leveque gives the following definition for *convergence*:

$$\lim_{\Delta t \to 0} \|E^N\| = 0 \text{ at time } T \text{ where } N\Delta t = T,$$

and he writes that a method is *accurate to order s* if

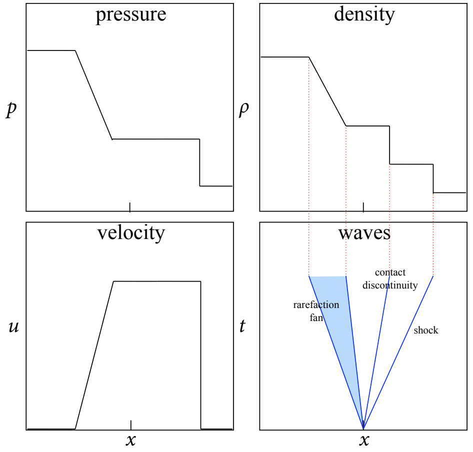
$$\|E^N\| = \mathcal{O}(\Delta t^s) \text{ as } \Delta t \to 0.$$

Very often you will hear people speak of a method as *convergent to order s* if

$$\frac{\|E\|_{k\Delta x}}{\|E\|_{\Delta x}} \sim k^s,$$

where $\|E\|_{\Delta x}$ is the error evaluated on a grid with spacing Δx. This latter notion enables convergence (or resolution) studies in the absence of knowledge of the true solution, but invites the danger that a method may be validated while converging to the wrong solution!

Variants of the classic shock tube problem are used for code validation

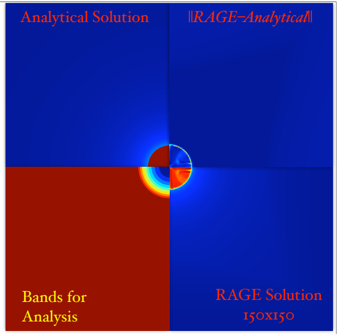


The one-dimensional shock tube problem that we've seen before.

Above is the exact analytical solution (resolved to a finite grid) for a *radial* shock tube in cylindrical geometry, showing 1/4 of the system, colours indicating density.

We ran simulations of this with the Sage/Rage code at Los Alamos.

An accuracy and convergence study on an analytical test problem

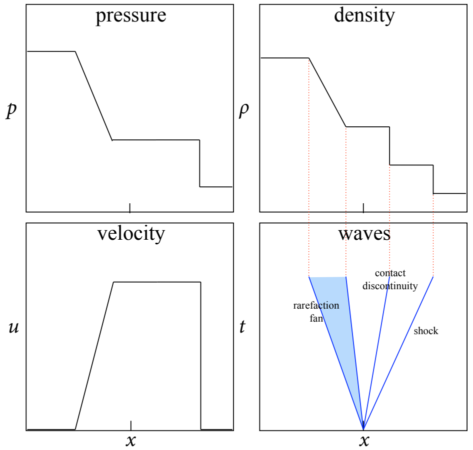


A calculation of the Noh spherical shock tube problem in two dimensions with the Rage code (Los Alamos, SAIC).

We used the *p*-1 norm, summed in bands instead of over the whole domain, to illustrate differences in convergence in different parts of the solution space.

Outside the shock we saw 2[nd] order convergence, as expected from the algorithm, but within the shock the convergence was at best linear. The position of the shock was accurately determined, and it was sharp at all resolutions.

Variants of the classic shock tube problem are used for code validation

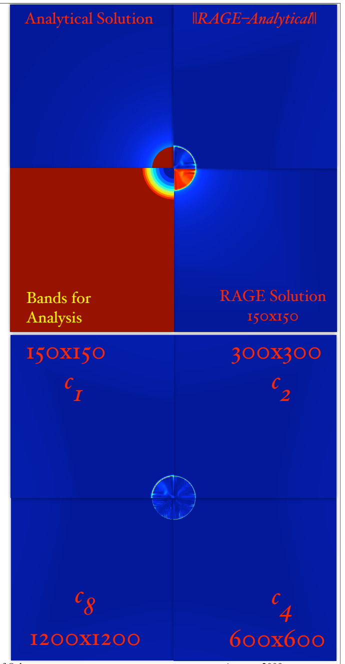


The one-dimensional shock tube problem that we've seen before.

Above is the exact analytical solution (resolved to a finite grid) for a *radial* shock tube in cylindrical geometry, showing 1/4 of the system, colours indicating density.

We ran simulations of this with the Sage/Rage code at Los Alamos.

An accuracy and convergence study on an analytical test problem



A calculation of the Noh spherical shock tube problem in two dimensions with the Rage code (Los Alamos, SAIC).

We used the *p*-1 norm, summed in bands instead of over the whole domain, to illustrate differences in convergence in different parts of the solution space.

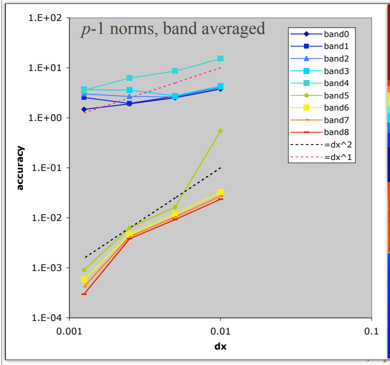
Outside the shock we saw 2[nd] order convergence, as expected from the algorithm, but within the shock the convergence was at best linear. The position of the shock was accurately determined, and it was sharp at all resolutions.

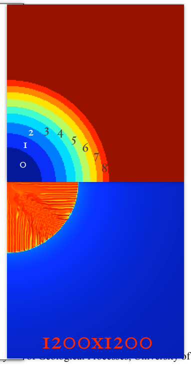
An accuracy and convergence study on an analytical test problem

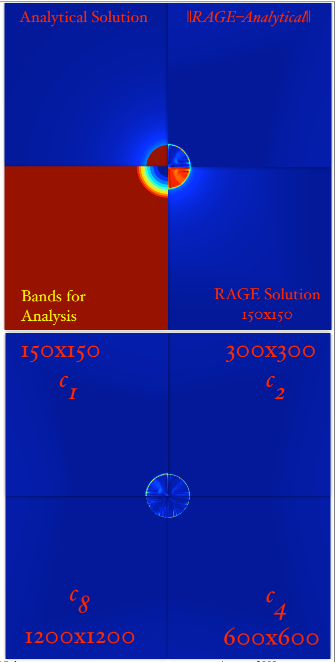
A calculation of the Noh spherical shock tube problem in two dimensions with the Rage code (Los Alamos, SAIC).

We used the p-1 norm, summed in bands instead of over the whole domain, to illustrate differences in convergence in different parts of the solution space.

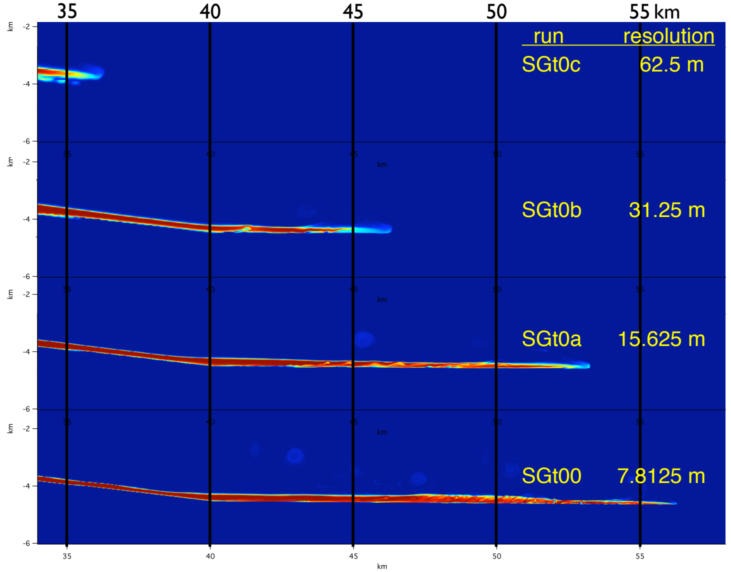
Outside the shock we saw 2nd order convergence, as expected from the algorithm, but within the shock the convergence was at best linear. The position of the shock was accurately determined, and it was sharp at all resolutions.



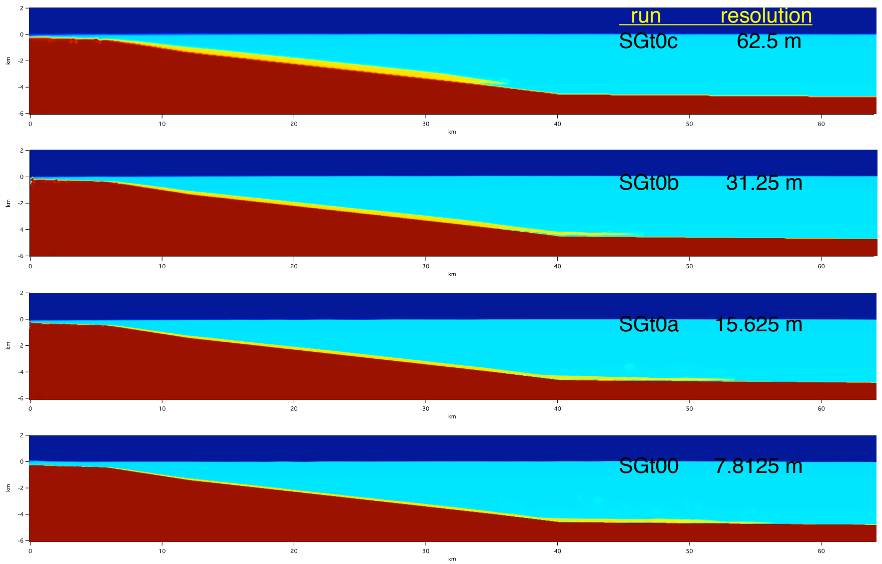




A resolution (not convergence!) study



A resolution (not convergence!) study



Consistency, robustness, and numerical stability

Errors occur at each time step as a simulation progresses. These are inevitable because of the approximations involved (*truncation error*) and because of machine accuracy (*round-off error*).

A method is *consistent* if the error introduced at a single time step is small.

Small single-step errors affect the calculation subsequently. After hundreds or thousands of time steps, these errors accumulate and the numerical solution diverges from the true solution.

The divergence will be *bounded* if the method is *stable*. If the method is unstable, the errors grow exponentially.

The method is *robust* if small changes in the input parameters produce reasonably small changes in the solution after many time steps.

Robustness is tested by running many simulations with different inputs; consistency and stability can be checked analytically.

The fundamental theorem of numerical methods

The Lax-Richtmeyer equivalence theorem states that:

A numerical method for a linear differential equation will converge if that method is *consistent* and *stable*.

Consistency means the single-step error is small.

Stability means that single-step errors do not grow catastrophically with time.

Assessing stability for linear methods

If at time step n we have an error $E^n = Q^n - q^n$, then at time step $n+1$ our error will be

$$E^{n+1} = \mathcal{N}(q^n + E^n) - q^{n+1}$$
$$= \underbrace{\mathcal{N}(q^n + E^n) - \mathcal{N}(q^n)}_{\text{propagation of previous errors}} + \underbrace{\Delta t \tau^n}_{\text{single-step error}} .$$

If the numerical solution operator $\mathcal{N}$ is *contractive,* that is if

$$\|\mathcal{N}(P) - \mathcal{N}(Q)\| \le \|P - Q\|$$

in an appropriate norm, then the operator is stable in that norm, and we get a bound on the global error:

$$\|E^N\| \le \|E^0\| + \sum_{n=1}^{N} \Delta t \|\tau^n\|.$$

Unfortunately, it may be difficult to prove contractivity for nonlinear systems. In that case, we insist on Total Variation Diminishing methods, which accomplishes the same thing.

Single-step errors

A single step of a numerical method can be written as

$$Q^{n+1} = \mathcal{N}(Q^n),$$

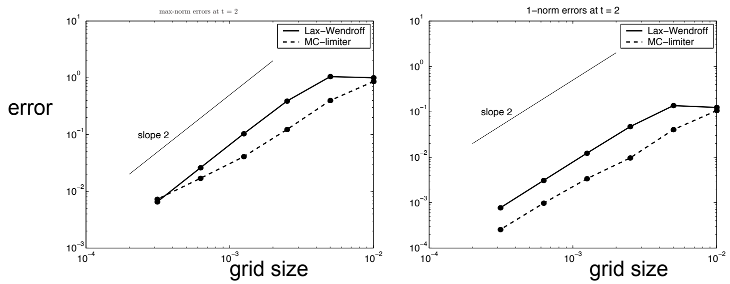
where the operator $\mathcal{N}$ maps the approximate solution at one time step to the next time step. We define the local truncation error by applying the same operator to the *true* solution at time step n and comparing it to the true solution (resolved to the grid) at time step $n+1$:

$$\text{truncation error} = \tau^n = \frac{1}{\Delta t}\mathcal{N}(q^n) - q^{n+1}.$$

If $\lim_{\Delta t \to 0} \tau^n = 0$ then the method is consistent with the differential equation.

The truncation error is calculated by doing a Taylor series expansion and estimating the size of all the terms that are omitted from the numerical approximation.

The formal order of accuracy isn't a guarantee of true accuracy!



Comparing the 2nd-order accurate Lax-Wendroff with a high-resolution method of lower *formal* order: the high-resolution method is more accurate for all but the finest grids in the max norm.

Variable Coefficients
(Chapter 9 in Leveque)

We'll look at the acoustics equations with variable coefficients

The equation is $q_t + A(x)q_x = 0$

with $q(x,t) = \begin{bmatrix} p(x,t) \\ u(x,t) \end{bmatrix}$ $A(x) = \begin{bmatrix} 0 & K(x) \\ 1/\rho(x) & 0 \end{bmatrix}$

and the eigensystem is

$$R(x) = \begin{bmatrix} -Z(x) & Z(x) \\ 1 & 1 \end{bmatrix} \quad \Lambda(x) = \begin{bmatrix} -c(x) & 0 \\ 0 & c(x) \end{bmatrix}$$

where

$$c(x) = \sqrt{\frac{K(x)}{\rho(x)}} \qquad Z(x) = \rho(x)c(x) = \sqrt{K(x)\rho(x)}$$

A brief-look at variable-coefficient problems

Before we get into nonlinear equations and systems, it's useful to consider systems of the form:

$$q_t + A(x)q_x = 0$$

or the similar conservation law:

$$q_t + \left(A(x)q\right)_x = 0$$

or the capacity-function form:

$$\kappa(x)q_t + \left(A(x)q\right)_x = 0$$

Real examples include: advection of a tracer in a pipe with variable cross-section, two conveyor belts with different speeds, traffic flow on a highway with a sudden change in the speed limit, or linear acoustics in a layered medium.

Piece-wise constant coefficients

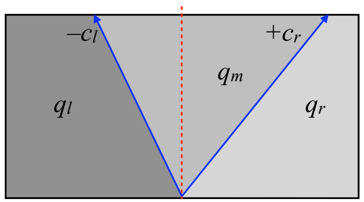
We consider a medium with a discontinuity in the coefficients:

$$\rho(x) = \begin{cases} \rho_l & \text{if } x < 0, \\ \rho_r & \text{if } x > 0, \end{cases} \qquad K(x) = \begin{cases} K_l & \text{if } x < 0, \\ K_r & \text{if } x > 0. \end{cases}$$

and consider it with the piecewise constant initial data

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

Much as before, we have the Riemann problem illustrated here:



Resolve the discontinuity, using the appropriate coefficients

For each wave the jump is:

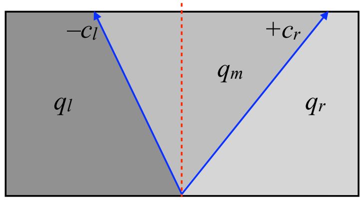
$$q_m - q_l = \alpha^1 \begin{bmatrix} -Z_l \\ 1 \end{bmatrix} \text{ and } q_r - q_m = \alpha^2 \begin{bmatrix} Z_r \\ 1 \end{bmatrix},$$

and the total discontinuity is therefore

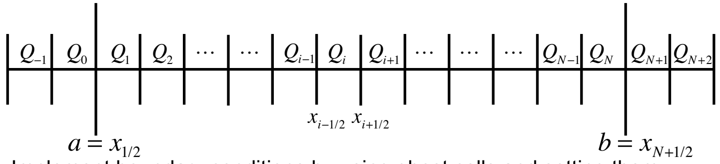
$$q_r - q_l = \alpha^1 \begin{bmatrix} -Z_l \\ 1 \end{bmatrix} + \alpha^2 \begin{bmatrix} Z_r \\ 1 \end{bmatrix}.$$

We can define the special mixed eigenvector matrix $R_{lr} = \begin{bmatrix} -Z_l & Z_r \\ 1 & 1 \end{bmatrix}$,

then $R_{lr}\alpha = q_r - q_l$.



Review: Boundary Conditions



Implement boundary conditions by using ghost cells and setting them appropriately.

Then treat the boundary cells with the normal Riemann solution:

$$Q_1 - Q_0 = \sum_{p=1}^{m} \mathcal{W}_{1/2}^p,$$

with incoming waves having strength zero. The outgoing waves may have nonzero strength.

The ghost cells are reset at the start of the next cycle.

Demonstration of variable coefficient acoustics

?

in $CLAW/book/chap9/acoustics/interface

Review: Accuracy and Convergence

A method is *consistent* if the single-step error is small.

Single-step errors include *truncation error* and (*round-off error*).

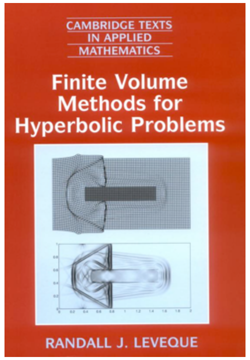
Small single-step errors accumulate and the numerical solution diverges from the true solution after many time steps.

If the method is *stable*, the divergence will be *bounded*. If the method is unstable, the errors grow exponentially.

The Lax-Richtmeyer equivalence theorem states that:

A numerical method for a linear differential equation will converge if that method is *consistent* and *stable*.

Assignment for next time



Read Chapter 7, Chapter 8, and 9.0-9.2, 9.6-9.9 and more of Chapter 9 if interested.

Work problems 7.2a and 8.2. Hand them in to me by Tuesday 29 September.

Next: Nonlinear Conservation Laws (Ch 11)