Einstein Toolkit Workshop

Numerical methods for spacetime and matter evolution

Peter Diener & Bruno Mundim

Apr 4 2012

Outline

- The BSSN formulation.
- Puncture data.
- Finite differencing.
- The method of lines.
- McLachlan.
- GRMHD - Valencia Formulation.
- Conservation Laws.
- Finite Volume Methods.
- Riemann Problem.
- Example: Burgers Equation.
- Approximate Riemann Solvers.
- Slope Limiters
- Algorithm Overview.
- Shocktube Test.

The BSSN formulation

The 3+1 ADM evolution equations are

$$(\partial_t - \mathcal{L}_\beta)\,\gamma_{ij} = -2\alpha K_{ij}, \tag{1}$$

$$(\partial_t - \mathcal{L}_\beta)\,K_{ij} = -D_i D_j\alpha + \alpha(R_{ij} + K K_{ij} - 2K_{ik}K^k{}_j), \tag{2}$$

and the constraints are

$$\mathcal{H} \equiv R + K^2 - K_{ij}K^{ij} = 0, \tag{3}$$

$$\mathcal{M}^i \equiv D_j(K^{ij} - \gamma^{ij}K) = 0. \tag{4}$$

These equations were used a lot in the early years of numerical relativity. However, it was later discovered that the ADM evolution equations are only weakly hyperbolic.
In the mid to late 90's a new formulation was introduced that proved to be much more robust and stable: The BSSN formulation.

The BSSN formulation (continued)

Introduce a conformal rescaling of the three metric

$$\gamma_{ij} = \psi^4 \tilde{\gamma}_{ij}. \tag{5}$$

We choose $\psi = \gamma^{1/12}$ such that the determinant of $\tilde{\gamma}_{ij}$ is 1

$$\det(\tilde{\gamma}_{ij}) = \det(\psi^{-4}\gamma_{ij}) = \det(\gamma^{-1/3}\gamma_{ij}) = \gamma^{-1}\det(\gamma_{ij}) = 1. \tag{6}$$

In addition we introduce a trace decomposition of the extrinsic curvature.

$$K = \gamma^{ij} K_{ij}, \tag{7}$$

$$A_{ij} = K_{ij} - \frac{1}{3}\gamma_{ij} K. \tag{8}$$

We then promote the following variables to evolution variables

$$\phi = \ln\psi = \frac{1}{12}\ln\gamma, \tag{9}$$

$$K = \gamma_{ij} K^{ij}, \tag{10}$$

$$\tilde{\gamma}_{ij} = e^{-4\phi}\gamma_{ij}, \tag{11}$$

$$\tilde{A}_{ij} = e^{-4\phi} A_{ij}. \tag{12}$$

The BSSN formulation (continued)

We finally additionally promote the conformal connection functions

$$\tilde{\Gamma}^i = \tilde{\gamma}^{jk}\tilde{\Gamma}^i{}_{jk} = -\partial_j\tilde{\gamma}^{ij}, \tag{13}$$

to evolved variables as well.
The final set of evolution variables are ϕ, K, $\tilde{\gamma}_{ij}$, $\tilde{A}_{ij}$ and $\tilde{\Gamma}^i$.
The BSSN evolution equations can be derived from the ADM equations.
As an example take the equation for ϕ.

$$\left(\partial_t - \mathcal{L}_\beta\right)\phi = \partial_0\phi = \partial_0\left(\frac{1}{12}\ln\gamma\right) = \frac{1}{12}\frac{1}{\gamma}\partial_0\gamma. \tag{14}$$

Using the expression for the derivative of the determinant of the metric in terms of the derivatives of the metric ($\partial_0\gamma = \gamma\gamma^{ij}\partial_0\gamma_{ij}$) we find

$$\left(\partial_t - \mathcal{L}_\beta\right)\phi = \partial_0\phi = \frac{1}{12}\gamma^{ij}\partial_0\gamma_{ij} = \frac{1}{12}\gamma^{ij}(-2\alpha K_{ij}) = -\frac{1}{6}\alpha K. \tag{15}$$

The BSSN formulation (continued)

The evolution equation for all the BSSN variables are[1]

$$\partial_t \tilde{\gamma}_{ij} = -2\alpha \tilde{A}_{ij} + \beta^k \partial_k \tilde{\gamma}_{ij} + \tilde{\gamma}_{ik} \partial_j \beta^k + \tilde{\gamma}_{jk} \partial_i \beta^k - \frac{2}{3} \tilde{\gamma}_{ij} \partial_k \beta^k, \tag{16}$$

$$\partial_t \phi = -\frac{1}{6} \alpha K + \beta^k \partial_k \phi + \frac{1}{6} \partial_k \beta^k, \tag{17}$$

$$\begin{aligned} \partial_t \tilde{A}_{ij} = {}& e^{-4\phi} [-D_i D_j \alpha + \alpha R_{ij}]^{TF} + \alpha (K \tilde{A}_{ij} - 2 \tilde{A}_{ik} \tilde{A}^k{}_j) \\ & + \beta^k \partial_k \tilde{A}_{ij} + \tilde{A}_{ik} \partial_j \beta^k + \tilde{A}_{jk} \partial_i \beta^k - \frac{2}{3} \tilde{A}_{ij} \partial_k \beta^k, \end{aligned} \tag{18}$$

$$\partial_t K = -D^i D_i \alpha + \alpha (\tilde{A}_{ij} \tilde{A}^{ij} + \frac{1}{3} K^2) + \beta^k \partial_k K, \tag{19}$$

$$\begin{aligned} \partial_t \tilde{\Gamma}^i = {}& \tilde{\gamma}^{jk} \partial_j \partial_k \beta^i + \frac{1}{3} \tilde{\gamma}^{ij} \partial_j \partial_k \beta^k + \beta^j \partial_j \tilde{\Gamma}^i - \tilde{\Gamma}^j \partial_j \beta^i + \frac{2}{3} \tilde{\Gamma}^i \partial_j \beta^j \\ & - 2 \tilde{A}^{ij} \partial_j \alpha + 2\alpha (\tilde{\Gamma}^i{}_{jk} \tilde{A}^{jk} + 6 \tilde{A}^{ij} \partial_j \phi - \frac{2}{3} \tilde{\gamma}^{ij} \partial_j K). \end{aligned} \tag{20}$$

[1]Note: the matter terms are left out.

The BSSN formulation (continued)

Here $R_{ij} = \tilde{R}_{ij} + R^{\phi}_{ij}$, where

$$\begin{aligned} R^{\phi}_{ij} = & -2\tilde{D}_i\tilde{D}_j\phi - 2\tilde{\gamma}_{ij}\tilde{D}^k\tilde{D}_k\phi + 4\tilde{D}_i\phi\,\tilde{D}_j\phi - 4\tilde{\gamma}_{ij}\tilde{D}^k\phi\,\tilde{D}_k\phi, \qquad (21) \\ \tilde{R}_{ij} = & -\frac{1}{2}\tilde{\gamma}^{lm}\partial_l\partial_m\tilde{\gamma}_{ij} + \tilde{\gamma}_{k(i}\partial_{j)}\tilde{\Gamma}^k + \tilde{\Gamma}^k\tilde{\Gamma}_{(ij)k} \\ & + \tilde{\gamma}^{lm}\left(2\tilde{\Gamma}^k{}_{l(i}\tilde{\Gamma}_{j)km} + \tilde{\Gamma}^k{}_{im}\tilde{\Gamma}_{klj}\right). \qquad (22) \end{aligned}$$

The BSSN formulation (continued)

The constraints are

$$\tilde{\mathcal{H}} \equiv R + \frac{2}{3}K^2 - \tilde{A}_{ij}\tilde{A}^{ij} = 0, \tag{23}$$

$$\tilde{\mathcal{M}}^i \equiv \tilde{D}_j\tilde{A}^{ij} + 6\tilde{A}^{ij}\partial_j\phi - \frac{2}{3}\tilde{\gamma}^{ij}\partial_j K = 0, \tag{24}$$

$$\tilde{\mathcal{G}} \equiv \tilde{\gamma} - 1 = 0, \tag{25}$$

$$\tilde{\mathcal{A}} \equiv \tilde{\gamma}^{ij}\tilde{A}_{ij} = 0, \tag{26}$$

$$\tilde{\mathcal{L}}^i \equiv \tilde{\Gamma}^i + \partial_j\tilde{\gamma}^{ij} = 0. \tag{27}$$

The constraints $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{A}}$ are enforced actively at each timestep.
The other constraints ($\tilde{\mathcal{H}}$, $\tilde{\mathcal{M}}^i$ and $\tilde{\mathcal{L}}^i$) are not enforced.
To improve stability and to help maintain $\tilde{\mathcal{L}}^i$ at a low level the following rule is employed in an implementation

- Where derivatives of $\tilde{\Gamma}^i$ are needed the evolved $\tilde{\Gamma}^i$ are used directly.
- Where $\tilde{\Gamma}^i$ are needed without taking derivatives $\tilde{\gamma}^{jk}\tilde{\Gamma}^i{}_{jk}$ are used instead.

The BSSN formulation (continued)

In order to evolve a spacetime with the BSSN equations, you have to specify the gauges α and β^i.
Most codes use the “moving puncture” gauges

$$\partial_t \alpha = -2\alpha K + \beta^i \partial_i \alpha, \tag{28}$$

$$\partial_t \beta^i = \frac{3}{4} B^i + \beta^j \partial_j \beta^i, \tag{29}$$

$$\partial_t B^i = \partial_t \tilde{\Gamma}^i - \eta B^i + \beta^j \partial_j (B^i - \tilde{\Gamma}^i) \tag{30}$$

or slight variations thereof for black hole spacetime evolutions.

Puncture data

The puncture approach to binary black hole initial data is

$$\gamma_{ij}^{\mathrm{ph}} = \psi^4 \gamma_{ab}, \qquad K_{ij}^{\mathrm{ph}} = \psi^{-2} K_{ij}, \tag{31}$$

where γ_{ij} is chosen to be the flat metric and K_{ij} is assumed to be tracefree. The constraint equations become

$$0 = \Delta\psi + \frac{1}{8} K^{ij} K_{ij} \psi^{-7} \tag{32}$$

$$0 = D_j K^{ij}. \tag{33}$$

The momentum constraint has an analytic solution

$$K_{\mathrm{BY}}^{ij} = \frac{3}{2r^2}\left(P^i n^j + P^j n^i - (\gamma^{ij} - n^i n^j) P^k n_k\right) + \frac{3}{r^3}\left(\epsilon^{ikl} S_k n_l n^j + \epsilon^{jkl} S_k n_l n^i\right). \tag{34}$$

P^i and S^i are the linear momentum and spin of the black hole.

Puncture data (continued)

For $K_{ij} = 0$ the Hamiltonian constraint has a simple solution for N black holes.

$$\psi = 1 + \sum_{i=1}^{N} \frac{m_i}{2|\vec{r} - \vec{r}_i|}. \tag{35}$$

Inspired by this, for $K_{ij} \neq 0$ we make the ansatz

$$\psi = \frac{1}{\alpha} + u, \qquad \frac{1}{\alpha} = \sum_{i=1}^{N} \frac{m_i}{2|\vec{r} - \vec{r}_i|}, \tag{36}$$

In which case the Hamiltonian constraint becomes an equation for u

$$\Delta u + \frac{1}{8}\alpha^7 K^{ij} K_{ij} (1 + \alpha u)^{-7} = 0. \tag{37}$$

It can be shown that u will be C^2 at the location of the 'punctures' and that ψ has a unique solution.

Finite differencing

With finite differencing we discretize a function by sampling it at a collection of grid points.
The grid points are usually (but not necessarily) equally spaced.
We can then approximate derivatives of a function at a grid point by a weigthed sum of function values at grid points in the neighbourhood (the stencil) of the grid point.
As an example consider a stencil containing the grid point (f_i) and it's two nearest neighbors (f_{i-1} and f_{i+1}) with $\Delta x = x_{i+1} - x_i = x_i - x_{i-1}$.

$$\left.\frac{df}{dx}\right|_{x_i} \approx \frac{1}{\Delta x} \sum_{j=-1}^{j=1} a_j f_{i+j}. \qquad (38)$$

Finite differencing (continued)

The coefficients a_j can be found be expanding f in a Taylor series around x_i for the grid points in the stencil

$$\begin{aligned}
f(x_{i-1}) &= f(x_i) - \left.\frac{df}{dx}\right|_{x_i} \Delta x + \frac{1}{2} \left.\frac{d^2 f}{dx^2}\right|_{x_i} \Delta x^2 + O((\Delta x)^3) \\
f(x_i) &= f(x_i) \\
f(x_{i+1}) &= f(x_i) + \left.\frac{df}{dx}\right|_{x_i} \Delta x + \frac{1}{2} \left.\frac{d^2 f}{dx^2}\right|_{x_i} \Delta x^2 + O((\Delta x)^3)
\end{aligned}$$

Requiring that the weighted sum approximates the derivative yields the following equations for a_{-1}, a_0 and a_1

$$\begin{aligned}
0 &= a_{-1} + a_0 + a_1 \\
1 &= -a_{-1} + a_1 \\
0 &= a_{-1} + a_1
\end{aligned}$$

with the solution $a_{-1} = -1/2, a_0 = 0, a_1 = 1/2$.

Finite differencing (continued)

Thus we find that

$$\left.\frac{df}{dx}\right|_{x_i} = \frac{f_{i+1} - f_{i-1}}{2\Delta x} + O((\Delta x)^2). \tag{39}$$

Similarly we find for the second derivative that

$$\left.\frac{d^2 f}{dx^2}\right|_{x_i} = \frac{f_{i+1} - 2f_i + f_{i-1}}{(\Delta x)^2} + O((\Delta x)^2). \tag{40}$$

These finite difference operators are second order accurate.
Higher order accuracy or higher order derivatives require larger stencils.
Another way of looking at finite differencing operators is through interpolating polynomials.
Either approach gives the same coefficients for the same stencil.
It is also clear from either approach that the error estimates are only correct if f is smooth enough.

The method of lines

Consider the set of hyperbolic PDE's

$$\partial_t \mathbf{q} + \mathbf{A}^i(\mathbf{q})\partial_i \mathbf{B}(\mathbf{q}) = \mathbf{S}(\mathbf{q}). \tag{41}$$

The idea then is to discretize in space first, i.e. write the equations as

$$\partial_t \mathbf{q} = \mathbf{L}(\mathbf{q}), \tag{42}$$

where $\mathbf{L}(\mathbf{q})$ is a discrete approximation to the equations (e.g. using finite differencing).
This then turns the equations into a set of coupled ODE's with respect to time.
If the spatial discretization (including boundary conditions) is stable we we can then evolve the system of equations using any stable ODE time integrator.
Often Runge-Kutta integration schemes are the scheme of choice.

The method of lines (continued)

Advantages of using the method of lines:

- It is easy to change the time integration scheme (e.g. going to higher order).
- It is easy to couple different evolution equations maintaining high order coupling.

The method of lines is implemented in Cactus in the thorn `CactusNumerical/MoL`.To use:

- Schedule a routine to register the evolution variables and RHS variables with `MoL`.
- Schedule routines to set the RHS variables.
- Set parameters when launching job to choose the time integration scheme.

McLachlan

`McLachlan` is the Einstein Toolkit implementation of BSSN.

`McLachlan` is named in honor of the Canadian Singer/Song writer Sarah McLachlan.

It is based on finite differencing and the method of lines.

It supports high order finite differencing (8th order).

Since it is generated from the tensor equations by Kranc it is easy to maintain and modify.

It is highly optimized (supports vectorization and OpenMP) and an effort is ongoing to make it be able to run on GPU's as well.

In the future it may be necessary to generate different versions optimized for specific computer architectures.

GRMHD - Valencia Formulation

Once we have established the spacetime evolution and initial data equations, we need to obtain the evolution equations for the matter fields and the magnetic field evolution equation in ideal MHD case.
These equations can be expressed as the local conservation laws of baryon number and energy momentum. For baryon number we have:

$$\nabla_\nu J^\nu = 0, \tag{43}$$

where $J^\mu = \rho u^\mu$ is the rest-mass current, ρ the rest-mass density and u^μ is the four-velocity of a fluid comoving observer.
The conservation of energy-momentum is given by:

$$\nabla_\nu T^{\mu\nu} = 0, \tag{44}$$

where $T^{\mu\nu}$ for a perfect fluid is given by:

$$T^{\mu\nu}_{\mathrm{Fluid}} = \rho h u^\mu u^\nu + p g^{\mu\nu}, \tag{45}$$

where $g_{\mu\nu}$ is the metric, p is the pressure, and h is the specific enthalpy, defined by $h = 1 + \varepsilon + p/\rho$, ε being the specific internal energy.

GRMHD - Valencia Formulation

In the case of a magnetized fluid in the ideal MHD approximation, the stress-energy-momentum tensor can be written as:

$$T^{\mu\nu} = \rho h^* u^\mu u^\nu + p^* g^{\mu\nu} - b^\mu b^\nu, \tag{46}$$

where we define $p^* = p + b^2/2$, $h^* = h + b^2/\rho$, $\varepsilon^* = \varepsilon + b^2/(2\rho)$, that results into $h^* = 1 + \varepsilon^* + p^*/\rho$.
Note that in the expression above we have used the magnetic field b^μ as measured by the comoving observer. It can expressed in terms of the magnetic field measured by the Eulerian observer, B^i as:

$$b^0 = \frac{W B^i v_i}{\alpha} \tag{47}$$

$$b^i = \frac{B^i + \alpha b^0 u^i}{W} . \tag{48}$$

Finally, the modulus of the magnetic field can be written as

$$b^2 = \frac{B^2 + \alpha^2 (b^0)^2}{W^2} , \tag{49}$$

GRMHD - Valencia Formulation

If we define the rest-mass density, the momentum density of the magnetized fluid in the *j*-direction, and its total energy density as measured by an Eulerian observer as

$$
\begin{aligned}
D &\equiv -J_\nu n^\nu = \rho W && (50)\\
S_j &\equiv -\mathbf{T}(\mathbf{n}, \mathbf{e}_{(j)}) = \rho h^* W^2 v_j - \alpha b^0 b_j && (51)\\
\tau &\equiv \mathbf{T}(\mathbf{n}, \mathbf{n}) = \rho h^* W^2 - p^* - \alpha^2 (b^0)^2 - D && (52)
\end{aligned}
$$

The GRMHD equations can be cast in conservative form:

$$\frac{1}{\sqrt{-g}}\left(\frac{\partial\sqrt{\gamma}\mathbf{F}^0}{\partial x^0} + \frac{\partial\sqrt{-g}\mathbf{F}^i}{\partial x^i}\right) = \mathbf{S}, \qquad (53)$$

where $\mathbf{F}^0$ is the state vector:

$$\mathbf{F}^0 = \begin{bmatrix} D \\ S_j \\ \tau \\ B^k \end{bmatrix}, \qquad (54)$$

GRMHD - Valencia Formulation

while $\mathbf{F}^i$ are the fluxes

$$\mathbf{F}^i = \left[\begin{array}{c} D\tilde{v}^i \\ S_j \tilde{v}^i + p^* \delta^i_j - b_j B^i / W \\ \tau \tilde{v}^i + p^* v^i - \alpha b^0 B^i / W \\ \tilde{v}^i B^k - \tilde{v}^k B^i \end{array} \right] \qquad (55)$$

with the corresponding sources $\mathbf{S}$ are given by

$$\mathbf{S} = \left[\begin{array}{c} 0 \\ T^{\mu\nu} \left(\frac{\partial g_{\nu j}}{\partial x^\mu} - \Gamma^\delta_{\nu\mu} g_{\delta j} \right) \\ \alpha \left(T^{\mu 0} \frac{\partial \ln \alpha}{\partial x^\mu} - T^{\mu\nu} \Gamma^0_{\nu\mu} \right) \\ 0^k \end{array} \right], \qquad (56)$$

where $\tilde{v}^i = v^i - \frac{\beta^i}{\alpha}$.

Conservation Laws

The simplest one dimensional conservation law (a class of hyperbolic equations):

$$q_t(x,t) + f(q(x,t))_x = \psi(q(x,t)), \tag{57}$$

where $q(x,t)$ is a vector of conserved quantities (densities), $f(q(x,t))$ is known as the flux function, while $\psi(q(x,t))$ are the source terms (they work as a sink or fountain).

Conservation Laws

The simplest one dimensional conservation law (a class of hyperbolic equations):

$$q_t(x,t) + f(q(x,t))_x = \psi(q(x,t)), \tag{57}$$

where $q(x,t)$ is a vector of conserved quantities (densities), $f(q(x,t))$ is known as the flux function, while $\psi(q(x,t))$ are the source terms (they work as a sink or fountain).

Solutions of conservation laws may contain discontinuities such as shock waves.

Conservation Laws

The simplest one dimensional conservation law (a class of hyperbolic equations):

$$q_t(x,t) + f(q(x,t))_x = \psi(q(x,t)), \tag{57}$$

where $q(x,t)$ is a vector of conserved quantities (densities), $f(q(x,t))$ is known as the flux function, while $\psi(q(x,t))$ are the source terms (they work as a sink or fountain).
Solutions of conservation laws may contain discontinuities such as shock waves.
Finite difference methods are expected to break down near discontinuities where the differential equation does not hold.

Conservation Laws

The simplest one dimensional conservation law (a class of hyperbolic equations):

$$q_t(x,t) + f(q(x,t))_x = \psi(q(x,t)), \tag{57}$$

where $q(x,t)$ is a vector of conserved quantities (densities), $f(q(x,t))$ is known as the flux function, while $\psi(q(x,t))$ are the source terms (they work as a sink or fountain).
Solutions of conservation laws may contain discontinuities such as shock waves.
Finite difference methods are expected to break down near discontinuities where the differential equation does not hold.
Integral form of Eq. 57 leads to finite volume methods.

Conservation Laws

The simplest one dimensional conservation law (a class of hyperbolic equations):

$$q_t(x,t) + f(q(x,t))_x = \psi(q(x,t)), \tag{57}$$

where $q(x,t)$ is a vector of conserved quantities (densities), $f(q(x,t))$ is known as the flux function, while $\psi(q(x,t))$ are the source terms (they work as a sink or fountain).

Solutions of conservation laws may contain discontinuities such as shock waves.

Finite difference methods are expected to break down near discontinuities where the differential equation does not hold.

Integral form of Eq. 57 leads to finite volume methods.

Finite Volume Methods

Rather than pointwise approximations at grid points, we break the domain into grid cells and approximate the total integral of q over each grid cell (or the cell average of q).

Finite Volume Methods

Rather than pointwise approximations at grid points, we break the domain into grid cells and approximate the total integral of q over each grid cell (or the cell average of q).
These values are modified in each time step by the fluxes through the edges of the cell.

Finite Volume Methods

Rather than pointwise approximations at grid points, we break the domain into grid cells and approximate the total integral of q over each grid cell (or the cell average of q).

These values are modified in each time step by the fluxes through the edges of the cell.

The primary problem is to calculate the numerical flux functions that approximate reasonably well the correct fluxes, based on the data available: the approximate cell average.

Finite Volume Methods

Let the ith cell be $\mathcal{C}_i = (x_{i-1/2}, x_{i+1/2})$.

Finite Volume Methods

Let the ith cell be $\mathcal{C}_i = (x_{i-1/2}, x_{i+1/2})$.
The average value over the ith interval at time t_n is:

$$Q_i^n \approx \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} q(x, t_n) dx \equiv \frac{1}{\Delta x} \int_{\mathcal{C}_i} q(x, t_n) dx, \tag{58}$$

where $\Delta x = x_{i+1/2} - x_{i-1/2}$ is the cell's length.

Finite Volume Methods

Let the ith cell be $\mathcal{C}_i = (x_{i-1/2}, x_{i+1/2})$.
The average value over the ith interval at time t_n is:

$$Q_i^n \approx \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} q(x, t_n) dx \equiv \frac{1}{\Delta x} \int_{\mathcal{C}_i} q(x, t_n) dx, \tag{58}$$

where $\Delta x = x_{i+1/2} - x_{i-1/2}$ is the cell's length.
The integral form of the conservation law gives:

$$\frac{d}{dt} \int_{\mathcal{C}_i} q(x, t) dx = f(q(x_{i-1/2}, t)) - f(q(x_{i+1/2}, t)) + \int \int_{\mathcal{C}_i} \psi(q(x, t)) dx dt \tag{59}$$

Finite Volume Methods

Integrating Eq.59 in time from t_n to t_{n+1} and dividing by the grid cell interval Δx yields an explicit time-marching algorithm:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(F_{i+1/2}^n - F_{i-1/2}^n) + \Psi_i^n, \tag{60}$$

where $F_{i-1/2}^n$ is some approximation to the average flux along $x = x_{i-1/2}$:

$$F_{i-1/2}^n \approx \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} f(q(x_{i-1/2}, t))dt, \tag{61}$$

and Ψ_i^n is the spacetime average of the sources:

$$\Psi_i^n \approx \frac{1}{\Delta x \Delta t}\int_{t_n}^{t_{n+1}} \int_{\mathcal{C}_i} \psi(q(x_i, t))dxdt. \tag{62}$$

Finite Volume Methods

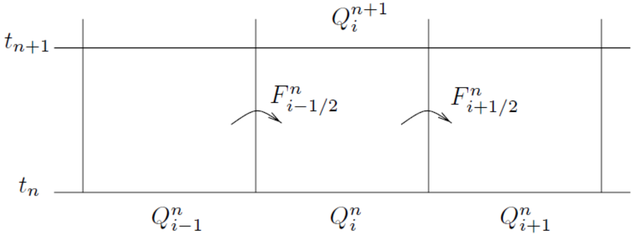

Fig. 4.1. Illustration of a finite volume method for updating the cell average Q_i^n by fluxes at the cell edges. Shown in x–t space.

Riemann Problem

How do we calculate the numerical fluxes at the cell interfaces?

Riemann Problem

How do we calculate the numerical fluxes at the cell interfaces?
By solving a colection of local Rieamann problems!

Riemann Problem

How do we calculate the numerical fluxes at the cell interfaces?
By solving a colection of local Rieamann problems!
This method of approximating the continous solution by a colection of local Riemann problems is also called Godunov's method.

Riemann Problem

How do we calculate the numerical fluxes at the cell interfaces?
By solving a colection of local Rieamann problems!
This method of approximating the continous solution by a colection of local Riemann problems is also called Godunov's method.
The Riemann problem consists of a conservation law equation with a piecewise constant data, with only a single jump discontinuity at some point ($x = 0$ for example):

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

Riemann Problem

How do we calculate the numerical fluxes at the cell interfaces?
By solving a colection of local Rieamann problems!
This method of approximating the continous solution by a colection of local Riemann problems is also called Godunov's method.
The Riemann problem consists of a conservation law equation with a piecewise constant data, with only a single jump discontinuity at some point ($x = 0$ for example):

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

By solving it with $q_l = Q_{i-1}$ and $q_r = Q_i$, we can obtain information that can be used to calculate the numerical flux and update the cell averages over a time step.

Riemann Problem

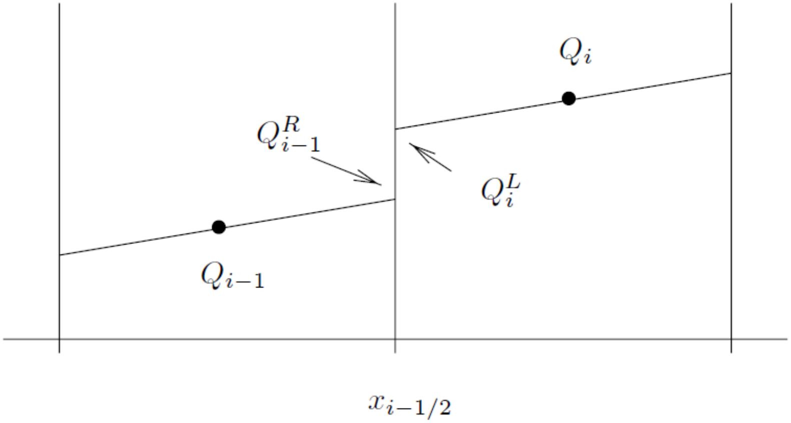

Fig. 5.1. The states used in solving the Riemann problem at the interface $x_{i-1/2}$.

Example: Burgers Equation

$$u_t + f(u)_x = 0, \qquad f(u) = \frac{1}{2}u^2 \tag{63}$$

with

$$u(x,0) = \begin{cases} u_l & \text{if } x < 0 \\ u_r & \text{if } x > 0 \end{cases}$$

Example: Burgers Equation

$$u_t + f(u)_x = 0, \qquad f(u) = \frac{1}{2}u^2 \tag{63}$$

with

$$u(x,0) = \begin{cases} u_l & \text{if } x < 0 \\ u_r & \text{if } x > 0 \end{cases}$$

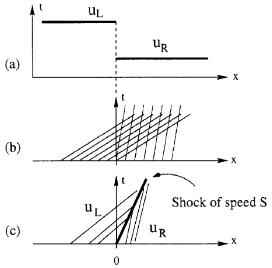

Fig. 2.13. (a) Compressive discontinuous initial data (b) picture of characteristics and (c) solution on x–t plane

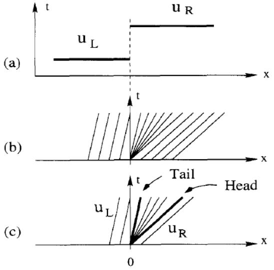

Fig. 2.16. Centred rarefaction wave: (a) expansive discontinuous initial data (b) picture of characteristics (c) entropy satisfying (rarefaction) solution on x–t plane

Approximate Riemann Solvers

Usually the exact solution of the Riemann problem is computationally very expensive.

Approximate Riemann Solvers

Usually the exact solution of the Riemann problem is computationally very expensive.
Fortunately we can obtain very good approximation for the solutions by approximating the conservation law as a quasi-linear system:

$$q_t + Aq_x = 0 \tag{64}$$

Approximate Riemann Solvers

Usually the exact solution of the Riemann problem is computationally very expensive.
Fortunately we can obtain very good approximation for the solutions by approximating the conservation law as a quasi-linear system:

$$q_t + Aq_x = 0 \tag{64}$$

where A is a diagonalizable matrix given by:

$$A(q_l, q_r) = \left.\frac{\partial f}{\partial q}\right|_{q=1/2(q_l+q_R)} \tag{65}$$

Approximate Riemann Solvers

Usually the exact solution of the Riemann problem is computationally very expensive.
Fortunately we can obtain very good approximation for the solutions by approximating the conservation law as a quasi-linear system:

$$q_t + Aq_x = 0 \tag{64}$$

where A is a diagonalizable matrix given by:

$$A(q_l, q_r) = \left.\frac{\partial f}{\partial q}\right|_{q=1/2(q_l+q_R)} \tag{65}$$

Roe solver:

$$F^{\mathrm{Roe}}_{i+1/2} = \frac{1}{2}\left[f(q^r_{i+1/2}) + f(q^l_{i-1/2}) - \sum_\alpha |\lambda_\alpha|\omega_\alpha r_\alpha\right] \tag{66}$$

where λ_α are the characteristics speeds, ω_α the jumps in the characteristics and r_α the right eigenvector of A.

Slope Limiters

How do we calculate the left and right states?

Slope Limiters

How do we calculate the left and right states?
For example if we want to construct a piecewise linear data instead of a piecewise constant data, we can use a slope limiting process:

$$\begin{aligned} q^l_{i+1/2} &= q_i + \sigma_i (x_{i+1/2} - x_i) \quad (67) \\ q^r_{i+1/2} &= q_{i+1} + \sigma_{i+1} (x_{i+1/2} - x_{i+1}) \quad (68) \end{aligned}$$

where $\sigma_i = \mathrm{minmod}(s_{i-1/2}, s_{i+1/2})$ and the slopes:

$$s_{i+1/2} = \frac{q_{i+1} - q_i}{x_{i+1} - x_i} \quad (69)$$

Slope Limiters

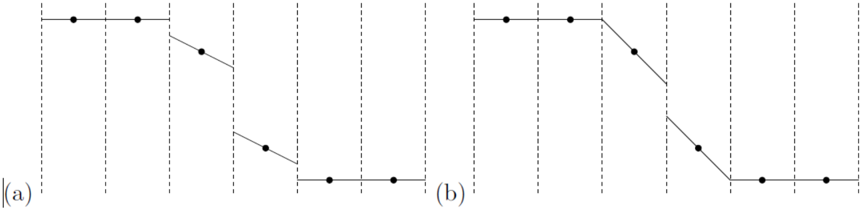

Fig. 6.5. Grid values Q^n and reconstructed $\tilde{q}^n(\cdot, t_n)$ using (a) minmod slopes, (b) superbee or MC slopes. Note that these steeper slopes can be used and still have the TVD property.

Algorithm Overview

- Set boundary conditions or ghost zone values

Algorithm Overview

- Set boundary conditions or ghost zone values
- Calculate numerical flux at every cell interface by solving an approximate Riemann problem

Algorithm Overview

- Set boundary conditions or ghost zone values
- Calculate numerical flux at every cell interface by solving an approximate Riemann problem
    - Reconstruct the primitive variables p_l and p_r at the cell interfaces (by using slope limiters for example).

Algorithm Overview

- Set boundary conditions or ghost zone values
- Calculate numerical flux at every cell interface by solving an approximate Riemann problem
  - Reconstruct the primitive variables p_l and p_r at the cell interfaces (by using slope limiters for example).
  - Find the conservative variables q_l and q_r by using their definition

Algorithm Overview

- Set boundary conditions or ghost zone values
- Calculate numerical flux at every cell interface by solving an approximate Riemann problem
    - Reconstruct the primitive variables p_l and p_r at the cell interfaces (by using slope limiters for example).
    - Find the conservative variables q_l and q_r by using their definition
    - Calculate the fluxes $f(q_l)$ and $f(q_r)$

Algorithm Overview

- Set boundary conditions or ghost zone values
- Calculate numerical flux at every cell interface by solving an approximate Riemann problem
  - Reconstruct the primitive variables p_l and p_r at the cell interfaces (by using slope limiters for example).
  - Find the conservative variables q_l and q_r by using their definition
  - Calculate the fluxes $f(q_l)$ and $f(q_r)$
  - Calculate A, r_α, λ_α and use them to find ω_α

Algorithm Overview

- Set boundary conditions or ghost zone values
- Calculate numerical flux at every cell interface by solving an approximate Riemann problem
  - Reconstruct the primitive variables p_l and p_r at the cell interfaces (by using slope limiters for example).
  - Find the conservative variables q_l and q_r by using their definition
  - Calculate the fluxes $f(q_l)$ and $f(q_r)$
  - Calculate A, r_α, λ_α and use them to find ω_α
  - Calculate the Roe formula for the numerical flux: $F^{\mathrm{Roe}}_{i+1/2}$

Algorithm Overview

- Set boundary conditions or ghost zone values
- Calculate numerical flux at every cell interface by solving an approximate Riemann problem
  - Reconstruct the primitive variables p_l and p_r at the cell interfaces (by using slope limiters for example).
  - Find the conservative variables q_l and q_r by using their definition
  - Calculate the fluxes $f(q_l)$ and $f(q_r)$
  - Calculate A, r_α, λ_α and use them to find ω_α
  - Calculate the Roe formula for the numerical flux: $F^{\mathrm{Roe}}_{i+1/2}$
- Evaluate the sources $\Psi(q)$.

Algorithm Overview

- Set boundary conditions or ghost zone values
- Calculate numerical flux at every cell interface by solving an approximate Riemann problem
  - Reconstruct the primitive variables p_l and p_r at the cell interfaces (by using slope limiters for example).
  - Find the conservative variables q_l and q_r by using their definition
  - Calculate the fluxes $f(q_l)$ and $f(q_r)$
  - Calculate A, r_α, λ_α and use them to find ω_α
  - Calculate the Roe formula for the numerical flux: $F^{\mathrm{Roe}}_{i+1/2}$
- Evaluate the sources $\Psi(q)$.
- Update the average quantities Q^n_i using method of lines.

Algorithm Overview

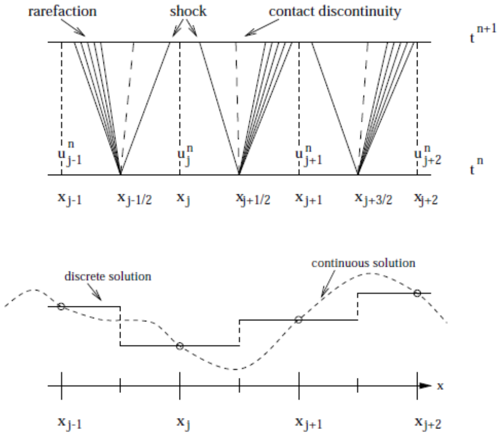

Figure 2: Godunov's scheme: local solutions of Riemann problems. At every interface, $x_{j-\frac{1}{2}}$, $x_{j+\frac{1}{2}}$ and $x_{j+\frac{3}{2}}$, a local Riemann problem is set up as a result of the discretization process (bottom panel), when approximating the numerical solution by piecewise constant data. At time t^n these discontinuities decay into three elementary waves, which propagate the solution forward to the next time level t^{n+1} (top panel). The timestep of the numerical scheme must satisfy the Courant–Friedrichs–Lewy condition, being small enough to prevent the waves from advancing more than $\Delta x/2$ in Δt.

Shocktube Test

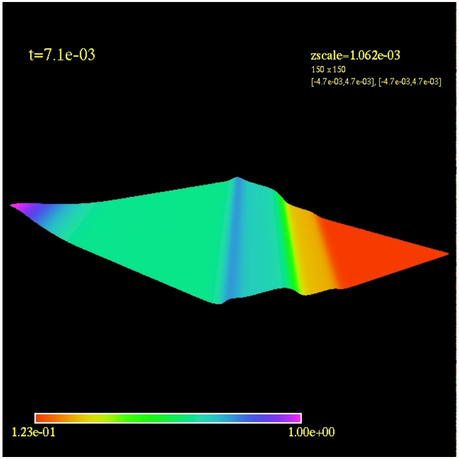

Figure: Diagonal shock along xy-plane: final ρ profile.

References

- Numerical 3+1 general relativistic magnetohydrodynamics: A Local characteristic approach. - by Luis Antón, Olindo Zanotti, Juan A. Miralles, José M. Martí, José M. Ibáñez, José A. Font and José A. Pons - Astrophys.J.637:296-312,2006. e-Print: astro-ph/0506063
- Numerical Hydrodynamics and Magnetohydrodynamics in General Relativity - by José A. Font - Living Rev. Relativity 11, (2008), 7
- Finite Volume Methods for Hyperbolic Problems - by Randall J. LeVeque
- Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction - by Eleuterio F. Toro